TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

SELF-AWARE RELIABLE MONITORING

Maximilian Götzinger

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
SARJA – SER. F OSA – TOM. 5 | TECHNICA – INFORMATICA | TURKU 2021

TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

SELF-AWARE RELIABLE MONITORING

Maximilian Götzinger

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
SARJA – SER F OSA – TOM. 5 | TECHNICA – INFORMATICA | TURKU 2021

University of Turku

Faculty of Technology
Department of Computing
Information and Communication Technology
Doctoral Programme in Mathematics and Computer Sciences

Supervised by

Professor Pasi Liljeberg
Department of Computing
University of Turku, Finland

Professor Axel Jantsch
Institute of Computer Technology
TU Wien, Austria

Associate Professor Amir M. Rahmani
Department of Computer Science
University of California Irvine, USA

Reviewed by

Professor Paul Pop
Department of Applied Mathematics and Computer Science
Technical University of Denmark, Denmark

Professor Martin Törngren
Department of Machine Design
KTH Royal Institute of Technology, Sweden

Opponent

Professor Selma Saidi
Department of Electrical Engineering and Information Technology
Technical University of Dortmund, Germany

ISBN 978-951-29-8617-0 (PRINT)
ISBN 978-951-29-8618-7 (PDF)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)
Painosalama, Turku, Finland, 2021

This work is dedicated to the memory of my family members Erika, Franz, and Vera, who passed away during my doctoral studies and, sadly, cannot share this special moment with me.

UNIVERSITY OF TURKU
Faculty of Technology
Department of Computing
Information and Communication Technology
GÖTZINGER, MAXIMILIAN: Self-Aware Reliable Monitoring
Doctoral dissertation, 239 pp.
Doctoral Programme in Mathematics and Computer Sciences
October 2021

ABSTRACT

Cyber-Physical Systems (CPSs) can be found in almost all technical areas where they constitute a key enabler for anticipated autonomous machines and devices. They are used in a wide range of applications such as autonomous driving, traffic control, manufacturing plants, telecommunication systems, smart grids, and portable health monitoring systems. CPSs are facing steadily increasing requirements such as autonomy, adaptability, reliability, robustness, efficiency, and performance.

A CPS necessitates comprehensive knowledge about itself and its environment to meet these requirements as well as make rational, well-informed decisions, manage its objectives in a sophisticated way, and adapt to a possibly changing environment. To gain such comprehensive knowledge, a CPS must monitor itself and its environment. However, the data obtained during this process comes from physical properties measured by sensors and may differ from the ground truth. Sensors are neither completely accurate nor precise. Even if they were, they could still be used incorrectly or break while operating. Besides, it is possible that not all characteristics of physical quantities in the environment are entirely known. Furthermore, some input data may be meaningless as long as they are not transferred to a domain understandable to the CPS. Regardless of the reason, whether erroneous data, incomplete knowledge or unintelligibility of data, such circumstances can result in a CPS that has an incomplete or inaccurate picture of itself and its environment, which can lead to wrong decisions with possible negative consequences.

Therefore, a CPS must know the obtained data's reliability and may need to abstract information of it to fulfill its tasks. Besides, a CPS should base its decisions on a measure that reflects its confidence about certain circumstances. Computational Self-Awareness (CSA) is a promising solution for providing a CPS with a monitoring ability that is reliable and robust — even in the presence of erroneous data. This dissertation proves that CSA, especially the properties abstraction, data reliability, and confidence, can improve a system's monitoring capabilities regarding its robustness and reliability. The extensive experiments conducted are based on two case studies from different fields: the health- and industrial sectors.

KEYWORDS: Abstraction, Data reliability, Computational Self-Awareness, Confidence, Robust and reliable monitoring

TURUN YLIOPISTO
Teknillinen tiedekunta
Tietotekniikan laitos
Tieto- ja viestintätekniikka
GÖTZINGER, MAXIMILIAN: Self-Aware Reliable Monitoring
Väitöskirja, 239 s.
Matemaattis-tietotekninen tohtoriohjelma
Lokakuu 2021

TIIVISTELMÄ

Kyberfyysiset järjestelmät (engl. Cyber-Physical Systems, CPSs) ovat tärkeitä monilla tekniikan aloilla, joilla käytetään ennakoitavia autonomisia koneita ja laitteita. Niitä sovelletaan laajasti esimerkiksi autonomisessa ajamisessa, liikennevalvonnassa, tehtaissa, telekommunikaatiossa, älykkäissä sähköverkoissa ja terveydenhuollossa. CPS-järjestelmiin kohdistuu lisääntyvässä määrin uusia vaatimuksia liittyen autonomiaan, mukautuvaisuuteen, luotettavuuteen, kestävyyteen, tehokkuuteen ja suorituskykyyn.

CPS-järjestelmä tarvitsee kattavaa tietoa itsestään ja ympäristöstään täyttääkseen nämä vaatimukset ja kyetäkseen tekemään rationaalisia päätöksiä, hallitakseen tavoitteitaan ja mukautuakseen ympäristön mahdollisiin muutoksiin. Tästä johtuen CPS-järjestelmän pitää monitoroida itseään ja ympäristöään. Monitoroinnin aikana kerätty data saattaa kuitenkin pitää sisällään virheitä. Sensorit eivät ole aina täysin tarkkoja ja vaikka ne olisivatkin, niitä voidaan silti käyttää virheellisesti tai ne voivat rikkoontua käytössä. Ympäristön kaikki fyysiset ominaisuudet eivät myöskään ole aina täysin tunnettuja. Lisäksi osa kerätystä datasta voi olla CPS- järjestelmälle merkityksetöntä kunnes data on käsitelty niitä ymmärtävällä hallinta-alueella. Oli syynä virheellinen data tai vaillinainen tieto, tällaiset olosuhteet voivat johtaa tilanteeseen jossa järjestelmällä on epätäydellinen kuva itsestään ja ympäristöstään. Tämä voi johtaa vääriin päätöksiin ja ei toivottuihin vaikutuksiin koko järjestelmälle.

Toimiakseen luotettavasti CPS-järjestelmän täytyy pystyä määrittelemään kerätyn datan luotettavuus ja mahdollisesti suodattaa tarvittavaa tietoa datasta. Lisäksi CPS-järjestelmän tulisi perustaa päätöksensä arviointiin, joka heijastaa sen varmuutta kaikissa olosuhteista. Itsetietoisuus on lupaava ratkaisu tarjoamaan CPS-järjestelmälle monitorointikyky, joka takaa toiminnalle luotettavuuden jopa virheellisen datan ilmaantuessa. Tämä väitöskirja todistaa, että itsetietoisuus, etenkin varmuus, ominaisuuksien erottaminen ja datan luotettavuus, voi parantaa järjestelmän monitorointikykyä liittyen sen kestävyyteen ja luotettavuuteen. Työssä toteutetut käytännön demonstraatiot pohjautuvat kahteen case-tutkimukseen terveys- ja teollisuusaloilta.

ASIASANAT: Autonominen järjestelmä, laskennallinen itsetietoisuus, kyberfyysinen järjestelmä, datan luotettavuus, monitorointi

UNIVERSITY OF TURKU
Faculty of Technology
Department of Computing
Information and Communication Technology
GÖTZINGER, MAXIMILIAN: Self-Aware Reliable Monitoring
Doctoral dissertation, 239 pp.
Doctoral Programme in Mathematics and Computer Sciences
October 2021

KURZFASSUNG

Cyber-Physical Systems (CPSs) spielen in fast allen technischen Bereichen eine wichtige Rolle für die gewünschte Automatisierung von Maschinen und Geräten. Das breite Spektrum ihrer Anwendungen umfasst autonomes Fahren, Verkehrssteuerungen, Produktionsanlagen, Stromnetze, Telekommunikationssysteme und mobile Gesundheitsüberwachungssysteme. Dadurch steigen Anforderungen an CPSs bezüglich ihrer Autonomie, Anpassungsfähigkeit, Zuverlässigkeit, Robustheit, Effizienz und Leistung.

Um diesen Anforderungen gerecht zu werden und auch rationale Entscheidungen treffen sowie Ziele ausgeklügelt verwalten und sich an eine verändernde Umgebung anpassen zu können, muss ein CPS umfassendes Wissen über sich selbst und seine Umgebung haben. Daher muss ein CPS sowohl sich selbst als auch sein Umfeld überwachen. Die hierfür gesammelten Sensordaten spiegeln aber möglicherweise nicht die Wahrheit wider. Sensoren sind weder absolut genau noch präzise. Weiters können sie falsch genutzt werden oder während des Betriebs Schaden nehmen. Möglicherweise sind auch nicht alle physikalischen Eigenschaften der Umgebung vollständig bekannt. Zudem können einige dieser Daten unverständlich sein, solange sie nicht in ein für das CPS verständliches Format übertragen werden. Solche Umstände (eine unvollständige Kenntnis oder fehlerhafte bzw. unverständliche Daten) können zu einem unvollständigen bzw. ungenauen Abbild von sich selbst sowie der Umgebung und — in weiterer Folge — zu Fehlentscheidungen, mit möglichen negativen Folgen, führen.

Deshalb muss ein CPS notwendiges Wissen aus den gewonnenen Daten abstrahieren und deren Reliabilität ermitteln können. Zudem sollten Entscheidungen mithilfe einer Metrik, die das Vertrauen zu gegebenen Umständen widerspiegelt, getroffen werden. Computational Self-Awareness (CSA) ist ein vielversprechendes Konzept, um ein CPS mit einer zuverlässigen und robusten Überwachungsfähigkeit auszustatten. Anhand von umfangreichen Experimenten in zwei unterschiedlichen Fallstudien (aus dem Gesundheits- und Industriesektor), zeigt diese Dissertation, dass CSA — im Besonderen seine Eigenschaften abstraction, data reliability und confidence — die Überwachungsfähigkeiten eines Systems robuster und zuverlässiger macht.

Stichwörter: Abstraction, Data reliability, Computational Self-Awareness, confidence, Robust and reliable monitoring

Acknowledgements

Actually, I never planned to do a Ph.D. Now, however, I find it difficult to express my gratitude to the many people who provided me with support and encouragement along the way without the acknowledgments becoming longer than the dissertation itself.

My Ph.D. studies were like an adventure trip that took me to another country, increased my knowledge in the field of computer science, and, last but not least, shaped me in very positive ways. Although it was cold, slippery, and dark when I arrived in Finland, I was very warmly welcomed by my supervisors and colleagues. Hence, I would like to thank them first.

Prof. Pasi Liljeberg, my principal supervisor, gave me the feeling of being part of the institute right after I arrived in Turku and helped me with many organizational matters throughout my studies, thank you for that! Without your help, I could not have succeeded. My co-supervisor, Prof. Axel Jantsch, always supported me, no matter whether I was in Turku or in Vienna during my research visits. Thank you for that and for the good ideas you kept contributing. My co-supervisor, Assoc. Prof. Amir Rahmani was, like me, on his own adventure trip and lived in Turku, Vienna, and California during my studies. Regardless of where he was, he has always supported me and provided me with valuable hints and tips — thank you for all your help!

My office colleagues also welcomed me straight away, making it extremely easy for me to settle in. Just like me, they were foreigners and, therefore, on a similar research expedition. In the course of my Ph.D., they evolved from travel companions to partners in crime and good friends with whom I definitely wish to stay in contact. Thank you Arman, Anil, Behailu, Hashem, Igor, Iman, Jorge, Mingzhe, Reza, and Tuan for being part of my journey and the good times we had!

I want to especially thank the following persons without whom I could not have finished my work. Lilian, thank you for not having given up after proofreading my master's thesis and agreeing to do the same for my dissertation. Big thanks also to my colleague and very good friend Nima. You have been such a tremendous help that I already see you as some kind of additional supervisor. My gratitude to you, Dávid. Whether you were in Sweden, Austria, or Hungary, you kept working with me over a long period and in a truly helpful

manner. To my friend Janne, special thanks for translating the abstract of this thesis into Finnish.

That my research trip came to a successful end, and that I returned safely home is also owed to the following three persons. I want to thank my two reviewers, Prof. Paul Pop and Prof. Martin Törngren, who helped improve my thesis with their valuable comments and feedback, and my opponent, Prof. Selma Saidi, who critically examined my work.

My research expedition would not have been possible without a solid and reliable infrastructure as well as financial support such as research scholarships. So I would like to thank the following institutions for supporting me during my studies: the University of Turku and its Doctoral Programme in Mathematics and Computer Sciences, the Academy of Finland, the HPY Research Foundation, the Nokia Foundation, and the Tekniikan Edistämissäätiö (Finnish Foundation for Technology Promotion).

My Finland experience would not have been the same without my Finnish friends most of whom I was lucky enough to meet thanks to the purchase of a second-hand microwave oven. Thank you, Emilia, Essi, Goran, Janne, Milja, and Sarah for the wonderful time we spent together. I also have to thank you for all the help with translating Finnish texts, which for me, despite doing two Finnish courses, were very hard to understand. I hope we all will stay in contact and see each other again as often as possible, whether in Finland, Austria, or anywhere else in the world.

However, without good friends at home, even the most exciting trip would only be half as good. Thank you, Alisha, Benedikt, Beni, Christian, Jakob, Marille, Mirjam, Niko, the two Philip(p)s, and Xandi for always being there for me, and it was fun that some of you even spent time with me in Finland.

Last but not least, I would like to thank my family and my girlfriend for all their enduring support. Thank you, Mama, Oma, and Vicky as well as Claudia, David, Elmar, and Heidi for letting me be part of this small but delicate circle and always being here for me.

October 2021
Maximilian Götzinger

List of Original Publications

This dissertation is based on the following original publications, which are referred to in the text by their Roman numerals:

Paper I	**Maximilian Götzinger**, Dávid Juhász, Nima TaheriNejad, Edwin Willegger, Benedikt Tutzer, Pasi Liljeberg, Axel Jantsch, and Amir M. Rahmani. RoSA: A Framework for Modeling Self-Awareness in Cyber-Physical Systems. *IEEE Access*, 2020; 8: 141373–141394.
Paper II	**Maximilian Götzinger**, Arman Anzanpour, Iman Azimi, Nima TaheriNejad, Axel Jantsch, Amir M. Rahmani, and Pasi Liljeberg. Confidence-Enhanced Early Warning Score Based on Fuzzy Logic. *Mobile Networks and Applications*, 2019; 8: 1–18.
Paper III	**Maximilian Götzinger**, Nima TaheriNejad, Hedyeh A. Kholerdi, Axel Jantsch, Edwin Willegger, Thomas Glatzl, Amir M. Rahmani, Thilo Sauter, and Pasi Liljeberg. Model-Free Condition Monitoring with Confidence. *International Journal of Computer Integrated Manufacturing*, 2019; 32(4-5): 466–481.
Paper IV	**Maximilian Götzinger**, Nima TaheriNejad, Hedyeh A. Kholerdi, and Axel Jantsch. On the Design of Context-Aware Health Monitoring without A Priori Knowledge; an AC-Motor Case-Study. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, IEEE, 2017; pages 1–5.
Paper V	**Maximilian Götzinger**, Edwin Willegger, Nima TaheriNejad, Axel Jantsch, Thilo Sauter, Thomas Glatzl, and Pasi Lilieberg. Applicability of Context-Aware Health Monitoring to Hydraulic Circuits. In *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, IEEE, 2018; pages 4712–4719.

Paper VI	Arman Anzanpour, Iman Azimi, **Maximilian Götzinger**, Amir M. Rahmani, Nima TaheriNejad, Pasi Liljeberg, Axel Jantsch, and Nikil Dutt. Self-Awareness in Remote Health Monitoring Systems Using Wearable Electronics. In *Design, Automation and Test in Europe (DATE)*, IEEE, 2017; pages 1056–1061.
Paper VII	**Maximilian Götzinger**, Nima TaheriNejad, Amir M. Rahmani, Pasi Liljeberg, Axel Jantsch, and Hannu Tenhunen. Enhancing the Early Warning Score System Using Data Confidence. In *Wireless Mobile Communication and Healthcare*, Cham. EAI, Springer International Publishing, 2016; pages 91–99.
Paper VIII	**Maximilian Götzinger**, Arman Anzanpour, Iman Azimi, Nima TaheriNejad, and Amir M. Rahmani. Enhancing the Self-Aware Early Warning Score System through Fuzzified Data Reliability Assessment. In *Wireless Mobile Communication and Healthcare*, Cham. EAI, Springer International Publishing, 2017; pages 3–11.

The original publications have been reproduced with the permission of the copyright holders. The author's contributions to each publication are listed in Chapter 8.

The following peer-reviewed papers were accepted for publication (Paper IX) or already published in the course of the author's doctoral studies but are not included in this thesis:

Paper IX	Daniel Hauer, **Maximilian Götzinger**, Axel Jantsch, Florian Kintzler. Context Aware Monitoring for Smart Grids. In *2021 IEEE 30th International Symposium on Industrial Electronics*, IEEE, 2021.
Paper X	**Maximilian Götzinger**, Amir M. Rahmani, Martin Pongratz, Pasi Liljeberg, Axel Jantsch, and Hannu Tenhunen. The Role of Self-Awareness and Hierarchical Agents in Resource Management for Many-Core Systems. In *2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC)*, IEEE, 2016; pages 53–60.
Paper XI	Alexander Wendt, **Maximilian Götzinger**, and Thilo Sauter. An Agent-Based Framework for Complex Networks. In *Artificial Intelligence Applications and Innovations*, Cham. Springer International Publishing, 2019; pages 559–570.
Paper XII	**Maximilian Götzinger**, Martin Pongratz, Amir M. Rahmani, and Axel Jantsch. Parallelized Flight Path Prediction Using a Graphics Processing Unit. In *12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, VISAPP, 2017; pages 386–393.

MAXIMILIAN GÖTZINGER

Maximilian Götzinger made his bachelor's degree in Electrical Engineering and Information Technology and completed his master's program in Computer Technology with distinction in 2015, both at TU Wien, Vienna, Austria. He has a keen and serious interest in computer science, engineering, and teaching. He published several peer-reviewed papers and served as reviewer for various conferences and journals. One of his papers was awarded a Best Paper Award, and for his work as a teacher, he received a Best Teacher Award and a Best Lecture Award.

Table of Contents

1 Introduction

Digitalization and automation have been playing a significant role in the industrial environment already for some time (Papadopoulos et al., 2021). Entire production lines are expected to run as autonomously as possible to save time and money (Kitajima and Sakurai, 2019). However, not only the industry sector makes usage of intelligent solutions and systems. By now, they have already entered private homes and lives (Matyi et al., 2020). Today, digitalization is playing an essential role in both industry and society. Driven by the Internet of Things (IoT) and System of Systems (SoS), the number of Cyber-Physical Systems (CPSs), including their small devices and sensors, is exponentially growing (Atzori et al., 2010; Rivera and van der Meulen, 2014). These systems have become omnipresent in almost every technical field. They are essential for a wide range of applications such as (mobile) telecommunication, automated manufacturing plants, autonomous driving, traffic control, health monitoring, and smart grids (Denker et al., 2012; Kim et al., 2017; Dafflon et al., 2021; Seshia et al., 2015; Zhao et al., 2017; Zhang et al., 2017; Oyewumi et al., 2019).

Due to the ever-expanding fields of intelligent applications as well as the expectations for full automation of nearly any machine and device, these systems face an increasing demand in various applications. A CPS should be autonomous, efficient, high performant, and durable (Barenji et al., 2020; Denker et al., 2012; Platzer, 2019; Stankovic et al., 2005; Ossamah et al., 2020; Wang et al., 2020). A CPS must also be able to achieve a controlled balance between these sometimes conflicting properties (Jantsch et al., 2018). Moreover, it needs the ability to adapt to a possibly changing environment and must be reliable but also robust against interferences and malfunctions (Denker et al., 2012). All these characteristics are essential to saving costs and improving the use of resources.

Independent of the application, CPSs and IoT connect the physical world with the digital world (Lee, 2006). As Figure 1 shows, CPSs are expected to carry out well-considered actions, which then influence or change the physical world's conditions in a targeted manner. More precisely, such a CPS is expected to achieve one or more goals through targeted actions (Anzanpour et al., 2019). To achieve such goals completely and appropriately, the CPS

must be aware of the environment's reactions to its own actions (Wang et al., 2010). However, an awareness of the interaction with the outside world alone is not sufficient. A CPS's internal components (hardware and software) have also constraints and limited resources (Törngren and Grogan, 2018). Therefore, in addition to awareness of the external environment, it must be aware of its internal environment as well where well-considered resource management must take place.

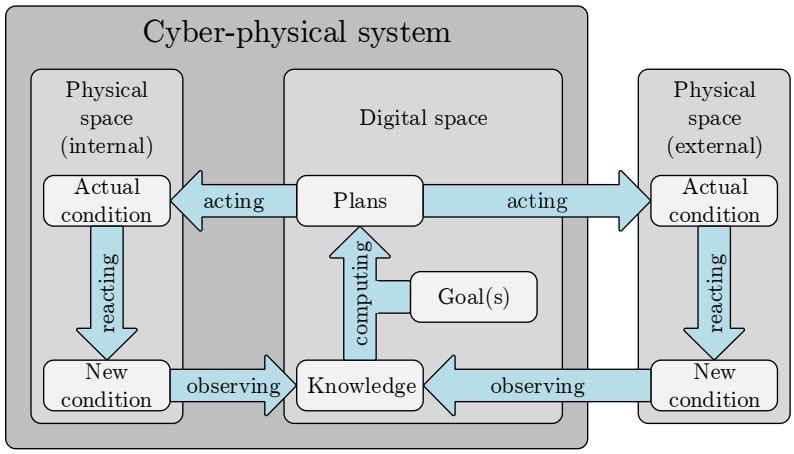


Figure 1. Schematic of a CPS connecting the physical with the digital world

The following example shall help in providing a better understanding. Let us assume that we use a CPS as a central control unit for a fully automated production line. The CPS must know of the condition of its external environment, which consists of all workpieces in the production line as well as the various manufacturing plant's machines. In other words, it must know the current status as well as the next production steps of each workpiece, while also considering the condition of each machine (such as state, workload, and capacities) in the production line. This knowledge is crucial to avoid wrong actions with possible negative consequences and keep an overview of the machines' degree of wear and tear. Disregarding these circumstances may lead to increased throw-outs, reduced throughput, unscheduled maintenance, defective machines, or several and more extended downtimes (Bousdekis et al., 2020; Ferreiro et al., 2016).

Besides the external-, the CPS must also monitor its internal environment, which comprises its own hardware and software, such as processing units, memory as well as hardware- and software sensors. The CPS must manage the available resources in the best possible way (Zhuge and Xing, 2012). Because of the constantly rising demands on such a CPS for higher performance, these systems' complexity (of hardware and software) increases with each new

technology generation (Kaliorakis et al., 2014; Haghbayan et al., 2014; Sangwan et al., 2008; Liggesmeyer and Trapp, 2009). Because of this increasing complexity, the era of many-core systems has long since started, and the number of transistors is continuously increasing (Kong et al., 2012). According to ITRS (2009) and Gupta et al. (2013), the continuously shrinking transistors as well as the increasing power- and thermal density leads to many problems that massively impair reliability and lifetime. Today, such a system has to cope with overheating, hotspots, aging, wear-outs, a significant unbalanced chip, under-utilization, and the dark-silicon phenomenon (Sonnenfeld et al., 2008; Haghbayan et al., 2016; Taylor, 2012).

In summary, there are many problems and uncertainties of both intrinsic and extrinsic nature that await solutions (Tao et al., 2020). A CPS must achieve its given goals as effectively and efficiently as possible while not jeopardizing its functionality to enable a long lifetime (Siegel et al., 2014). Besides, CPSs mostly work in a continuously changing environment (both internal and external), combined with possibly changing goals (Denker et al., 2012). This unsteadiness in the real world affects both the system as well as the applications it runs. For example, unidentified software and hardware faults or changes in the environment can lead to a malfunctioning of the system or the application (Georgeff and Ingrand, 1989; Goswami and Iyer, 1993; Leveson et al., 1991).

In these cases, a CPS must adapt well to new circumstances to make the right decisions and take proper actions (Denker et al., 2012). It may also decide which goals have priority and which are negligible. Existing research recognizes the critical role played by context-awareness to improve decision-making ability (Azimi et al., 2016).

Thus, there is a need for novel autonomous decision-making solutions for making systems intelligent. In this context, Albus (1991) defines *intelligence* as "the ability of a system to act appropriately in an uncertain environment, where appropriate action is that which increases the probability of success, and success is the achievement of behavioral subgoals that support the system's ultimate goal." Together with the definition of Wang (1995), according to which *intelligence* "is the ability for an information processing system to adapt to its environment with insufficient knowledge and resources," the above-mentioned requirements for a CPS are best summarized.

Both the United States and the European Research Council have designated intelligent CPSs as a significant research priority (Al-Ars et al., 2019; Duranton et al., 2019). Various learning techniques and big data analytic algorithms have been proposed to enable building intelligent systems (Farouk and Zhen, 2019; Qiu et al., 2016). As an example of intrinsic challenges for a CPS to overcome resource management issues and rectify the dark-silicon

phenomena, various techniques have been developed, such as dynamic mapping, core allocation, lifetime balancing as well as power-, thermal-, reliability-, and communication management (Chaturvedi et al., 2014; Guang et al., 2012; Haghbayan et al., 2016; Haghbayan et al., 2016; Winter et al., 2010; Yang and Shu-Min Li, 2009). Besides getting over such intrinsic problems and uncertainties, there are also efforts in other areas, such as in autonomous driving, which is one of the hottest topics in various academic communities (Lidberg and Müller, 2019). Here, too, an all-encompassing management is needed that covers all the aforementioned requirements. However, to the best of the author's knowledge, no fully self-adaptive solution exists so far that is totally aware of its own state, its environment, all possible uncertainties, and all its (possibly competing) goals (Alidoost Nia et al., 2020; De Lemos et al., 2017; Tao et al., 2020; Törngren et al., 2018). Tavčar and Horváth (2019) assume that the way to fully self-adaptive systems will be in numerous small increments rather than in one large radical shift. Current methods are limited as they do not have a complete picture of themselves and their environment. They only optimize one objective while neglecting others or combine several goals into a weighed objective function (Rahmani et al., 2018).

1.1 Motivation and Research Challenges

As mentioned, a CPS can only be effective, robust, reliable, and adaptable if it has a comprehensive knowledge of itself and its environment (Tavčar and Horváth, 2019). According to definitions of Kephart and Chess (2003) as well as Lewis et al. (2011), a system can make its own decisions and adapt to new situations if it is aware of itself and its environment. Hoffmann et al. (2010) and Rinner et al. (2015) add that such awareness enables a system to also adapt to possibly changing goals. It is self-evident that a CPS must monitor itself and its environment to gain a comprehensive understanding of both. However, an error-prone monitoring could prevent a CPS from making meaningful decisions and acting reliably and autonomously (Berk et al., 2019). Incorrect monitoring is incompatible with having comprehensive self- and environmental knowledge. Such incorrectnesses produce a false image of the CPS's environment (physical space) in its digital space (Figure 1). This is also in line with the concept of dependability as defined by Laprie (1992) and Avizienis et al. (2001), according to which a fault (e.g., a corrupted sensory data) can lead to an error (e.g., an incorrectly abstracted value) and subsequently to a system failure (a deviation of the delivered service from the correct service). In other words, without reliable monitoring, a system cannot have a complete and accurate image of itself and its environment. These data mostly come from physical properties measured by sensors (Alexopoulos et al., 2016). In general, however, it is

not easy to either evaluate or guarantee the correctness of such sensory data (TaheriNejad et al., 2016). Sensors themselves are neither entirely precise nor accurate (Taylor et al., 1994). Besides, depending on the application, sensors can be incorrectly used, e.g., incorrect installation or usage outside their intended working conditions. Of course, also a change in the environment can cause a correctly used sensor to suddenly not being used correctly anymore or becoming broken. There are numerous possibilities why a value provided by the sensor may not reflect the truth. Therefore, it is crucial to have a measure that classifies sensor readings into reliable or unreliable so that the CPS can use this information to make the right decisions even if the representation of reality is not entirely true.

Furthermore, it is also possible that not all correlations among physical quantities in the environment are entirely known. Nevertheless, a CPS must make the right decisions even without this complete knowledge to prevent negative consequences. Therefore, it is also essential that a CPS can also base its decisions on a measure that reflects the CPS's confidence about certain circumstances (Varshney and Alemzadeh, 2017).

Besides, quantities measured by sensors often have to be transformed into a domain to make them understandable and applicable for a CPS (Jantsch et al., 2017). In other words, the CPS must usually abstract required information from raw sensory data to be able to make the right decisions. There are different levels of abstraction, from less abstract to highly abstract (Sadighi et al., 2018). Decisions must be made on the right level of hierarchy, comparable to an organization's hierarchical structure where the CEO needs to decide based on the big picture, and employees take smaller actions based on more detailed knowledge.

Computational Self-Awareness (CSA) is a promising solution for providing a comprehensive assessment of the system's state and its surroundings to render it more reliable, intelligent, and autonomous (TaheriNejad and Jantsch, 2019; Dutt and TaheriNejad, 2016). It has already been studied and used in many applications such as cloud computing, network management, health monitoring, and mobile applications (Spathis and Bicudo, 2010; Psaier and Dustdar, 2011; Mercati et al., 2014; Chen et al., 2014; Jennings and Stadler, 2015; Preden et al., 2015; Kounev et al., 2015; Lewis et al., 2015, 2016; Dutt and TaheriNejad, 2016; Forooghifar et al., 2019; Bellman et al., 2020).

This work hypothesizes that CSA can also help to overcome the issues mentioned above of monitoring in the case of corrupted sensory data, (partially) unknown physical quantities of an environment, and data unintelligible in the context of the application running on the CPS. To investigate this research question, the self-awareness properties abstraction, data reliability, and confidence are chosen. In particular, the property abstraction is used to transform

gathered input data into a domain understandable and applicable for a CPS. The second property, data reliability, provides metadata describing the trustworthiness of the input data to enable the CPS to act accordingly. Finally, the property confidence provides metadata describing the trustworthiness of an algorithm, analysis, function, or (sub-)system so that the CPS is aware of the quality of its own computations and outputs.

1.2 Objectives

The main objective of this thesis is to investigate whether CSA can render a system's monitoring capability more robust and reliable. However, the focus of this work is not on whether internal or external parts of the CPS are monitored. It is generally about monitoring physical sensor signals and their further processing with the help of CSA and its properties. Since CSA is a promising solution for providing a comprehensive assessment of both the system's state as well as its surroundings (TaheriNejad and Jantsch, 2019; Dutt and TaheriNejad, 2016), the term "self" in Computational Self-Awareness relates rather to the research in this field and to the properties, visions, and techniques elaborated (TaheriNejad and Jantsch, 2019; Dutt and TaheriNejad, 2016; Spathis and Bicudo, 2010; Psaier and Dustdar, 2011; Mercati et al., 2014; Chen et al., 2014; Jennings and Stadler, 2015; Preden et al., 2015; Kounev et al., 2015; Lewis et al., 2015, 2016; Dutt and TaheriNejad, 2016; Forooghifar et al., 2019; Bellman et al., 2020).

Regarding the schematic shown in Figure 1, this work's focus is solely on observing, knowledge[1] extraction, and computing. So that these three steps will lead to a more correct monitoring result — even in the presence of erroneous data — self-awareness and its properties are leveraged.

As already mentioned in Section 1.1, the main focus is on the self-awareness properties: abstraction, data reliability, and confidence. This thesis investigates several subproblems and pursues the following research objectives:

Research Objective I
Investigate and analyze the capabilities of CSA to find a way of designing and evaluating self-awareness methods that tackle the challenges a CPS faces when monitoring itself or its environment.

Research Objective II
Develop an environment for implementing self-awareness methods and self-aware applications in a meaningful and user-friendly way.

[1] It should be noted that knowledge is not only extracted, rather, a system's setup can already contain some knowledge, such as thresholds, parameters, and fuzzy functions.

Research Objective III
Propose formal definitions as well as methods and implementations of the self-awareness properties abstraction, data reliability, and confidence.

Research Objective IV
Investigate how self-awareness (especially abstraction, data reliability, and confidence) enhances the reliability and robustness of a CPS's monitoring abilities.

1.3 Contributions

To enable these contributions[2] and prove this work's hypothesis, two case studies from different areas were chosen and existing literature about monitoring techniques currently used in these fields was reviewed. The first case study is related to the health sector. It describes a mobile system that monitors a patient's vital signs, calculates an Early Warning Score (EWS) indicating the patient's condition, and alerts when this condition is deteriorating. The second case study involves a Condition Monitoring System (CMS) that monitors another unknown system (black box) to determine its condition (its state and whether it is working properly or malfunctioning). The outcome of the two case studies are two different systems (independent of each other) the author has developed. These systems are equipped with self-awareness functionalities also developed by the author. Sections 2.5 and 2.6 present the two case studies and show that they are related to the topics of dependability, uncertainty, and fault detection. Fault diagnosis is not covered in this thesis because the implemented systems do not perform this task. Subsequently, the development stages of the implemented systems are shown in the various chapters about the different self-awareness properties (Chapters 4, 5, and 6). The same chapters also present the results derived from extensive experiments conducted with these systems (based on the case studies), and show that CSA can provide a CPS with a reliable and robust monitoring ability — even in the presence of erroneous data. An overview of the various development steps of the two systems, the evaluations of the corresponding experiments, and the rationales behind these experiments is shown in Section 1.4.1.

Contribution I
Propose and demonstrate the Research on Self-Awareness (RoSA) framework, which was developed and implemented in the course of this thesis. This framework supports modeling and evaluating different self-

[2]This dissertation is based on eight original publications, which are the results of collaborations. These publications can be found in the appendix. The author's contributions to each publication are listed in Chapter 8.

awareness concepts in hierarchical agent systems. RoSA is a middleware providing novel features for developing and evaluating self-aware applications. Its agents consist of self-awareness functionalities and fulfill their assigned tasks. The framework helps overcome the current practice of developing self-aware systems from scratch for each application, which is highly redundant, inefficient, and uneconomical.

Contribution II
Leverage CSA, especially the properties abstraction, data reliability, and confidence, to improve a system's monitoring capabilities in terms of robustness and reliability. Besides, definitions of these three properties are proposed. Moreover, possible methods of these properties are developed and implemented[3], and their usage is demonstrated in two selected case studies. The results of the extensive experiments conducted here based on these case studies prove the increased robustness and reliability provided by these properties.

Contribution III
Propose and demonstrate a Self-Aware Early Warning Score (SA-EWS) system that uses the self-awareness properties abstraction, data-reliability, and confidence. This SA-EWS system is much more robust than a non-self-aware EWS system. It provides a more reliable EWS even under adverse monitoring conditions such as noisy signals, disconnected sensors, and non-nominal monitoring conditions. This is achieved without redundancy (like redundant sensors), but exclusively with data reliability and confidence methods based on fuzzy logic.

Contribution IV
Propose and demonstrate a CMS that can detect the state of a black box system. In this context, state detection includes both working state as well as health state[4]. In its current implementation, the CMS is limited to black-box systems that behave like a bijective function (having a one-to-one correspondence between inputs and outputs). The CMS does not require deep expertise nor cumbersome customization to monitor different black-box systems. Moreover, the state identification process through a fuzzy-logic-based confidence metric makes the CMS more robust and its state detection more reliable.

[3]It is not guaranteed that the proposed methods are the best possible implementations of the selected self-awareness properties. The dissertation's focus is on showing that self-awareness can improve a system's monitoring capabilities regarding its robustness and reliability.

[4]It must be noted that, in this context, the term "health" describes the condition of a machine or device; i.e., whether it is working properly or malfunctioning.

1.4 Organization

This dissertation is a collection of eight original publications, three of which were published in international peer-reviewed journals and five in international peer-reviewed conference proceedings. All eight papers are the result of collaborations with other scientists.

This work starts with a research summary (Chapters 1-8) to provide an overview of the author's research. It aims to elaborate the big picture of which all the individual publications are a part. To avoid exceeding the scope of this research summary, the author sometimes refers to the original publications, appended to this thesis, for more detailed aspects and more detailed explanations. To provide a proper orientation, each chapter begins with a short introduction and refers to the original publications that play a central role in it.

In Chapter 2, the author reviews relevant existing literature with the main focus on existing CSA architectures and frameworks as well as on current monitoring techniques used in the fields of the selected case studies. Chapter 3 describes the development of a software framework that serves as a generic tool to develop self-aware applications. Chapters 4, 5, and 6 propose the formal definitions and possible implementations of each of the self-awareness properties, abstraction, data reliability, and confidence. These chapters also present the different development stages based on the two case studies and analyze the results of the relevant experiments. Chapter 7 discusses the results, presents the conclusions, and shows directions for possible future research. Finally, Chapter 8 presents a summarized overview of all original publications and states the author's contributions to each of them.

All original publications that form the basis for this dissertation are described in the following as they relate to each chapter. Although all eight papers contribute to some degree to Chapter 2, it is mainly based on Papers I, II, and III. These papers also form the basis for some other chapters. Paper I also covers Chapter 3 and contains the formal definition presented in Chapter 4. Paper II shows the formal definitions given in Chapters 5 and 6. Together with Paper III, it also completes Chapter 6. Papers IV and V complete the content of Chapter 4, while Papers VI, VII, and VIII are the basis for Chapter 5. All eight papers jointly provide the basis for Chapters 7 and 8.

1.4.1 Evaluations Performed

In the course of this work, several experiments and evaluations have been conducted based on the two case studies. Since these and the related results are addressed in the individual chapters on the respective CSA properties,

Table 1. Overview of the case studies and the relevant experiments

Section	CSA properties	System	Rationale	Experimental data
4.4	Abstraction	CMS	Evaluating a possible method to abstract the condition of a black-box system (its working states and whether it works correctly or malfunctions)	AC motor data derived from simulations and physical experiments as well as data of a water pipe system[a] derived from physical experiments
5.4.1	Abstraction and Data reliability (binary method)	EWS system	Proof of concept: assessing the input data's reliability (in experiments of limited complexity) without redundancy but based on the three measures plausibility, consistency, and cross-validity	Vital sign data recorded from a 35-year-old healthy male subject, whereas the measured body temperature was replaced with incorrect temperature data to obtain one corrupted vital sign signal
5.4.2	Abstraction and Data reliability (fuzzy method)	EWS system	Evaluating a fuzzified reliability method that assesses the reliability of the system's input and output data as well as proving that the calculated reliability values correlate with the truthfulness of the input data	Vital sign data recorded from a 36-year-old male subject with diastolic hypertension where different real sensor errors were introduced that corrupted the vital sign signals body temperature, heart rate, and blood pressure
6.3	Abstraction, Data reliability (fuzzy method), and Confidence	EWS system	Evaluating a system that makes confidence-based decisions and autonomously corrects its output, comparing it with a non-self-aware system, and proving that it is more robust against erroneous data and that its output is more reliable	Vital sign data recorded from eight participants with a variety of sensors to obtain the same recordings in different qualities (good quality with low noise as well as bad quality containing a lot of errors and noise)
6.4	Abstraction and Confidence	CMS	Evaluating a system that makes confidence-based decisions, comparing it with the system without confidence (Section 4.4), and proving that confidence increases both the reliability of the system's output as well as the system's robustness	For reasons of comparison, the same data used in the experiments of the CMS without confidence (Section 4.4): AC motor data[b] derived from simulations and physical experiments as well as data of a water pipe system[a] derived from physical experiments

[a] More precisely, the water pipe system is a Heating, Ventilation, and Air Conditioning (HVAC) system described in more detail in Papers III and V.

[b] In addition to the motor data used in Section 4.4, more complex data was simulated and processed.

this section is intended to serve as a guide through these various development steps. While the underlying architecture of the EWS system is described in Section 3.2, Section 3.3 covers the CMS architecture. Since this is just the basic structure, these sections do not include experiments regarding the self-awareness methods and the performance of both systems. The evaluations of these experiments are presented starting in Chapter 4. Table 1 shows the CSA properties used in the different developments steps of the two systems and the rationales behind these experiments. Additionally, the table also shows the data used for the respective experiments.

2 Background and Related Works

This work is motivated by the ever-increasing importance and relevance of reliable and robust Cyber-Physical Systems (CPSs) (Barenji et al., 2020; Romero-Silva and Hernández-López, 2020; Sha et al., 2008), which are expected to make the right decisions even under challenging circumstances (Platzer, 2019; Stankovic et al., 2005), such as in a changing environment or in the presence of erroneous input (Denker et al., 2012). This dissertation shows that Computational Self-Awareness (CSA), especially its properties abstraction, data reliability, and confidence, is a powerful tool to provide a CPS with a reliable and robust monitoring function. To explore these properties and test possible self-awareness methods, the author selected two case studies from different areas.

This chapter will provide a detailed review of existing research literature. Section 2.1 gives a short overview of autonomic computing before CSA is introduced and related fields are presented in Section 2.2. Since this thesis is about implementing self-aware applications in the two case studies, the review of already published reference architectures for CSA is essential and shown in Section 2.3. Section 2.4 presents existing CSA frameworks and explains why a separate framework was developed in the course of this work. For further details about related work presented in these four sections, the author refers to Paper I (Götzinger et al., 2020)[1]. Section 2.5 introduces the first of the two case studies, which derives from the health sector. It describes the case study and current monitoring techniques. It also presents related works regarding the assessment of data reliability and refers to related fields such as dependability and uncertainty (Laprie, 1992; Taylor et al., 1994). Section 2.6 shows the same for the second case study, which derives from the industrial sector. In addition, it discusses detection techniques and refers to related research as well as to related fields such as uncertainty (Baraldi et al., 2014; Taylor et al., 1994).

While a more detailed literature review relating to the health sector case study is given in Paper II (Götzinger et al., 2019a)[1] and VI (Anzanpour et al.,

[1]This dissertation is based on eight original publications, which are the results of collaborations. These publications can be found in the appendix. The author's contributions to each publication are listed in Chapter 8.

2017)[1], Papers III, IV, and V (Götzinger et al., 2019b, 2017b, 2018)[1] cover the industrial case study.

2.1 Autonomic Computing

To enable a CPS to process its complex tasks in an automatized manner and adapt to a possibly changing environment, it requires a high degree of autonomy (Schlingensiepen et al., 2016). The term autonomy is derived from ancient Greek and means self-administration, respectively, having its own laws (Abeywickrama and Ovaska, 2017). Later, in the 1960s, it became an object of investigation in psychology before it found its way as a concept into computer technology in the 1990s (Rinner et al., 2015; Dutt et al., 2016). Inspired by biological systems, academia and industry started several initiatives around autonomous computing (Parashar and Hariri, 2005).

One of the early applications comes from the military sector. According to Randall and Walter (2003) as well as Huebscher and McCann (2008), the Defense Advanced Research Projects Agency (DARPA) developed a communication and tracking system that enabled soldiers to share information among themselves and with the command center. The system consisted of mobile devices with various positioning- and sensor capabilities, which, combined with the soldiers' input regarding their own situation, could compile and spread relevant details about the combat field.

Also in the 1990s, but outside the military sector, the National Aeronautics and Space Administration (NASA) launched projects such as Mars Pathfinder and Deep Space 1 (Rahman et al., 2011). Remote control of these faraway spacecrafts was associated with noticeable delays and, therefore, highly impractical. Hence, NASA's goal was to render these spacecrafts more autonomous, that is, to enable them to operate, navigate, and manage deep-space probes with reduced human intervention.

Shortly thereafter, in the early 2000s, Intel wanted embedded systems to become more autonomous, independent, and self-adaptive. Computing should become proactive so that people would no longer be needed "in the loop" (Tennenhouse, 2000). International Business Machines Corporation (IBM) also announced similar statements around that time (Kephart and Chess, 2003). In 2001, IBM declared the complexity of Information Technology (IT) systems being one of the most challenging factors for an industry's progress in the upcoming decades (Kephart, 2005). To accelerate progress in autonomous computer systems, IBM introduced the autonomic computing initiative and formulated the following maturity levels (sorted by ascending autonomy): basic, managed, predictive, adaptive, and autonomic (Ganek and Corbi, 2003; Kephart and Chess, 2003; IBM Corporation, 2006; Lalanda et al., 2013). A

system classified as basic is in the lowest maturity level of autonomic computing and must be monitored and manually modified by highly qualified personnel (Huebscher and McCann, 2008). In contrast, a computer system classified as autonomic is in the highest level and must be completely self-managing and fulfill high-level objectives defined by humans (Parashar and Hariri, 2005; Cámara et al., 2017). To some extent, this is in line with the levels of automation proposed by Parasuraman et al. (2000). As they defined ten different levels, this scale of automation is finer-grained, but still describes the various stages between the two extremes: fully manual performance and full automation.

In addition to the autonomy gradations, IBM also introduced the four self-* properties of autonomic computing, often referred to as "self-chop" (IBM Corporation, 2006; Gurgen et al., 2013).

- self-configuration: the ability to autonomously adjust parameters or change the software to achieve high-level goals,
- self-healing: the ability to autonomously detect and diagnose problems to solve them independently if possible,
- self-optimization: the ability to autonomously optimize the utilization of resources, and
- self-protection: the ability to autonomously protect against malicious attacks or an unintentional misapplication by the system's user.

For a detailed description of these self-* properties, the author refers to the works of Kephart and Chess (2003) as well as Bantz et al. (2003). These four properties are the most frequently cited in the autonomic computing field. However, the number of such properties has been steadily increasing; Abeywickrama and Ovaska (2017) as well as Parashar and Hariri (2005) list the most prominent examples in their works.

2.2 Computational Self-Awareness

Among these self-* properties of the autonomic computing initiative is also self-awareness (Kephart and Chess, 2003; Salehie and Tahvildari, 2009). A self-aware CPS observes itself and its environment to be able to act according to the observations. Thus, CSA could also be called "computational reflection" since this describes a system's ability to reflect on its capabilities, limitations, and resources (Cámara et al., 2017; Bellman, 1991; Landauer and Bellman, 2017). A self-aware CPS requires sensors to perceive both its internal and

external environment as well as actuators to adapt to possible environmental changes (Parashar and Hariri, 2005).

Figure 2 shows the hierarchy of self-* properties, according to which self-awareness and context-awareness are the basis for self-adaptiveness. Hence, a system must be self-aware to be self-adaptive (or autonomous). However, CSA has come to the fore in recent years and is no longer seen as a supporting feature for advanced adaptive behavior but rather encompasses all relevant self-* properties, including self-adaptivity. In other words, the pyramid in Figure 2 has been turned upside down. CSA is not just a collection of state variables but must also include the system's goals and adequately reflect the effects of its actions on itself and its environment. In contrast to autonomic computing, a fully self-aware system is not only reactive but proactive, which means that it must be able to learn, draw conclusions, and act accordingly (Giese et al., 2017).

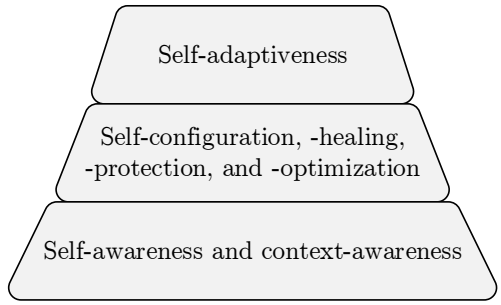


Figure 2. Self-awareness is the base for self-adaptiveness (Salehie and Tahvildari, 2009)

The recent past has shown that CSA can help solve many problems of CPSs and System on Chips (SoCs). Various aspects of CSA have proven to be essential to make these systems more intelligent and efficient, such as self-monitoring, situation-awareness, and attention (Dutt et al., 2015; Faniyi et al., 2014; Guang et al., 2012; Dutt et al., 2016; Bouajila et al., 2006; Teich et al., 2011; Bouajila et al., 2011; Jafri et al., 2012; Kornaros and Pnevmatikatos, 2013; TaheriNejad et al., 2017). In this context, self-monitoring means the activity of sampling system properties (e.g., chip temperature in the work of TaheriNejad et al. (2017)) as well as transforming and filtering the sampled data so that the system can use the collected information. This transformation process is called abstraction and constitutes one of the self-awareness properties investigated in this thesis and presented in Chapter 4. Situation-awareness assesses the observations made, and attention provides a reasonable allocation of resources through dynamic prioritization of the system's various tasks and objectives.

CSA has been applied in both software (Kephart and Chess, 2003) and hardware (Dutt et al., 2016). The following applications have benefited from CSA concepts (some of them under other terms such as adaptivity, autonomy, and goal-oriented systems): mobile applications (Mercati et al., 2014), object tracking with smart cameras (Rinner et al., 2015; Lewis et al., 2015), artificial intelligence (Forooghifar et al., 2018), cloud computing (Jennings and Stadler, 2015), networks (Spathis and Bicudo, 2010), operating systems (Wanner et al., 2013), web (Strassner et al., 2009), adaptive and dynamic compilation environment (Baek and Chilimbi, 2010), Multi-Processor System-on-Chip (MPSoC) resource management (Hoffmann et al., 2010; Shamsa et al., 2020), (cyber-physical) SoC (Dutt et al., 2016), mobile robots (Akbar and Lewis, 2018), industrial systems (Siafara et al., 2017, 2018), health monitoring (Chen et al., 2014) as well as single- and multi-user active music environments (Nymoen et al., 2016).

It should be noted that self-awareness and its encompassed self-* properties also have similarities with other fields. The fields described in the following (without any guarantee of completeness) are beyond the scope of this work, but mentioning them should give the reader a broader view of the overall subject.

For example, Model Predictive Control (MPC) from the field of control theory has similar characteristics as self-adaptiveness since it is an optimizing feedback control loop for linear and nonlinear systems (Grüne and Pannek, 2017; Angelopoulos et al., 2018). In other words, MPC is a control strategy that computes a sequence of future actions to optimize for one or more specific control objectives (Alamir, 2013). Feedback control loops in general share properties with self-awareness. Such a loop computes, based on measurements, appropriate control parameters and adapts the behavior of a system to given requirements (Salzmann et al., 2000). These loops rely on models of a process that shall be optimized, and techniques exist to do this also in real-time (Grüne and Pannek, 2017; Samet et al., 1998; Stankovic et al., 1999). Proportional–Integral–Derivative (PID) controllers may be the oldest control loop mechanisms, yet, they are still widely researched. Thanks to their simplicity and efficiency, they are still the most widely used controllers in this field (Borase et al., 2020; Samet et al., 1998). Since usually no perfect model exists in the real world, the optimization plan (the sequence of future actions) must be updated at each step (Angelopoulos et al., 2018).

Another related field from industry is Digital Twins (DTs). The term DTs was already introduced about 20 years ago, but the technique is only slowly becoming more popular these days, and a common concept for it does not yet exist (Grieves, 2014; Batty, 2018; Brosinsky et al., 2019). DTs has become a widely used marketing term that describes the ability to remotely monitor,

control, or simulate another system (also in real-time) to optimize workflows (Brosinsky et al., 2019; Lee et al., 2015; Lim et al., 2020; Tao et al., 2019). This is accomplished through the ability to reflect the physical conditions of a process, machine, or system. In other words, a DT is a virtual copy of an entire process, machine, or system (Tao and Qi, 2019) and is thus able to bridge the gap between physical and digital worlds (Brosinsky et al., 2019). Because of its monitoring characteristics, a DT could even be the basis for a self-aware system (Goossens, 2017). However, since DTs are highly sophisticated models, their implementation is usually time consuming and expensive, therefore, they are only used in high-end systems (Qi and Tao, 2018; Batty, 2018). Batty (2018) even argues that a DT most likely cannot be an absolutely identical reflection because "an exact mirror is an idealization that will never be achieved." Neither is the use of such a highly complex model compatible with the idea that a CPS can be self-aware even if it does not fully know its own internals or its environment. Despite the associated constraints, DTs can be used for self-aware systems.

2.3 Computational Self-Awareness Architectures

Several reference architectures concerning CSA exist (Kramer and Magee, 2007; Aßmann et al., 2014; Lewis et al., 2015; Giese et al., 2017). One of them is the MAPE-K (Monitor-Analyze-Plan-Execute over a shared Knowledge) loop from the autonomic computing field (IBM Corporation, 2006; Kephart and Chess, 2003). In this autonomous control loop, different phases are processed consecutively. First comes the monitoring phase, in which information from sensors is collected, followed by the phase, in which the collected information is analyzed. This is followed by the planning of actions to achieve goals or solve problems, which are then executed in the next phase (Giese et al., 2017). All four processes share one common knowledge about the hardware infrastructure, its execution environment, context, goals, states, historical logs, and policies (Nguyen et al., 2015; Arcaini et al., 2015).

This MAPE-K loop is very similar to the LRA-M (Learn-Reason-Act-Model) architecture, which describes a model-based learning and reasoning loop (Kounev et al., 2017). This is an architecture for self-aware computer systems that are driven by their goals and making decisions based on observations collected. The gathered data is used in a continuous learning process to generate a model that decides on possible future actions. The LRA-M loop is a model-based formulation of the Observe-Decide-Act (ODA) loop implemented in this work (Section 3.1).

Already in the early days of Artificial Intelligence (AI), decentralized architectures were proposed, consisting of several independent rational units

(agents) that can interact with each other and work in parallel on their various tasks (Giese et al., 2017). According to Russell and Norvig (2010), the agent-based architecture, in which agents perform the best possible action according to their information and capabilities, is fundamental to AI. The self-awareness reference architecture of Guang et al. (2010) is also based on agents, although they describe these agents rather vaguely as a design abstraction. Wooldridge and Jennings (1995) define agents as pieces of software that are autonomous (no human intervention), social (communicate with other agents or humans), reactive (react to changes in their environment), and proactive (take the initiative).

Such a system consisting of several agents that work together to achieve one or more common goals is called a Multi-Agent System (MAS) (Cámara et al., 2017). MAS corresponds to the actor model, a programming paradigm known for scalable, parallel, and distributed computing (Hewitt, 2017). Usually, it is advantageous to divide complex applications into several smaller subtasks to enable better handling. Often, a hierarchical structure, in which these subtasks are divided into different levels, helps a system to cover both the overall picture and the small details. This is similar to the nature-inspired hierarchical system of IBM's autonomic computing initiative and has already been studied in the context of self-aware systems (Guang, 2012; Sadighi et al., 2018).

2.4 Computational Self-Awareness Frameworks

While some approaches to render CPSs more intelligent through CSA exist, this area has remained largely unexplored so far (Guang et al., 2012; Faniyi et al., 2014). Many aspects of this hot topic still need to be researched, but techniques and methods are being developed at a moderate pace and lack convergence. This slow progress is due to the rather fragmented research community and the high costs (mostly development time) to implement self-aware applications. To overcome today's practice of implementing each self-aware application from scratch, a common framework to investigate CSA would be advantageous and could generate cooperation and synergy among researchers.

While several frameworks focus on specific self-* properties, to the best of the author's knowledge, there is no satisfactory common tool to accelerate research on CSA. SAPERE (Viroli et al., 2012), ACOSO (Savaglio et al., 2016), and BIONETS (Carreras et al., 2007) are platforms that support the self-organization of autonomic nodes in distributed environments. However, SAPERE and ACOSO are implemented on top of JADE (Bellifemine et al., 2005), which is Java-based and hence has a high resource requirement that exceeds the capacity of some Embedded Systems (ESs) (Chang et al., 2019;

Tanigawa et al., 2019). The concepts of BIONETS, on the other hand, are implemented only in simulation models, which limits its deployability in real systems. The Collective Adaptive Systems approach of the ALLOW Ensembles project (Bucchiarone, 2019) supports collaborative self-adaptation of agents within groups called ensembles. The DeMOCAS (Bucchiarone et al., 2017) framework uses this ALLOW approach but is also implemented in Java with limited deployability. SEEC (Hoffmann et al., 2010) is a framework for self-aware resource allocation based on the concept of application heartbeats, which allows monitoring and adjusting program performance. However, SEEC does not correspond to the agent-based architecture needed for its flexibility and scalability. All these platforms offer somewhat specific design proposals for different self-aware systems, but none of them represent a complete modeling framework.

Table 2. Multi-agent modeling systems (Götzinger et al., 2020)

Framework	License-free	Multi-agent simulation	Deployable actor system	Implementation / runtime	CPS / ES
Akka (Hunt, 2014)	Yes	Possible[b]	Yes	Java/Scala	Partially[d]
CAF (Charousset et al., 2016)	Yes	Possible[b]	Yes	Native C++	Partially[e]
GAMA-Platform (Taillandier et al., 2019)	Yes	Yes	No	Java	No
JADE (Bellifemine et al., 2005)	Yes	Possible[b]	Yes	Java	Partially[d]
Mobile-C (Chen et al., 2006)	Partially[a]	Possible[b]	Yes	Native C / C++/ Ch[c]	Yes
Repast for HPC (Collier and North, 2013)	Yes	Yes	No	Native C++	No
Repast Simphony (North et al., 2013)	Yes	Yes	No	Java	No
RoSA	Yes	Yes	Yes	Native C++	Yes

[a] Proprietary dependency: Embedded Ch, free for non-commercial use on ARM-based systems.
[b] Not designed for simulation but might be configured for the purpose with considerable effort.
[c] Ch is a scripting language with C/C++ syntax.
[d] Depending on the ES's Java-support and the actual performance requirements.
[e] CAF poses a relatively large footprint because of its extensive non-configurable set of features.

There are also general agent-based frameworks unrelated to CSA that are summarized in Table 2 and discussed in the following. Akka (Hunt, 2014), JADE (Bellifemine et al., 2005), Repast Simphony (North et al., 2013) are Java-based frameworks and, therefore, cannot be used for all ESs due to high resource requirements (Chang et al., 2019; Tanigawa et al., 2019). On the other hand, other simulation frameworks, such as GAMA-Platform (Taillandier et al., 2019) and Repast for HPC (Collier and North, 2013) are not deployable actor systems. This means that these frameworks can be used for simulations but cannot run the implemented agent-based application in ESs with real sensors and actuators. In contrast, Mobile-C (Chen et al., 2006) and CAF (Charousset et al., 2016) are deployable actor systems with native implementation and can, therefore, support execution on ES hardware. Although Mobile-C is a distributed actor system with a small footprint, it has limited applicability due to a proprietary dependency and a custom native API. On the other hand, CAF is an open-source distributed actor system with standard C++ implementation and can work on a wide range of hardware platforms. However, its extensive non-configurable feature set makes it less suitable for ES. A stripped-down version for resource-constrained systems remains a promise to date.

As a result, the Research on Self-Awareness (RoSA) framework was developed in the course of this work. Besides meeting the required demands, RoSA is intended to serve as a common framework for the research community. Chapter 3 shows its architecture, implementation, and usage as well as the implementation of the two case studies' systems in RoSA.

2.5 Early Warning Score System

According to the WHO (2017), chronic diseases such as cardiovascular disease are among the leading causes of death worldwide. These ailments can lead to a sudden deterioration in patients' health and become a threat to their lives. McGaughey et al. (2007) state that a deterioration in a patient's health condition is already visible early on in the patient's vital signs — sometimes even up to 24 hours in advance. Kyriacos et al. (2011) continue to point out that the early detection of such a health deterioration effectively increases a patient's chances of survival. This means that periodic monitoring and evaluation of a patient's vital signs can enable early detection of deteriorating health conditions and consequently save lives.

For this purpose, Morgan et al. (1997) published the Early Warning Score (EWS) method, which has become common practice among healthcare professionals in hospitals, especially in intensive care units. They manually monitor

and classify the patient's vital signs, such as heart rate, respiratory rate, body temperature, blood pressure, oxygen saturation, and level of consciousness.

Table 3. A conventional EWS chart (Urban et al., 2015)

Vital sign score	**3**	**2**	**1**	**0**	**1**	**2**	**3**
Heart rate (beats/min.)	0 - 39	40 - 50	51 - 59	60 - 100	101 - 110	111 - 129	≥ 130
Systolic blood pressure (mmHg)	0 - 69	70 - 80	81 - 100	101 - 149	150 - 169	170 - 179	≥ 180
Respiratory rate (breaths/min.)		0 - 8		9 - 14	15 - 20	21 - 29	≥ 30
Body temperature (°C)		≤ 35		35.1 - 38		38.1 - 39.5	≥ 39.6
Blood oxygen saturation (%)	0 - 84	85 - 89	90 - 94	95 - 100			
AVPU score[a]				A	V	P	U

[a] AVPU (the level of consciousness): A = alert, V = reacting to voice, P = reacting to pain and U = unresponsive

This classification is done by assigning scores to the corresponding vital signs, similar to a lookup table. Table 3 is an example of such an EWS classification table (used in the work of Urban et al. (2015)) and shows that each vital sign can have a score ranging[2] from 0 to 3. Score 0 stands for a vital sign in perfect condition, e.g., a heart rate between 60 and 100 or systolic blood pressure from 101 to 149. If a vital sign is not in perfect condition but a bit too low or too high, it is classified with the next higher score. Medical researchers created and categorized this table according to the degree of deterioration of the various vital signs. In other words, the main focus is on mapping the severity of the possible medical consequences and not on the symmetry of the table. Thus, such a table does not need to assign each score (in both directions) to a range of vital signs. Table 3 shows as an example that if the respiratory rate deteriorates from 9 breaths per minute to 8 breaths per minute, score 2 is assigned to the respiratory rate instead of score 1. If the value of a vital parameter is in an even worse condition (even higher or lower), it is again classified with the next higher score. The highest score of a vital sign is score 3 and means that it is in the worst condition.

Afterward, the various vital sign scores, abstracted from the patient's vital signs, are then summed up to form the EWS. While a low EWS indicates a patient is in a good health condition, a high EWS corresponds to a high risk of critical medical conditions (Royal College of Physicians, 2017). The resulting EWS can then be classified as one of three different risk levels that indicate whether the patient requires acute medical care. Table 4 shows the three risk

[2]It should be noted that there are several different EWS classification tables from other studies besides this one, e.g., the works from Groarke et al. (2008) or Smith et al. (2013). While the vital sign value ranges may differ in the different tables, score range (0 to 3) and method of calculating the EWS are always the same.

levels defined by Kyriacos et al. (2014) supplemented with healthcare actions suggested by the National Clinical Effectiveness Committee et al. (2013) and Holbery and Newcombe (2016).

Table 4. The three risk levels defined by Kyriacos et al. (2014) supplemented with healthcare actions suggested by the National Clinical Effectiveness Committee et al. (2013) and Holbery and Newcombe (2016)

EWS	**Interpretation**	**Suggested action**
0 - 3	Uncritical / low risk	Periodically monitoring of the patient's vital signs
4 - 6	Equivocal / medium risk	Urgently informing a medical team
≥ 7	Critical / high risk	Triggering an urgent clinical response

Although the EWS method is being used successfully in hospitals, two main aspects limit this manual approach. Firstly, interpretation of the vital signs and, consequently, the determined EWS could be incorrect due to inaccuracies and latency caused by manual data acquisition. However, from a more practical point of view, the second reason is the more limiting factor. This non-automatic technique is solely designed for patients who are already in the hospital. It does not work if persons are not cared for around-the-clock by professional staff able to conduct measurements and interpret the patient's vital signs. It would be a big step forward if elderly persons or those suffering from chronic health problems could be at home and continue with their daily routines while still receiving needed medical care as quickly as possible. A mobile device could tackle this demand and monitor the patient unsupervised by medical personnel. Such a system could, in case of an increasing EWS, trigger a rapid response team to further examine the patient; similar to an approach, approved by the U.S. Food and Drug Administration (FDA), for predicting a patient's potential sudden death (Excel Medical, 2018). In addition to a potentially higher survival rate, a mobile early warning system could also reduce healthcare costs and reduce the length of hospital stays.

For some time now, efforts have been undertaken to develop such a portable device that independently reads a person's vital parameters and automatically calculates the early warning system, independent of medical personnel. In this context, Anzanpour et al. (2015b) propose an Internet of Things (IoT)-based health monitoring system that autonomously monitors vital parameters and, if necessary, transmits data to healthcare professionals or alerts them. The ever-

increasing number of IoT devices also speaks for this. Hung (2017) estimates that the ratio between the world population and IoT devices will soon be one to four. These small devices and wearables form a good and cost-efficient basis for a well-structured EWS system that autonomously monitors a patient and thus reduces the mortality rate (Dohr et al., 2010; Atzori et al., 2010; Miorandi et al., 2012; Anzanpour et al., 2015a).

Besides the performance of IoT devices, it is, of course, also essential to keep energy consumption as low as possible to ensure a long runtime. Hafshejani et al. (2020) show in their experiments that they found out how to increase the power efficiency of wireless Photoplethysmogram (PPG) and Electrocardiogram (ECG) devices.

Azimi et al. (2016) confirm that IoT is an excellent enabler for a portable EWS system. Besides, they propose an approach that calculates the EWS based on the patient's activity since a patient's different activities will lead to different vital sign values. For example, a person's vital signs (especially, heart rate, blood pressure, and respiratory rate) will be significantly higher while running than while sleeping. However, while running, increased vital signs are not necessarily a sign of an inferior health state. Therefore, they adapt the EWS to the patient's activity, which is necessary so as not to cause false alarms. Their experiments prove what Jantsch and Tammemäe (2014) have already proposed before, namely, that knowledge of situations and circumstances improves the system's ability to make decisions.

Manual monitoring of a patient lying more or less motionless in a hospital bed is much less problematic than automatic monitoring of a patient performing daily tasks at home (TaheriNejad, 2019). A widely recognized challenge for portable devices is the occurrence of motion artifacts. Pollreisz and TaheriNejad's (2019) research focused on techniques to make measurements with PPG sensors more reliable by removing measurement artifacts caused by the patient's movements. Subsequently, Pollreisz and TaheriNejad (2020a,b) also showed that the respiratory rate can be extracted from the PPG signal of the smartwatch the test subject is wearing.

However, all these works cannot provide entirely reliable measurement results. Some of them even neglect the possibility of falsely monitored vital signs resulting from incorrectly attached or detached sensors, broken sensors, or noisy signals. Because of the potentially dangerous consequences, data reliability of such EWS systems is of utmost importance. Erroneous data may lead to a wrong score of the corresponding vital sign. An incorrectly classified vital sign leads to a wrong EWS calculation. If the EWS deviates enough from the truth, it leads, in further consequence, to a false or — even worse — to a missing alarm (Parego et al., 2017). This in line with the works on *dependability* of Laprie (1992) and Avizienis et al. (2001), where a *fault* (e.g., a

broken sensor) leads to an *error* (e.g., erroneous vital sign data) and in further consequence to a system failure (e.g., a miscalculated EWS). As mentioned in Section 2.2, CSA offers the possibility to make computer systems more autonomous, intelligent, and, most importantly here, more reliable (TaheriNejad and Jantsch, 2019; Dutt and TaheriNejad, 2016). Therefore, CSA is also a promising solution to overcome or improve these problems in monitoring and make the EWS assessment more robust and reliable.

In the course of this thesis, abstraction, data reliability, and confidence are leveraged to render the autonomous EWS system more reliable and robust against faulty data acquisition. While Section 3.2 describes the system architecture implementation, the subsequent chapters give a more detailed insight into the self-awareness properties utilized and show how the quality of the EWS assessment benefits from them. It starts with Chapter 4, in which the abstraction steps from the various vital signs to the system's output (the EWS) are explained to initially model the EWS method (Morgan et al., 1997). Chapter 5 then deals with the process of assessing the reliability of the input data as well as that of the calculated EWS. Since the task of extracting a signal's reliability is rather about giving a statement regarding its trustfulness than detecting or diagnosing a specific error, this assessment is not comparable with works such as that of Avižienis (1967).

However, other works exist that are related to assessing input data reliability in medical applications. The work of Cao et al. (1999) describes the different measures, namely, *plausibility* (Section 5.2.1), *consistency* (Section 5.2.2), and *cross-validity* (Section 5.2.3) to analyze the input data to detect artifacts in these data. In another work, Wolf et al. (1996) use fuzzy logic to detect artifacts in the observed vital sign data to minimize false alarms. However, in both of these works, newborn infants are monitored, which is not covered by the EWS method. In addition, the set of vital signs is different from that used in the EWS method. For example, the former includes ECG data, which is not practicable in a mobile EWS system (Liu et al., 2008).

The closest work to the one presented in this thesis is from Liu et al. (2008). Here, the data reliability of a patient's heart rate and respiratory rate is assessed with a fuzzy-based algorithm. The features are a bit different from the ones described in Section 5.4.2, and the number of vital signs considered is quite limited compared to the set used in the EWS method. This limited set is sufficient for their work, which is designed to monitor soldiers in combat zones but not to early detect a patient's deteriorating health conditions. Moreover, since soldiers are also equipped with other sensors to obtain their position and motion, Liu et al. (2008) classify the plausibility of vital signs as unreliable as soon as they are so critical that they become incompatible with a healthy person. However, if a patient's condition is very poor, the vital signs monitored

are not necessarily unreliable. Furthermore, in their experiments, unlike in this thesis, the errors in the data (e.g., spikes, noise, and abnormal slopes) were superimposed individually and not caused by real sensor faults or faulty usage (described in Sections 5.4.2 and 6.3.3).

Finally, Chapter 6 deals with the uncertainties in data acquisition. This is in line with the work on uncertainty as defined by Taylor et al. (1994)[3] where the uncertainty of a component can arise from a *random* or *systematic* effect. According to Apostolakis (1990) and Baraldi et al. (2014), uncertainty can be caused by the inherent variability of a physical system or the lack of precise knowledge. In the case of the EWS system, uncertainties are in the data acquisition or caused by the lack of complete knowledge about the monitored subject. The former can describe, for example, a slight measurement error when a vital sign value is exactly in-between two different scores. In contrast, the latter could, for example, describe the different characteristics of different persons, for whom the boundaries between the different scores of a vital sign could vary. To overcome these issues, all the abstraction steps, from the input data (the vital signs) to the system's output (the EWS), are based on various confidence values to increase the reliability of the EWS in the presence of erroneous input data.

2.6 Condition Monitoring Systems

Most systems and machines need regular maintenance and adjustments to properly operate over a certain period (Bousdekis et al., 2020; Ferreiro et al., 2016). Especially in industrial systems (e.g., a production machine in a factory), these actions are necessary to avert repairs and long downtimes and the high costs associated with them (Selcuk, 2017). Regardless of the application area, maintenance work on a system most likely leads to its prolonged durability (Salvia et al., 2015). However, such maintenance work is also associated with costs and possible downtimes (Susto et al., 2015). Consequently, high costs are expected in both cases, whether a system is maintained too seldom or too often. Thus, the goal is to carry out maintenance work as seldom as possible but as often as necessary.

In industry, especially in automated production plants, there exists a high interest in reliable autonomous monitoring systems that observe such a system/machine, in the following called System under Observation (SuO). Such a monitoring system could trigger an alarm if the SuO deteriorates or shows a malfunction (Thomson and Gilmore, 2003). An alarm would indicate that

[3]It must be noted that the work in this thesis only touches on the field of uncertainty. Elements, such as risk management, the possibility theory, the evidence theory, etc. (Aven et al., 2013), are beyond the scope of this thesis.

the SuO needs maintenance or repair before its condition worsens. Thus, an autonomous condition monitoring system can optimize the scheduling of maintenance work and minimize downtime as well as costs. This shows that besides medical applications (Section 2.5), monitoring and accompanying sensory data processing are relevant also in various other areas, such as in industrial processes.

A major challenge of such automated monitoring is the amount of time and labor required for the increased design and engineering effort. Implementing the necessary models, pattern recognition, or machine learning algorithms for a SuO is a complex, time-consuming, and costly endeavor. Therefore, from-scratch implementation of a tailor-made monitoring system for each SuO is neither economical nor feasible. To reduce development efforts and costs, generic methods are desired. These can be applied in a variety of different SuOs to detect their status and alert in case of deterioration.

Research on Automated Fault Detection and Diagnostic (AFDD) methods has steadily increased in recent years (Woohyun and Srinivas, 2018). These methods are distinguishable into qualitative- and quantitative models and models based on process history (Katipamula and Brambley, 2005). Qualitative and quantitative models require a representation (knowledge of structural properties or exact mathematical relationships) of the SuO (Van Harmelen et al., 2008; Lunze, 2016). Thus, they do not overcome the problem mentioned above of avoiding excessive development efforts for tailor-made monitoring systems for each SuO. Therefore, process-history-based approaches are of interest and discussed in the following. According to Woohyun and Srinivas (2018), around 62% of implemented AFDD methods are process-history-based and can be further divided into black-box and grey-box approaches.

In contrast to a black box, some specific knowledge is available about a grey box. This means that in grey-box monitoring, usually, knowledge of physical phenomena (physical knowledge about the system) is combined with knowledge of statistical information (information from the monitored data) (Macarulla et al., 2021; Massano et al., 2019a). This means that a grey-box monitoring system has some kind of model of the SuO (Massano et al., 2019b). Grey-box monitoring systems are beyond the scope of this thesis, but two related works are presented in the following to provide a broader view.

The work of Weyer and Hangos (1997) suggests a grey-box model-based method to detect faults in a heat exchanger. For this purpose, they make use of a first-principle model of the SuO (the heat exchanger) combined with a grey-box model of the possible fault, which is based on the knowledge about signal jumps that are small and infrequent at the beginning of the deterioration of the SuO's condition and become larger and more frequent as the SuO's condition further deteriorates. Another work, Pulido et al. (2019), proposes a hybrid

solution based on a Neural Network (NN) to monitor a beet sugar factory. The hybrid solution involves the building of a grey-box model of the SuO (the sugar factory) with knowledge about structural information that is linked to measurements with a priori known equations that describe the SuO. In other words, as in the previously mentioned work, the structural knowledge again consists of a first-principle model of the SuO.

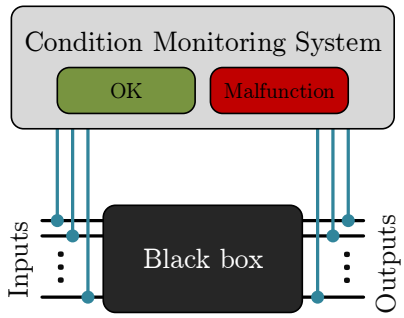


Figure 3. Block diagram of a black-box monitoring system

Figure 3 shows a block diagram of such a desirable black-box monitoring system that could monitor any SuO without needing any a priori knowledge of it. In the following, the author reviews some works about black-box methods. However, a more detailed review of related work is given in Papers III, IV, and V (Götzinger et al., 2019b, 2017b, 2018).

According to Katipamula and Brambley (2005), three different black-box methods have been studied so far: Artificial Neural Network (ANN)-based techniques, statistical methods, and other pattern recognition techniques.

Many research projects in this area are based on the use of Artificial Neuronal Networks. For example, He et al. (2011) proposed a fault detection system in which the centerpiece is an ANN-based on the hierarchical Adaptive Resonance Theory (ART). The ANN was trained with information obtained from a verified solar hot water system provided by TRNSYS (2019). In experiments, their system could detect rudimentary failures appearing in a Solar Hot Water (SHW) system. Du et al. (2014) propose another approach, namely integrating a dual neural network and subtractive clustering analysis into Heating, Ventilation, and Air Conditioning (HVAC) systems. They conclude that their approach's performance highly depends on the quantity and quality of the training data. In another work using an ANN, Nejjari and Benbouzid (2000) follow the so-called Park's vector approach to detect failures in an induction motor. For this purpose, they had to train Park's vector patterns to the ANN, which then enabled their detection system to identify malfunctioning with an accuracy of 97%. Besides these works, there exist many other ANN-based approaches such as those from Silva et al. (2006), Fan

et al. (2010), Hou et al. (2006), and Postolache et al. (1998). An ANN's advantage is its ability to model nonlinear systems without detailed knowledge (Dexter and Pakanen, 2001). However, modeling complex systems typically requires massive amounts of training data, which might cause incorrect output if incomplete. Furthermore, ANNs usually require considerable resources and computational power, which renders them usually unsuited for use in resource-constrained systems. Moreover, it is rather complicated to extract physical knowledge from an ANN, therefore, most researchers limit their work to extracting just the most comprehensible rules (Averkin and Yarushev, 2021; Dexter and Pakanen, 2001).

Among those applying statistical methods is the work of Guo et al. (2013), in which a Hidden Markov Model (HMM) encodes probabilistic relationships among variables of interest to detect faults of the HVAC system of a commercial building. Their results show that their system can identify failures that have been trained to the detection system in advance during the training phase. Hatzipantelis and Penman (1993) present another work of this kind. In their work, they also use an HMM for statistical pattern recognition applied to the diagnostic function of condition monitoring of electrical machines. After training, their approach detected trained faults with an accuracy of at least 80% accuracy. Besides these works, other publications also proposed systems based on statistical methods to deal with fault detection, such as Srivastav et al. (2013), Najafi et al. (2012), Li and Wen (2014), and Sharifi and B. (2011).

Examples of fuzzy schemes for failure detection come from Kang et al. (1991); Marcu and Voicu (1992); Frelicot and Dubuisson (1993); Sauter et al. (1994). Dexter and Ngo (2001) explain that failure detection based on fuzzy logic can also model nonlinear behavior. Furthermore, expert knowledge and knowledge learned from measurement data can be easily combined. Moreover, they usually require less computational power than other learning-based approaches. However, these fuzzy-based systems for failure detection also have their disadvantages. Their results are less precise compared to other approaches, and they may require application-specific assumptions or models. In addition, the rule-based descriptions used can be quite long.

In summary, all the methods and techniques reviewed here have various disadvantages. They either require application-specific models or considerable resources. Because of the demand for high computational power, most of them are only applicable to large-scale systems but not to ESs with limited resources. Moreover, most of them need a lot of training data that likely is not available for each SuO. Furthermore, the training data dependency can result in false detections if the training datasets are incomplete.

The system the author developed in the course of this case study is a lightweight black-box Condition Monitoring System (CMS). It does not require any model or knowledge of the SuO and uses only contextual knowledge (gained from sensory data) to make its decisions. The CMS is designed to monitor sensory data (the SuO's signals) and its changes to detect system states, state changes, normal behavior, and anomalies of the SuO. Since the system has no information about the SuO, the relationships between its various signals are also unknown. Only two assumptions are made here:

1. The SuO describes a bijective function, which means that a unique input dataset corresponds exactly to one and only one output dataset and vice versa. This definition implies abnormal behavior of the SuO if solely the input or output data change.

2. The SuO is in a steady state; transient states (e.g., sinusoidal) are ignored. When the SuO works properly (shows normal behavior), it is in a steady state or changes into another steady state.

The assumption that the SuO resembles a bijective function could seem somewhat contradictory to the statements that no knowledge of the SuO is required and that it, therefore, corresponds to a black box. However, the knowledge of the SuO being a bijective function does not provide any specific information about the SuO in particular, e.g., which variables are interconnected. It is only known to which group of systems the SuO belongs. From another point of view, it could be considered a black-box detection, which can only be applied to the group of systems that correspond to a bijective function.

Neither theoretical nor functional models of the SuO are available. The only knowledge needed is to identify the system's inputs and outputs that have a causal connection. Arguably, this could be seen as an indication that CMS is dealing with grey-box models. On the other hand, the input-output relationships are assumptions possibly requiring some technical common sense (i.e., the input current of a motor is connected to its rotational speed; the pressure in a chemical reactor is connected to the temperature) but no expert domain knowledge. Furthermore, assumptions are made only about the system's boundaries and not its internal details. According to Hauth (2008), all model development processes, whether white-, grey-, or black-box models, are "always driven by both prior knowledge and experimental data." Because of this very limited knowledge, the CMS can be described as a black-box monitor (Palm, 2007). In contrast to a black-box model, a grey-box model consists of expert knowledge (qualitative knowledge) and data knowledge (quantitative knowledge) (Hauth, 2008).

However, a worthwhile goal of future research would be to overcome the limiting constraints of being a bijective function and to replace the setup with an auto adjustment during an initial setup process of the CMS.

Besides detecting these inconsistencies between input and output data changes that indicate a defective device, the CMS should also detect slight deviations (drifts). Such drifts can occur when a machine is wearing out (Chammas et al., 2013). Identification of this circumstance would enable repairing the SuO before the device's condition further deteriorates.

In the course of this work, the author implemented two development stages of the CMS, which differ most noticeably in their decision-making process. The first version is called Context-Aware Health Monitoring (CAH)[4] and bases all its decisions (e.g., whether the SuO works well or shows a malfunction) on certain thresholds. This development stage is mostly about the abstraction of the states of a SuO out of its various signals. Therefore, this first development stage is explained in Chapter 4, which describes abstraction. The second version (the second development stage) of the CMS is called Confidence-based Context-Aware condition Monitoring (CCAM). It follows the same work principles as CAH, but its decisions are based on various assessed confidence values to deal with uncertainties in the data acquisition resulting from noise and the lack of knowledge about the monitored black box. It is introduced in Chapter 6, which describes confidence. Finally, Chapter 6 deals with the uncertainties in data acquisition. As with the EWS system (Section 2.5), this is in line with the work on uncertainty as defined by Taylor et al. (1994)[5]. In the case of the CMS system, uncertainties are in the data acquisition where data can be noisy, which makes it hard to distinguish between noise and a signal change, or because of a lack of knowledge about the SuO which, of course, is a black box. To overcome these issues, all decisions of the CMS are based on various confidence values to increase the reliability of the CMS output and make it more robust against noise.

Both systems (development stages) of this case study were tested on two different SuOs (two sub-case studies): on an Alternating Current (AC) Motor (used in a conveyor belt) and on a water pipe system[6]. The results obtained in these experiments are shown in Chapters 4 and 6.

[4]It must be noted that, in this context, the term "health" describes the condition of a machine or device; i.e., whether it is working properly or malfunctioning.

[5]It must be noted that the work in this thesis only touches on the field of uncertainty. Elements, such as risk management, the possibility theory, the evidence theory, etc. (Aven et al., 2013), are beyond the scope of this thesis.

[6]More precisely, the water pipe system is a HVAC system, described in more detail in Papers III and V.

3 Development of a Framework as Research Environment

Computational Self-Awareness (CSA) is a highly topical subject in science, industry, and academia (Guang et al., 2012; Faniyi et al., 2014). It has various properties that can help to make computer systems more autonomous as well as more intelligent and reliable (TaheriNejad and Jantsch, 2019; Dutt and TaheriNejad, 2016). However, this area is still widely unexplored and many aspects of CSA still need to be researched. Yet, the slow pace of development of self-aware systems results from the lack of a common framework for exploring CSA.

While Section 2.4 shows the disadvantages of existing frameworks, this chapter describes the development of the Research on Self-Awareness (RoSA) framework. RoSA is a standalone actor framework with an open-source standard native implementation[1] programmed in standard C++. The purpose of RoSA is to serve as a generic tool to help exploiting CSA and simplifying the development of self-aware systems. It (i) provides a high-level modeling interface for application developers, (ii) allows the same application code to be used for both simulation and deployment, and (iii) creates small-footprint software that can be used in resource-constrained Embedded Systems (ESs).

Section 3.1 describes the architecture of RoSA, including its rationale and principles as well as its uses. RoSA was used in all development steps of the systems related to the two selected case studies; Sections 3.2 and 3.3 show the implementation of these systems' underlying architectures. Finally, Section 3.4 demonstrates the usefulness of the RoSA framework. For further details on this chapter, the author refers to the appended Paper I (Götzinger et al., 2020)[2].

The methods developed for the corresponding self-awareness properties, abstraction, data reliability, and confidence, are presented in Chapters 4, 5, and 6.

[1] This open-source implementation is available at https://phabricator.ict.tuwien.ac.at/source/SoC_Rosa_repo.git.

[2] It should be noted that the author shares the first authorship of this paper with the second-listed author (Dávid Juhász). The author's contributions to this publication are listed in Section 8.1.

3.1 Architecture of RoSA

RoSA is a middleware providing novel features for developing and evaluating agent-based self-aware applications. The RoSA architecture sets the conceptual basis at the application level (i.e., agents and self-aware features). In other words, RoSA combines the agent-based actor model with self-awareness properties, and it is executable on ESs. In this context, the term *agent* describes a design abstraction, which Russell and Norvig (2010) define as "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators." An agent-based architecture helps to divide complex applications into smaller subtasks (Section 2.3).

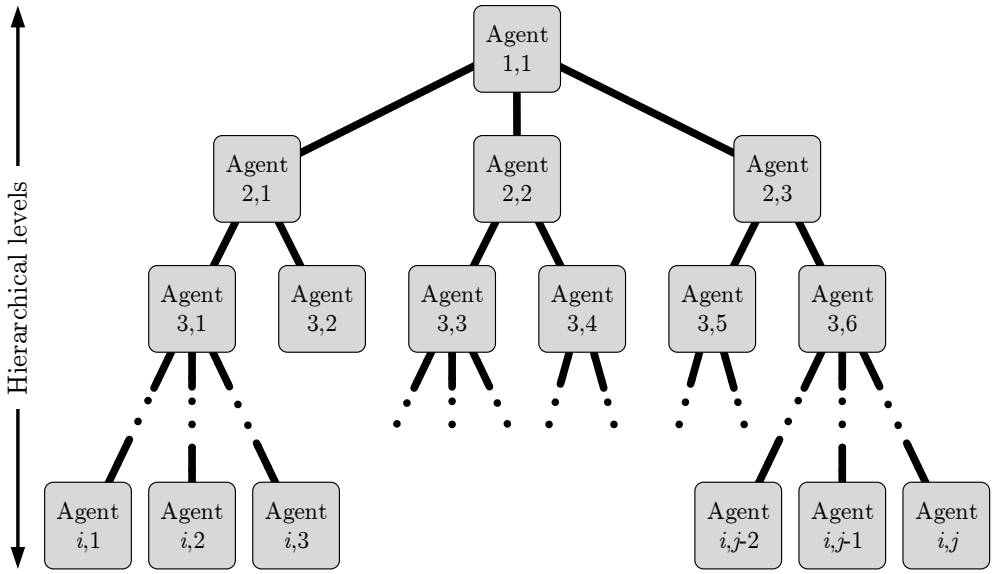


Figure 4. A RoSA application is implemented as a hierarchical agent-based model; in this example, the first digit of an agent's name refers to its hierarchical level and the second digit to its position within this level

Figure 4 shows that the RoSA agents are organized in a hierarchical structure to handle different application tasks on different abstraction levels. Different agents receive fine- or coarse-grained knowledge according to the hierarchical level on which they are located. Such a distribution of knowledge helps self-aware systems to work more efficiently and achieve their goals (Faniyi et al., 2014). In RoSA, connected agents are in master-slave relationships and can thus communicate with each other. An agent can be the master of any number of other agents, but can have at most one master agent. However, agents can also interact with their environment (via sensors and actuators). While possible sensors and actuators were faded out for the sake of simplicity in Figure 4, Figure 5 shows a possible system architecture in which also sensors

and actuators are used. A RoSA agent can be connected to as many agents, sensors, and actuators as needed for the particular application.[3] A sensor is a data source and sends sensor inputs as slave-to-master data messages to its master (the agent connected to the sensor). Thus, a sensor has exactly one master but cannot have any slaves. On the other hand, an actuator is a data sink and is controlled by slave-to-master data messages. Therefore, an actuator has one slave but cannot have any masters. This data flow from a sensor to actuators is the reason why actuators are on top of agents in Figure 5.

However, since agents and actuators work essentially in a data-driven manner, the use of the term slave-master to describe the connections between sensors and actuators might not be optimal. It can be better described as a client-server connection where a sensor sends a request to its connected agent (its server) in which it asks the agent to process the data it sends. Regarding the connection between an agent and an actuator, the situation is the same, except that the agent is the client and the actuator is the server.

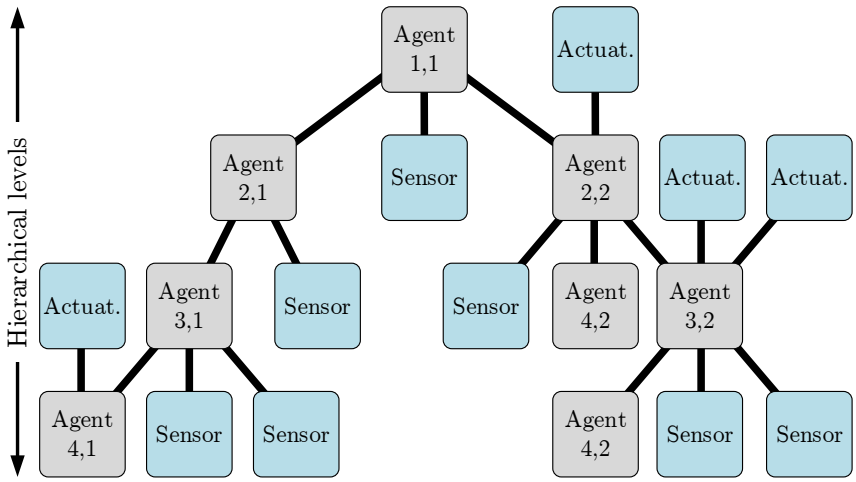


Figure 5. A RoSA agent can be connected with other agents as well as with sensors and actuators

Figure 6 corresponds to a small section of the exemplary agent system (from Figure 4) and shows that each RoSA agent is implemented as an Observe-Decide-Act (ODA) loop. The ODA loop belongs to the group of control loops that are also common in autonomic computing systems (Section 2.3), and instructs an agent to perform the following steps iteratively: (i) The system monitors its own behavior and that of its environment, (ii) then decides on specific actions based on the collected data, and (iii) acts accordingly (Parashar and Hariri, 2005; Dutt et al., 2016). In this example, assuming that the data

[3]The number of agents, sensors, and actuators is limited based on the hardware platform on which RoSA is running.

flow only goes from bottom to top, Agent 2,3 (Figure 6) is first observing (reading messages from Agents 3,5 and 3,6), then making decisions based on the data obtained as well as on its own goals, state, and knowledge, and then acting based on these decisions (e.g., sending relevant results to Agent 1,1). In real systems, of course, an agent can also send messages to its slave, and thus, Agent 2,3 might also have to read messages from Agent 1,1 and, if necessary, send messages to Agents 3,5 and 3,6.

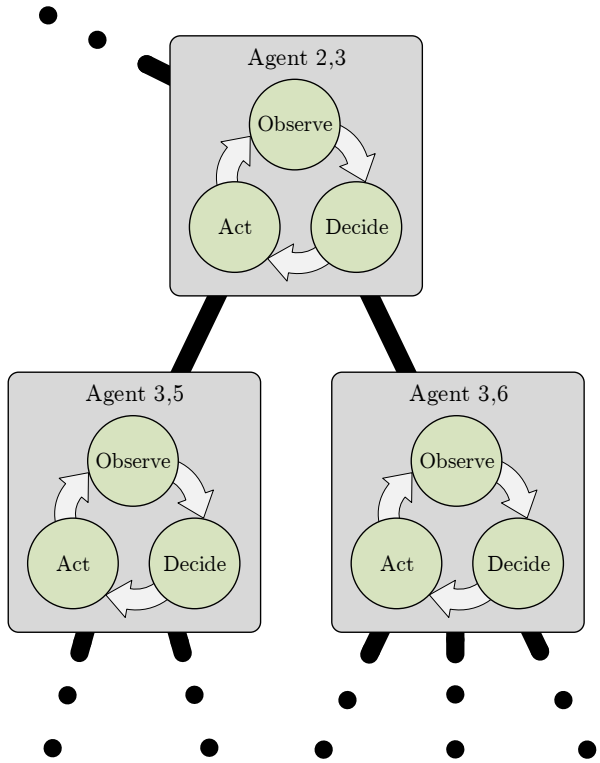
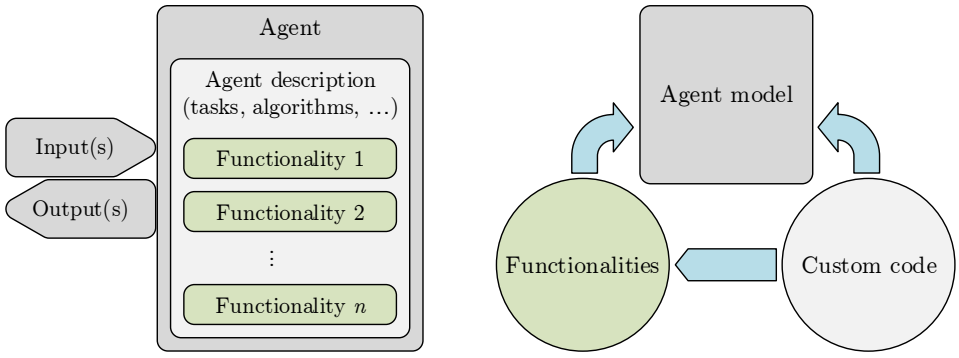


Figure 6. Each RoSA agent is implemented as an ODA loop

As mentioned in Section 2.3, the agent-based architecture offers the possibility to break down a self-aware application (acting as an ODA loop) into interacting ODA loops (located in the various agents) of lower complexity. The RoSA user can freely determine how many agents are to be created, which of them are to communicate directly with each other, and how many hierarchical levels should exist.

The user also defines the individual tasks of each agent. Figure 7a shows that an agent, respectively, its task(s) are defined by functionalities. Concerning RoSA, the term functionality describes a reusable self-awareness component or method that an agent can use to accomplish its task. The functionalities an agent is equipped with depends on its role in the application. In other

words, the application developer can choose which agent uses which functionalities. RoSA offers a library of predefined functionalities and thus enables fast implementation of self-aware applications. In the current implementation of RoSA, functionalities of the self-awareness properties abstraction, data reliability, confidence, and (in a very limited way) history are available. All these functionalities (self-awareness methods) have been developed by the author in the course of this thesis and are presented in the next chapters dedicated to each of the self-awareness properties.



(a) An agent's behavior defined by self-awareness functionalities

(b) An agent modeled with self-awareness functionalities and custom code; custom code can get promoted to functionalities

Figure 7. An agent is modeled based on available functionalities and custom code (Götzinger et al., 2020)

However, the modeling of an agent is not only based on functionalities but also on custom code. Since it is impossible for RoSA to offer all imaginable functionalities (self-aware or not), the application designer can also use custom code to describe an agent's behavior. RoSA, however, places great emphasis on modularity and reusability. Thus, every method, and even smaller components of it, should be reusable for other methods. Figure 7b illustrates an agent consisting of functionalities as well as custom code, and this custom code can also be integrated into new functionalities. The procedure of making functionalities out of custom code was also performed in the course of this work. In other words, the individual self-awareness methods (described in Chapters 4, 5, and 6) were initially custom code, and following successful experiments, they were encapsulated in reusable functionalities. It is to be expected that the number of functionalities will grow over time thanks to the research community's usage and help.

An agent works in the scheme of an ODA loop, in which the different self-awareness functionalities can be assigned to the loop's different phases. Abstraction (Chapter 4) improves the observation result so that the self-aware

system can understand or make better use of the data. On the other hand, data reliability (Chapter 5) and confidence (Chapter 6) help to make the right decisions in the decision phase.

Furthermore, it has to be mentioned that while the main objective of RoSA is to facilitate the development of applications and concepts related to CSA, the applicability of the framework is not limited to this. In principle, any application (also non-self-aware) that benefits from hierarchical agent-based modeling can be implemented within RoSA. Thus, the application designer has just to define the agents with an application-specific code without using self-awareness methods. Besides, it is also possible to migrate an existing application to RoSA or add RoSA as a self-aware component to an existing application. For further details, the author refers to Paper I (Götzinger et al., 2020).

The principles and rationales of the RoSA architecture are summarized in Table 5.

Table 5. The principles and rationales of the RoSA architecture

Principle	**Rationale**
Hierarchical agent system	Appropriate distribution of knowledge and division of complex applications into smaller subtasks (Faniyi et al., 2014; Guang et al., 2012)
ODA loop as agent control	Use in autonomic computing to enable appropriate Cyber-Physical System (CPS) actions as well as to adapt to changes in the environment (Parashar and Hariri, 2005; Dutt et al., 2016)
Agent tasks as self-awareness functionalities	Inclusion of functionalities that enable a comprehensive assessment of the system's state and its surroundings in a straightforward manner (TaheriNejad and Jantsch, 2019; Dutt and TaheriNejad, 2016)

3.2 Early Warning Score System in RoSA

Each development step of the Early Warning Score (EWS) system was done within the RoSA framework. In a first step, an agent-based system was modeled in which the different self-awareness functionalities and the custom code

can be integrated. This basic structure of the EWS system is shown in Figure 8. It consists of three hierarchical levels, whereby the lowest level consists only of sensors and the two levels above only of agents. In the following, the agent on the top level is called high-level agent and the agents on the level below are called low-level agents.

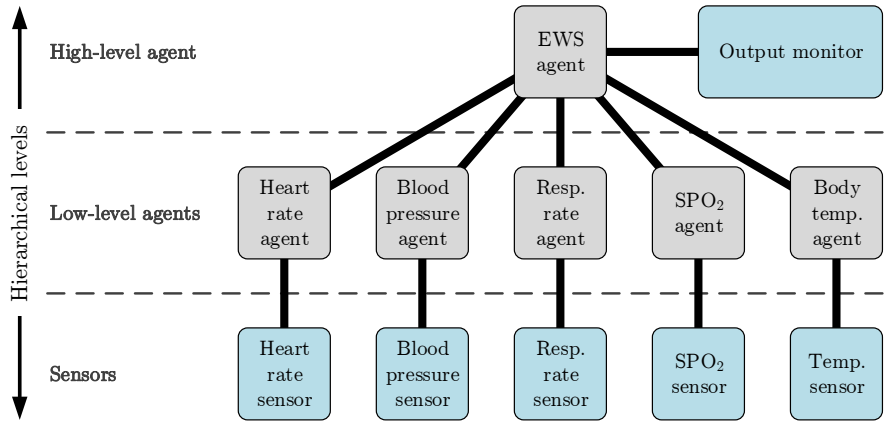


Figure 8. Architecture of the EWS system; the agents are in gray boxes and the various sensors as well as the output monitor are in blue boxes

The five low-level agents are assigned to the corresponding vital signs[4]: heart rate, blood pressure, respiratory rate, oxygen saturation (SPO_2), and body temperature. In principle, a low-level agent has two tasks: (i) read the raw values of the assigned vital sign sensors[5], and (ii) determine the appropriate vital sign score (according to Table 3).

In contrast, the high-level agent is responsible for calculating and displaying the patient's EWS. The steps of abstraction and its methods are explained in Chapter 4. In the course of this work, the EWS system was also equipped with the concepts of data reliability and confidence to make the EWS more robust and reliable. As described in Chapter 5, different data reliability methods require knowledge of different abstraction levels. Thus, in the hierarchy, one part of the logic of data reliability is located on the low level and the other on the high level. The same goes for confidence, described in Chapter 6, which can also be found on both hierarchical levels of agents.

[4]The level of consciousness (Table 3) is excluded because it is not applicable in out-of-hospital monitoring.

[5]The experimental measurements' vital signs were stored in Comma-Separated Values (CSV) files, which are loaded one after the other by the virtual RoSA sensors. These virtual sensors are modeled as agents in RoSA.

3.3 Condition Monitoring System in RoSA

As described in Section 2.6, the second case study is a Condition Monitoring System (CMS) that monitors a black-box system — the System under Observation (SuO). The CMS is designed to monitor the SuO's sensory data and its changes to detect system states, state changes, normal behavior, and anomalies.

There are two different systems or development stages of the CMS, which differ in their decision-making process. The first one, the Context-Aware Health Monitoring (CAH)[6] system, has a threshold-based decision-making process. Since it is essentially about abstracting system states from the SuO signals, the specific functional principle of CAH is explained in Chapter 4, which covers the self-awareness property abstraction. In contrast, in Confidence-based Context-Aware condition Monitoring (CCAM), the second system (second development step), all decisions are based on different confidence values. Therefore, Chapter 6, which covers the self-awareness property confidence, deals with the exact functional principle in more detail.

Nevertheless, the underlying agent-based model (the basic architecture) is the same in both systems. Figure 9 also shows that it is very similar to the EWS system architecture (Section 3.2). Agents are, again, located on two different hierarchical levels. While the low-level agents read the raw signal values from their assigned sensors[7] and recognize signal states, the high-level agent determines the system state and determines whether the SuO is functioning or malfunctioning. However, in contrast to the EWS architecture (Figure 8), it is noticeable that the low-level agents do not have exact names; they are simply enumerated. This is because the CMS monitors a black box and lacks efficient knowledge of its signals, which might either be pressures, voltages, currents, or many other signals. In other words, neither physical properties of the signals nor possible relations among them need to be known.

Moreover, Figure 9 does not show an exact number of low-level agents since the CMS was tested on two different SuOs (in two different sub-case studies): an Alternating Current (AC) motor used in a conveyor belt, and a water pipe system[8] driven by a Direct Current (DC) water pump. Both systems have a different number of signals, which leads to a different number of low-level agents. The AC motor of the first sub-case study has the following signals:

[6] It must be noted that, in this context, the term "health" describes the condition of a machine or device; i.e., whether it is working properly or malfunctioning.

[7] The experimental data was obtained from simulations and real measurements. It was stored in CSV files, which are loaded one after the other by virtual RoSA sensors. These virtual sensors are modeled as agents in RoSA.

[8] More precisely, the water pipe system is a Heating, Ventilation, and Air Conditioning (HVAC) system, described in more detail in Papers III and V

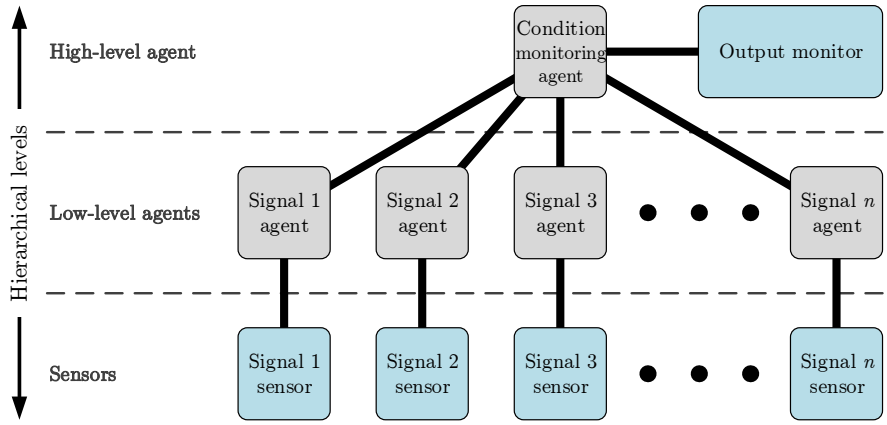


Figure 9. Architecture of the CMS, capable of monitoring a black box with n signals; the agents are in gray boxes and the various sensors as well as the output monitor are in blue boxes

voltage (AC), current (AC), load torque, electrical torque, and rotating velocity. Whereas the water pipe system has the following signals: voltage (DC), water temperature, and various water flow signals.

3.4 Evaluation of RoSA

The self-awareness methods were first developed with custom code and subsequently became reusable components. This way, it was possible to identify reusable components and separate the runtime system from the application code. However, as mentioned in Section 3.1, RoSA does not only simplify the implementation of a self-aware application because of its self-awareness functionalities. It also enables a simple and straightforward implementation of a hierarchical agent-based model whether self-aware or not.

To obtain a quantitative measure of how much development effort is simplified by using RoSA, a comparison between the number of non-comment lines of code of the original custom-written applications (implemented without RoSA) and the RoSA-based implementation was made. If the custom-written applications represent 100%, the RoSA-based implementations make up only 3.46% to 6.24%, depending on the application. On average, across all implemented applications, this is about 5.15%. On the other hand, the framework comes with a slight overhead. The framework code itself has, on average, 20.75% more lines of code than a custom application. However, the framework code needs to be implemented only once and can then be reused as often as desired. In other words, the implementation of this framework pays off as it can be used for multiple applications. This reusability is exactly the purpose of RoSA and was proved in this work.

Another requirement of RoSA was to work also on ESs, which have limited resources. To show the advantage of RoSA, the memory footprint of RoSA's binaries was compared to that of the CAF framework. The binary size of the RoSA libraries on an x86-64 Linux machine is 300kB, while that of the CAF (Section 2.4) core library (version 0.17.5) is 7248kB. This significant (24 times) difference in favor of RoSA indicates that the framework is applicable to considerably smaller systems than CAF. To prove this hypothesis, the applications were also executed on an ODROID XU4 (Hardkernel, 2017) and Raspberry Pi 3 (Raspberry Pi (Trading) Ltd., 2019). The ODROID board has an eight-core big.LITTLE (ARM, 2013) configuration with Cortex-A7 and Cortex-A15 cores, and the Raspberry Pi 3 has a quad-core configuration with Cortex-A53 cores. Thus, three different configurations could be tested. The applications created a moderate memory footprint well below 4MB and, therefore, fit the typically limited memory of an ES. Moreover, depending on the application and the ES on which it was running, samples were processed around 5 to 4500 times faster than required for real-time execution. For further details, the author refers to Paper I (Götzinger et al., 2020).

However, despite all these achievements, there are still several improvements that could be made to RoSA. While it can run on workstations and ESs, conceptually, RoSA is also intended for running on distributed systems. However, while the current architecture would not be an impediment, the current implementation is rather limited, therefore, some implementation work is still needed to enable this feature. Moreover, RoSA as a middleware still lacks important capabilities (e.g., load balancing, resilience, message ordering) that would make the system dependable in a distributed setup. However, these features could also be added to the existing codebase. These implementation details are beyond the scope of this thesis, which focuses on a reliable self-aware monitoring ability.

4 Abstraction

The self-awareness property abstraction is highly versatile and can be done in many different ways, as described in this thesis. Some of the implemented methods are closely related to the other two self-awareness properties (data reliability and confidence) presented in Chapters 5 and 6.

This chapter starts with the formal definition of abstraction (Section 4.1), based on Paper I (Götzinger et al., 2020)[1]. Section 4.2 presents implemented abstraction methods, followed by Section 4.3, which elaborates the usage of abstraction in the Early Warning Score (EWS) case study, based on Papers VII and VIII (Götzinger et al., 2016, 2017a)[1]. Finally, Section 4.4 deals with the usage of abstraction in the field of the Condition Monitoring System (CMS) case study, based on Papers IV and V (Götzinger et al., 2017b, 2018))[1].

4.1 Formal Definition of Abstraction

While Giunchiglia and Walsh (1992) defines the property abstraction as "the process of mapping a representation of a problem onto a new representation," TaheriNejad et al. (2016) formulates abstraction as "an appropriate selection of the representation of the information in order to obtain compact knowledge relevant to a particular purpose." According to Dutt et al. (2016), abstraction is defined as "the primary input data into a semantic domain which is meaningful for the system at hand." As these definitions are very broadly formulated, abstraction can be summarized as a mapping between formal systems and shall make sense and be useful in the context of a system. This mapping can be mathematically described as

$$f\colon X \longrightarrow Y, \tag{1}$$

where elements of set X are mapped onto elements of set Y.

The range of abstraction possibilities is very large, and a wide variety of approaches is possible. For instance, raw input data could be transferred to a semantic domain that the self-aware system understands. While it is

[1]This dissertation is based on eight original publications, which are the results of collaborations. These publications can be found in the appendix. The author's contributions to each publication are listed in Chapter 8.

important that abstraction should be meaningful, efficient, and have a well-defined structure, it does not even matter where such an abstraction takes place. It could be done at any hierarchical level of a system, and even whether it is top-down or bottom-up is not specified (TaheriNejad et al., 2016).

4.2 Abstraction Methods

In the different development stages of the two case studies, a set of different abstraction methods come into use. These are as follows[2]:

1. **Lookup table:** A lookup table assigns a symbol to given input data. Such a symbol could be, for example, a number, a character, or a string. Section 4.3 demonstrates the usage of this abstraction method.

2. **Overlapping lookup table:** While a standard lookup table maps input data (e.g., an input value) to exactly one symbol, an overlapping lookup table potentially assigns several symbols to such given input data. This is necessary if the boundaries between the symbols cannot be sharply defined, e.g., due to insufficient knowledge about the environment. Both the abstraction step itself and the subsequent processing step are characterized by the self-awareness properties data reliability and confidence and are, therefore, described in detail in Chapters 5 and 6, respectively.

3. **Signal state detector:** The function of a signal state detector is to abstract steady states from a signal course (i.e., a sequence of input values). In other words, it detects stable phases in a potentially changing signal. Stable states are characterized by a small distance between the values of several sequential samples of an input signal. For each signal state found, representative data is stored in the signal state detector. Two different approaches are chosen to recognize such states:

    - Threshold-based detection: An average value of all input samples belonging to a state is stored as a state's representative. Whether a sample belongs to a state is decided with respect to a threshold for the relative distance[3] between the sample's value and the average value of the state. An application of this abstraction method is shown in Section 4.4.

[2]It should be noted that this list does not guarantee the completeness of all possible abstraction methods. It only shows the abstraction methods used in the course of this thesis.

[3]Distance here means a difference in any dimension, e.g., performance- or success-rate difference, geometric distance, time difference, or absolute values difference.

- Confidence-based detection: Evaluation whether a sample belongs to a state is based on confidence-based decisions (see Chapter 6). In addition, states are not stored just as average values, but in a sliding window history that also serves as a basis for above-mentioned evaluation.

4. **System state detector:** A system state detector abstracts system states out of the various signal states of a System under Observation (SuO). In the current implementation, only stateless SuOs are considered. These are systems that express their states depending only on their inputs and not on their internal state memory. Section 4.4 explains the usage of this abstraction method.

5. **General mathematical representations:** The set of possible functions is huge. Therefore, only the implemented functions are listed here, for example, a function to abstract the amplitude from a sinusoidal signal. Additions or suchlike operations are other examples of such functions. Besides, signal filters will also be considered here because they enable to abstract a clean signal from a noisy input stream. Implementations in Sections 4.3 and 4.4 make usage of such methods.

4.3 Abstraction in Early Warning Score System

The very nature of the application, in which vital signs are to provide information about the health of a patient, makes it clear that abstraction must be used. In fact, this process consists of two abstraction steps, one in the lower, one in the upper hierarchical level of the EWS system's architecture (Section 3.2). Figure 10 shows a simple schematic of the abstraction procedure for one low-level agent. To simplify, the other low-level agents were faded out. Firstly, each vital sign agent (located in the lower hierarchical level) receives a raw vital sign value from its connected vital sign sensor. Then the first abstraction takes place. Each of these low-level agents abstracts the raw vital sign to the corresponding vital sign score using a lookup table (Section 4.2). The lookup table of a low-level agent equals the row assigned to the specific vital sign in Table 3. Subsequently, all vital sign agents send the abstracted vital sign scores to the connected EWS agent (located in the higher hierarchical level).

After the EWS agent has obtained all five vital sign scores, it can abstract them to the EWS. This is done by adding up all these scores. That a simple addition is also considered to be an abstraction process may be a bit surprising at first. However, as already explained in Section 4.1, the property abstraction is very broadly formulated. The formation of the EWS is, after all, a

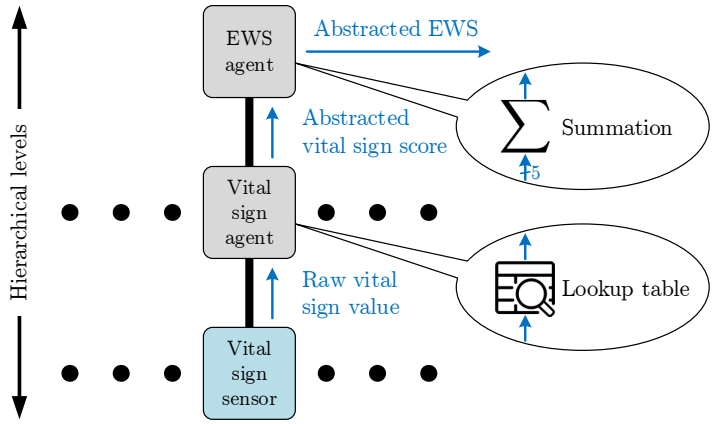


Figure 10. The two abstraction processes transform the raw vital sign values to the abstracted vital sign scores and subsequently to the abstracted EWS

mapping of one domain (the states of the different vital signs) onto another (the patient's state of health).

This section shows that abstraction methods need not be very complex or complicated. Moreover, an abstraction method does not necessarily bring about improvements of every application. In some cases, abstraction methods are just necessary for the execution of the application itself. Abstraction is used to determine the EWS based on the patient's vital signs. This procedure reproduces the normal EWS system used in hospitals (Morgan et al., 1997), but it does not make it more reliable or robust. Therefore, no meaningful results in the context of this thesis and its assumptions can be presented here. Improvements of the EWS system are achieved with the self-awareness properties data reliability and confidence. In Chapters 5 and 6, these two properties will be discussed respectively and results presented.

4.4 Abstraction in Condition Monitoring System

The first development stage of the CMS is called Context-Aware Health Monitoring (CAH)[4], and its decisions are threshold-based. In the following, the author gives an overview of this system, but refers for more details to Papers IV and V (Götzinger et al., 2017b, 2018).

The task of the CMS is to detect the SuO's system states (normal working states) as well as its health condition. For example, if the CMS observes a motor, the different system states could be: switched off, running slow, running fast. Similar to the fact that the CMS does not know what it monitors (in

[4]It must be noted that, in this context, the term "health" describes the condition of a machine or device; i.e., whether it is working properly or malfunctioning.

this case a motor), it also does not know what the recorded states mean. The CMS only records the various signals (inputs and outputs) of a SuO and recognizes changes in them; i.e., in this motor example, the CMS would not know whether the different states mean that the motor is switched off, running slow, or running fast. It would just know that the states differ from each other. Thus, the CMS cannot store meaningful names of the recorded SuO states but simply numbers them according to the time of occurrence (1, 2, 3, etc.). This means that a system state has no physical relation to the number assigned to it.

A system state detector detects system states which are described by the individual signal states. In other words, it is the task of the system state detector to abstract system states from the states of the different SuO signals. In contrast, signal state detectors abstract these needed signal states from the various SuO signals. Figure 11 shows a simple schematic of this abstraction procedure for one low-level agent. The other low-level agents of this architecture (Section 3.3) are faded out for the sake of simplicity. Because every signal has to be abstracted separately, there is one signal state detector for each signal. Since a signal state detector needs direct (unabstracted) knowledge about the signal assigned to it, it is placed in the lower hierarchical level. In contrast, a system state detector needs already higher abstracted knowledge (the signal states) and is, therefore, placed in the hierarchical top level.

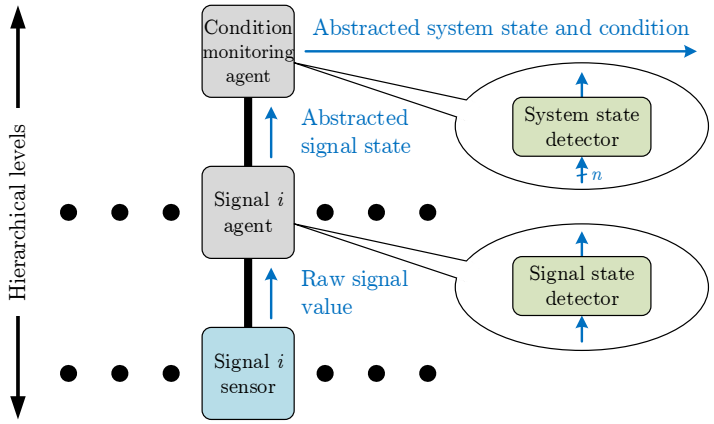


Figure 11. The abstraction process is completed in two steps: abstracting signal states out of the SuO signals and abstracting a system state out of the signal states

Figure 12 is a very simplified example to explain the function of both the signal state detector (Section 4.4.1) and the system state detector (Section 4.4.2). In this simplified example, the CMS monitors a SuO that has only one input and one output.

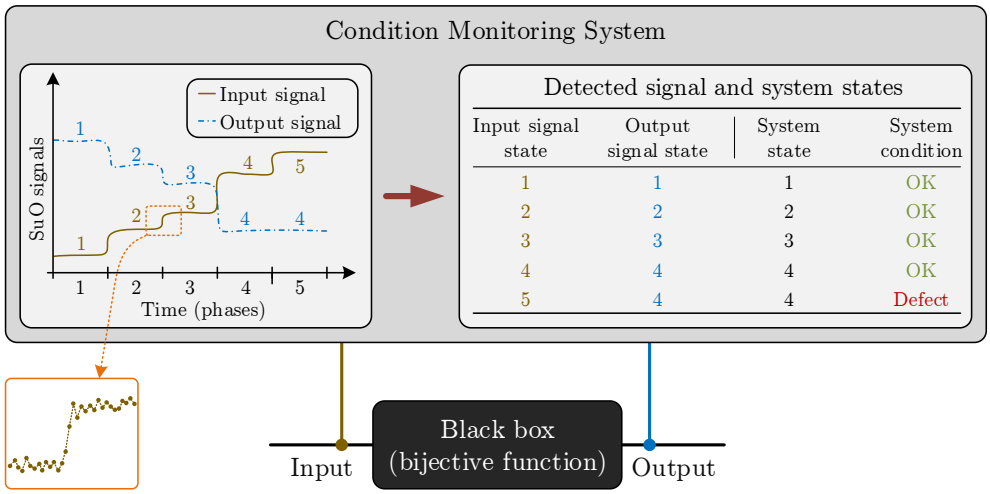


Figure 12. A simplified example in which a CMS monitors a SuO that has only one input and one output

4.4.1 Threshold-based Signal State Detection

A signal state detector (located in a low-level agent, Figure 11) analyzes the signal samples that its agent obtained from the corresponding sensor. If several samples are in close proximity to each other, they belong to the same signal state. If a signal state was found, a representative is stored in the signal state detector. Each signal state detector has a set $S = \langle s_1, ..., s_n \rangle$ in which all n signal states found, s_1 to s_n, are stored. A stored signal state, s_i, is represented by its name (its state number, e.g., "1", "2", and so forth) and the average value, v_{avg}, of all samples belonging to this state.

When a new sample, v_{new}, arrives, the signal state detector must find out to which existing signal state it belongs. Whether v_{new} belongs to the state $s_i \in S$ is determined by the relative distance between it and the average value, $v_{avg,i}$ of s_i. This distance is calculated by

$$d = \left| \frac{v_{new} - v_{avg,i}}{v_{new}} \right|, \tag{2}$$

and if it is smaller than a certain threshold, d_{th}, the v_{new} belongs to s_i. If v_{new} belongs to s_i ($d < d_{th}$), it is inserted in s_i. The insertion is done by a recalculation of v_{avg} but now also including v_{new}. If v_{new} does not belong to any of the existing states in S, the signal state detector creates a new state and adds it to S so that the size of S increases by 1.

In the example of Figure 12, the two SuO signals, which change over time, can be seen on the left side. These changes indicate that the SuO has changed its state (Section 4.4.2). The first two columns of the table in Figure 12 show

the corresponding abstracted signal states (detected in the changing signals). As mentioned before, for each signal, there exists one signal state detector. This means each of these two columns contains the output of one signal state detector. For better understanding, these outputs (in the table on the right side of Figure 12) are shown in the same color as the corresponding graphs of the signals (on the left side of Figure 12).

Usually, the measurement of a signal is not ideal but contains noise (illustrated in the orange frame in Figure 12). Therefore, the threshold, d_{th}, must be set[5] so that this noise does not lead to the detection of a state change while real (noticeable) signal changes are detected correctly.

One of the assumptions mentioned in Section 2.6 is that the SuO is in a steady state, and therefore, transient states shall be ignored. Thus, the signal state detector considers a new signal sample only if it is sufficiently close to a set of the most recent samples (stored in a sliding window history). In other words, the input signal needs to be sufficiently stable. This stability check is also based on the relative distance calculation (Equation 2) and a corresponding threshold, and prevents the inclusion of signal samples belonging to transient states. For more details, the author refers to Paper IV (Götzinger et al., 2017b)[6].

One of the two sub-case studies was an Alternating Current (AC) motor of a conveyor belt, so two signals (electrical voltage and current) were sinusoidal. Since transient states are to be ignored and a sinusoidal signal is, by definition, never steady, a further abstraction step is necessary ahead of the signal state detection. In this additional abstraction process (Section 4.2), the amplitudes are abstracted out of these sinusoidal signals. In further consequence, changes in such a signal's amplitude are traceable and indicate signal state changes.

4.4.2 System State Detection

A system state mirrors the simultaneously occurring signal states of all signals (input and output). The recognition of the SuO's condition corresponds to the assumption that the SuO behaves like a bijective function. Thus, a unique input dataset leads to exactly one unique output dataset. Consequently, the output dataset must change if there are changes in the input dataset and vice versa. If only the input dataset changes while the output dataset remains unchanged (or vice versa), the SuO is malfunctioning. Therefore, the CMS

[5]Since the SuO is a black box, the CMS has no knowledge about it, and all parameters (such as d_{th}) are set arbitrarily (Section 4.4.4).

[6]In this paper, the term injective instead of bijective was chosen. This is imprecise but not wrong because a bijective function is also always injective. However, using bijective is more precise and means both injective and surjective.

must consider the input dataset (set of input signal states) and the output dataset (set of output signal states) separately.

The table in Figure 12 shows not only the detected signal states but also the system states abstracted out of this information as well as the determined system condition (last two columns). In this example, the SuO only has one input and one output. When both corresponding signal state detectors have detected the first signal state (state 1 in the two left columns), the first system state is detected. Since the numbering of system states is also based on the first occurrence, this first state has the number 1 (shown in the third column). In continuation of this example, the SuO signals change into other states and, therefore, further system states are detected. With all state changes from phase 1 to phase 4 in this example, there are changes in both the input and output signal. Thus, according to the definition of a bijective function, the SuO works correctly, as shown in the last column. However, when changing from phase 4 to phase 5, the input signal changes but not the output signal. Since this behavior does not correspond to a bijective function, a malfunction of the SuO is determined.

It should be noted that the system condition in the first state is only an assumption because the CMS is not aware of any changes when entering the first system state. So if the CMS is initialized while monitoring a malfunctioning SuO, it is possible that the condition monitoring does not work correctly due to wrongly learned facts.

In all physical processes, certain inertia and delays are present. Thus, the SuO's output also needs a certain time until it reacts to a change in the input. Therefore, the CMS allows for a delay between a change of the input and the output. If the measured delay is higher or equal to a threshold, the SuO is classified as defective. In contrast, if the delay is below this threshold, the SuO works properly.

4.4.3 Drift Detection

The CMS is not only expected to recognize whether the SuO is already defective, but also whether a deterioration of its condition has occurred. Such deterioration could be indicated by drifting signals (Chammas et al., 2013). For this reason, the CMS must also detect drifts. Since such a drift is about a change in a signal, this detection is handled by the assigned signal state detector (located in the low-level agent in Figure 11).

A drifting signal changes slowly but continuously. In other words, the changing sample values of a drifting signal belong to the same state, but the signal is gradually deviating from its normal expected range. Since the insertion of samples of a drifting signal leads to a state's average value following the

drift, a drift cannot be detected by comparing a new sample with the average value. Therefore, not only the average value is stored as a representation of a state, but also so-called Discrete Average Blocks (DABs). A DAB is an average value of a certain number of consecutive samples. When this number of samples is reached, a new DAB is created. Thus, future samples will not change the completed DAB anymore. New samples are only inserted in the latest (incomplete) DAB. All DABs are stored in a DAB history, and if a new DAB is significantly different from a previous one, a drifting signal is detected. In other words, if the relative distance between these two DABs is below a certain threshold, no drift is detected. On the other hand, if this distance is greater than or equal to the threshold, the CMS informs about a drift (a possible deterioration).

4.4.4 Experiments Conducted

The CMS was tested on two different SuOs: on an AC motor used in a conveyor belt and on a water pipe system[7] driven by a Direct Current (DC) water pump. The AC motor in the first sub-case study has the following signals: voltage (AC), current (AC), load torque, electrical torque, and rotating velocity. The motor datasets were obtained through simulations as well as real measurements. In contrast, the water pipe system datasets came solely from real measurements and contained the following signals: voltage (DC), water temperature, and various water flow signals.

In both sub-case studies, experiments were conducted in which the respective SuO (i) functioned normally and performed regular state changes, (ii) showed malfunctions as well as (iii) drifting signals. For each of these cases, one experiment is shown below. For further information and more experiments, the author refers to Papers IV and V (Götzinger et al., 2017b, 2018). In the upcoming Figures 13, 14, and 15, the SuO's input signals are dotted, the SuO's output signals are dashed, and the CMS's output (SuO's state and condition) is represented through solid lines.

Figure 13 shows a test scenario of the AC motor at normal operation with two regular state changes. In this scenario, the motor is started and stays in its state for the first 2s. Due to the power-on process, the SuO output signals show strong oscillations at the very beginning, and after 1s, the CMS has found the first stable state (state 1). As already mentioned, in the beginning, the CMS just assumes that the SuO operates correctly. After 2s, one input signal, the load torque (M-torque), increases from 0Nm to 10Nm. While the input voltage remains unchanged, all output signals react to the changed load.

[7]More precisely, the water pipe system is a Heating, Ventilation, and Air Conditioning (HVAC) system, described in more detail in Papers III and V

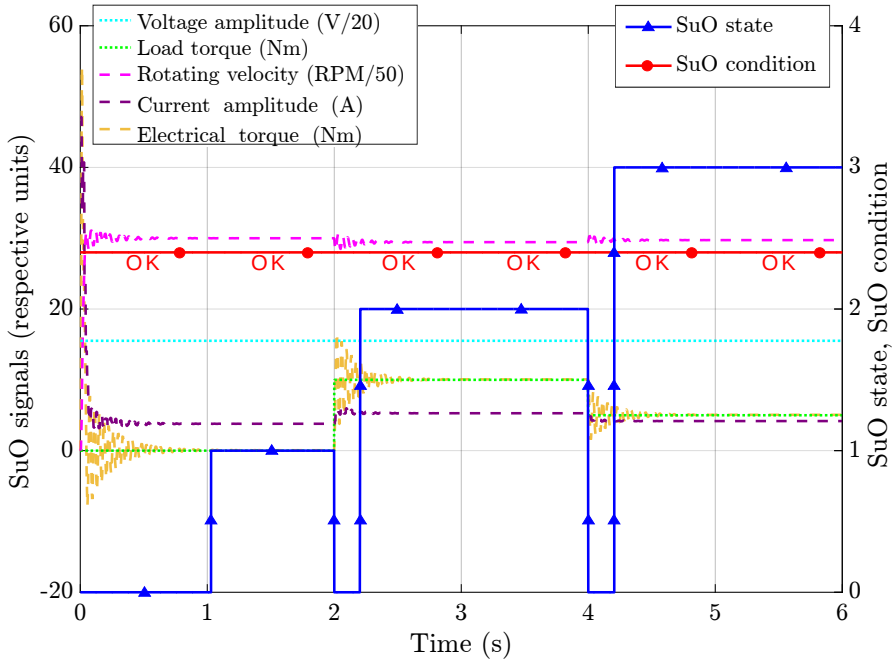


Figure 13. The output of the threshold-based CMS when monitoring a normally operating AC motor while two load changes happen

During their settlement process, the output signals again show oscillations, but these are less distinct. Approximately 0.2s after the load change, the CMS recognizes a new state (state 2), and since both the input as well as the output datasets have changed, the CMS considers the SuO's condition still to be OK. At 4s, another load change (from 10Nm to 5Nm) is performed. Because of the smaller lift of the load torque, the signal oscillations are slightly less distinct. At 4.2s, the CMS found state 3 of the SuO. The SuO is still classified as working correctly since its inputs and outputs have both changed.

In the second experiment (Figure 14), the CMS monitors the same AC motor. However, in this scenario, the motor does not run normally but shows a drift that could indicate a wear-out. In other words, the motor is still working, and input and output signals do not change states, but one or more signals are slowly drifting away from their nominal value. At about 0.9s, the CMS has found the first (and only) state in this scenario. However, the magnification (shown in the orange frame) shows that the rotational speed changes very slowly (around 0.013RPM/s). The CMS detects this slight change at about 2.3s and shows this change through its SuO condition output. After about one more second, the SuO condition output again changes and now displays a malfunction. Such behavior occurred for all available experimental data. An

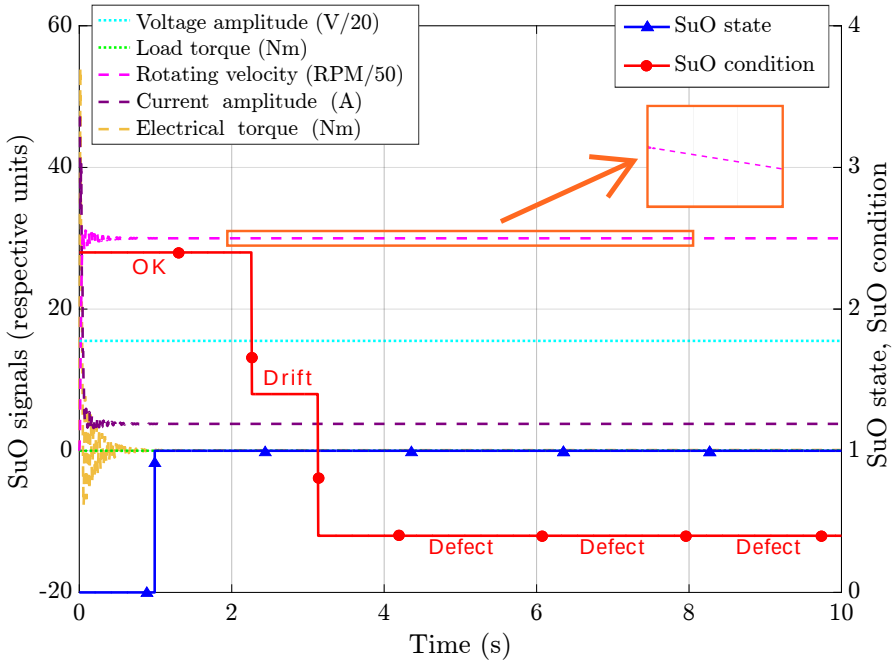


Figure 14. The output of the threshold-based CMS when monitoring an AC motor showing a drift

adjustment of the CMS parameters (e.g., thresholds) could only cause a delay of this phenomenon but not prevent it. There are setups with extreme threshold values that prevent this phenomenon, but they also lead to non-detection of a malfunctioning SuO in other experiments. Therefore, these setups are not appropriable for the correct detection of a SuO's condition. In summary, the target has been achieved to a certain extent, but there is considerable potential for improvement. The follow-up system, Confidence-based Context-Aware condition Monitoring (CCAM), makes all decisions based on confidence values, which leads to much better results in this respect (presented in Section 6.4.2).

Figure 15 shows a scenario of the other sub-case study, where a water pipe system is malfunctioning. The scenario begins with the start-up of the water pump, which then runs at constant speed throughout the entire experiment. Due to the water pipe system's inertia, it takes several seconds until the output signals are more or less stable. After about 40s, the CMS finds a steady state in this scenario, but at 310s, the abrupt opening of several valves simulated the behavior of a burst pipe. Therefore, several output signals suddenly change their state while the only input signal (voltage of the pump) remains constant. According to the definition of a bijective function, such behavior is not allowed. Thus, at around $245s$ the CMS classifies the SuO as malfunctioning. While this result is quite satisfying, there is still potential for improvement. Chapter 6

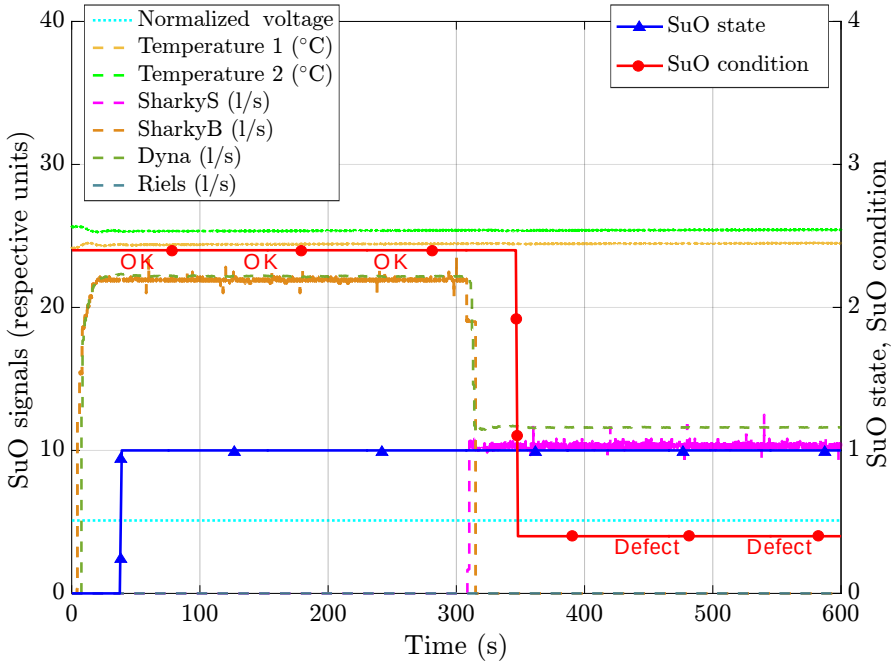


Figure 15. The output of the threshold-based CMS when monitoring a malfunctioning water pipe system

shows that confidence-based decision-making can significantly shorten the time between the event's occurrence and the CMS's alert.

Besides the successful recognition of the states and conditions of the SuO, other results and insights were also obtained. As already mentioned in Section 3.4, it was possible to run the CMS on different Embedded Systems (ESs), which were able to process samples faster than required for real-time execution. This is possible especially since the CMS is able to read signal data with a down-sampling factor. In the examples shown, only every 50th signal sample was evaluated. This behavior enables to save considerable computational power. It should be noted that most likely such down-sampling also leads to a delay in recognition. Moreover, in certain situations, it may obscure the behavior of the monitored signals between sampling times. Therefore, it depends on the application whether down-sampling should be used or how large it should be at maximum.

For the sake of completeness, it should be mentioned that, despite thresholds that need to be set, the CMS can be considered a black-box monitoring system. This setup is exclusively about internal CMS thresholds, which do not require any knowledge about the processes in a SuO. In the experiments, these thresholds were set completely arbitrarily. Regarding this setup process, in Paper V (Götzinger et al., 2018), the author presents also a sensitivity anal-

ysis of the CMS. The CMS displays a certain robustness against alterations of the CMS parameter (e.g., thresholds) setup. However, this analysis also shows that the time allowed between a SuO's input change and output change very much depends on the type of SuO. While several parameter sets of the CMS exist, with which the scenarios of both SuOs (AC motor and water pipe system) could be recognized correctly, the time parameter had to be different because of these two systems' vastly different inertias. Delays are purely phenomenological, and the respective setup was also done arbitrarily. However, a worthwhile goal of future research would be an automatic adjustment during the initial setup of the CMS to render all current manual settings completely obsolete.

Besides the already mentioned problem of drift detection, another disadvantage of the threshold-based CMS was identified. Due to the partly strong oscillations of the motor output signals, a low-pass filter to smooth them out had to be used as an additional abstraction step. For more details, the author refers to Paper IV (Götzinger et al., 2017b). Such additional filtering step is not necessary when using the confidence-based CMS (Section 6.4).

5 Data Reliability

This chapter starts with the formal definition of data reliability in Section 5.1. Section 5.2 then presents the various concepts developed in the course of this work, followed by Section 5.3 in which the causes of unreliable data are discussed. Finally, Section 5.4 shows the different ways of applying data reliability in the various development stages of the Early Warning Score (EWS) system as well as the results of the experiments performed. While the formal definitions of data reliability and the methods used in the latest development step of the EWS system are originally derived from Paper II (Götzinger et al., 2019a)[1], the enhancement of the EWS system by this self-awareness property is based on Papers VII and VIII (Götzinger et al., 2016, 2017a)[1,2]. For more details, the author refers to these three publications.

5.1 Formal Definition of Data Reliability

In contrast to the self-awareness property abstraction (Chapter 4), data reliability is defined much more precisely. According to TaheriNejad et al. (2016), data reliability is "the extent to which a measuring procedure yields the same results on repeated trials." Data reliability is metadata that describes the trustworthiness of a given dataset. It consists of accuracy, precision, and truthfulness. This relationship can be mathematically described as

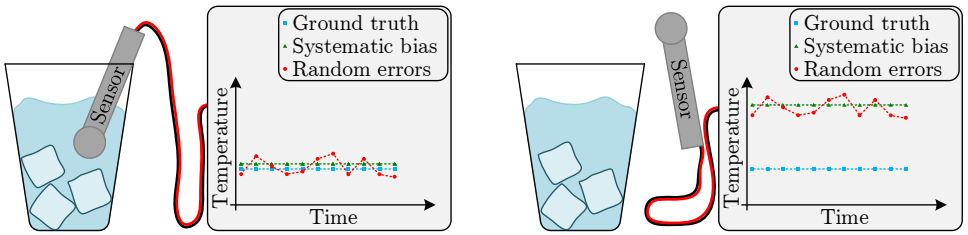
$$R_f(X') = f(A(X'), P(X'), T(X')), \tag{3}$$

where f determines the role of each of the three parameters and, consequently, how well R would fit its purpose. $X' = \langle x'_0, ..., x'_n \rangle$ is the input dataset at hand, in contrast to its ground truth dataset, $X = \langle x_0, ..., x_n \rangle$, which an ideal sensor would provide. $A(X')$ stands for accuracy which describes the systematic deviation (systematic bias) of the measured data values, X', compared to the actual values of the observed quantity X. On the other hand, precision, $P(X')$ is a stochastic variable that accounts for the random errors that would be different for each repetition of the same measurement under the same conditions.

[1] This dissertation is based on eight original publications, which are the results of collaborations.

[2] In Paper VII (Götzinger et al., 2016), which constitutes the author's first thesis-relevant publication, data reliability was erroneously called confidence.

Finally, $T(X')$ represents the truthfulness of the input dataset X'. A sensor may be accurate and precise, but may be used outside its assumed working conditions. In this case, a sensor may be entirely accurate and precise but would still fail to provide truthful data.



(a) A correct water temperature measurement setup

(b) An incorrect water temperature measurement setup

Figure 16. A simplified illustration of two different water temperature measurement setups

To visually demonstrate these relationships, Figure 16 serves as a simplified illustration of a theoretical experiment where the water temperature in an ice-cooled glass of water is measured. The blue data points (square line markers) indicate the ground truth of the water temperature, which may be measured by an ideal sensor. Since in reality ideal sensors do not exist, the actual measurement would never be absolutely correct.

In Figure 16a, the green data points (triangle line markers) indicate measurement results that would occur if the sensor was entirely precise and inside its assumed working conditions but had a limited accuracy, $A(X')$. This would mean that there was only a systematic bias. In contrast, an entirely accurate sensor that is inside its assumed working conditions but has limited precision, $P(X')$, may provide the red data points (circle line markers). It would only produce random errors during the measurements. However, actually measured data points would include both systematic and random errors (Equation 3).

Because this theoretical experiment aims to measure the water temperature, Figure 16a shows a correct measurement setup[3]. The water temperature is the assumed measurement output. Therefore, to use the sensor in its assumed working condition, it has to be located in the water glass[4]. In contrast, Figure 16b shows the sensor outside the water during the measurements. Hence it is outside its assumed working conditions since it measures the air temperature. If the air temperature is unequal to the water temperature, the result cannot be truthful — no matter how accurate and precise the sensor.

[3]Valid on the assumption of a homogeneous temperature distribution throughout the water.

[4]Valid on the assumption of a water temperature that does not exceed the sensor's temperature range.

All graphs in Figure 16b mean the same as in Figure 16a, but are from a sensor outside its assumed working condition. They show a possible result of the experiment if the air temperature is higher than the water temperature.

While Equation 3 defines the relationship between $A(X')$, $P(X')$, and $T(X')$ based on a set of data, the data reliability of one single datum, $x'_i \in X'$, is calculated by

$$r_f(X') = c_1 A(X') + c_2 p(x'_i) + c_3 t(x'_i), \tag{4}$$

where $A(X')$ is the systematic bias of the measured data values X', $p(x')$ is the random error of a single datum at hand, x'_i, and $t(x'_i) = |x'_i - x_i|$ is the truthfulness of a single datum, i.e., the distance of one value at hand, x'_i, to the corresponding ground truth value x_i. Moreover, c_1, c_2, and c_3 are relative weights for the three components of data reliability. Naturally, the data reliability mapping range should be between zero and one, i.e., $R, r \colon X' \to [0, 1] \in \mathbb{R}$.

5.2 Principal Concepts of Data Reliability

For Cyber-Physical Systems (CPSs), a sensor manufacturer usually provides information about a sensor's accuracy and precision, $A(X')$ and $P(X')$ respectively. The system using the sensor must calculate or estimate the truthfulness, $T(X')$, as well as the relationship, f, between these parameters.

Although accuracy and precision provide general measures of a sensor's overall quality, they do not offer explicit metadata for each data point. However, it may be that a self-aware system will have to make decisions based on just a few — or even only a single — data points. Under such circumstances, information such as accuracy and precision is not sufficient to make a statement about the data's reliability. If this is the case or the values of $A(X')$, $P(X')$, or $T(X')$ are unknown, the designer of a system must estimate the overall reliability of these data points so that the system can make the right decisions. In other words, custom methods are necessary in order to estimate r and R by r' and R', respectively.

The following subsections show three measures that can give an insight into the reliability of a given dataset: consistency, plausibility, and cross-validity of the data (Cao et al., 1999). The three following measures are the basis for the methods the author has developed to determine the input data's reliability. These measures can be applied to both low-level data (e.g., directly obtained from sensors) and higher-level data (e.g., gained from processes and algorithms within a system).

Another possible measure would be a redundant verification, but this requires redundant hardware (TaheriNejad et al., 2016). Such additional hardware would possibly mean higher costs, higher energy consumption, and less

compactness of a system equipped with it. All these are significant disadvantages, especially for mobile devices such as an autonomous EWS system. Moreover, a vast amount of research results on redundant verification (Osaki and Nakagawa, 1971; Marquet et al., 2002; Jie Han et al., 2005; Yu et al., 2007) already exists. Therefore, redundant verification is not considered here.

Again, a simplified example shall serve as a theoretical experiment to explain and describe the other three measures. Figure 17 shows a water tank from which water can be drained to drive a small turbine. The turbine's rotational speed and the volume of water contained in the tank are measured[5]. Simple assumptions have been made for the experiment so that further explanations are not only understandable but also valid: (i) the exact capacity of the water tank and the maximum possible flow of its valve are known, (ii) the only way for the water leads through the turbine (no other water loss is possible), (iii) there is no other power driving the turbine except the water from the tank, (iv) water pressure and the turbine's rotational speed are related, (v) the mechanical properties (e.g., mass inertia) of the turbine are known, (vi) at the beginning of the experiment, the tank is full of water, and (vii) no water is refilled during the experiment.

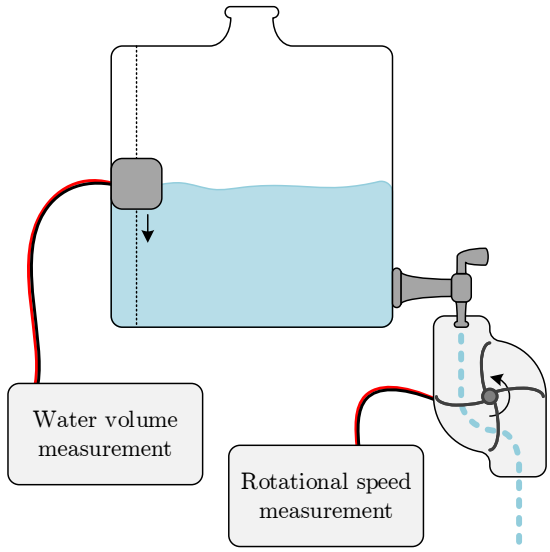


Figure 17. A simplified illustration of a setup for measuring a turbine's rotational speed and the volume of the water

[5]The theoretical experimental setup shown in this example does not claim to be the most sophisticated solution. This example is only intended to provide a better understanding of the data reliability concepts discussed here.

5.2.1 Plausibility

Plausibility is the first measure to evaluate the input data's reliability by determining if it is in its expected domain, eg., whether an input datum's absolute value is within a plausible range (Cao et al., 1999).

In the thought experiment (Figure 17), two different quantities are measured: the amount of water and the rotational speed. A plausible amount of water measured must be between 0 liters and the water tank's maximum capacity. The tank can neither be filled negatively nor with more water than it can hold. Accordingly, a measured value outside this range must be an incorrect measurement. For the second measured quantity, the rotational speed, there is a plausible range from 0 Revolution per Minute (RPM) to the maximum rotational speed limited by the maximum possible water pressure and the turbine mechanics. Because of the example's setup, the turbine can neither rotate in the opposite direction nor is it driven by a second force besides the water. Therefore, a measured value outside this range would be implausible.

5.2.2 Consistency

Consistency is the second measure in data reliability. In a set of data points, a certain consistency can often be observed, especially when representing natural phenomena, i.e., data collected by a sensor in the real world. Such signals are limited in the speed of their changes from one sample to the next. In other words, a signal (i.e., several consecutive data points) has a maximum possible slope. A measured signal with a physically impossibly steep slope indicates an incorrect measurement (Cao et al., 1999). Since more than just one measuring point is needed to determine the signal's slope, historical data are required.

For the measured amount of water stored in the tank (Figure 17), the maximum rate of change is equal to the water tank valve's maximum flow rate. If the measurement shows that more water disappears than the valve can let through in a specific time, the measurement must be wrong.

The rate of change of turbine speed depends on its physical properties. Due to its mass inertia, it cannot start or stop infinitely fast. Only if the rate of change of the measured rotational speed is within these limits, the measurement is reliable.

5.2.3 Cross-Validity

Cross-validity (or co-existence plausibility) is another aspect of data reliability. Often, a correlation between the values of two different datasets can be identified. This can then be used to evaluate whether a dataset matches other

datasets as expected. If two dependent variables do not match, possibly the data (at least one of the two variables) is not reliable (Cao et al., 1999; Liu et al., 2008).

With regard to the thought experiment (Figure 17), a clear correlation between the rotational speed of the turbine and the rate of change of the water volume is evident. When water flows out of the tank, the turbine rotates — slowly with small and fast with large quantities of water passing. Conversely, this means that measurements must not indicate a rotating turbine when the water volume is not changing. On the other hand, a measurement indicating a still-standing turbine while the water volume is decreasing is also impossible. Furthermore, measurements in which the turbine rotates very fast although the water volume decreases only slightly, or vice versa, are also incorrect. However, for the sake of completeness, it must be mentioned that in this example certain delay effects would need to be considered. The turbine itself is inert and will not start or stop at the exact same moment in which the water starts or stops running.

Furthermore, it should be mentioned that in this example it would probably be difficult to establish which of the two measured values is wrong if they do not match. Besides, measurement errors could remain undetected if both measured variables are faulty and compensate each other.

5.3 Causes of Unreliable Data

Data reliability indicates the trustworthiness of a measured quantity and has some similarities with the work on dependability[6] as defined by Avizienis et al. (2001) and Laprie (1992), which is briefly summarized in the following. According to Avizienis et al. (2001), dependability is a system's "ability to deliver service that can justifiably be trusted." A system *failure* describes the fact that a system's delivered service is incorrect; e.g., when the system outputs incorrect results. For example, regarding the EWS system, such incorrect output would be a miscalculated EWS. Such a miscalculation is called a *value failure*, in contrast to a *timing failure*, which is about the timing of the system's service and beyond the scope of this thesis. A system failure is caused by an *error*, which is an incorrect system state. For example, regarding the EWS system, this could be erroneous vital sign data. An error, in turn, is caused by a *fault*; i.e., a broken sensor. However, the sequence that a fault leads to an error and an error to a failure can be a long repeating chain. A broken sensor (a physical fault), for example, could be caused by a physical

[6]It must be noted that the work in this thesis only touches on the field of dependability. Elements, such as fault removal, fault forecasting, etc. (Laprie, 1992; Avizienis et al., 2001), are beyond the scope of this thesis.

failure (e.g., a short circuit occurring in the sensor) which, in turn, could be caused by an error of one of the sensor's components, and so on.

As the definition of data reliability and the three measures elaborated above show, the use of a measurand is unreliable (e.g., because of a sensor defect) if the measured value fails to meet specific criteria. However, from a different perspective, data considered as unreliable could indicate a malfunctioning of the System under Observation (SuO) instead of faulty measuring.

The theoretical experiment from Section 5.2, respectively, the reason for making these assumptions around this experiment already shows what is meant by this statement. For example, in terms of data reliability, measurements of a changing water volume while the turbine stands still implies that either the water level or the rotational speed is not measured correctly. However, such measurements could also be caused by a leak in the water tank or a blocked turbine where the water runs off via some kind of bypass.

Since fault diagnosis to determine the potential cause of the observed error is beyond the scope of this thesis, there remains a contradiction between the implemented data reliability methods and the Condition Monitoring System (CMS). As described in Section 2.6, this application assumes correct measurements in order to detect a defect in the SuO. However, this does not definitively exclude the possibility of the coexistence of a data reliability assessment with the CMS. Yet, this requires a separate study and many further development steps. For example, an exciting research question that was excluded from this research (see Section 5.2) would be whether this is possible without any redundant hardware. Since this thesis is about improving monitoring applications through Computational Self-Awareness (CSA), such questions are beyond its scope.

Because of the contradiction between the CMS and data reliability, the latter was applied in this thesis exclusively in the EWS case study (Section 2.5). Nevertheless, this is a worthwhile topic for future research on monitoring applications.

5.4 Data Reliability in Early Warning Score System

Data reliability provides an understanding of the validity of the data that goes beyond that of the sensors. In the context of EWS calculation, for example, a sensor could simply become detached from the test person's body. Data supplied by such a sensor may be accurate and precise but still invalid in the application's context (as explained in Section 5.1). Therefore, the EWS system should not consider data obtained in such a manner, even if it has been measured accurately and precisely.

Figure 18 shows a simple schematic of the data reliability assessment for one low-level agent. The other agents (Section 3.2) are faded out for the sake of simplicity. Because plausibility and consistency are measures of a single vital sign and can be calculated independently of other vital signs, this assessment takes place in the low-level agent assigned to this vital sign. This means that plausibility and consistency assessments are done in the same place as the abstracting of the vital sign score (Section 4.3). In contrast to plausibility and consistency, the cross-validity check requires already abstracted knowledge of the different vital signs (explained in Sections 5.4.1 and 5.4.2). Therefore, this evaluation takes place on the higher hierarchical level of the EWS system's architecture (Section 3.2), where the EWS agent also abstracts the EWS out of the various vital sign scores (Section 4.3).

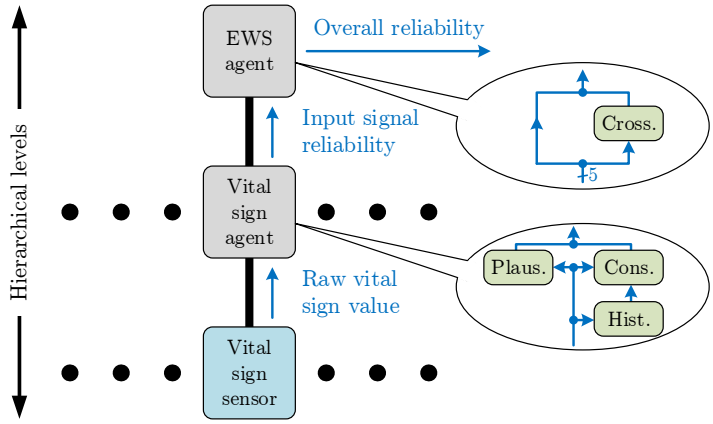


Figure 18. The three reliability measures (plausibility, consistency, and cross-validity) assessing the overall reliability of the calculated EWS

After applying the self-awareness property abstraction to the EWS system, the author implemented three more development steps, all including a data reliability assessment. The first two steps differ in their procedure of assessing the measured vital signs' data reliability: Boolean logic (Section 5.4.1) and fuzzy logic (Section 5.4.2). The third step also includes a confidence-based decision-making process besides the data reliability assessment and will, therefore, be dealt with in Chapter 6.

5.4.1 Binary Data Reliability Assessment

Section 3.2 describes the design of the EWS system's basic architecture. The first development step of the data reliability assessment continues the work described there and is binary, which means that the reliability of an input signal can only have two different states (either reliable or unreliable). This

first development step here is comparable to a proof of concept, and thus, the quantitative statements about results are less important than finding possible methods for the data reliability concepts (Section 5.2) and testing their functionality. Hence, the experiments and the input data are rather limited in contrast to the development stages that follow (Sections 5.4.2 and 6.3).

Since vital signs can react differently from person to person, it is impossible to clearly indicate each vital sign regarding the three measures to be determined (plausibility, consistency, and cross-validity). Results from other scientific fields determine body temperature behavior very precisely compared to other vital signs, therefore, it is the perfect candidate for these initial implementation tests.

Plausibility of the Body Temperature

According to Omics International (2016), Europe's temperature extremes are -58.1°C and 48.0°C. Therefore, these boundaries were used to define the lower and upper limits of temperature. The measured body temperature is considered valid if it lies within this range and invalid if it lies outside this range. Table 6 shows a simple lookup table (as described in Section 4.2) with three different ranges used for this purpose, which means that the vital signs' plausibility classification is similar to an abstraction process.

Table 6. A simple lookup table to classify body temperature in terms of plausibility

Temperature range	**Classification**
$<$ -58.1°C	Invalid measurement
-58.1°C – 48.0°C	Valid measurement
$>$ 48.0°C	Invalid measurement

The selected values in Table 6 may initially appear too extreme, considering that the body temperature, despite open bounds, is defined in Table 3 within a much smaller range. However, the person being monitored may have already died and is lying unprotected in an extremely cold or hot place. Whether EWS calculations make sense in such a case is debatable. However, the fact that the person's body temperature would be either extremely low or high is beyond doubt. Such a measurement result would, therefore, be reliable in terms of the application.

It is important to note that while these chosen limits allow testing this plausibility check's behavior, it may not be safe to use these values for monitoring the health condition of a patient in real life.

Consistency of the Body Temperature

In the studies of Oberhammer et al. (2008), Putzer et al. (2010), and Pasquier et al. (2015), cooling rates of persons wholly buried under an avalanche are reported. These rates range from 6°C to 9.4°C per hour, depending on the study considered. Assuming that a person's body temperature cannot increase faster than it can decrease, the maximum possible rate of change is 9.4°C/h. This parameter's time base must be the same as that of the data obtained by the EWS system. Since the measured data's sampling frequency is 1Hz, the maximum rate of change had to be converted to a time base of one second. Table 7 shows a simple lookup table (as described in Section 4.2) with values to classify the measured body temperature's consistency. This means that, as with plausibility, the vital signs' consistency classification is similar to an abstraction process.

Table 7. A simple lookup table to classify the body temperature in terms of consistency

Temperature's rate of change	**Classification**
$<$ -0.003°C/s	Invalid measurement
-0.003°C/s – 0.003°C/s	Valid measurement
$>$ 0.003°C/s	Invalid measurement

The measurement of body temperature is valid if the rate of change is within the defined boundaries. If the perceived signal's slope is outside this range, the measured body temperature is not trustworthy and will, therefore, not be considered.

Cross-validity and Body Temperature

The studies of Fauci et al. (2008) and Brown et al. (2012) show that humans' heart rate, blood pressure, and respiratory rate change with their body temperature, i.e., if it is too high or too low. Mild hypothermia can lead to tachycardia and tachypnea, while moderate hypothermia can already show signs of hypotonia and bradycardia. As the body temperature continues to drop, the vital signs become weaker and weaker until they finally cease entirely (McCullough and Arora, 2004). While hyperthermia does not trigger completely identical changes in the other vital signs, it does show similar behavior, namely, a general deterioration (Fauci et al., 2008). Therefore, a measurement showing a critical body temperature cannot be correct if, at the same time, the other vital signs are in good condition. If the body tempera-

ture score deviates from the other vital sign scores, the EWS system classifies the body temperature measurement as unreliable[7].

It is important to note that while this configuration allows testing of the cross-validity check's behavior, it may not be safe to use it for monitoring the health condition of a patient in real life.

Experimental Setup

The vital sign datasets used for the experiments are the same as those recorded by Azimi et al. (2016). In their experiments, they also used these vital sign datasets, obtained from a 35-year-old healthy male subject. The datasets include all necessary vital signs (Table 3)[8], sampled once a second[9], namely, heart rate, systolic blood pressure, respiratory rate, body temperature, and oxygen saturation. However, to test the implemented data reliability checks, the measured body temperature was replaced with incorrect temperature data. This approach then led to various experiments with a body temperature that was too high or too low, or increasing too fast, or not correlating with the other vital signs. As mentioned above, these initial experiments were only a proof of concept and do not replace a more extensive study of an autonomous EWS system.

Results of the Binary Data Reliability Assessment

The results of this proof-of-concept work showed that such a solution could be further pursued. The EWS system always detected the erroneous temperature data. Despite the limited complexity of these experiments, the results nevertheless confirmed that the basic concept of the three measures for data reliability calculation is reasonable and useful. Additionally, the EWS system's hierarchical architecture proved to be very beneficial as it allows well-structured processing of the individual steps necessary for the EWS calculation. In other words, the EWS system can house the different modules exactly on the appropriate levels of abstraction where they optimally fit.

All experiments were conducted with all three data reliability modules running simultaneously and with only single modules (the others were disabled). The cross-validity check – at least for the tested datasets – revealed all incorrect temperature data. The other two only worked on the data they

[7]Since Table 3 shows only one possible score (score 2) for hypothermia, the author supplemented it with the different stages of accidental hypothermia from the work of Brown et al. (2012).

[8]The level of consciousness (Table 3) is excluded because it is not applicable in out-of-hospital monitoring.

[9]The blood pressure was not measured each second, but approximately every two minutes.

were designed for (too extreme absolute value or too steep signal slope). Consequently, cross-validity was the most universally working module. This fact can be explained easily because, in contrast to body temperature, all other vital signs were in very good condition. So if the body temperature is faulty, it is easily noticeable since its condition does not match the other vital signs, regardless of whether the body temperature is highly abnormal or has changed too fast. More complex data, in which the vital signs do not match each other so perfectly (all in the same good condition), do not achieve this effect as clearly.

Somewhat apart from this topic, it is noticeable that abstraction is not only an enabler for the actual (not self-aware) EWS system (Section 4.3) but also for parts of the data reliability assessments. This way, plausibility and consistency are implemented with the help of lookup tables, which are part of the abstraction methods (Section 4.2). Thus, strictly speaking, these parts of the data reliability assessment could also be called abstraction as the reliability of the input data is abstracted from the input data.

Besides all these findings, a significant disadvantage of a binary data reliability assessment became visible. Such natural measurement data often have no sharp limits. It can be defined that the oxygen saturation of a person's blood must be between 0% and 100%, but it is not exactly defined how high the person's maximum blood pressure can be. However, if the ranges for the various data reliability measures are too large, they could theoretically obscure faulty measurement data. For example, this might occur in cases when measurements are rather unrealistic yet still within the rather wide range set for a vital sign (e.g., body temperatures just within Europe's extreme temperatures). The next section deals with a solution to this problem.

5.4.2 Fuzzified Data Reliability Assessment

As described above, binary decision-making in combination with natural phenomena has a blatant disadvantage. An input data's reliability is a much more continuous value and should, therefore, not be treated in a binary way. Although binary decision making simplifies the analysis of data reliability, it can cause a loss of information.

A solution to this problem lies in the fuzzification of the data reliability assessment (Bowles and Pelaez, 1995). The use of fuzzy logic instead of binary logic covers these fuzzy areas in which vital signs could actually be reliable or unreliable (Maimon et al., 2001). This means that a fuzzified reliability validation can also work in the absence of complete knowledge about the various vital signs and their interactions. A fuzzy evaluation does not clearly define

whether a datum is reliable or not but indicates the degree of its reliability within a range of $[0, 1]$ (Maimon et al., 2001).

Fuzzified Plausibility and Consistency

Similar to the binary reliability assessment, there are the measures plausibility and consistency. However, they no longer tell precisely whether a measured vital sign is reliable or not; rather, they show the vital sign's degree of reliability (a value between 0 and 1).

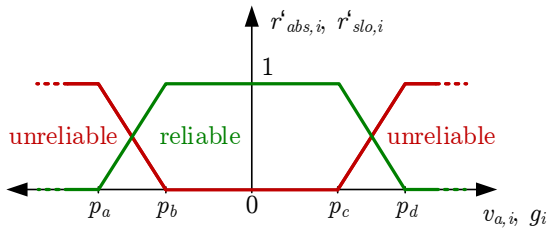


Figure 19. A possible fuzzy membership function to assess a vital sign's plausibility and consistency (Götzinger et al., 2019a)

The estimates of both plausibility, $r'_{abs,i}$, and consistency, $r'_{slo,i}$, for one vital sign, i, are calculated by a fuzzy membership function appropriately configured (as shown in Figure 19). Thus, the plausibility for the vital sign (the reliability of its absolute value) is calculated by

$$r'_{abs,i} = \begin{cases} \frac{v_{a,i}-p_a}{p_b-p_a} & \text{if } p_a < v_{a,i} < p_b \\ 1 & \text{if } p_b \leq v_{a,i} \leq p_c \\ \frac{p_d-v_{a,i}}{p_d-p_c} & \text{if } p_c < v_{a,i} < p_d \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $v_{a,i}$ is the actual value of the vital sign i. Points p_a, p_b, p_c, and p_d (i.e., the intervals between them) are configured to match the assigned vital sign's characteristics. Hence, if a vital sign's value lies in the interval $[p_b, p_c]$, it is seen as absolutely plausible. If the value is lying in one of the intervals, (p_a, p_b) or (p_c, p_d), it is — depending on the absolute value — more or less plausible. Absolute values lying outside of these intervals are considered as absolutely unreliable.

Analogously, a vital sign's consistency (the reliability of its slope) is calculated by

$$r'_{slo,i} = \begin{cases} \frac{g_i-p_a}{p_b-p_a} & \text{if } p_a < g_i < p_b \\ 1 & \text{if } p_b \leq g_i \leq p_c \\ \frac{p_d-g_i}{p_d-p_c} & \text{if } p_c < g_i < p_d \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where g_i is the gradient of the signal of the vital sign i, and calculated by

$$g_i = \frac{v_{p,i} - v_{a,i}}{t}, \tag{7}$$

where $v_{a,i}$ is again the vital sign's actual value, $v_{p,i}$ is the previous value of the same vital sign, and t is the time passed between these two samples.

The counterparts of Equation 5 and 6 are the fuzzy membership functions of the estimated implausibility, $u'_{abs,i}$, and inconstancy, $u'_{slo,i}$, respectively. These serve to indicate how much a signal is implausible or inconsistent and are calculated by

$$u'_{abs,i} = 1 - r'_{abs,i} \tag{8}$$

and

$$u'_{slo,i} = 1 - r'_{slo,i}, \tag{9}$$

which leads to an overlapping of these two functions (Figure 19) with their counterparts, plausibility and consistency. In contrast to the normal lookup table used in the binary data reliability assessment (Section 5.4.1), Equations 8 and 9 show an overlapping lookup table (Section 4.2), which can lead to more than one abstracted symbol. In this particular case, a vital sign can be to some extent plausible and implausible at the same time, instead of just being either plausible or implausible.

Fuzzified Cross-Validity

The studies of Davies and Maconochie (2009), Zila and Calkovska (2011) as well as Reule and Drawz (2012) show that like the body temperature also the other vital signs show correlations with each other. The Cross-validity reliability is calculated (based on these correlations) for each possible pair of vital signs by

$$r'_{cro,i,j} = \begin{cases} 1 & \text{if } s_i = s_j \\ \frac{1}{p_{cro,i,j}|s_i - s_j|} & \text{if } s_i \neq s_j, \end{cases} \tag{10}$$

where $p_{cro,i,j} \in (0, \infty)$ denotes a coefficient of the strength of the correlation between two different vital signs of which s_i and s_j are their abstracted scores. Because cross-validity reliability, $r'_{cro,i,j}$, must be in the interval of $[0, 1]$, it has to be limited to these maximum values; even if $p_{cro,i,j}$ is less than 1, which could lead to a higher $r'_{cro,i,j}$ than 1.

Reliability of the Early Warning Score

Plausibility (Equation 5) and consistency (Equation 6) are exclusively dealing with the data reliability of one single input signal (vital sign) i. They do this

without considering any interaction with other vital signs. Thus, a combination of the two can be called input data reliability, $r'_{in,i}$, of signal i, and is given by

$$r'_{in,i} = r'_{abs,i} \wedge r'_{slo,i}, \tag{11}$$

where the conjunction means that an input signal is reliable if both its absolute value as well as its slope are reliable. In this context, it must be considered that conjunction in fuzzy logic is equivalent to a minimum function (Ross, 2009), which then would be $r'_{in,i} = \min(r'_{abs,i}, r'_{slo,i})$. However, the crucial statement is that a vital sign must be plausible and consistent, and, therefore, the conjunction (the logical *and)* operator is used here.

The input data reliability of all vital signs together is a conjunction of all those and calculated by

$$r'_{in} = \bigwedge_{i=1}^{5} r'_{in,i}, \tag{12}$$

according to the principle that the entire input data is reliable if each vital sign's data is reliable. Similarly, the cross-validity reliability of all vital signs is a conjunction of all cross-validities (Equation 10) and calculated by

$$r'_{cro} = \bigwedge_{i=1}^{5} (\bigwedge_{j=1}^{5} r'_{cro,i,j}), \tag{13}$$

where $r'_{cro,i,j} = 1$ for $i = j$ (Equation 10). These equations finally lead to the reliability, r', of the EWS (the overall reliability), which is calculated by

$$r' = r'_{in} \wedge r'_{cro}. \tag{14}$$

Experimental Setup

In contrast to the experimental data used in the binary data reliability assessment development step (proof-of-concept experiments in Section 5.4.1), real sensor errors were emulated for the experiments here. The goal was to determine whether the EWS system can correctly classify the EWS's reliability; i.e., whether the EWS outputs a reliability value that correlates with the truthfulness of the input data. For this purpose, several 15-minute vital signs datasets (containing heart rate, systolic blood pressure, respiratory rate, body temperature, and oxygen saturation)[10] were recorded with a sampling frequency of 1Hz[11]. The vital signs were recorded from a 36-year-old man with diastolic

[10]The level of consciousness (Table 3) is excluded because it is not applicable in out-of-hospital monitoring.

[11]The blood pressure was not measured every second, but approximately every two minutes.

hypertension, and different errors were introduced at some points during the various experiments, namely:

1. The test subject contracted his biceps during blood pressure measurement.

2. The sensor measuring heart and respiratory rate (a chest strap) was loosened so that it had only partial contact with the body.

3. The body temperature sensor was detached from the body and attached to an object that was either cold, at room temperature, or hot.

The author configured the various fuzzy membership functions and the different correlation coefficients based on the works of Song and Lehrer (2003), Fauci et al. (2008) Davies and Maconochie (2009), Zila and Calkovska (2011), Brown et al. (2012), Reule and Drawz (2012), and Pasquier et al. (2015). In addition, expert opinions of various physicians, information on the accuracy of the sensors used, and the test subject's medical condition in the system's configuration were also taken into account.

Paper VIII (Götzinger et al., 2017a) lists the exact sensor types used for these experiments and provides more detailed information on the experiments' parameter setup.

Results of the Fuzzyfied Data Reliability Assessment

The experiments conducted here show that fuzzified data reliability assessment works. The reliability of the EWS matches the condition of the measurement environment — faulty data lead to reduced data reliability. Figures 20 and 21 show two representative results from the experiments performed.

In the first experiment (Figure 20), the body temperature sensor was detached from the test subject after about 350s. The body temperature sensor started to decrease from this time on because the room temperature was measured instead of the body temperature. After the signal had stabilized around room temperature, the sensor was reattached to the body after about 680s. As a result, the measured temperature rose again. At the beginning and the end of the faulty measurement phase, data reliability was reduced due to the consistency evaluation. In the intervening period, the cross-validity evaluation caused the low data reliability value because the low body temperature did not correlate with the other vital signs. In addition to these intentionally induced errors, there is also a small decrease in reliability at 710s. This can be explained by the fact that the body temperature sensor shifted slightly during loosening and refastening of the chest strap. Due to the sensor's decreased

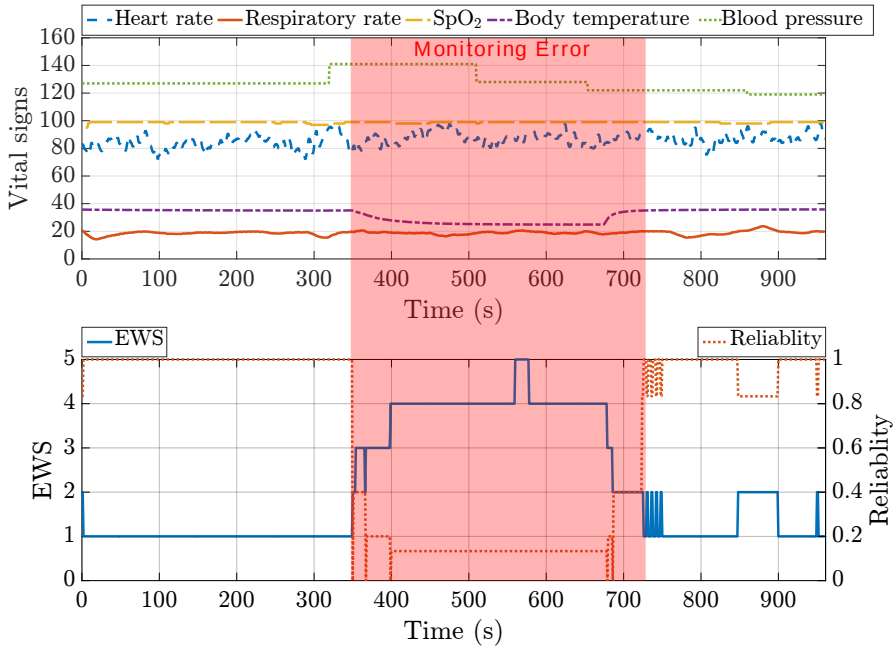


Figure 20. The monitored vital signs, the resulting EWS and its reliability when the temperature signal is temporarily incorrect due to a detached sensor

contact with the test subject, the measured temperature failed to measure the correct temperature (it was slightly too low).

Figure 21 shows an experiment in which the chest strap (housing a heart- and respiratory sensor) was loosened from the test subject after 330s. Since from then on the chest strap was no longer appropriately fastened, the sensor had only partial contact with the subject. After about 700s, the chest belt was properly refastened. Between these points in time, an unstable measurement with unreliable readings from this sensor can be observed. The measured heart rate signal sporadically shows errors (drops to 0 beats per minute). As can be seen, the EWS system detected these wrong measurements and indicated this faulty sensor setup by showing a very low reliability value (drop to 0) in instances of erroneous sensor readings.

These two experiment results show that reliability assessment works well. However, the EWS is still calculated according to standard rules independent of reliability values and would, therefore, lead to false alarms because of a wrong (too high) EWS in instances of corrupted sensory data (Table 4). Correcting the EWS based solely on reliability would be unsafe because of the uncertainty of reasonably defining a minimum reliability value (e.g., 0.5 or 1) at which the calculated EWS could be trusted. Such a decision cannot be made easily because, despite the fuzzification of the data reliability assessment, no

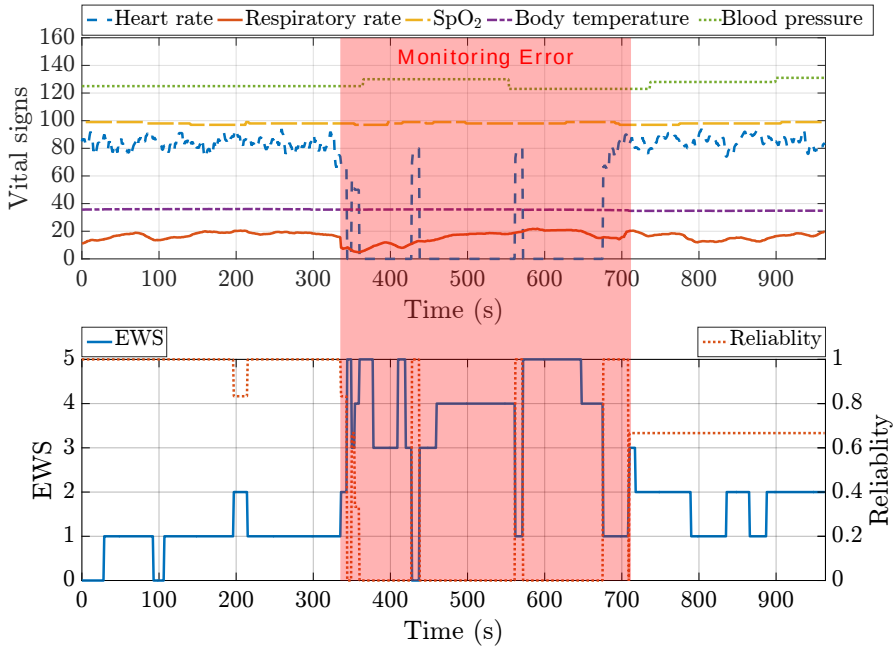


Figure 21. The monitored vital signs, the resulting EWS and its reliability in case of frequent breakdowns of the heart rate signal due to a loosened chest strap sensor

complete knowledge about the vital signs and their interaction are available. EWS corrective decisions have to be based on a number of considerations. Chapter 6 deals with a possible approach and shows how confidence-based decisions can be made.

Moreover, the implementation of the reliability assessment showed that the property abstraction (Section 4) also plays a significant role in assessing the (fuzzyfied) data reliability (overlapping lookup table).

6 Confidence

Confidence is the self-awareness property that was part of the systems' final development stage in both case studies. Like data reliability (Chapter 5), confidence is metadata and has significant similarities with the former. The difference is that data reliability describes the trustworthiness of the available data, while confidence describes the trustworthiness of an algorithm, analysis, function, or (sub-)system. Because of these similarities, the methods of confidence assessment presented in this chapter are inspired by those of data reliability presented in Section 5.4.2.

According to TaheriNejad and Jantsch (2019), the significance of confidence lies in its ability to enable a self-aware system to make decisions based on the reliability of its sub-processes and sub-algorithms. Kholerdi et al. (2018) show a possible application of this concept, a system that switches between different algorithms based on their confidences.

This chapter starts starts with the formal definition of confidence in Section 6.1, followed by Section 6.2 showing the intertwining of confidence and data reliability. Sections 6.3 and 6.4 show how confidence was applied in the systems of the two case studies. There, experiment results show that a confidence-based decision-making process improves their robustness and reliability. The formal definition of confidence and its usage in the two case studies is based on Papers II and III (Götzinger et al., 2019a,b)[1]

6.1 Formal Definition of Confidence

According to TaheriNejad et al. (2016), confidence is "the extent to which a procedure may yield the same results on repeated trials." It describes the trustworthiness of an algorithm, analysis, function, or (sub-)system, while the input data is assumed to be entirely trustworthy. This means, confidence provides a measure of how much the results of a system's data manipulation can be trusted or, in other words, how close the obtained output is to an ideal output (reflecting the ground truth). However, knowing the ground

[1]This dissertation is based on eight original publications, which are the results of collaborations. These publications can be found in the appendix. The author's contributions to each publication are listed in Chapter 8.

truth (gaining such complete knowledge) is mostly impossible because many properties and correlations (e.g., physical dependencies) of the environment are usually unknown. The property confidence is a measure that can deal with uncertainties arising from a lack of precise knowledge about the variables or processes of a system or the environment (Apostolakis, 1990). This is in line with the work on uncertainty as defined by Taylor et al. (1994). For example, in the case of the Early Warning Score (EWS) system, uncertainties could be caused by a lack of knowledge about physical differences in different people. Confidence is often described in the relevant literature as a probability measure (Mozelli et al., 2021). However, according to Baudrit et al. (2008), the probabilistic approach to uncertainties tends to make assumptions that are not necessarily justified in the case of incomplete knowledge. Therefore, in the course of this work, an approach was pursued in which confidence can be understood as distance. Namely, as the distance[2] between an ideal function $I(X)$ and an unideal function $G(X)$ at hand, which are both defined over the dataset $X = \langle x_1, \ldots, x_n \rangle$. The function $\Delta\Big(\iota(x_i), g(x_i)\Big)$ represents the distance between $\iota(x_i)$ and $g(x_i)$, the ideal- and unideal functions respectively, for one member of X. Consequently, the confidence $c\big(g(x_i)\big)$, for the function $g(x_i)$, is calculated by

$$c\big(g(x_i)\big) = 1 - \Delta\Big(\iota(x_i), g(x_i)\Big), \tag{15}$$

where Δ is based on an application-specific metric for distance, which is normalized such that $0 \leq \Delta \leq 1$.

The overall confidence, $C(G(X))$, is the average confidence of all $g(x_i)$ over the whole set X, calculated by

$$C(G(X)) = \frac{1}{n} \sum_{i=1}^{n} c(g(x_i)), \tag{16}$$

where it must be noted that $0 \leq c(g(x_i)), C(G(X)) \leq 1$ and $c(\iota(x_i)) = C(I(X)) = 1$ applies. Furthermore, the methods for calculating c and C are case-specific, i.e., they depend on the application. However, the ground truth of ι and I is often unknown, which leads to the situation that the distance function Δ cannot be calculated. Thus, the function Δ' must be used to estimate Δ. In the further course of this work, the heuristics c' and C' are confidence estimates based on Δ'. The implemented methods for calculating the respective confidences with respect to the two case studies' systems are presented in Sections 6.3 and 6.4.

[2]Distance denotes a difference that can be of any dimension: performance or success rate difference, geometric distance, time difference, difference in absolute values, and so forth.

6.2 Relation between Confidence and Data Reliability

As set out, confidence describes the trustworthiness of an unideal function used for data manipulation. Such a function is unideal because having complete knowledge about all properties and correlations (e.g., physical dependencies) of the environment is rather unlikely. Besides this lack of knowledge, the input data are not likely to be entirely trustworthy either.

Up to this point, data reliability (Chapter 5) and confidence have been considered separately and discussed independently. However, in the real world, these two terms are often closely intertwined with each other. Assuming an ideal process that consumes unideal (not entirely correct) data, the process's output data cannot be entirely reliable. Hence, reduced output reliability is exclusively caused by input data with reduced reliability. In other words, the degree of output data reliability is exclusively determined by the degree of input data reliability. On the other hand, a process's output data reliability would be solely affected by this process's confidence if it is unideal but fed with absolutely correct input data. However, most likely, both input data and processes are not ideal. Therefore, the output data reliability, $r_g(x'_i)$, depends on both and is calculated by the function ϕ as follows[3]

$$r_g(x'_i) = \phi\Big(r_f(x'_i), c\big(g(x_i)\big)\Big), \tag{17}$$

where $c\big(g(x_i)\big)$ (Equation 15) is the confidence for the unideal function $g(x_i)$ processing one single datum, x_i, and $r_f(x'_i)$ (Equation 4) is the data reliability of one single datum, x'_i. Here, x'_i belongs to the set $X' = \langle x'_0, ..., x'_n \rangle$, which is the dataset at hand (i.e., the unideal values), corresponding to the ground truth values $X = \langle x_0, ..., x_n \rangle$. Since the relations $c\big(g(x_i)\big) \leq c\big(\iota(x_i)\big)$ and $r_f(x'_i) \leq r_f(x_i)$ hold for all $x_i \in X$, we can conclude that

$$r_g(x'_i) \leq r_\iota(x'_i) \leq r_\iota(x_i), \tag{18}$$

where $r_\iota(x'_i)$ is the output reliability if the data manipulation process is ideal but its input data is not. In contrast, $r_\iota(x_i)$ is the output reliability if both are ideal.

Calculating the output data reliability of a system's process is much more difficult when data reliability or confidence are obtained by estimation functions since the ground truth is unknown. Besides, a system usually consists of many different processes which consume and produce data. So it should be noted that the output data reliability of a process could be, in turn, the input data reliability of another process.

[3]Function ϕ depends on the application but must meet monotonicity criteria.

6.3 Confidence in Early Warning Score System

Confidence is intended to help making well-considered decisions. This is especially necessary when a choice is uncertain. Such uncertainty may be caused by incomplete knowledge of the environment or by a slightly distorted image (Taylor et al., 1994; Baraldi et al., 2014); e.g., slight measurement errors that still result in reliable values. In the EWS system, the author detected two of these uncertainties that can benefit from a confidence-based decision. Figure 22 shows the locations of the confidence-based decision-making processes, which are explained in Sections 6.3.1 and 6.3.2. For the sake of simplicity, this figure shows only one low-level agent, the other agents (Section 3.2) were faded out.

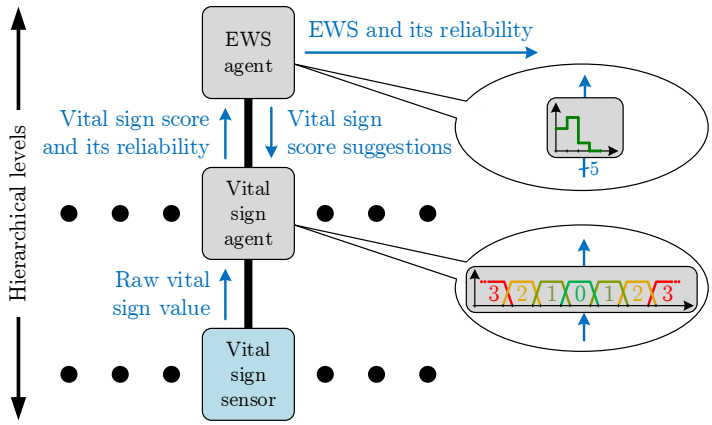


Figure 22. The confidence-based vital sign score abstraction and the cross-validity confidence assessment

6.3.1 Confidence-Based Vital Sign Score Abstraction

The first uncertainty relates to the abstraction of a vital sign score and has two different origins. On the one hand, each person's body has different characteristics. The boundaries between the different scores of a vital sign could vary from person to person. In this case, a value close to a boundary (defined in Table 3) would lead to a wrong score should the certain boundary of the monitored patient be drawn differently. However, a simple lookup table (Section 4.2) has sharp boundaries, leading to the same vital sign score abstracted for each patient. The other reason for this uncertainty could be caused by a measurement that is not entirely correct. Since the ground truth is usually unknown, a measured value that deviates only very slightly from the truth can still be evaluated as reliable (Section 5.2). Thus, if a vital sign value is

close to the boundary between two scores, a slightly deviating measurement can lead to the wrong vital sign score while still considered reliable.

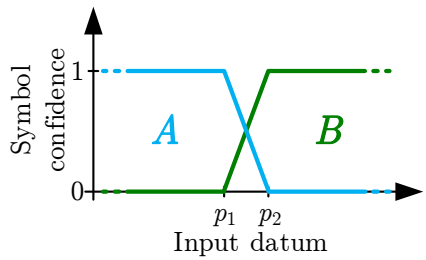


Figure 23. A confidence-based abstraction of an input datum into symbols *A* and *B*

To counteract these two causes of abstraction uncertainties, the lookup table was replaced by an overlapping lookup table (Section 4.2). In such an overlapping lookup table, an input datum can be abstracted into more than one symbol (score). For each symbol resulting from the abstraction process, one confidence value exists. For example, confidence $c'_{s,i}$ indicates how likely it is that a sample value of signal i should be abstracted to symbol (score) s. In other words, $c'_{s,i}$ represents the trustworthiness of the abstraction method to deliver the right output when abstracting the actual sample value of i to score s.

Figure 23 shall serve as a simplified example of such an overlapping lookup table (a confidence-based abstraction). While a conventional EWS system abstracts a vital sign into one of four different symbols (vital sign scores: 0, 1, 2, and 3), in this simplified example, only two different symbols exist, A and B. As can be seen, there are no clear boundaries, only a fuzzy transition between A and B. An input value in the range $(-\infty, p1]$ is abstracted to symbol A, and if it is the range $[p2, \infty)$, it is abstracted to symbol B. However, if it is in the range $(p1, p2)$, it is abstracted into both symbols simultaneously. Each symbol available to the abstraction process is assigned a confidence value, which indicates how much this abstraction can be trusted. For symbol A, the confidence, c'_A, is calculated by

$$c'_A = \begin{cases} 1 & \text{if } v_a \leq p_1 \\ \frac{p_2 - v_a}{p_2 - p_1} & \text{if } p_1 < v_a < p_2 \\ 0 & \text{if } v_a \geq p_2, \end{cases} \tag{19}$$

where v_a is the actual value which is abstracted into a symbol. On the other hand, the confidence, c_B, whether the abstraction to symbol B can be trusted,

is calculated by

$$c'_B = \begin{cases} 0 & \text{if } v_a \leq p_1 \\ \frac{v_a - p_1}{p_2 - p_1} & \text{if } p_1 < v_a < p_2 \\ 1 & \text{if } v_a \geq p_2, \end{cases} \tag{20}$$

whereby this function is the exact counterpart to Equation 19.

As information about the raw vital signs is needed for this procedure, it is located in the low-level agent (Figure 22). The low-level agent sends the vital sign score with the highest score reliability to the high-level agent. As explained in Section 6.2, this reliability (output reliability) combines a process's input data reliability and confidence. However, information sent by the high-level agent shall also be considered in this application. Thus, the calculation of reliability of score s, for vital sign i, consists of three terms, and is calculated by

$$r'_{out,s,i} = r'_{in,i} \wedge c'_{s,i} \wedge r'_{sug,s,i}, \tag{21}$$

where $r'_{in,i}$ is given by Equation 11, $c'_{s,i}$ is the confidence of the vital sign i being correctly abstracted to score s, and $r'_{sug,s,i}$ constitutes the opinion of the high-level agent (Figure 22) how much this score, s, can be trusted from a top view. How $r'_{sug,s,i}$ is determined also depends on the cross-validity confidence and is explained in the next section.

6.3.2 Cross-Validity Confidence

Besides the uncertainty of the boundaries between the individual vital sign scores, the correlations between the vital signs are also different from person to person. As in Equation 10, $c'_{cro,i,j}$ indicates how well the scores, s_i and s_j, of two different vital signs, i and j, fit together. However, the difference is that this is not a generally formulated reliability assessment but person-specific knowledge. In other words, it reflects more or less a partial aspect of a process taking place in a specific person. Figure 24 shows a possible example of a $c'_{cro,i,j}$ function that gives information on how well the scores of i and j fit together. In other words, $c'_{cro,i,j}$ represents the trustworthiness that the difference between the abstracted scores of i and j is valid. In this example, the patient has most likely a difference of 1 between the scores s_i and s_j, followed by a difference of 0.

The overall reliability of the calculated EWS is still calculated using Equation 14, whereas the calculation of r'_{cro}, in contrast to Equation 13, is now calculated by

$$r'_{cro} = \bigwedge_{i=1}^{5} \left(\bigwedge_{j=1}^{5} (r'_{cro,i,j} \vee c'_{cro,i,j}) \right), \tag{22}$$

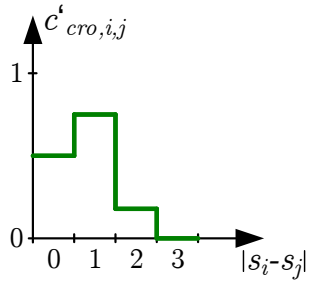


Figure 24. An example for a cross-validity confidence function, which uses available knowledge to indicate how confident the cooccurrence of the scores of two vital signs is

where the disjunction means that the values of two different vital signs (i and j) fit together if their simultaneous occurrence is appropriate ($r'_{cro,i,j}$) or known ($c'_{cro,i,j}$, knowledge about the specific person). In this context, it must be considered that disjunction in fuzzy logic is equivalent to a maximum function (Ross, 2009).

Because this process requires already higher abstracted knowledge, it is located in the high-level agent (Figure 22). However, this agent does not only output the EWS and its reliability but also sends a score suggestion needed for Equation 21 to the low-level agent. This suggestion, $r'_{sug,s,i}$, is calculated by

$$r'_{sug,s,i} = \bigwedge_{j=1}^{5} (r'_{cro,i,j} \vee c'_{cro,i,j}), \tag{23}$$

and indicates how much the score s of a vital sign fits the scores of all other vital signs (based on $r'_{cro,i,j}$ and $c'_{cro,i,j}$). Reliability $r'_{sug,s,i}$ is calculated for each possible score $s \in [0, 1, 2, 3]$ to which signal i could be abstracted.

6.3.3 Experimental Setup

This section gives an overview of the experiments conducted. The author refers to Paper II (Götzinger et al., 2019a) for more details about the sensors used and detailed descriptions of the different scenarios in which the various test subjects were monitored.

The vital signs data were collected from eight different participants, four women and four men. The test subjects were 23 to 37 years old, with an average age of 27.25 years. As in the experiments described in Section 5.4, the

same five vital signs[4] were collected once a second[5], namely, heart rate, systolic blood pressure, respiratory rate, body temperature, and oxygen saturation.

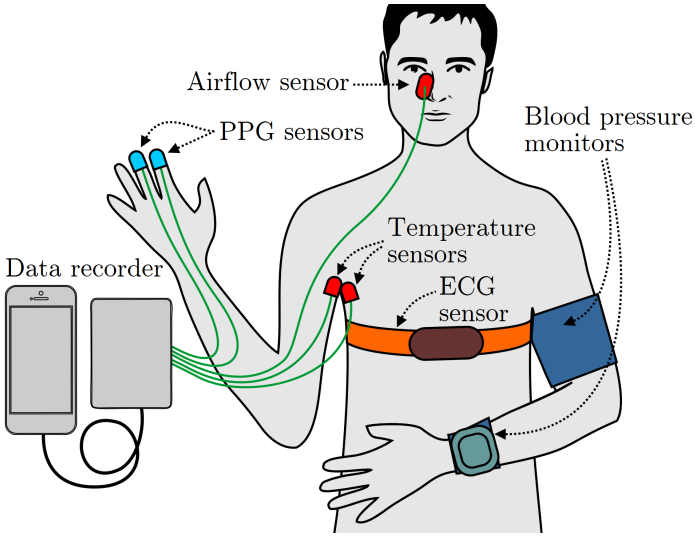


Figure 25. The measurement setup includes multiple data sources for each required vital sign (Götzinger et al., 2019a)

Figure 25 shows that each participant was monitored with a variety of different sensors. To be precise, each of these five parameters was abstracted from different vital sign signals: (i) three different signals for heart rate, (ii) three for respiratory rate, (iii) two for oxygen saturation, (iv) two for body temperature, and (v) two for blood pressure.

The procedure of having more than just one signal for each vital sign provides datasets of different quality for the same measurement. In other words, one set of signals represents high-quality sources for each vital sign, and the other signals constitute a low-quality source. For reasons described in Section 5.1, it was not possible to obtain the ground truth of the vital sign signals. However, for each vital sign, the signal with the highest quality will be called in the following ground truth signal since this is the closest possible approximation[6]. This means that the datasets contain a ground truth signal for each measured vital sign and one or two additional signals, some of which contain errors. Resulting from all these different signals, are a total of 72

[4]The level of consciousness (Table 3) is excluded because it is not applicable in out-of-hospital monitoring.

[5]The test subjects' blood pressure was not measured each second, but approximately every two minutes.

[6]To have the ground truth signals as error-free as possible, they were additionally filtered to remove any possible noise.

different signal setups, of which one single setup contains five ground truth signals. All other setups contain at least one signal that might contain errors.

Four different test scenarios were used. In all of these scenarios, the test subjects were sitting quietly (without performing any physical activities). In one of these scenarios, no additional error was introduced. In the other three scenarios: (i) the temperature sensor was detached, (ii) the biceps were tensed during blood pressure measurements, and (iii) the temperature sensor was detached and the biceps tensed. The additional (intentional) errors were only applied to signals that do not serve as ground truth since the latter is necessary as reference for comparison purposes.

Six of the eight participants were monitored twice; the other two persons were monitored four times. Thus, 20 different measurements with 72 signal setups each were performed. Unfortunately, in two of the measurements, the ground truth setup was erroneous and, therefore, these measurements had to be excluded from further experiments. Hence, 18 measurements with 72 signal setups each, which means 1296 different record datasets, were available. 18 of them are ground truth datasets (one for each measurement), and the remaining 1278 may contain one or more faulty signals.

6.3.4 EWS Systems Validation

The experiments aimed to prove that Computational Self-Awareness (CSA) enables more robust and reliable monitoring, in this case applied in an EWS system. Therefore, a conventional (non-self-aware) EWS system (described in Section 4.3) was compared with a Self-Aware Early Warning Score (SA-EWS) system (equipped with abstraction, data reliability, and confidence). To compare the EWS and SA-EWS systems' performances with each other, their outputs (EWS signals) were compared with a ground truth EWS signal. This ground truth EWS signal was generated by the conventional EWS system processing the ground truth vital sign signals. The conventional system was used here because, in contrast to the SA-EWS system, it does not manipulate the EWS calculation with self-awareness methods. Since no errors are expected in the ground truth vital sign signals, the conventional EWS system should produce an absolutely correct output signal (the ground truth EWS signal). It should be mentioned that this procedure can lead to an advantage of the conventional EWS system over the SA-EWS system if the ground truth vital signals still contain small errors, which could lead to a slightly distorted ground truth EWS.

Three different metrics are used to compare the outputs of the two EWS systems with the ground truth. The first one is the calculation of the Root-Mean-Square Deviation (RMSD), which indicates how close two different sig-

nals are to each other. However, the RMSD is not the best way to compare the two EWS systems. A signal that deviates slightly over a long period of time (e.g., a deviation of only one value) may have a worse RMSD than a signal with a much larger deviation but only for a short time. Most likely, however, the former will not lead to negative consequences, while the latter will lead to false or missing alarms due to its temporarily high deviation (Table 4).

The second metric is based on the absolute error, ε, between the EWS calculated by the EWS system or the SA-EWS system and its ground truth. For sample i it is calculated by

$$\varepsilon_i = \left| E'_i - E_i \right|, \tag{24}$$

between the output EWS sample, $E'_i \in [0, 15] \subset \mathbb{Z}$, and its ground truth, $E_i \in [0, 15] \subset \mathbb{Z}$. Because of the range within which the output of an EWS system and the ground truth can be, ε_i can only be between 0 (no deviation) and 15 (largest deviation). The maximum absolute error, ε_{max}, of one measurement is calculated by

$$\varepsilon_{max} = max(\varepsilon_i \colon i \in [1, n] \subset \mathbb{Z}), \tag{25}$$

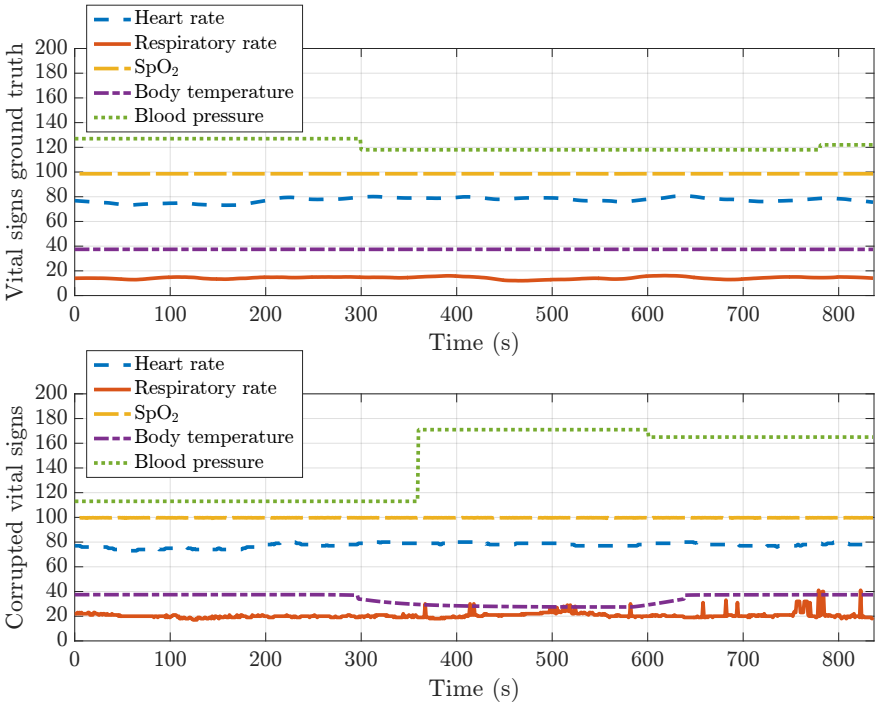
where ε_i is the absolute error of the i^{th} of n samples of one measurement.

The third metric is about the number of false and missing alarms (based on the ε). As shown in Table 4, the calculated EWS of a patient can be divided into three different risk classes: low risk, medium risk, and high risk. If ε of an EWS signal's sample (output of an EWS system) is so high that it does not belong to the same risk class as its ground truth, a false or missing alarm is the result. If several consecutive samples meet this criterion, the false/missed alarm is counted only once as this is a continuous state of false classification. In principle, this corresponds to the way it is handled by medical personnel. They raise one alarm at the beginning of a medical emergency and not every second, i.e., a large number of alarms.

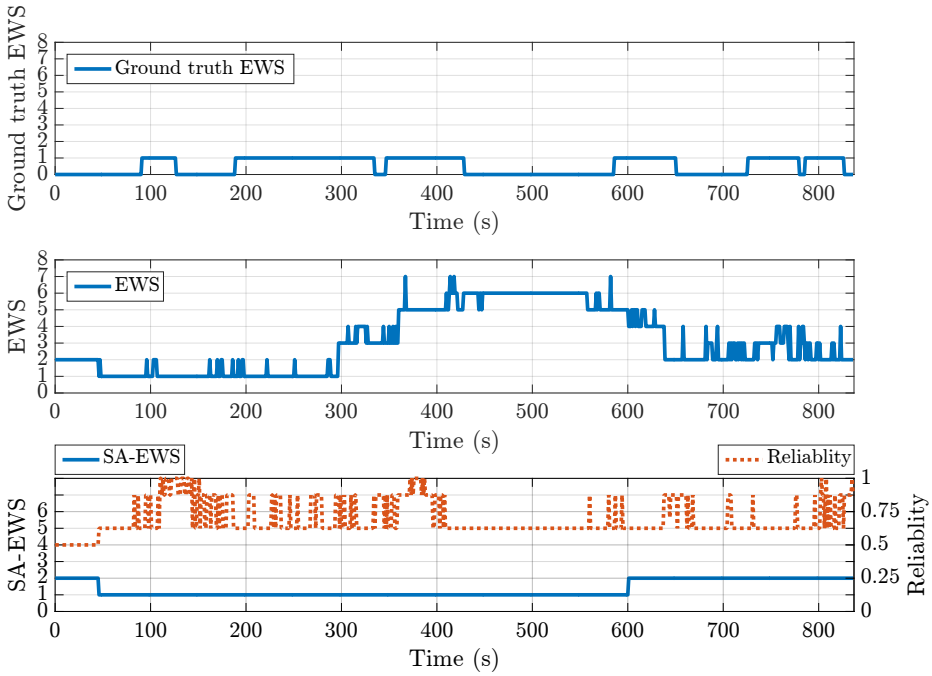
6.3.5 Experimental Results

This section gives an overview of the experiment results and the comparison of the conventional (non-self-aware) EWS with the SA-EWS. For more details and further results, the author refers to Paper II (Götzinger et al., 2019a).

Figure 26 illustrates one of the experiments conducted (Section 6.3.3). Figure 26a shows the ground truth vital sign signals (in the upper diagram) and the low-quality signals (in the lower diagram) of the same measurement. Up to three low-quality signals simultaneously show errors. These include the respiratory rate, which can be recognized by the many spikes in the signal. Besides, two additional (intentional) errors were introduced in the middle of



(a) The ground truth vital signs and the corrupted vital signs



(b) The ground truth EWS, as well as the EWS and SA-EWS systems' output

Figure 26. An experiment in which up to three vital sign errors simultaneously occur (Götzinger et al., 2019a)

the measurement: the temperature sensor was detached from the test subject's body, and the subject tensed the biceps. In comparison, Figure 26b shows the corresponding EWS outputs; from top to bottom: (i) the ground truth EWS (based on the ground truth vital signs), (ii) the output of the conventional EWS system, and (iii) the output of the SA-EWS system. The outputs of the EWS and the SA-EWS system were both based on the same corrupted vital signs. These results show that the difference between the EWS of the conventional system and the ground truth is sometimes huge (absolute errors up to 7 points), while the SA-EWS shows only absolute errors of 1 or 2 in the worst case.

Similar to this experiment (Figure 26), all 1278 datasets (containing low-quality signals) were processed with the conventional EWS system as well as with the SA-EWS system. The results were then compared with the respective ground truth. Figure 27 shows for all experiments the frequency of absolute errors of the EWS calculated by these two systems. While the abscissa shows all absolute errors that occurred (from $\varepsilon = 0$ to $\varepsilon = 7$), the ordinate shows the number of EWS samples deviating from the ground truth by ε. As can be seen, the SA-EWS produced slightly more samples without errors ($\varepsilon = 0$) but also slightly more where $\varepsilon = 1$. However, in both cases, the results (number of samples) of the two EWS systems are very similar. A larger gap can already be seen at $\varepsilon = 2$ and $\varepsilon = 3$, where the SA-EWS system apparently performs better. Overall, the SA-EWS system produces significantly fewer errors than the EWS system, especially when it comes to larger error sizes. The SA-EWS system never produces absolute errors greater than 5, whereas the conventional EWS system is responsible for more than 1000 of these errors. Even with error sizes of $\varepsilon = 4$ and $\varepsilon = 5$, the SA-EWS system produces significantly (about 1000) fewer errors than its counterpart. The errors with a high ε are especially crucial because larger errors imply a deviation from the ground truth risk class, which leads to false or missing alarms.

Overall, Figure 27 shows that the SA-EWS system is more reliable and robust against false vital sign measurements than its conventional counterpart. Moreover, the difference in the occurrence of false alarms is considerable. While none of the experiments resulted in a missed alarm in any of the systems, there were several false alarms. While the conventional EWS system caused a total of 5034 false alarms, the SA-EWS system caused only 64 — about 80 times less. This outperformance of the SA-EWS system clearly shows that CSA is capable of making an EWS system more robust and its output more reliable.

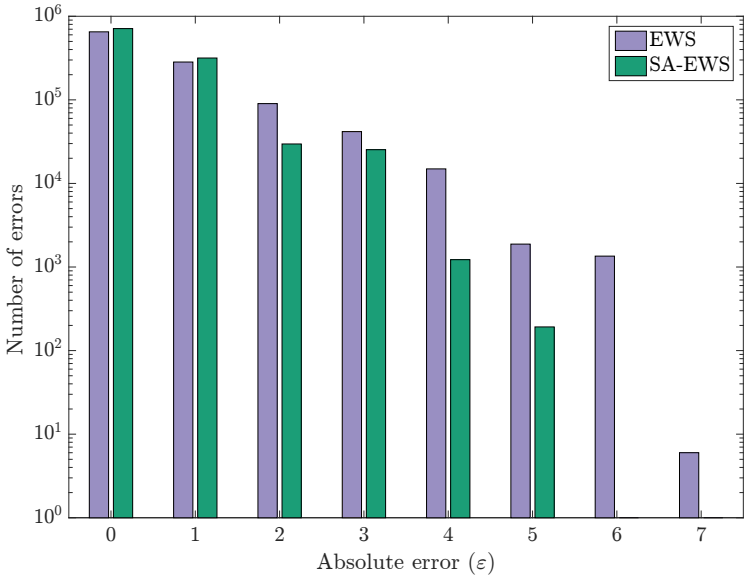


Figure 27. Occurrence frequency of absolute errors of different sizes (Götzinger et al., 2019a)

6.4 Confidence in Condition Monitoring System

Like the EWS system (Section 6.3), also the Condition Monitoring System (CMS) has to cope with uncertainties as defined by Taylor et al. (1994). It is usually unknown how noisy the signals of a System under Observation (SuO) are or how big the average difference between two signal states is. The difference between two signal states equals the difference between their sample values and will be called signal delta in the following. The sensitivity analysis (in Paper V) has shown that the threshold-based CMS (Section 4.4.4) output was quite robust against changes in the CMS parameters (e.g., thresholds). However, as discussed, there were problems with drift detection, where the CMS classified the SuO as malfunctioning after a short time. Besides, more complex motor data were simulated in the later stage of this work. Motor data with a set of different gradients of a drifting signal were simulated. Other motor datasets show significantly more state changes and much bigger ranges of signal delta sizes. With such complex data, it was impossible to select a suitable CMS parameter set with which the different states and conditions of the SuO could be correctly detected.

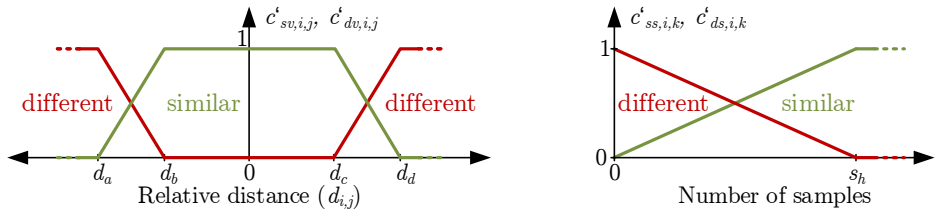
Since these are only uncertainties of the signal state detector, the basic CMS architecture does not need to be adapted (Section 4.4). Also the system state detector (located in the high-level agent) is more or less the same. However, the threshold-based signal state detector was replaced by a confidence-based signal state detector.

6.4.1 Confidence-Based Signal State Detection

This section is intended to give an overview of the confidence-based signal state detection used in the system called Confidence-based Context-Aware condition Monitoring (CCAM). For more details about this system, the author refers to Paper III (Götzinger et al., 2019b).

In the threshold-based signal state detector used previously (Section 4.4.1), a state is represented by the average value of all its signal samples. Simultaneously, there is one sliding window history in which the last signal samples are stored. A newly read sample is compared with the average values to determine to which state it belongs, but only if it is part of a stable signal phase. This stability is given if it matches a certain number of the sliding window history samples.

In the case of the confidence-based signal state detector presented here, this extra stability check is unnecessary because each state is represented by a sliding window history (containing the most recently added samples) instead of just an average value. When a new signal sample arrives, it is compared with this historical data to find out whether it belongs to a state or not. If the new sample is similar enough to a great enough set of history samples, it belongs to that state. However, it is not easy to determine which distance between two samples is close enough or how big the number of matching samples must be. Since it was already established in Section 5.4.2 that fuzzy logic can help counteract such uncertainties, four different fuzzy functions were designed to overcome these uncertainties (Figure 28).



(a) Confidences of a new sample being similar or different to an existing datum, based on the relative distance between them

(b) Confidences of a new sample being similar or different to an existing dataset, based on the number of similar samples

Figure 28. Fuzzy functions showing the confidences of a new sample fitting an already existing dataset

Figure 28a shows the two functions that determine whether a new sample value, $v_{i,new}$, of a variable of interest, i, matches or mismatches the j^{th} sample, $v_{h_{i,j}}$, of the i^{th} state's history. The confidence, $c'_{sv,i,j}$, of these two samples

matching each other is calculated by

$$c'_{sv,i,j} = \begin{cases} \frac{d_{i,j}-d_a}{d_b-d_a} & \text{if } d_a < d_{i,j} < d_b \\ 1 & \text{if } d_b \leq d_{i,j} \leq d_c \\ \frac{d_d-d_{i,j}}{d_d-d_c} & \text{if } d_c < d_{i,j} < d_d \\ 0 & \text{otherwise,} \end{cases} \tag{26}$$

where $d_{i,j}$ is the the relative distance between these two samples and calculated by

$$d_{i,j} = \frac{v_{i,new} - v_{h_{i,j}}}{v_{i,new}}, \tag{27}$$

which is also the base for determining the confidence, $c'_{dv,i,j}$, of these two samples mismatching each other, calculated by

$$c'_{dv,i,j} = \begin{cases} \frac{d_b-d_{i,j}}{d_b-d_a} & \text{if } d_a < d_{i,j} < d_b \\ 0 & \text{if } d_b \leq d_{i,j} \leq d_c \\ \frac{d_{i,j}-d_c}{d_d-d_c} & \text{if } d_c < d_{i,j} < d_d \\ 1 & \text{otherwise.} \end{cases} \tag{28}$$

Figure 28b shows the two functions that determine whether $v_{i,new}$ matches or mismatches a set of samples of a state's history. The confidence, $c'_{ss,i,k}$, for a new sample belonging to an existing dataset is calculated by

$$c'_{ss,i,k} = \begin{cases} 1 & \text{if } k \geq s_h \\ \frac{k}{s_h} & \text{if } 0 \leq k < s_h, \end{cases} \tag{29}$$

where k is the number of matching samples and s_h is the size of the sliding window history (maximum number of stored samples). In contrast, the confidence, $c'_{ds,i,k}$, for a new sample not belonging to an existing dataset is calculated by

$$c'_{ds,i,k} = \begin{cases} 0 & \text{if } k \geq s_h \\ \frac{s_h-k}{s_h} & \text{if } 0 \leq k < s_h, \end{cases} \tag{30}$$

whereby it is not specified how high these confidences (Equations 26, 28, 29, and 30) must be to decide whether a sample matches a state or not. Summarized, the two key questions are (i) how many history samples must match $v_{i,new}$, and (ii) how well must they match. Since answers to these questions would require a priori knowledge, which would not be consistent with black-box monitoring, these questions are simply left unanswered. Instead, a procedure is applied, in which only two confidences (belonging or not belonging to a state) remain at the end, and a decision is made on the basis of the higher one. However, the number of possible answers to one of the two questions can be narrowed down more easily, namely, to the number of history samples

that must match $v_{i,new}$. This sample set's size, k, can only be between 1 and s_h. Thus, instead of determining which is the right number, all possible cases, that is, s_h, are calculated for the confidence, $c'_{b,i,k}$, for $v_{i,new}$ belonging to a set of k history samples, by

$$\text{case k: } c'_{b,i,k} = (\bigwedge_{j=1}^{k} c'_{sv,i,j}) \wedge c'_{ss,i,k}, \tag{31}$$

where the $c'_{sv,i,j}$ is sorted by descending confidence ($c_{sv,i,j} \geq c_{sv,i,k}$; $\forall j \leq k$) to select a set with the best fitting samples. In further consequence, the case with the highest confidence $c'_{b,i} = \max(c'_{b,i,k}), i \in [1, s_h]$ is selected, indicating how well $v_{i,new}$ fits the state. Similarly, the confidence $c'_{n,i} = \min(c'_{n,i,k}), i \in [1, s_h]$ is selected, which indicates to what extent $v_{i,new}$ does not belong to a state. For this purpose, s_h cases for $c'_{n,i,k}$ (the confidence for $v_{i,new}$ not belonging to a set of k history samples) are calculated by

$$\text{case k: } c'_{n,i,k} = (\bigvee_{j=1}^{k} c'_{dv,i,j}) \vee c'_{ds,i,k}, \tag{32}$$

where the $c'_{dv,i,j}$ is sorted by descending confidence ($c_{dv,i,j} \geq c_{dv,i,k}$; $\forall j \leq k$) to select a set with the worst fitting samples.

Finally, $v_{i,new}$ matches a signal state if $c'_{b,i} > c'_{n,i}$ and mismatches it if $c'_{b,i} \leq c'_{n,i}$.

6.4.2 Experiments Conducted

Like the threshold-based CMS (Section 4.4), also the confidence-based CMS was tested on the two different SuOs: an Alternating Current (AC) motor used in a conveyor belt, and a water pipe system[7] driven by a Direct Current (DC) water pump. The AC motor of the first subcase study has the following signals: voltage (AC), current (AC), load torque, electrical torque, and rotating velocity. In contrast, the water pipe system datasets contained the following signals: voltage (DC), water temperature, and various water flow signals.

In both subcase studies, experiments were performed in which the respective SuO (i) functioned normally and performed regular state changes, (ii) showed malfunctions, and (iii) drifting signals. For each of these cases, one experiment is shown below (Figures 29, 30, and 31). The experiments shown here were selected so that the confidence-based CMS's outputs can be effectively compared with those of the threshold-based CMS (Section 4.4.4).

[7]More precisely, the water pipe system is a Heating, Ventilation, and Air Conditioning (HVAC) system, described in more detail in Papers III and V

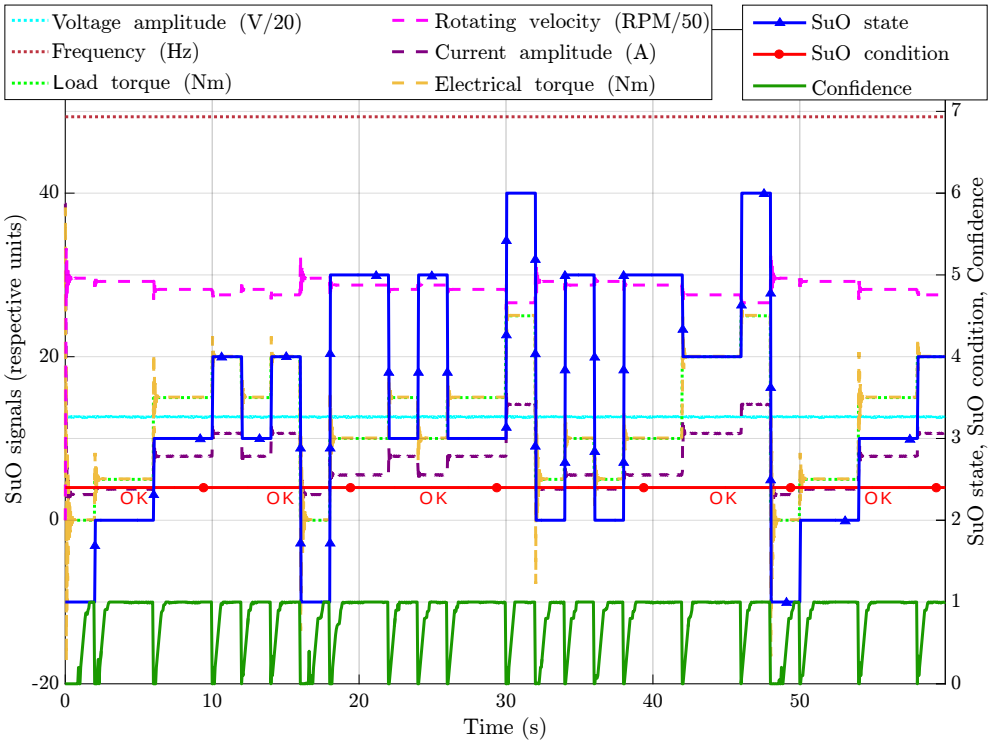


Figure 29. Output of the confidence-based CMS when monitoring a normally operating AC motor while 21 load changes take place (Götzinger et al., 2019b)

For further information and more experiments, the author refers to Paper III (Götzinger et al., 2019b). In the three upcoming figures, the graphs of the SuO's input signals are dotted, the graph of its output signals are dashed, and the output of the CMS (SuO's state and condition) is represented through solid lines.

Figure 29 shows one of the more complex motor scenarios mentioned, namely, an AC motor working normally while 21 external load changes happen. These load changes are of different sizes (load changes from 5Nm to 20Nm), leading to output signals with different signal delta sizes and signal oscillations directly after a load change. As can be seen, all states were detected correctly. The CMS could recognize states when they appeared for the first time and when the SuO changed back to an already known state.

When using the threshold-based signal state detector (Section 4.4.1), it was not possible to detect the motor's states and its condition in such complex scenarios. Another difference to the results of Section 4.4.4 is that the confidence-based CMS provides the overall confidence in addition to the SuO's states and its condition. This output indicates how confident the CMS is about

the correctness of its decisions (the classification of the SuO). It can be observed that the confidence at the beginning of each state is always around the value 0 and increases in the course of the same state until it is eventually around 1. This uncertainty at the beginning of a state comes from the SuO's output signals that are still unstable after the load changes. It can also be seen that the confidence value takes the longest time to reach 1 after switching on the motor and during significant load changes (e.g., 20Nm at 16s and 48s). The reason for this is that the output signals of the SuO show longer and more significant oscillations after such big load changes.

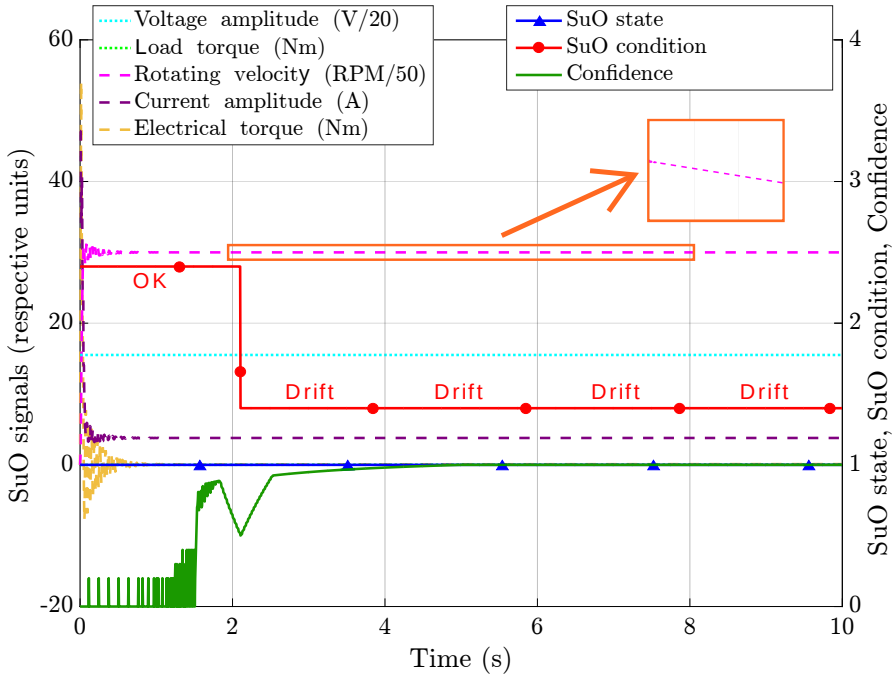


Figure 30. Output of the confidence-based CMS when monitoring an AC motor showing a drift

Figure 30 shows the same drifting motor scenario that was used in the experiment conducted with the threshold-based CMS (Figure 14). While the SuO signals are exactly identical, the CMS's output is different because the CMS is now making all decisions based on confidence. Comparing these two figures, the confidence-based CMS detects the drift approximately 0.2s earlier (at around 2.1s). However, the most crucial difference is that the SuO condition classified by the confidence-based CMS does not change from drift to malfunctioning; it remains with its decision. This decision of classifying the SuO as drifting also reflects the truth since the SuO obviously does not perform a state change.

As in this experiment (Figure 30), another 12 different drift scenarios were simulated, in which the rotational speed showed rates of change from 0.0025RPM/s to 0.022RPM/s. Besides the already discussed problem of changing the classification from drift to malfunction, the threshold-based CMS could only detect 4 out of 12 drifts (33% success rate). In contrast, the confidence-based CMS could detect all drifts and maintained its decision all along.

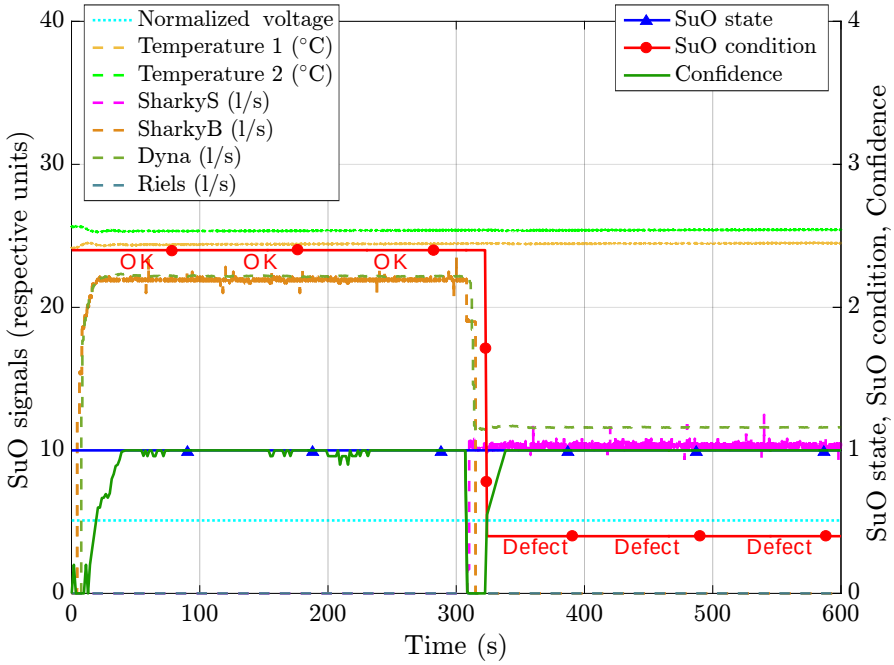


Figure 31. Output of the confidence-based CMS when monitoring a malfunctioning water pipe system (Götzinger et al., 2019b)

Figure 31 shows the malfunctioning water pipe scenario which was also processed by the threshold-based CMS (Figure 15). As before, the SuO signals are precisely the same, but the CMS output is a different one since it is now generated by decisions based on confidence. It can be seen that the confidence-based CMS can also recognize a malfunctioning SuO. The confidence-based CMS even classifies the SuO as malfunctioning about 25s earlier (at about 320s). It should be noted that the delay between the occurrence of an event and its detection by the CMS (threshold- as well as confidence-based) depends on the CMS's setup (e.g., thresholds, fuzzy functions). It has turned out that, if appropriate setups are chosen[8], the confidence-based CMS usually detects

[8]The CMS source code including parameter setup files can be found on the Research on Self-Awareness (RoSA) repository, available at https://phabricator.ict.tuwien.ac.at/source/SoC_Rosa_repo.git. Besides, Paper III provides a sensitivity analysis that shows all possible parameter setups.

such events earlier. This results from the threshold-based CMS's stability check (Section 4.4.1) which leads to delays in the decision-making process.

A total of 26 experiments (14 with different drifts, 12 functioning and malfunctioning) were conducted, whereby the confidence-based CMS correctly classified the SuO in all of them. In contrast, the threshold-based CMS correctly classified only 15 scenarios.

Another advantage of the confidence-based CMS is that the AC motor data could be used unfiltered. No extra preprocessing step of filtering was needed. As described in Section 4.4.4, this was necessary for the threshold-based CMS to correctly classify the AC motor's states and its condition.

In summary, confidence (i) increased the performance of the CMS (fast detection), (ii) made it more reliable (better detection), and (iii) made it more robust against noise (no filter needed). With all these enhancements through a confidence-based decision process, the CMS was still able to process all input samples faster than required for real-time execution (Section 3.4).

Additionally, a sensitivity analysis was performed (for details see Paper III (Götzinger et al., 2019b)) that shows that the results are reasonably robust against changes in the CMS parameters (e.g., fuzzy functions). However, for the future it would be desirable if the CMS learned these parameters by itself.

7 Discussion and Conclusion

Cyber-Physical Systems (CPSs) can be found in almost any technological field and are used in a wide range of applications needing autonomous solutions (Denker et al., 2012; Kim et al., 2017; Dafflon et al., 2021; Seshia et al., 2015; Zhao et al., 2017; Zhang et al., 2017; Oyewumi et al., 2019). To meet the requirement of being autonomous, adaptive, reliable, robust, efficient, and performant, a CPS needs comprehensive knowledge about itself and its environment (Denker et al., 2012; Kephart and Chess, 2003; Lewis et al., 2011). Without this knowledge, it is impossible for a CPS to make well-informed decisions, manage its objectives in a sophisticated way, and adapt to a possibly changing environment (Jantsch et al., 2018; Hoffmann et al., 2010; Rinner et al., 2015). To gain such comprehensive knowledge, a CPS must monitor itself and its environment. Data obtained during this process come from physical properties measured by sensors (Alexopoulos et al., 2016). However, the obtained data may not reflect the truth since sensors are neither completely accurate nor precise (TaheriNejad et al., 2016). Moreover, they could still be used incorrectly or break while operating. Besides, not all physical properties and environmental correlations may be well known. Additionally, in some cases, input data may be meaningless as long as they are not transferred to a domain understandable to the CPS (Jantsch et al., 2017). Regardless of the reason, such circumstances can result in a CPS that has an incomplete or inaccurate picture of itself and its environment, which can lead to wrong decisions with possible negative consequences (Azimi et al., 2016).

The work presented in this thesis is motivated by the need for a reliable and robust monitoring capability for CPSs that works correctly even in the presence of erroneous data (Berk et al., 2019). Besides the various research objectives and contributions, the central question of this work is whether Computational Self-Awareness (CSA), particularly its properties abstraction, data reliability, and confidence, enables monitoring to be more reliable and robust.

To answer this main question as well as to test various self-awareness methods, two suitable case studies from different areas were chosen. The first case study is related to the health sector. It describes a mobile system that monitors a patient's vital signs and alerts when the patient's condition is deteriorating. This so-called Early Warning Score (EWS) system calculates a value (the

EWS) that indicates the patient's condition. The EWS system is an optimal candidate for this thesis since it is highly dependent on reliable monitoring. As already mentioned, sensors have limited accuracy and precision. Besides, the system's user might use them incorrectly or they could break while in operation. Erroneous data may lead to a wrong classification of a patient's health condition and, in further consequence, to a false or — even worse — missing alarm.

The second case study is about a Condition Monitoring System (CMS) that monitors another unknown system (black box) to determine its condition, i.e., whether it is properly functioning or not. The CMS should be able to monitor various machines or devices in order to detect malfunctions so that maintenance work can be carried out before the condition of the System under Observation (SuO) deteriorates. Maintenance work as well as defective machines with associated repairs or long downtimes are usually highly cost-intensive. The goal is to carry out maintenance work as seldom as possible but as often as necessary. For this purpose, reliable monitoring that can also cope with a high level of noise is essential.

Among these three self-awareness properties, abstraction has the least precise definition and the broadest range of complexity among the methods developed here. While assigning a symbol to an input value is a very uncomplex task, finding signal states in a changing signal is already a complex task. Furthermore, it turned out that the use of abstraction does not necessarily make applications more reliable or robust. In both case studies, usage of abstraction was mandatory to enable the execution of the application itself. However, while the CMS equipped with abstraction already shows some robustness against different parameter settings, the EWS system could only reproduce the normal EWS system used in hospitals but was not more reliable or robust. Nevertheless, abstraction was fundamental in both applications, which clearly would not work without it. However, some abstraction methods were even enablers for the other self-awareness properties described in this thesis. Several data reliability and confidence methods were based on abstraction methods.

Because of the contradiction between erroneous data and a possible system failure, data reliability was applied exclusively in the EWS system. Due to resource and energy constraints in portable devices, three measures were used to assess the reliability of input data without requiring redundant hardware: consistency, plausibility, and cross-validity of the data. Those measures indicate whether data is within a plausible range, its observed changes appear reasonable, and different correlating variables match each other. Initial tests with a binary decision-making process, whether input data is reliable or unreliable, showed promising results but also revealed problems. The lack of complete knowledge of the human body and its processes render the drawing

of clear boundaries regarding vital signs' reliability impossible. A decision-making process based on fuzzy logic overcomes this shortcoming and makes it possible to cover these fuzzy areas in which vital signs can be to some extent plausible and implausible at the same time instead of just being either plausible or implausible. In all experiments conducted, the EWS system equipped with abstraction and data reliability could correctly indicate to which degree the calculated EWS can be trusted. The reliability value was significantly reduced in all situations where erroneous data were present and led to a false EWS.

To autonomously correct the calculated EWS solely based on the data reliability assessment is unsafe since it is not feasible to reasonably define a minimum reliability value at which the calculated EWS can be trusted. In a next development step, the self-awareness property confidence is used to make well-considered decisions regarding the EWS calculation and the assessed reliability. Extensive experiments were conducted, in which various participants' vital signs were recorded with different sensors of different quality. In this way, the output of the EWS system processing erroneous data could be compared with the output expected in the absence of faulty sensory data. These experiments proved that a self-aware EWS system (equipped with abstraction, data-reliability, and confidence) provides a more reliable EWS that is more robust against erroneous vital sign measurements than a conventional (non-self-aware) EWS system. The conventional EWS system caused a total of 5034 false alarms, while the self-aware EWS system raised only 64 false alarms — about 80 times less.

A similar situation was also evident in the CMS after it was enhanced with a confidence-based decision-making process. Confidence enables the system to correctly process even more complex scenarios of the SuO, which was impossible without confidence. While the CMS without confidence falsely classified the SuO's condition, the confidence-based CMS no longer showed these errors. Furthermore, the additional preprocessing step of filtering signals of the SuO became obsolete through the usage of confidence. In other words, confidence made the CMS's output more reliable and its detection process more robust against noise.

In summary, this means that the results derived from the extensive experiments conducted based on both case studies prove the hypothesis of this research work, namely that CSA, especially the properties abstraction, data reliability, and confidence, can improve a system's monitoring capabilities regarding its robustness and reliability — even in the presence of erroneous data.

To arrive at these results, other steps besides conducting the experiments were necessary. Due to the lack of a common framework, research on different

self-awareness properties and methods as well as development of self-aware applications used to be rather slow. After an extensive review of existing literature, it became evident that no existing framework met the desired requirements. Applications implemented with the help of such a framework must be deployable and executable on Embedded Systems (ESs) with limited resources in order to be applicable in the two chosen case studies. Therefore, the Research on Self-Awareness (RoSA) framework was developed in the course of this work. With this framework, self-aware hierarchical agent-based applications can be implemented in a simple and straightforward manner. RoSA offers both ready-to-use self-awareness functionalities (methods) as well as the possibility to integrate custom code.

To obtain a quantitative measure of the reduced development effort resulting from the usage of RoSA, a comparison between the number of lines of code of the original custom-written applications and the RoSA-based implementation was made. The RoSA-based implementations make up only 3.46% to 6.24% (on average about 5.15%) non-comment lines of code of the original custom-written applications. Moreover, RoSA was also tested on different ESs, where it could process input samples, depending on the application and the hardware platform on which it was running, around 5 to 4500 times faster than required for real-time execution. Furthermore, RoSA could easily cope with memory constraints since the implemented applications posed a moderate memory footprint well below 4MB, which fits typical ESs. It is to be hoped that thanks to its good usability, RoSA will serve as a common framework for the research community to explore uncharted aspects of CSA and speed up development in the field.

Besides the reusability of small software modules, RoSA's hierarchical agent-based architecture was a clear advantage for the implemented systems. The application of all three self-awareness properties investigated in this thesis benefited from this well-structured approach. Tasks can be divided into different less-complex subtasks located on different levels of abstracted knowledge.

All self-awareness functionalities and development steps of the two self-aware applications (EWS system and CMS) were implemented in RoSA. The documentation[1] of RoSA shows how to set up the framework. The source code of all applications including parameter setup files can be found on the RoSA repository[2]. In addition, the various publications included in this thesis describe the implementations of the two systems and also provide information about all possible parameter setups to sensitivity analyses (Papers III and V).

[1] A documentation that shows how to set up RoSA is available at https://www.rosa.ict.tuwien.ac.at/docs/html/.

[2] RoSA is an open-source implementation that is available at https://phabricator.ict.tuwien.ac.at/source/SoC_Rosa_repo.git.

7.1 Open Directions

Future research can take several directions. They can be divided into two parts: further research on CSA and further development of the case studies' systems, whereby both parts are interconnected.

There is still a set of mostly unexplored self-awareness properties that could be investigated. Besides investigating other properties, methods proposed in this work could be improved. As this thesis did not have the goal to find the best possible solution, it could be a motivation to find better solutions.

In the author's opinion, one of the self-awareness properties not covered in this work could enhance both case studies' systems, namely, learning. For example, the EWS system could learn over time from the monitored patient about their body functions. Thus, it would be possible to have a highly personalized EWS system that recognizes the patient's condition based on their specific needs.

The CMS would also benefit from the self-awareness property learning. Currently, the CMS's parameters have to be set during an initial setup process. Although this setup does not need to be particularly accurate, overcoming it would be a considerable advantage. With the ability to learn, this setup could be done autonomously by the CMS itself.

However, the CMS could also benefit from other enhancements. It would be an advantage if the CMS could also monitor black boxes that do not behave like bijective functions. Furthermore, unsteady states should also be detected, for example, transient phases of a signal. It would also be useful to find a way for the CMS to simultaneously verify the SuO's working condition as well as the reliability of observed input data.

8 Publications Overview and Author's Contributions

This chapter presents a summarized overview of all original publications included in this thesis and states the author's contributions to each of them.

8.1 Paper I – RoSA: A Framework for Modeling Self-Awareness in Cyber-Physical Systems

Paper I proposes the Research on Self-Awareness (RoSA) framework that supports modeling and evaluating various Computational Self-Awareness (CSA) concepts in hierarchical agent systems where agents are made up of self-awareness functionalities. The paper also presents the framework's design principles and discusses the use cases of RoSA-based modeling for different scenarios. Additionally, its capabilities are demonstrated by showing the process of implementation in the case studies from the health- and industrial sectors. Hereby, also the usage of the currently provided self-awareness functionalities (abstraction, data reliability, and confidence) is shown. The paper also describes the implementation process of these applications in RoSA to show the framework's modeling power and applicability. RoSA is meant as a vehicle for researchers to study various concepts related to CSA and the relations among them. It also aims to assist engineers in prototyping and evaluating self-awareness features. The helpfulness of RoSA is manifested in the fact that the non-comment lines of code of the RoSA-based implementations make up only 3.46% to 6.24% of the equivalent custom-written applications. Besides, RoSA-based applications generate a moderate memory footprint well below 4MB. These applications were also tested on different Embedded Systems (ESs). Depending on the application and the ES on which they were running, they could process input samples around 5 to 4500 times faster than required for real-time execution.

Author's contributions: The author shares the first authorship with the second-listed author (Dávid Juhász). The author had a significant role in the predevelopment phase, in planning requirements, and in prototyping RoSA. Furthermore, he was responsible for designing and developing the various self-

awareness functionalities. Besides, he was one of the main contributors in writing the manuscript, on a par with Dávid Juhász.

8.2 Paper II – Confidence-Enhanced Early Warning Score Based on Fuzzy Logic

In Paper II, an Early Warning Score (EWS) system enhanced with the self-awareness properties abstraction, data-reliability, and confidence is proposed. Additionally, the properties data reliability, confidence, and history are formalized and implementations of corresponding methods are shown. A confidence metric based on fuzzy logic provides information about the input data's correctness and, in further consequence, about the reliability of the system's output (the EWS). Because the output reliability is a combination of the input data's reliability and the system's confidence, a method for combining these two measures is proposed. Extensive experiments prove that the proposed system provides equally good or better results than a similar system that does not use reliability and confidence metrics. These experiments demonstrate that the proposed system provides — under adverse monitoring circumstances (such as noisy signals, detached sensors, and non-nominal monitoring conditions) — a more reliable EWS than a conventional EWS system (without self-awareness properties).

Author's contributions: The author developed and implemented the self-aware EWS system with all its self-awareness methods. Furthermore, he designed the participants' measurement setup. With the measurement datasets (recorded and abstracted by the coauthors), he processed all experiments with a conventional as well as with the self-aware EWS system. He then compared the performances of both systems by using appropriate metrics. Besides, he was the main contributor in writing the manuscript.

8.3 Paper III – Model-Free Condition Monitoring with Confidence

Paper III proposes the model-free Confidence-based Context-Aware condition Monitoring (CCAM) system which uses only contextual information to recognize the condition (working state and health status[1]) of a black-box system that behaves like a bijective function. A fuzzy logic-based confidence metric for the quality assessment of observation is introduced and leveraged to improve the correct identification of an observed system's states. CCAM requires neither in-depth knowledge of the field nor any cumbersome effort to adjust its

[1]It must be noted that, in this context, the term "health" describes the condition of a machine or device; i.e., whether it is working properly or malfunctioning.

parameters to the given application. Various experiments show that CCAM can properly monitor and assess working conditions of two diverse systems (an industrial motor and a waterpipe system) without the need for model building or any other customization. These experiments also show that confidence not only improves the quality of system performance but also enhances its robustness. It is shown that CCAM is better at identifying the correct system condition compared to a similar system without confidence. Additionally, a sensitivity analysis shows that CCAM is robust against small variations of parameter settings.

Author's contributions: The author developed and implemented CCAM with all its self-awareness methods. Furthermore, he designed the measurement and simulation setup for both systems (industrial motor and waterpipe system) and simulated some additional motor measurement datasets. With the obtained datasets, he processed all experiments with CCAM and a comparable system without confidence. He then compared the performances of both systems and performed a sensitivity analysis regarding CCAM's parameters. Besides, he was the main contributor in writing the manuscript.

8.4 Paper IV – On the Design of Context-Aware Health Monitoring without A Priori Knowledge; an AC-Motor Case-Study

Paper IV proposes a monitoring system, which uses context-awareness to assess the condition of a black-box system that behaves like a bijective[2] function. The proposed system, Context-Aware Health Monitoring (CAH)[3], can identify normal modes of operation, change of states (operation modes), deviation from a state, and abnormal functional operation. Compared to other methods, such as deep learning and data mining, the proposed system is designed to have a small footprint. It is intended for use under resource constraints to be also suitable for implementation in smaller gadgets with limited computing power. To verify the validity of the proposed approach, CAH was tested on Alternating Current (AC) motor datasets, where it could successfully identify different operation modes of the AC motor and whether the motor worked correctly or was malfunctioning.

Author's contributions: The author developed and implemented CAH with all its methods. Furthermore, he designed the measurement and sim-

[2]In this paper, the term injective instead of bijective was chosen. This is imprecise but not wrong because a bijective function is also always injective. However, using bijective is more precise and means both injective and surjective.

[3]It must be noted that, in this context, the term "health" describes the condition of a machine or device; i.e., whether it is working properly or malfunctioning.

ulation setup for the AC motor experiments. With the obtained datasets (recorded and simulated by the coauthors), he processed all experiments with CAH. Besides, he was the main contributor in writing the manuscript.

8.5 Paper V – Applicability of Context-Aware Health Monitoring to Hydraulic Circuits

In Paper V, the scope of Context-Aware Health Monitoring (CAH)[4], introduced in an earlier publication, was extended to Heating, Ventilation, and Air Conditioning (HVAC) applications. In other words, in contrast to the earlier work, CAH was now used in a hydraulic circuit (a waterpipe system) — an entirely different industrial application. While the CAH algorithm remained unchanged, the only necessary adaptation concerned the time parameter because of the significant inertia differences between the application of the earlier work (an Alternating Current (AC) motor) and the waterpipe system. The results show the potential for considerable benefits in monitoring HVAC systems. Like in the AC-motor case study, CAH could identify normal modes of operation, change of states (operation modes), deviation from a state, and abnormal functional operation. Moreover, since CAH would be used in different applications that may need a different setup of parameters, a sensitivity analysis of the values of different CAH parameters was performed. The results of this sensitivity analysis show the robustness of CAH concerning the values of these parameters.

Author's contributions: The author designed the measurement setup for the waterpipe system experiments. With the obtained datasets (recorded by the coauthors), he processed all experiments with CAH. Furthermore, he conducted a sensitivity analysis of the values of different parameters of CAH regarding both applications: the waterpipe system and the AC motor (using the datasets of the earlier publication). Besides, he was the main contributor in writing the manuscript.

8.6 Paper VI – Self-Awareness in Remote Health Monitoring Systems Using Wearable Electronics

Paper VI proposes a self-aware Early Warning Score (EWS) system that provides personalization (reflecting parameters such as age, body mass index, and gender), self-organization, and autonomy for remote monitoring scenarios. Moreover, it offers an intelligent decision-making process for patients in different situations (sleeping, walking, running, and resting). Furthermore,

[4]It must be noted that, in this context, the term "health" describes the condition of a machine or device; i.e., whether it is working properly or malfunctioning.

self-awareness properties are used to improve the system's energy efficiency and the reliability of the calculated EWS. Additionally, a proof of concept implementation of an EWS system, from cloud services development to hard- and software demonstration, is proposed in the paper.

Author's contributions: The author developed and implemented the algorithms which improve the EWS system's reliability when erroneous sensory data are present. Besides, he contributed to writing and reviewing the manuscript.

8.7 Paper VII – Enhancing the Early Warning Score System Using Data Confidence

Paper VII proposes an Early Warning Score (EWS) system that has a hierarchical agent-based architecture and is equipped with a binary data reliability[5] assessment. The data reliability assessment's goal is to detect erroneous vital signal measurement. This is done by analyzing the sensory data concerning its plausibility, consistency, and cross-validity. This work is comparable to a proof of concept, and the experiments demonstrate that the proposed system correctly identifies erroneous data. They also showed that the system's hierarchical agent-based architecture perfectly matches the data processing flow from lower to higher abstraction levels.

Author's contributions: The author developed and implemented the EWS system with all its self-awareness methods. Furthermore, he processed the obtained datasets (recorded by the coauthors) with the proposed EWS system. Besides, he was the main contributor in writing the manuscript.

8.8 Paper VIII – Enhancing the Self-Aware Early Warning Score System through Fuzzified Data Reliability Assessment

Paper VIII presents an enhancement of the hierarchical agent-based Early Warning Score (EWS) system proposed in an earlier publication. This enhanced EWS system contains a data reliability validation technique based on plausibility, consistency, and cross-validity. However, while the earlier work's system used a binary decision-making process, the data reliability assessment proposed here is based on fuzzy logic. Boolean logic cannot cover all natural measurement data, which usually lack sharp limits. In contrast, fuzzy logic enables the EWS to cover also the fuzzy ranges in which vital signs can be to some extent plausible and implausible at the same time instead of just being either plausible or implausible. The experiments showed that the proposed system

[5] In this publication, data reliability was falsely called confidence.

successfully detected erroneous vital-sign data caused by incorrect measurements due to loose sensors, detached sensors, or the test subject's abnormal behavior (e.g., biceps contraction during a blood pressure measurement). To indicate this, the EWS system decreased the output data's reliability during such events. The experiments' outcomes show that self-awareness techniques such as the proposed data reliability assessment can provide a more robust EWS system with a more reliable EWS output.

Author's contributions: The author developed and implemented the EWS system with all its self-awareness methods. Furthermore, he designed the test subject's measurement setup. With the measurement datasets (recorded and abstracted by the coauthors), he processed all experiments with the proposed EWS system. Besides, he was the main contributor in writing the manuscript.

Abbreviations

AC Alternating Current 10, 31, 40, 41, 49, 51–53, 55, 90–92, 94, 103, 104
AFDD Automated Fault Detection and Diagnostic 27
AI Artificial Intelligence 18, 19
ANN Artificial Neural Network 28, 29
ART Adaptive Resonance Theory 28

CAH Context-Aware Health Monitoring 31, 40, 46, 103, 104
CCAM Confidence-based Context-Aware condition Monitoring 31, 40, 53, 88, 102, 103
CMS Condition Monitoring System 7, 8, 10, 11, 30, 31, 40, 41, 43, 46–55, 63, 87, 90–94, 96–99
CPS Cyber-Physical System v, vii, ix, 1–7, 13–16, 18–20, 38, 59, 95
CSA Computational Self-Awareness v, ix, 5–11, 13, 15–19, 21, 25, 33, 38, 63, 83, 86, 95, 97–99, 101
CSV Comma-Separated Values 39, 40

DAB Discrete Average Block 51
DARPA Defense Advanced Research Projects Agency 14
DC Direct Current 40, 41, 51, 90
DT Digital Twin 17, 18

ECG Electrocardiogram 24, 25
ES Embedded System 19–21, 29, 33, 34, 42, 54, 98, 101
EWS Early Warning Score 7, 8, 10, 11, 21–26, 31, 38–40, 43, 45, 46, 57, 60, 62–68, 71–74, 76, 78–81, 83–87, 95–99, 102, 104–106

FDA U.S. Food and Drug Administration 23

HMM Hidden Markov Model 29
HVAC Heating, Ventilation, and Air Conditioning 10, 28, 29, 31, 40, 51, 90, 104

IBM International Business Machines Corporation 14, 15, 19
IoT Internet of Things 1, 23, 24

IT Information Technology 14

MAS Multi-Agent System 19
MPC Model Predictive Control 17
MPSoC Multi-Processor System-on-Chip 17

NASA National Aeronautics and Space Administration 14
NN Neural Network 28

ODA Observe-Decide-Act 18, 35–38

PID Proportional–Integral–Derivative 17
PPG Photoplethysmogram 24

RMSD Root-Mean-Square Deviation 83, 84
RoSA Research on Self-Awareness 7, 8, 20, 21, 33–42, 93, 98, 101
RPM Revolution per Minute 61

SA-EWS Self-Aware Early Warning Score 8, 83–86
SHW Solar Hot Water 28
SoC System on Chip 16, 17
SoS System of Systems 1
SuO System under Observation 26–31, 40, 45–55, 63, 87, 90–94, 96, 97, 99

List of References

Abeywickrama, D. B. and Ovaska, E. (2017). A survey of autonomic computing methods in digital service ecosystems. *Service Oriented Computing and Applications*, 11(1):1–31.

Akbar, A. and Lewis, P. R. (2018). Self-adaptive and self-aware mobile-cloud hybrid robotics. In *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, pages 262–267.

Al-Ars, Z., Basten, T., de Beer, A., Geilen, M., Goswami, D., Jääskeläinen, P., Kadlec, J., de Alejandro, M. M., Palumbo, F., Peeren, G., Pomante, L., van der Linden, F., Saarinen, J., Säntti, T., Sau, C., and Zedda, M. K. (2019). The fitoptivis ecsel project: Highly efficient distributed embedded image/video processing in cyber-physical systems. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*, CF '19, page 333–338, New York, NY, USA. Association for Computing Machinery.

Alamir, M. (2013). *A Pragmatic Story of Model Predictive Control: Self Contained Algorithms and Case-studies*. CreateSpace Independent Publishing Platform.

Albus, J. S. (1991). Outline for a theory of intelligence. *IEEE transactions on systems, man, and cybernetics*, 21(3):473–509.

Alexopoulos, K., Makris, S., Xanthakis, V., Sipsas, K., and Chryssolouris, G. (2016). A concept for context-aware computing in manufacturing: the white goods case. *International Journal of Computer Integrated Manufacturing*, 29(8):839–849.

Alidoost Nia, M., Kargahi, M., and Faghih, F. (2020). Probabilistic approximation of runtime quantitative verification in self-adaptive systems. *Microprocessors and Microsystems*, 72:102943.

Angelopoulos, K., Papadopoulos, A. V., Souza, V. E. S., and Mylopoulos, J. (2018). Engineering self-adaptive software systems: from requirements to model predictive control. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 13(1):1–27.

Anzanpour, A., Azimi, I., Götzinger, M., Rahmani, A. M., TaheriNejad, N., Liljeberg, P., Jantsch, A., and Dutt, N. (2017). Self-awareness in remote health monitoring systems using wearable electronics. In *Design, Automation and Test in Europe (DATE), 2017*, pages 1056–1061. IEEE.

Anzanpour, A., Rahmani, A.-M., Liljeberg, P., and Tenhunen, H. (2015a). Context-aware early warning system for in-home healthcare using internet-of-things. In *International Internet of Things Summit*, pages 517–522. Springer.

Anzanpour, A., Rahmani, A.-M., Liljeberg, P., and Tenhunen, H. (2015b). Internet of things enabled in-home health monitoring system using early warning score. In *Proceedings of the 5th EAI International Conference on Wireless Mobile Communication and Healthcare*, MOBIHEALTH'15, page 174–177, Brussels, BEL. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Anzanpour, A., Rashid, H., Rahmani, A. M., Jantsch, A., Dutt, N., and Liljeberg, P. (2019). Energy-efficient and reliable wearable internet-of-things through fog-assisted dynamic goal management. *Procedia Computer Science*, 151:493–500. The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd

International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops.

Apostolakis, G. (1990). The concept of probability in safety assessments of technological systems. *Science*, 250(4986):1359–1364.

Arcaini, P., Riccobene, E., and Scandurra, P. (2015). Modeling and analyzing mape-k feedback loops for self-adaptation. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 13–23.

ARM (2013). big.LITTLE Technology: The Future of Mobile. Technical report, ARM.

Aßmann, U., Götz, S., Jézéquel, J.-M., Morin, B., and Trapp, M. (2014). A reference architecture and roadmap for models@ run. time systems. In Bencomo, N., France, R., Cheng, B. H. C., and Aßmann, U., editors, *Models@run.time: Foundations, Applications, and Roadmaps*, pages 1–18. Springer International Publishing, Cham.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805.

Aven, T., Baraldi, P., Flage, R., and Zio, E. (2013). *Uncertainty in risk assessment: the representation and treatment of uncertainties by probabilistic and non-probabilistic methods*. John Wiley & Sons.

Averkin, A. and Yarushev, S. (2021). Fuzzy rules extraction from deep neural networks. In *Proceedings of the of the XXIII International Conference "Enterprise Engineering and Knowledge Management" (EEKM 2020)*.

Avižienis, A. (1967). Design of fault-tolerant computers. In *Proceedings of the November 14-16, 1967, fall joint computer conference*, pages 733–743.

Avizienis, A., Laprie, J.-C., and Randell, B. (2001). Fundamental concepts of dependability. *Department of Computing Science Technical Report Series*.

Azimi, I., Anzanpour, A., Rahmani, A. M., Liljeberg, P., and Tenhunen, H. (2016). Self-aware early warning score system for iot-based personalized healthcare. In *eHealth 360°*, pages 49–55, Cham. Springer International Publishing.

Baek, W. and Chilimbi, T. M. (2010). Green: a framework for supporting energy-conscious programming using controlled approximation. In *ACM Sigplan Notices*, volume 45, pages 198–209. ACM.

Bantz, D. F., Bisdikian, C., Challener, D., Karidis, J. P., Mastrianni, S., Mohindra, A., Shea, D. G., and Vanover, M. (2003). Autonomic personal computing. *IBM Systems Journal*, 42(1):165–176.

Baraldi, P., Compare, M., and Zio, E. (2014). Uncertainty treatment in expert information systems for maintenance policy assessment. *Applied Soft Computing*, 22:297–310.

Barenji, A. V., Li, Z., Wang, W. M., Huang, G. Q., and Guerra-Zubiaga, D. A. (2020). Blockchain-based ubiquitous manufacturing: a secure and reliable cyber-physical system. *International Journal of Production Research*, 58(7):2200–2221.

Batty, M. (2018). Digital twins. *Environment and Planning B: Urban Analytics and City Science*, 45(5):817–820.

Baudrit, C., Dubois, D., and Perrot, N. (2008). Representing parametric probabilistic models tainted with imprecision. *Fuzzy sets and systems*, 159(15):1913–1928.

Bellifemine, F., Bergenti, F., Caire, G., and Poggi, A. (2005). Jade —- a java agent development framework. In Bordini, R. H., Dastani, M., Dix, J., and El Fallah Seghrouchni, A., editors, *Multi-Agent Programming: Languages, Platforms and Applications*, pages 125–147. Springer US, Boston, MA.

Bellman, K., Landauer, C., Dutt, N., Esterle, L., Herkersdorf, A., Jantsch, A., TaheriNejad, N., Lewis, P. R., Platzner, M., and Tammemäe, K. (2020). Self-aware cyber-physical systems. *ACM Trans. Cyber-Phys. Syst.*, 4(4).

Bellman, K. L. (1991). An approach to integrating and creating flexible software environments supporting the design of complex systems. In *1991 Winter Simulation Conference Proceedings.*, pages 1101–1105.

Berk, M., Schubert, O., Kroll, H.-M., Buschardt, B., and Straub, D. (2019). Reliability assessment of safety-critical sensor information: Does one need a reference truth? *IEEE Transactions on Reliability*, 68(4):1227–1241.

Borase, R. P., Maghade, D., Sondkar, S., and Pawar, S. (2020). A review of pid control, tuning methods and applications. *International Journal of Dynamics and Control*, pages 1–10.

Bouajila, A., Zeppenfeld, J., Stechele, W., Bernauer, A., Bringmann, O., Rosenstiel, W., and Herkersdorf, A. (2011). Autonomic system on chip platform. In Müller-Schloer, C., Schmeck, H., and Ungerer, T., editors, *Organic Computing - A Paradigm Shift for Complex Systems*, Autonomic Systems, chapter 4.7, pages 413–425. Birkhäuser.

Bouajila, A., Zeppenfeld, J., Stechele, W., Herkersdorf, A., Bernauer, A., Bringmann, O., and Rosenstiel, W. (2006). Organic computing at the system on chip level. In *2006 IFIP International Conference on Very Large Scale Integration*, pages 338–341.

Bousdekis, A., Apostolou, D., and Mentzas, G. (2020). Predictive maintenance in the 4th industrial revolution: Benefits, business opportunities, and managerial implications. *IEEE Engineering Management Review*, 48(1):57–62.

Bowles, J. and Pelaez, C. (1995). Application of fuzzy logic to reliability engineering. *Proceedings of the IEEE*, 83(3):435–449.

Brosinsky, C., Song, X., and Westermann, D. (2019). Digital twin - concept of a continuously adaptive power system mirror. In *International ETG-Congress 2019; ETG Symposium*, pages 1–6.

Brown, D. J., Brugger, H., Boyd, J., and Paal, P. (2012). Accidental hypothermia. *New England Journal of Medicine*, 367(20):1930–1938.

Bucchiarone, A. (2019). Collective Adaptation through Multi-Agents Ensembles: The Case of Smart Urban Mobility. *ACM Trans. Auton. Adapt. Syst.*, 14(2):6:1–6:28.

Bucchiarone, A., De Sanctis, M., Marconi, A., and Martinelli, A. (2017). DeMOCAS: Domain Objects for Service-Based Collective Adaptive Systems. In *Serv. Comput. – ICSOC 2016 Work.*, pages 174–178.

Cámara, J., Bellman, K. L., Kephart, J. O., Autili, M., Bencomo, N., Diaconescu, A., Giese, H., Götz, S., Inverardi, P., Kounev, S., et al. (2017). Self-aware computing systems: Related concepts and research areas. In Kounev, S., Kephart, J. O., Milenkoski, A., and Zhu, X., editors, *Self-Aware Computing Systems*, pages 17–49. Springer International Publishing, Cham.

Cao, C., Kohane, I. S., and McIntosh, N. (1999). Artifact detection in cardiovascular time series monitoring data from preterm infants. In *Proceedings of the AMIA Symposium*, page 207. American Medical Informatics Association.

Carreras, I., Chlamtac, I., De Pellegrini, F., and Miorandi, D. (2007). BIONETS: Bio-Inspired Networking for Pervasive Communication Environments. *IEEE Trans. Veh. Technol.*, 56(1):218–229.

Chammas, A., Traore, M., Duviella, E., Sayed-Mouchaweh, M., and Lecoeuche, S. (2013). Drift detection and characterization for condition monitoring: application to dynamical systems with unknown failure modes. *IMA Journal of Management Mathematics*, 26(2):225–243.

Chang, W., Zhao, S., Wei, R., Wellings, A., and Burns, A. (2019). From java to real-time java: A model-driven methodology with automated toolchain. In *Proceedings of the 20th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, pages 123–134.

Charousset, D., Hiesgen, R., and Schmidt, T. C. (2016). Revisiting Actor Programming in C++. *Computer Languages, Systems & Structures*, 45:105–131.

Chaturvedi, V., Singh, A. K., Zhang, W., and Srikanthan, T. (2014). Thermal-aware task scheduling for peak temperature minimization under periodic constraint for 3d-mpsocs.

In *2014 25nd IEEE International Symposium on Rapid System Prototyping*, pages 107–113.

Chen, B., Cheng, H. H., and Palen, J. (2006). Mobile-C: A Mobile Agent Platform for Mobile C-C++ Agents. *Softw. Pract. Exper.*, 36(15):1711–1733.

Chen, T., Faniyi, F., Bahsoon, R., Lewis, P. R., Yao, X., Minku, L. L., and Esterle, L. (2014). The handbook of engineering self-aware and self-expressive systems. *Computing Research Repository (CoRR)*, abs/1409.1793.

Collier, N. and North, M. (2013). Parallel agent-based simulation with Repast for High Performance Computing. *SIMULATION*, 89(10):1215–1235.

Dafflon, B., Moalla, N., and Ouzrout, Y. (2021). The challenges, approaches, and used techniques of cps for manufacturing in industry 4.0: a literature review. *The International Journal of Advanced Manufacturing Technology*, pages 1–18.

Davies, P. and Maconochie, I. (2009). The relationship between body temperature, heart rate and respiratory rate in children. *Emerg Med J.*, 26(9):641–3.

De Lemos, R., Garlan, D., Ghezzi, C., Giese, H., Andersson, J., Litoiu, M., Schmerl, B., Weyns, D., Baresi, L., Bencomo, N., Brun, Y., Camara, J., Calinescu, R., Cohen, M. B., Gorla, A., Grassi, V., Grunske, L., Inverardi, P., Jezequel, J.-M., Malek, S., Mirandola, R., Mori, M., Müller, H. A., Rouvoy, R., Rubira, C. M. F., Rutten, E., Shaw, M., Tamburrelli, G., Tamura, G., Villegas, N. M., Vogel, T., and Zambonelli, F. (2017). Software engineering for self-adaptive systems: Research challenges in the provision of assurances. In de Lemos, R., Garlan, D., Ghezzi, C., and Giese, H., editors, *Software Engineering for Self-Adaptive Systems III. Assurances*, pages 3–30, Cham. Springer International Publishing.

Denker, G., Dutt, N., Mehrotra, S., Stehr, M.-O., Talcott, C., and Venkatasubramanian, N. (2012). Resilient dependable cyber-physical systems: a middleware perspective. *Journal of Internet Services and Applications*, 3(1):41–49.

Dexter, A. and Pakanen, J. (2001). Demonstrating automated fault detection and diagnosis methods in real buildings.

Dexter, A. L. and Ngo, D. (2001). Fault diagnosis in air-conditioning systems: A multi-step fuzzy model-based approach. *HAC&R Research*, 7:83–102.

Dohr, A., Modre-Opsrian, R., Drobics, M., Hayn, D., and Schreier, G. (2010). The internet of things for ambient assisted living. In *2010 seventh international conference on information technology: new generations*, pages 804–809. Ieee.

Du, Z., Fan, B., Jin, X., and Chi, J. (2014). Fault detection and diagnosis for buildings and hvac systems using combined neural networks and subtractive clustering analysis. *Building and Environment*, 73:1–11.

Duranton, M., De Bosschere, K., Coppens, B., Gamrat, C., Gray, M., Munk, H., Ozer, E., Vardanega, T., and Zendra, O. (2019). *The HiPEAC Vision 2019*. HiPEAC CSA.

Dutt, N., Jantsch, A., and Sarma, S. (2015). Self-aware cyber-physical systems-on-chip. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 46–50. IEEE.

Dutt, N., Jantsch, A., and Sarma, S. (2016). Toward smart embedded systems: A self-aware system-on-chip (soc) perspective. *ACM Transactions on Embedded Computing Systems (TECS)*, 15(2):22.

Dutt, N. and TaheriNejad, N. (2016). Self-awareness in cyber-physical systems. In *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, pages 5–6.

Excel Medical (2018). Excel Medical's WAVE Clinical Platform Receives FDA Clearance. http://excel-medical.com/wp-content/uploads/RELEASE_FDA_Clearance_AchievedbyExcelMedical_1-8-18_RevMB6A.pdf.

Fan, B., Du, Z., Jin, X., Yang, X., and Guo, Y. (2010). A hybrid fdd strategy for local system of ahu based on artificial neural network and wavelet analysis. *Building and environment*, 45(12):2698–2708.

Faniyi, F., Lewis, P. R., Bahsoon, R., and Yao, X. (2014). Architecting self-aware software systems. In *2014 IEEE/IFIP Conference on Software Architecture*, pages 91–94.

Farouk, A. and Zhen, D. (2019). Big data analysis techniques for intelligent systems. *J. Intell. Fuzzy Syst.*, 37:3067–3071.

Fauci, A. S. et al. (2008). *Harrison's principles of internal medicine*, volume 2. McGraw-Hill, Medical Publishing Division.

Ferreiro, S., Konde, E., Fernández, S., and Prado, A. (2016). Industry 4.0: predictive intelligent maintenance for production equipment. In *European Conference of the Prognostics and Health Management Society, no*, pages 1–8.

Forooghifar, F., Aminifar, A., and Atienza, D. (2019). Resource-aware distributed epilepsy monitoring using self-awareness from edge to cloud. *IEEE Transactions on Biomedical Circuits and Systems*, 13(6):1338–1350.

Forooghifar, F. et al. (2018). Self-aware wearable systems in epileptic seizure detection. In *Euromicro Conference on Digital System Design (DSD)*.

Frelicot, C. and Dubuisson, B. (1993). An adaptive predictive diagnostic system based on fuzzy pattern recognition. *IFAC Proceedings Volumes*, 26(2):565–568.

Ganek, A. G. and Corbi, T. A. (2003). The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18.

Georgeff, M. P. and Ingrand, F. (1989). Decision-Making in an Embedded Reasoning System. In *International Joint Conference on Artificial Intelligence*, Detroit, United States.

Giese, H., Vogel, T., Diaconescu, A., Götz, S., Bencomo, N., Geihs, K., Kounev, S., and Bellman, K. L. (2017). State of the art in architectures for self-aware computing systems. In Kounev, S., Kephart, J. O., Milenkoski, A., and Zhu, X., editors, *Self-Aware Computing Systems*, pages 237–275. Springer International Publishing, Cham.

Giunchiglia, F. and Walsh, T. (1992). A theory of abstraction. *Artificial intelligence*, 57(2-3):323–389.

Goossens, P. (2017). Industry 4.0 and the power of the digital twin, adopt a systems approach to machine design and survive the next industrial revolution.

Goswami, K. and Iyer, R. (1993). Simulation of software behavior under hardware faults. In *FTCS-23 The Twenty-Third International Symposium on Fault-Tolerant Computing*, pages 218–227.

Götzinger, M., Anzanpour, A., Azimi, I., TaheriNejad, N., Jantsch, A., Rahmani, A. M., and Liljeberg, P. (2019a). Confidence-enhanced early warning score based on fuzzy logic. *Mobile Networks and Applications*, 8:1–18.

Götzinger, M., Anzanpour, A., Azimi, I., TaheriNejad, N., and Rahmani, A. M. (2017a). Enhancing the self-aware early warning score system through fuzzified data reliability assessment. In *Wireless Mobile Communication and Healthcare*, pages 3–11, Cham. EAI, Springer International Publishing.

Götzinger, M., Juhász, D., TaheriNejad, N., Willegger, E., Tutzer, B., Liljeberg, P., Jantsch, A., and Rahmani, A. M. (2020). Rosa: A framework for modeling self-awareness in cyber-physical systems. *IEEE Access*, 8:141373–141394.

Götzinger, M., TaheriNejad, N., Kholerdi, H. A., and Jantsch, A. (2017b). On the design of context-aware health monitoring without a priori knowledge; an ac-motor case-study. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–5. IEEE.

Götzinger, M., TaheriNejad, N., Kholerdi, H. A., Jantsch, A., Willegger, E., Glatzl, T., Rahmani, A. M., Sauter, T., and Liljeberg, P. (2019b). Model-free condition monitoring with confidence. *International Journal of Computer Integrated Manufacturing*, 32(4-5):466–481.

Götzinger, M., TaheriNejad, N., Rahmani, A. M., Liljeberg, P., Jantsch, A., and Tenhunen, H. (2016). Enhancing the early warning score system using data confidence. In *Wireless Mobile Communication and Healthcare*, pages 91–99, Cham. EAI, Springer International Publishing.

Götzinger, M., Willegger, E., TaheriNejad, N., Jantsch, A., Sauter, T., Glatzl, T., and Lilieberg, P. (2018). Applicability of context-aware health monitoring to hydraulic circuits. In *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, pages 4712–4719. IEEE.

Grieves, M. (2014). Digital twin: manufacturing excellence through virtual factory replication. *White paper*, 1:1–7.

Groarke, J., Gallagher, J., Stack, J., Aftab, A., Dwyer, C., McGovern, R., and Courtney, G. (2008). Use of an admission early warning score to predict patient morbidity and mortality and treatment success. *Emergency Medicine Journal*, 25(12):803–806.

Grüne, L. and Pannek, J. (2017). *Nonlinear Model Predictive Control : Theory and Algorithms. 2nd Edition*. Communications and Control Engineering. Springer, Cham, Switzerland.

Guang, L. (2012). *Hierarchical Agent-based Adaptation for Self-Aware Embedded Computing Systems*. PhD thesis, University of Turku, Finland.

Guang, L., Nigussie, E., Isoaho, J., Rantala, P., and Tenhunen, H. (2010). Interconnection alternatives for hierarchical monitoring communication in parallel socs. *Microprocessors and Microsystems*, 34(5):118 – 128. Special issue on selected papers from {NORCHIP} 2008.

Guang, L., Nigussie, E., Plosila, J., Isoaho, J., and Tenhunen, H. (2012). Survey of self-adaptive nocs with energy-efficiency and dependability. *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, 3(2):1–22.

Guo, Ying, P., Wall, Josh, P., Li, Jiaming, P., and West, S. (2013). Intelligent model based fault detection and diagnosis for hvac system using statistical machine learning methods. *ASHRAE Transactions*, 119:O1–O8. Name - International Energy Agency; Copyright - Copyright American Society of Heating, Refrigeration and Air Conditioning Engineers, Inc. 2013; Dokumentbestandteil - Illustrations; Graphs; Diagrams; ; Zuletzt aktualisiert - 2013-06-09; CODEN - ASHTAG.

Gupta, P., Agarwal, Y., Dolecek, L., Dutt, N., Gupta, R. K., Kumar, R., Mitra, S., Nicolau, A., Rosing, T. S., Srivastava, M. B., Swanson, S., and Sylvester, D. (2013). Underdesigned and opportunistic computing in presence of hardware variability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(1):8–23.

Gurgen, L., Gunalp, O., Benazzouz, Y., and Gallissot, M. (2013). Self-aware cyber-physical systems and applications in smart buildings and cities. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1149–1154.

Hafshejani, E. H., Elmi, M., TaheriNejad, N., Fotowat-Ahmady, A., and Mirabbasi, S. (2020). A low-power signal-dependent sampling technique: Analysis, implementation, and applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, pages 1–14.

Haghbayan, M., Miele, A., Rahmani, A. M., Liljeberg, P., and Tenhunen, H. (2016). A lifetime-aware runtime mapping approach for many-core systems in the dark silicon era. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 854–857.

Haghbayan, M.-H., Rahmani, A.-M., Liljeberg, P., Plosila, J., and Tenhunen, H. (2014). Online testing of many-core systems in the dark silicon era. In *17th International Symposium on Design and Diagnostics of Electronic Circuits Systems*, pages 141–146.

Haghbayan, M.-H., Rahmani, A.-M., Miele, A., Fattah, M., Plosila, J., Liljeberg, P., and Tenhunen, H. (2016). A power-aware approach for online test scheduling in many-core architectures. *IEEE Transactions on Computers*, 65(3):730–743.

Hardkernel (2017). ODROID-XU4 Manual. Technical report, Hardkernel.

Hatzipantelis, E. and Penman, J. (1993). The use of hidden markov models for condition monitoring electrical machines. In *1993 Sixth International Conference on Electrical Machines and Drives (Conf. Publ. No. 376)*, pages 91–96.

Hauth, J. (2008). *Grey-box modelling for nonlinear systems*. PhD thesis, Technical University of Kaiserslautern, Germany.

He, H., Menicucci, D., Caudell, T., and Mammoli, A. (2011). Real-time fault detection for solar hot water systems using adaptive resonance theory neural networks. In *Energy Sustainability*, volume 54686, pages 1059–1065.

Hewitt, C. (2017). Actor Model of Computation for Scalable Robust Information Systems. In *Symposium on Logic and Collaboration for Intelligent Applications,*.

Hoffmann, H., Maggio, M., Santambrogio, M. D., Leva, A., and Agarwal, A. (2010). SEEC: A framework for self-aware computing. Technical Report MIT-CSAIL-TR-2010-049, MIT.

Holbery, N. and Newcombe, P. (2016). *Emergency Nursing at a Glance*. Wiley.

Hou, Z., Lian, Z., Yao, Y., and Yuan, X. (2006). Data mining based sensor fault diagnosis and validation for building air conditioning system. *Energy Conversion and Management*, 47(15-16):2479–2490.

Huebscher, M. C. and McCann, J. A. (2008). A survey of autonomic computing-degrees, models, and applications. *ACM Comput. Surv.*, 40(3).

Hung, M. (2017). Leading the iot, gartner insights on how to lead in a connected world. *Gartner Research*, pages 1–29.

Hunt, J. (2014). *Introduction to Akka Actors*, pages 383–398. Springer International Publishing.

IBM Corporation (2006). An architectural blueprint for autonomic computing. IBM White Paper.

ITRS (2009). The international technology roadmap for semiconductors. Technical report, ITRS.

Jafri, S. M., Guang, L., Jantsch, A., Paul, K., Hemani, A., and Tenhunen, H. (2012). Self-adaptive noc power management with dual-level agents-architecture and implementation. In *PECCS*, pages 450–458.

Jantsch, A., Anzanpour, A., Kholerdi, H., Azimi, I., Siafara, L. C., Rahmani, A. M., TaheriNejad, N., Liljeberg, P., and Dutt, N. (2018). Hierarchical dynamic goal management for IoT systems. In *2018 19th International Symposium on Quality Electronic Design (ISQED)*, pages 370–375.

Jantsch, A., Dutt, N., and Rahmani, A. M. (2017). Self-awareness in systems on chip— a survey. *IEEE Design Test*, 34(6):8–26.

Jantsch, A. and Tammemäe, K. (2014). A framework of awareness for artificial subjects. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, pages 1–3. ACM.

Jennings, B. and Stadler, R. (2015). Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management*, 23(3):567–619.

Jie Han, Gao, J., Jonker, P., Yan Qi, and Fortes, J. A. B. (2005). Toward hardware-redundant, fault-tolerant logic for nanoelectronics. *IEEE Design Test of Computers*, 22(4):328–339.

Kaliorakis, M., Psarakis, M., Foutris, N., and Gizopoulos, D. (2014). Accelerated online error detection in many-core microprocessor architectures. In *2014 IEEE 32nd VLSI Test Symposium (VTS)*, pages 1–6.

Kang, H., Cheng, J., Kim, I., and Vachtsevanos, G. (1991). An application of fuzzy logic and dempster-shafer theory to failure detection and identification. In *[1991] Proceedings of the 30th IEEE Conference on Decision and Control*, pages 1555–1560. IEEE.

Katipamula, S. and Brambley, M. (2005). Methods for fault detection, diagnostics, and prognostics for building systems - a review, part i. *HVAC&R Research*, 11(1):1–24.

Kephart, J. O. (2005). Research challenges of autonomic computing. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pages 15–22.

Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

Kholerdi, H. A., TaheriNejad, N., and Jantsch, A. (2018). Enhancement of classification of small data sets using self-awareness - an iris flower case-study. In *To be published in the proceedings of the International Symposium on Circuit and Systems (ISCAS)*, Florence, Italy.

Kim, H., Kang, J., and Park, J. H. (2017). A light-weight secure information transmission and device control scheme in integration of cps and cloud computing. *Microprocessors and Microsystems*, 52:416–426.

Kitajima, M. and Sakurai, A. (2019). Colmina: manufacturing digital place to create new value in manufacturing industry. *Fujitsu Scientific and Technical Journal*, 55(1):14–19.

Kong, J., Chung, S. W., and Skadron, K. (2012). Recent thermal management techniques for microprocessors. *ACM Computing Surveys (CSUR)*, 44(3):13.

Kornaros, G. and Pnevmatikatos, D. (2013). A survey and taxonomy of on-chip monitoring of multicore systems-on-chip. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 18(2):1–38.

Kounev, S., Peter Lewis, Bellman, K., Bencomo, N., Camara, J., Diaconescu, A., Esterle, L., Geihs, K., Giese, H., Götz, S., Inverardi, P., Kephart, J., and Zisman, A. (2017). The notion of self-aware computing. In Kounev, S., Kephart, J. O., Milenkoski, A., and Zhu, X., editors, *Self-Aware Computing Systems*, pages 3–16. Springer.

Kounev, S., Zhu, X., Kephart, J. O., and Kwiatkowska, M. (2015). Model-driven Algorithms and Architectures for Self-Aware Computing Systems (Dagstuhl Seminar 15041). *Dagstuhl Reports*, 5(1):164–196.

Kramer, J. and Magee, J. (2007). Self-managed systems: an architectural challenge. In *Future of Software Engineering (FOSE '07)*, pages 259–268.

Kyriacos, U., Jelsma, J., James, M., and Jordan, S. (2014). Monitoring vital signs: development of a modified early warning scoring (mews) system for general wards in a developing country. *PloS one*, 9(1):e87073.

Kyriacos, U., Jelsma, J., and Jordan, S. (2011). Monitoring vital signs using early warning scoring systems: a review of the literature. *Journal of nursing management*, 19(3):311–330.

Lalanda, P., McCann, J. A., and Diaconescu, A. (2013). *Autonomic computing: principles, design and implementation*. Springer Science & Business Media.

Landauer, C. and Bellman, K. L. (2017). An architecture for self-awareness experiments. In *2017 IEEE International Conference on Autonomic Computing (ICAC)*, pages 255–262.

Laprie, J.-C. (1992). Dependability: Basic concepts and terminology. In *Dependability: Basic Concepts and Terminology*, pages 3–245. Springer.

Lee, E. A. (2006). Cyber-physical systems-are computing foundations adequate. In *Position paper for NSF workshop on cyber-physical systems: research motivation, techniques and roadmap*, volume 2, pages 1–9. Citeseer.

Lee, J., Bagheri, B., and Kao, H.-A. (2015). A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, 3:18–23.

Leveson, N., Cha, S., and Shimeall, T. (1991). Safety verification of ada programs using software fault trees. *IEEE Software*, 8(4):48–59.

Lewis, P. R., Chandra, A., Faniyi, F., Glette, K., Chen, T., Bahsoon, R., Torresen, J., and Yao, X. (2015). Architectural aspects of self-aware and self-expressive computing systems: From psychology to engineering. *Computer*, 48(8):62–70.

Lewis, P. R., Chandra, A., Parsons, S., Robinson, E., Glette, K., Bahsoon, R., Torresen, J., and Yao, X. (2011). A survey of self-awareness and its application in computing systems. In *2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 102–107.

Lewis, P. R., Platzner, M., Rinner, B., Tørresen, J., and Yao, X. (2016). *Self-aware Computing Systems: An Engineering Approach*. Natural Computing Series. Springer International Publishing : Imprint: Springer, Cham, 1st ed. 2016. edition.

Li, S. and Wen, J. (2014). Application of pattern matching method for detecting faults in air handling unit system. *Automation in Construction*, 43:49–58.

Lidberg, M. and Müller, S. (2019). Special issue on motion control for automated driving and autonomous functions on road vehicles.

Liggesmeyer, P. and Trapp, M. (2009). Trends in embedded software engineering. *IEEE Software*, 26(3):19–25.

Lim, K. Y. H., Zheng, P., and Chen, C.-H. (2020). A state-of-the-art survey of digital twin: techniques, engineering product lifecycle management and business innovation perspectives. *Journal of Intelligent Manufacturing*, 31(6):1313–1337.

Liu, J., McKenna, T. M., Gribok, A., Beidleman, B. A., Tharion, W. J., and Reifman, J. (2008). A fuzzy logic algorithm to assign confidence levels to heart and respiratory rate time series. *Physiological Measurement*, 29(1):81–94.

Lunze, J. (2016). *Künstliche Intelligenz für Ingenieure: Methoden zur Lösung ingenieurtechnischer Probleme mit Hilfe von Regeln, logischen Formeln und Bayesnetzen*. De Gruyter Studium. De Gruyter Oldenbourg, Berlin, 3 edition.

Macarulla, M., Casals, M., Forcada, N., and Gangolells, M. (2021). *Use of Grey-Box Modeling to Determine the Air Ventilation Flows in a Room*, pages 449–461.

Maimon, O., Kandel, A., and Last, M. (2001). Information-theoretic fuzzy approach to data reliability and data mining. *Fuzzy Sets and Systems*, 117(2):183–194.

Marcu, T. and Voicu, M. (1992). Fuzzy weighted clustering and classification for process fault detection. In *Proceedings of the IFACIIFIPSIIMACS international symposium on AI in real-time control, Delft, Netherlands*, pages 275–280.

Marquet, F., Company, O., Krut, S., and Pierrot, F. (2002). Enhancing parallel robots accuracy with redundant sensors. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 4, pages 4114–4119 vol.4.

Massano, M., Macii, E., Patti, E., Acquaviva, A., and Bottaccioli, L. (2019a). A grey-box model based on unscented kalman filter to estimate thermal dynamics in buildings. In *2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe)*, pages 1–6.

Massano, M., Macii, E., Patti, E., Acquaviva, A., and Bottaccioli, L. (2019b). A grey-box model based on unscented kalman filter to estimate thermal dynamics in buildings. In *2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe)*, pages 1–6.

Matyi, H., Veres, P., Bányai, T., Demin, V., and Tamás, P. (2020). Digitalization in industry 4.0: the role of mobile devices. *Journal of Production Engineering*, 23(1):75–78.

McCullough, L. and Arora, S. (2004). Diagnosis and treatment of hypothermia. *American family physician*, 70(12):2325–2332.

McGaughey, J., Alderdice, F., Fowler, R., Kapila, A., Mayhew, A., and Moutray, M. (2007). Outreach and early warning systems (ews) for the prevention of intensive care admission

and death of critically ill adult patients on general hospital wards. *Cochrane Database of Systematic Reviews*, 3.

Mercati, P., Bartolini, A., Paterna, F., Rosing, T. S., and Benini, L. (2014). A linux-governor based dynamic reliability manager for android mobile devices. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–4.

Miorandi, D., Sicari, S., De Pellegrini, F., and Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad hoc networks*, 10(7):1497–1516.

Morgan, R., Williams, F., and Wright, M. (1997). An early warning score for the early detection of patients with impending illness. *Clin Intensive Care*, 8:100.

Mozelli, A., Nejad, N. T., and Jantsch, A. (2021). A study on confidence: an unsupervised multi-agent machine learning experiment. *IEEE Design Test*, pages 1–1.

Najafi, M., Auslander, D. M., Haves, P., and Sohn, M. D. (2012). A statistical pattern analysis framework for rooftop unit diagnostics. *HVAC&R Research*, pages 406–416.

National Clinical Effectiveness Committee et al. (2013). National early warning score national clinical guideline no. 1.

Nejjari, H. and Benbouzid, M. E. H. (2000). Monitoring and diagnosis of induction motors electrical faults using a current park's vector pattern learning approach. *IEEE Transactions on Industry Applications*, 36(3):730–735.

Nguyen, T. A., Aiello, M., Yonezawa, T., and Tei, K. (2015). A self-healing framework for online sensor data. In *2015 IEEE International Conference on Autonomic Computing*, pages 295–300.

North, M. J., Collier, N. T., Ozik, J., Tatara, E. R., Macal, C. M., Bragen, M., and Sydelko, P. (2013). Complex adaptive systems modeling with Repast Simphony. *Complex Adaptive Systems Modeling*, 1(1):3.

Nymoen, K., Chandra, A., and Torresen, J. (2016). Self-awareness in active music systems. In Lewis, P. R., Platzner, M., Rinner, B., Tørresen, J., and Yao, X., editors, *Self-aware Computing Systems: An Engineering Approach*, pages 279–296. Springer International Publishing, Cham.

Oberhammer, R., Beikircher, W., Hörmann, C., Lorenz, I., Pycha, R., Adler-Kastner, L., and Brugger, H. (2008). Full recovery of an avalanche victim with profound hypothermia and prolonged cardiac arrest treated by extracorporeal re-warming. *Resuscitation*, 76(3):474 – 480.

Omics International (2016). List of weather records. http://http://research.omicsgroup.org/index.php/List_of_weather_records.

Osaki, S. and Nakagawa, T. (1971). On a two-unit standby redundant system with standby failure. *Operations Research*, 19(2):510–523.

Ossamah, A., Meshari, A., Yazed, A., and Norah, A. (2020). Cloud based cyber physical system for factory automation. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pages 1–7.

Oyewumi, I. A., Jillepalli, A. A., Richardson, P., Ashrafuzzaman, M., Johnson, B. K., Chakhchoukh, Y., Haney, M. A., Sheldon, F. T., and de Leon, D. C. (2019). Isaac: The idaho cps smart grid cybersecurity testbed. In *2019 IEEE Texas Power and Energy Conference (TPEC)*, pages 1–6.

Palm, R. (2007). Multiple-step-ahead prediction in control systems with gaussian process models and ts-fuzzy models. *Engineering Applications of Artificial Intelligence*, 20(8):1023–1035.

Papadopoulos, T., Singh, S. P., Spanaki, K., Gunasekaran, A., and Dubey, R. (2021). Towards the next generation of manufacturing: implications of big data and digitalization in the context of industry 4.0. *Production Planning & Control*, 0(0):1–4.

Parashar, M. and Hariri, S. (2005). Autonomic computing: An overview. In Banâtre, J.-P., Fradet, P., Giavitto, J.-L., and Michel, O., editors, *Unconventional Programming Paradigms*, pages 257–269, Berlin, Heidelberg. Springer Berlin Heidelberg.

Parasuraman, R., Sheridan, T., and Wickens, C. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 30(3):286–297.

Parego, P., Rahmani, A. M., and TaheriNejad, N. (2017). *Wireless Communication and Mobile Healthcare*. ecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer International Publishing.

Pasquier, M., Moix, P.-A., Delay, D., and Hugli, O. (2015). Cooling rate of 9.4 °C in an hour in an avalanche victim . *Resuscitation*, 93:e17 – e18.

Platzer, A. (2019). The logical path to autonomous cyber-physical systems. In Parker, D. and Wolf, V., editors, *Quantitative Evaluation of Systems*, pages 25–33, Cham. Springer International Publishing.

Pollreisz, D. and TaheriNejad, N. (2019). Detection and removal of motion artifacts in ppg signals. *Mobile Networks and Applications*, pages 1–11.

Pollreisz, D. and TaheriNejad, N. (2020a). Efficient respiratory rate extraction on a smartwatch. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 5988–5991.

Pollreisz, D. and TaheriNejad, N. (2020b). Reliable respiratory rate extraction using ppg. In *2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS)*, pages 1–4.

Postolache, O., Dias Pereira, J. M., Cretu, M., and Girao, P. S. (1998). An ann fault detection procedure applied in virtual measurement systems case. In *IMTC/98 Conference Proceedings. IEEE Instrumentation and Measurement Technology Conference. Where Instrumentation is Going (Cat. No.98CH36222)*, volume 1, pages 257–260 vol.1.

Preden, J. S., Tammemäe, K., Jantsch, A., Leier, M., Riid, A., and Calis, E. (2015). The benefits of self-awareness and attention in fog and mist computing. *Computer*, 48(7):37–45.

Psaier, H. and Dustdar, S. (2011). A survey on self-healing systems: approaches and systems. *Computing*, 91(1):43–73.

Pulido, B., Zamarreño, J. M., Merino, A., and Bregon, A. (2019). State space neural networks and model-decomposition methods for fault diagnosis of complex industrial systems. *Engineering Applications of Artificial Intelligence*, 79:67–86.

Putzer, G., Schmid, S., Braun, P., Brugger, H., and Paal, P. (2010). Cooling of six centigrades in an hour during avalanche burial. *Resuscitation*, 81:1043 – 1044.

Qi, Q. and Tao, F. (2018). Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison. *Ieee Access*, 6:3585–3593.

Qiu, J., Wu, Q., Ding, G., Xu, Y., and Feng, S. (2016). A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, 2016(1):1–16.

Rahman, M., Ranjan, R., Buyya, R., and Benatallah, B. (2011). A taxonomy and survey on autonomic management of applications in grid computing environments. *Concurrency and Computation: Practice and Experience*, 23(16):1990–2019.

Rahmani, A. M., Jantsch, A., and Dutt, N. (2018). Hdgm: Hierarchical dynamic goal management for many-core resource allocation. *IEEE Embedded Systems letters*, 10(3):61–64.

Randall, A. L. and Walter, R. C. (2003). Overview of the small unit operations situational awareness system. In *IEEE Military Communications Conference, 2003. MILCOM 2003.*, volume 1, pages 169–173 Vol.1.

Raspberry Pi (Trading) Ltd. (2019). Raspberry Pi Compute Module 3+ Datasheet. Technical report, Raspberry Pi (Trading) Ltd.

Reule, S. and Drawz, P. (2012). Heart rate and blood pressure: Any possible implications for management of hypertension? *Curr Hypertens Rep*, 14(6):478–84.

Rinner, B., Esterle, L., Simonjan, J., Nebehay, G., Pflugfelder, R., Fernández Domínguez, G., and Lewis, P. R. (2015). Self-aware and self-expressive camera networks. *Computer*, 48(7):21–28.

Rivera, J. and van der Meulen, R. (November 2014). Gartner says 4.9 billion connected 'things' will be in use in 2015. *Press release*.

Romero-Silva, R. and Hernández-López, G. (2020). Shop-floor scheduling as a competitive advantage: A study on the relevance of cyber-physical systems in different manufacturing contexts. *International Journal of Production Economics*, 224:107555.

Ross, T. J. (2009). *Fuzzy logic with engineering applications*. John Wiley & Sons.

Royal College of Physicians (London: RCP, 2017). National early warning score (news) 2: Standardising the assessment of acute-illness severity in the nhs. updated report of a working party.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach (3rd Edition)*. Pearson.

Sadighi, A., Donyanavard, B., Kadeed, T., Moazzemi, K., Mück, T., Nassar, A., Rahmani, A. M., Wild, T., Dutt, N., Ernst, R., Herkersdorf, A., and Kurdahi, F. (2018). Design methodologies for enabling self-awareness in autonomous systems. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1532–1537.

Salehie, M. and Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):14:1–14:42.

Salvia, G., Cooper, T., Fisher, T., Harmer, L., and Barr, C. (2015). What is broken? expected lifetime, perception of brokenness and attitude towards maintenance and repair.

Salzmann, C., Gillet, D., and Huguenin, P. (2000). Introduction to real-time control using labviewtm with an application to distance learning. *Int. J. Engng Ed*, 16(5):372–384.

Samet, L., Masmoudi, N., Kharrat, M., and Kamoun, L. (1998). A digital pid controller for real time and multi loop control: a comparative study. In *1998 IEEE International Conference on Electronics, Circuits and Systems. Surfing the Waves of Science and Technology (Cat. No.98EX196)*, volume 1, pages 291–296 vol.1.

Sangwan, R. S., Vercellone-Smith, P., and Laplante, P. A. (2008). Structural epochs in the complexity of software over time. *IEEE Software*, 25(4):66–73.

Sauter, D., Mary, N., and Sirou, F. (1994). Fault diagnosis in systems using fuzzy logic. In *1994 Proceedings of IEEE International Conference on Control and Applications*, pages 883–888 vol.2.

Savaglio, C., Fortino, G., and Zhou, M. (2016). Towards interoperable, cognitive and autonomic IoT systems: An agent-based approach. In *2016 IEEE 3rd World Forum Internet Things*, pages 58–63. IEEE.

Schlingensiepen, J., Nemtanu, F., Mehmood, R., and McCluskey, L. (2016). Autonomic transport management systems — enabler for smart cities, personalized medicine, participation and industry grid/industry 4.0. In *Intelligent transportation systems–problems and perspectives*, pages 3–35. Springer.

Selcuk, S. (2017). Predictive maintenance, its implementation and latest trends. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 231(9):1670–1679.

Seshia, S. A., Sadigh, D., and Sastry, S. S. (2015). Formal methods for semi-autonomous driving. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–5.

Sha, L., Gopalakrishnan, S., Liu, X., and Wang, Q. (2008). Cyber-physical systems: A new frontier. In *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008)*, pages 1–9.

Shamsa, E., Kanduri, A., TaheriNejad, N., Proebstl, A., Chakraborty, Rahmani, A. M., and Liljeberg, P. (2020). User-centric resource management for embedded multi-core

processors. In *The 33rd International Conference on VLSI Design and The 19th International Conference on Embedded Design*, pages 1–6.

Sharifi, R. and B., D. (2011). Fault detection in lighting systems - first phase results. Technical note, Philips Research North-America.

Siafara, L. C., Kholerdi, H., Bratukhin, A., TaheriNejad, N., and Jantsch, A. (2018). SAMBA – an architecture for adaptive cognitive control of distributed cyber-physical production systems based on its self-awareness. *e & i Elektrotechnik und Informationstechnik*, 135(3):270–277.

Siafara, L. C., Kholerdi, H. A., Bratukhin, A., TaheriNejad, N., Wendt, A., Jantsch, A., Treytl, A., and Sauter, T. (2017). SAMBA: a self-aware health monitoring architecture for distributed industrial systems. In *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 3512–3517.

Siegel, H. J., Khemka, B., Friese, R., Pasricha, S., Maciejewski, A. A., Koenig, G. A., Powers, S., Hilton, M., Rambharos, R., Okonski, G., and Poole, S. (2014). Energy-aware resource management for computing systems. In *2014 Seventh International Conference on Contemporary Computing (IC3)*, pages 7–12.

Silva, K. M., Souza, B. A., and Brito, N. S. D. (2006). Fault detection and classification in transmission lines based on wavelet transform and ann. *IEEE Transactions on Power Delivery*, 21(4):2058–2063.

Smith, G. B., Prytherch, D. R., Meredith, P., Schmidt, P. E., and Featherstone, P. I. (2013). The ability of the national early warning score (news) to discriminate patients at risk of early cardiac arrest, unanticipated intensive care unit admission, and death. *Resuscitation*, 84(4):465–470.

Song, H. and Lehrer, P. (2003). The effects of specific respiratory rates on heart rate and heart rate variability. *Appl Psychophysiol Biofeedback*, 28(1):13–23.

Sonnenfeld, G., Goebel, K., and Celaya, J. R. (2008). An agile accelerated aging, characterization and scenario simulation system for gate controlled power transistors. In *2008 IEEE AUTOTESTCON*, pages 208–215.

Spathis, P. and Bicudo, M. (2010). ANA: Autonomic Network Architecture. In *Autonomic Network Management Principles: From Concepts to Applications*, pages 49–65. Academic Press.

Srivastav, A., Tewari, A., and Dong, B. (2013). Baseline building energy modeling and localized uncertainty quantification using gaussian mixture models. *Energy and Buildings*, 65:438–447.

Stankovic, J., Lee, I., Mok, A., and Rajkumar, R. (2005). Opportunities and obligations for physical computing systems. *Computer*, 38(11):23–31.

Stankovic, J., Lu, C., Son, S., and Tao, G. (1999). The case for feedback control real-time scheduling. In *Proceedings of 11th Euromicro Conference on Real-Time Systems. Euromicro RTS'99*, pages 11–20.

Strassner, J., Kim, S.-S., and Hong, J. W.-K. (2009). The design of an autonomic communication element to manage future internet services. In *Management Enabling the Future Internet for Changing Business and New Computing Services*, pages 122–132. Springer.

Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., and Beghi, A. (2015). Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3):812–820.

TaheriNejad, N. (2019). Wearable medical devices: Challenges and self-aware solutions. In *IEEE Life Sciences Newsletter*, pages 5–6.

TaheriNejad, N. and Jantsch, A. (2019). Improved machine learning using confidence. In *IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, Edmonton, Canada.

TaheriNejad, N., Jantsch, A., and Pollreisz, D. (2016). Comprehensive observation and its role in self-awareness; an emotion recognition system example. In *Proceedings of the Federated Conference on Computer Science and Information Systems*, pages 117–124, Gdansk, Poland.

TaheriNejad, N., Shami, M. A., and Manoj, P. (2017). Self-aware sensing and attention-based data collection in multi-processor system-on-chips. In *2017 15th IEEE International New Circuits and Systems Conference (NEWCAS)*, pages 81–84.

Taillandier, P., Gaudou, B., Grignard, A., Huynh, Q.-N., Marilleau, N., Caillou, P., Philippon, D., and Drogoul, A. (2019). Building, composing and experimenting complex spatial models with the GAMA platform. *GeoInformatica*, 23(2):299–322.

Tanigawa, I., Hisazumi, K., Ogura, N., Sugaya, M., Watanabe, H., and Fukuda, A. (2019). Rtcop: Context-oriented programming framework based on c++ for application in embedded software. In *Proceedings of the 2019 2nd International Conference on Information Science and Systems*, ICISS 2019, page 65–72, New York, NY, USA. Association for Computing Machinery.

Tao, F. and Qi, Q. (2019). Make more digital twins.

Tao, F., Qi, Q., Wang, L., and Nee, A. (2019). Digital twins and cyber–physical systems toward smart manufacturing and industry 4.0: Correlation and comparison. *Engineering*, 5(4):653–661.

Tao, X., Broo, D. G., Törngren, M., and Chen, D. (2020). Uncertainty management in situation awareness for cyber-physical systems: State of the art and challenge. In *Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence*, pages 424–430.

Tavčar, J. and Horváth, I. (2019). A review of the principles of designing smart cyber-physical systems for run-time adaptation: Learned lessons and open issues. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(1):145–158.

Taylor, B. N., Kuyatt, C. E., et al. (1994). Guidelines for evaluating and expressing the uncertainty of nist measurement results.

Taylor, M. B. (2012). Is dark silicon useful? harnessing the four horsemen of the coming dark silicon apocalypse. In *DAC Design Automation Conference 2012*, pages 1131–1136. IEEE.

Teich, J., Henkel, J., Herkersdorf, A., Schmitt-Landsiedel, D., Schröder-Preikschat, W., and Snelting, G. (2011). Invasive computing: An overview. In Hübner, M. and Becker, J., editors, *Multiprocessor System-on-Chip – Hardware Design and Tool Integration*, pages 241–268. Springer, Berlin, Heidelberg.

Tennenhouse, D. (2000). Proactive computing. *Commun. ACM*, 43(5):43–50.

Thomson, W. and Gilmore, R. (2003). Motor current signature analysis to detect faults in induction motor drives–fundamentals, data interpretation, and industrial case histories. pages 145–156.

Törngren, M., Zhang, X., Mohan, N., Becker, M., Svensson, L., Tao, X., Chen, D.-J., and Westman, J. (2018). Architecting safety supervisors for high levels of automated driving. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1721–1728. IEEE.

TRNSYS (2019). Transient system simulation tool. http://www.trnsys.com.

Törngren, M. and Grogan, P. T. (2018). How to deal with the complexity of future cyber-physical systems? *Designs*, 2(4).

Urban, R. W., Mumba, M., Martin, S. D., Glowicz, J., and Cipher, D. J. (2015). Modified early warning system as a predictor for hospital admissions and previous visits in emergency departments. *Advanced emergency nursing journal*, 37(4):281–289.

Van Harmelen, F., Lifschitz, V., and Porter, B. (2008). *Handbook of knowledge representation*. Elsevier.

Varshney, K. R. and Alemzadeh, H. (2017). On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big data*, 5(3):246–255.

Viroli, M., Pianini, D., Montagna, S., and Stevenson, G. (2012). Pervasive Ecosystems: A Coordination Model Based on Semantic Chemistry. In *Proc. 27th Annu. ACM Symp. Appl. Comput.*, pages 295–302. ACM Press.

Wang, E. K., Ye, Y., Xu, X., Yiu, S. M., Hui, L. C. K., and Chow, K. P. (2010). Security issues and challenges for cyber physical system. In *2010 IEEE/ACM Int'l Conference on Green Computing and Communications Int'l Conference on Cyber, Physical and Social Computing*, pages 733–738.

Wang, G.-G., Cai, X., Cui, Z., Min, G., and Chen, J. (2020). High performance computing for cyber physical social systems by using evolutionary multi-objective optimization algorithm. *IEEE Transactions on Emerging Topics in Computing*, 8(1):20–30.

Wang, P. (1995). On the working definition of intelligence. Technical report, Citeseer.

Wanner, L., Elmalaki, S., Lai, L., Gupta, P., and Srivastava, M. (2013). VarEMU: An emulation testbed for variability-aware software. In *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2013 International Conference on*, pages 1–10.

Weyer, E. and Hangos, K. (1997). Grey box fault detection in heat exchanger networks. *IFAC Proceedings Volumes*, 30(18):191–196.

WHO (2017). Chronic diseases and health promotion. http://www.who.int/chp/en/.

Winter, J. A., Albonesi, D. H., and Shoemaker, C. A. (2010). Scalable thread scheduling and global power management for heterogeneous many-core architectures. In *2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 29–39.

Wolf, M., Keel, M., von Siebenthal, K., Bucher, H.-U., Geering, K., Lehareinger, Y., and Niederer, P. (1996). Improved monitoring of preterm infants by fuzzy logic. *Technology and health care*, 4(2):193–201.

Woohyun, K. and Srinivas, K. (2018). A review of fault detection and diagnostics methods for building systems. *Science and Technology for the Built Environment*, 24:3–21.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152.

Yang, Y.-W. and Shu-Min Li, K. (2009). Temperature-aware dynamic frequency and voltage scaling for reliability and yield enhancement. In *2009 Asia and South Pacific Design Automation Conference*, pages 49–54.

Yu, H., Chu, C., Châtelet, È., and Yalaoui, F. (2007). Reliability optimization of a redundant system with failure dependencies. *Reliability Engineering & System Safety*, 92(12):1627–1634.

Zhang, Y., Qiu, M., Tsai, C.-W., Hassan, M. M., and Alamri, A. (2017). Health-cps: Healthcare cyber-physical system assisted by cloud and big data. *IEEE Systems Journal*, 11(1):88–95.

Zhao, H., Sun, D., Yue, H., Zhao, M., and Cheng, S. (2017). Using cstpns to model traffic control cps. *IET Software*, 11(3):116–125.

Zhuge, H. and Xing, Y. (2012). Probabilistic resource space model for managing resources in cyber-physical society. *IEEE Transactions on Services Computing*, 5(3):404–421.

Zila, I. and Calkovska, A. (2011). Effects of elevated body temperature on control of breathing. *Acta Medica Martiniana*, 11.

List of Figures

List of Tables

Original Publications

Maximilian Götzinger, Dávid Juhász, Nima TaheriNejad, Edwin Willegger, Benedikt Tutzer, Pasi Liljeberg, Axel Jantsch, and Amir M. Rahmani

RoSA: A Framework for Modeling Self-Awareness in Cyber-Physical Systems

IEEE Access, 2020; 8: 141373–141394

I

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 28, 2020, accepted July 8, 2020, date of publication July 29, 2020, date of current version August 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3012824

RoSA: A Framework for Modeling Self-Awareness in Cyber-Physical Systems

MAXIMILIAN GÖTZINGER[1,2], (Member, IEEE), DÁVID JUHÁSZ[2,5], NIMA TAHERINEJAD[2], (Member, IEEE), EDWIN WILLEGGER[2], BENEDIKT TUTZER[2], PASI LILJEBERG[1], (Member, IEEE), AXEL JANTSCH[2], (Senior Member, IEEE), AND AMIR M. RAHMANI[3,4], (Senior Member, IEEE)

[1]Department of Future Technologies, University of Turku, 20014 Turku, Finland
[2]Institute of Computer Technology, TU Wien, 1040 Vienna, Austria
[3]Department of Computer Science, University of California at Irvine, Irvine, CA 92697, USA
[4]School of Nursing, University of California at Irvine, Irvine, CA 92697, USA
[5]Imsys AB, 194 61 Stockholm, Sweden

Corresponding authors: Maximilian Götzinger (maxgot@utu.fi) and Dávid Juhász (david.juhasz@tuwien.ac.at)

*The first two authors contributed equally to the paper.

This work was supported in part by Federal Ministry Republic of Austria for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMVIT)/Austrian Research Promotion Agency (FFG) under the program Production of the Future in the project SAVE under Grant FFG 864883; in part by the European Union's Horizon 2020 Framework Programme for Research and Innovation under Grant 674875 (oCPS Marie Curie Network); and in part by the Tekniikan Edistämissäätiö (Finnish Foundation for Technology Promotion).

ABSTRACT The role of smart and autonomous systems is becoming vital in many areas of industry and society. Expectations from such systems continuously rise and become more ambitious: long lifetime, high reliability, high performance, energy efficiency, and adaptability, particularly in the presence of changing environments. Computational self-awareness promises a comprehensive assessment of the system state for sensible and well-informed actions and resource management. Computational self-awareness concepts can be used in many applications such as automated manufacturing plants, telecommunication systems, autonomous driving, traffic control, smart grids, and wearable health monitoring systems. Developing self-aware systems from scratch for each application is the most common practice currently, but this is highly redundant, inefficient, and uneconomic. Hence, we propose a framework that supports modeling and evaluation of various self-aware concepts in hierarchical agent systems, where agents are made up of self-aware functionalities. This paper presents the Research on Self-Awareness (RoSA) framework and its design principles. In addition, self-aware functionalities abstraction, data reliability, and confidence, which are currently provided by RoSA, are described. Potential use cases of RoSA are discussed. Capabilities of the proposed framework are showcased by case studies from the fields of healthcare and industrial monitoring. We believe that RoSA is capable of serving as a common framework for self-aware modeling and applications and thus helps researchers and engineers in exploring the vast design space of hierarchical agent-based systems with computational self-awareness.

INDEX TERMS Computational self-awareness, framework, agent-based, hierarchical, modeling, development, monitoring, observe-decide-act.

I. INTRODUCTION

The number of Cyber-Physical Systems (CPSs) with embedded sensors and actuators is growing exponentially [1], [2]. These systems enable a wide range of applications like automated manufacturing plants [3], telecommunication systems [4], [5], autonomous driving [6], traffic control [7], smart grids [8], and mobile health monitoring systems [9] — just to name a few examples. No matter the actual application, CPSs connect their physical environment (the real world) with the digital (i.e., cyber) space under ever-increasing expectations and requirements [10]. Some system properties that are needed for meeting application requirements are adaptivity, autonomy, reliability, robustness, long lifetime, high performance, and energy efficiency. A controlled balance among those sometimes contradictory properties is a must as well [11].

Because of these requirements, a complex interaction between a CPS and its environment is necessary. The system needs to know how its environment behaves and how its

The associate editor coordinating the review of this manuscript and approving it for publication was Mark Kok Yew Ng.

own actions may affect the environment. Besides, a CPS may have limited resources and need to consider system properties like the growing process variability, thermal limitations, and wear-out effects of System on Chip (SoC) solutions. These could lead to an unbalanced lifetime, overheating, hotspots, rapid aging, and under-utilization [12]–[14], which requires sophisticated resource management. Thus, a comprehensive assessment of the system's state and that of its environment is needed and allows prediction of future events, better planning of actions, and hence optimized operation [15].

Computational Self-Awareness (CSA) has been studied in a wide range of applications [16]–[18], and proved to be a key enabler of efficient resource management in different domains (e.g., sensor networks [19] and health monitoring systems [20]). It has also been proposed to tackle the challenges of comprehensive assessment in different CPSs [18], [21]–[24]. However, the community researching on self-awareness is fractioned, and research proceeds rather slow. To the best of our knowledge, so far, there is no satisfactory common tool to speed up research on self-awareness. We propose a software framework, *Research on Self-Awareness (RoSA)*,[1] for modeling self-awareness concepts and applications. RoSA is based on a hierarchical agent-based model and provides facilities to implement, adapt, customize, and evaluate self-aware applications. The framework itself is a three-fold software engineering exemplar [25]: it can be used in the engineering process to model applications as well as it serves as a testbed and library (i.e. infrastructure for conducting research and a set of reusable models or code, respectively). We hope that RoSA can serve as a common framework for the community to explore uncharted aspects of self-awareness and speed up development in the field.

This paper provides an overview of the framework, how it works and how it can be used. The applicability and flexibility of RoSA are demonstrated by two case studies from the fields of human health monitoring [26]–[28] and industrial machine monitoring [29]–[31]. The main contributions of the paper are:

1) we propose a framework, RoSA, which facilitates the modeling and evaluation of self-awareness concepts by means of modeling self-aware applications as *hierarchical agent systems* and modeling agents based on self-aware *functionalities*;
2) we provide an initial set of self-aware *functionalities*, namely abstraction, data reliability, and confidence, implemented in RoSA; and
3) we describe use cases for RoSA-based modeling, whose utility has been proven by our case studies.

The rest of the paper is organized as follows. Section II highlights the motivation and research challenges of this study. Section III then summarizes the state of the art in the field of CSA with interest in modeling and implementation frameworks. Section IV introduces terminology as well as the architecture and implementation of RoSA. Possible use cases of the framework are discussed in Section V. Self-aware functionalities that are currently available in RoSA are described in Section VI, whereas Section VII presents case studies, which use those functionalities and general RoSA facilities. Section VIII discusses which lessons have been learned while developing this framework, and finally, Section IX concludes the paper. We include a list of abbreviations at the end of the paper for the reader's reference.

[1]Open-source implementation is available at `https://phabricator.ict.tuwien.ac.at/source/SoC_Rosa_repo.git`.

II. MOTIVATION AND RESEARCH CHALLENGES

CSA is a hot topic, and some approaches that make CPSs intelligent exist [32], [33]. Still, the field is widely unexplored, and many aspects of self-awareness are yet to be researched.

While studying open questions of self-awareness (case studies in Section VII), we made an effort to implement experiments in a sustainable modular fashion. It was possible to separate a runtime system from application code and identify reusable components by systematizing our experimental codebases. A retrospective realization showed that much work could have been saved if it were for a framework that provided the application-agnostic parts of our custom code.

We also realized that lacking a reusable framework is not a specific issue for us but must be a general one. Despite being a hot topic, techniques and methods around self-awareness are developed at a moderate pace and lack convergence. A major obstacle that is to be overcome is the high cost (mostly development time) of implementing self-aware systems. Lacking a common framework makes each system to be developed from scratch. This results in a considerable amount of work being done redundantly, inefficiently, and uneconomically. Using a common framework would enable cooperation and synergy among researchers and practitioners from a diverse spectrum of expertise.

So we set off to make a framework based on our experience and considering the following goals:

- use a compositional application model,
- provide reusable features and facilitate customization,
- support both simulation and deployment of applications,
- have a low-footprint realization to enable the framework in resource-constrained Embedded Systems (ESs),
- make a future-proof and sustainable framework (e.g., standard and stable technology, platform independence, low overhead, open architecture).

Selecting a proper architecture and implementation fitting our goals was a fundamental question. We concluded with a hierarchical agent-based architecture, whose details are discussed in Section IV-B. As none of the available agent-based frameworks can fully cover our goals (detailed evaluation in Section III-F), we implemented RoSA as a new framework.

Identifying and implementing self-aware functionalities so that they can be reused in different applications is a

storehouse of challenges. We spent the most time with functionalities abstraction, data reliability, and confidence, whose reusable implementations are featured in RoSA.

III. BACKGROUND AND RELATED WORK

Our work is motivated by the ever-growing importance and relevance of self-awareness in CPSs and SoCs. In Section III-A, we give a short inside of Autonomic Computing (AC), while we throw a bridge to Self-Awareness in Section III-B. Since we propose a modeling framework for self-aware systems to facilitate collaboration in the field and go beyond the state of the art, we discuss existing self-aware architectures in Section III-C and review frameworks implementing self-aware systems in Section III-D. For the technical background of our proposed implementation, decentralized architectures are reviewed in Section III-E and available implementations of our choice of architecture, agent-based frameworks, in Section III-F.

A. AUTONOMIC COMPUTING

Smart systems require high degrees of automation and autonomy [34]. The word autonomy originates from ancient Greece and means to be self-governing, in other words, to have own laws [35]. In the context of computer systems, the concept of autonomy came up in the 1990s and was inspired by biological systems [36]. Both academia and industry started some initiatives at that time [35].

As often, very early attempts were made in the military field. Defense Advanced Research Projects Agency (DARPA) had a project in which they developed a communication and location device for soldiers [37], [38]. Soldiers could give information about the situation of themselves and their environment. Together with locating and sensing abilities of the device, relevant details on the battlefield were spread between the soldiers.

Besides, in the 1990s, the National Aeronautics and Space Administration (NASA) started projects such as Mars Path Finder and Deep Space 1. The goal of these projects was that space crafts should become more autonomous to operate, navigate, and manage deep-space probes with less intervention of humans [39]. The fact of becoming more autonomous was important because remote control of these space crafts is associated with a clearly noticeable delay and therefore was highly impractical.

The complexity and dynamic changing environments call for autonomic systems [35]. In 2001, the International Business Machines Corporation (IBM) declared that the complexity of Information Technology (IT) systems would be one of the biggest challenges for the progress of the industry in the coming decades [40]. To make computer systems autonomous and having less need for human interventions, IBM started the AC initiative and introduced five levels of maturity: basic, managed, predictive, adaptive, and autonomic [41]–[44]. The lowest level (basic) describes a system that is managed by highly skilled staff which monitor these systems and manually modify them based on the gathered information [37]. In contrast, the highest level describes fully autonomic systems (or applications) that totally manage themselves in order to fulfill high-level goals which could be given by humans [36]. In other words, an AC system manages itself according to high-level objectives given by humans [45].

Furthermore, IBM introduced in [42] the four self-* properties of AC (often referred to as "self-chop" [10], [37]):

- self-configuration (autonomous configuration, such as adjusting parameters or changing software, in order to fulfill high-level goals),
- self-healing (autonomous detection and diagnostic for discovering problems and trying to fix them autonomously),
- self-optimization (autonomous resource usage optimization), and
- self-protection (autonomous protection against malicious attacks and unintentional misapplication by the system's user).

These self-* properties (in details described in [44], [46]) are the most cited ones in the AC domain, but the number of them has continuously grown; for the most prominent examples, we refer to [35], [36].

B. SELF-AWARENESS

Self-awareness, which is one of the self-* properties, was proposed originally in the IBM initiative on autonomic computing [44], [47]. Computational reflection and self-awareness are very close to each other. Computational Reflection is the ability of a system to reason about its capabilities, limitations and resources [45]. A self-aware system observes itself as well as its environment and changes its behaviour according to the observations it has made. Thus, self-awareness could also be called computational reflection [48], [49]. A self-aware computer system needs sensors to sense the internal as well as the external environment and actuators to self-adapt to the changing environment [36]. In an effort to improve flexibility and adaptivity of systems, the self-* properties are organized into a hierarchy with self-awareness and context-awareness at the base (Figure 1). In other words, a system has to be self-aware to be self-adaptive (or autonomous). A correlation between the usage of self-* properties and the quality of complex software systems has been shown in [50].

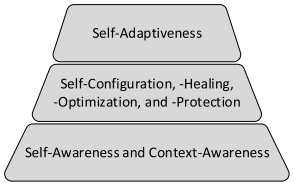


FIGURE 1. Pyramid of self-* properties.

Self-awareness has recently moved up in prominence. Initially, it was at the bottom of the self-* pyramid (Figure 1) as a supporting feature for more advanced adaptive behavior. Recently, self-awareness is used quite often to encompass all relevant self-* properties, including self-adaptiveness. The pyramid has been turned upside down because of the realization: self-awareness is not just a collection of state variables. It must include the goals of the system and properly reflect the effects of actions on itself and the environment. However, in contrast to other self-* solutions or AC, a fully self-aware system operates not only *reactively* but *proactively*. This means that such a system needs to be able to learn, make conclusions, and act accordingly [51].

For example, in the recent past, self-awareness showed to be a key enabler to tackle many challenges SoCs face such as growing process variability, thermal limitations, and wear-out effects [17], [18], [21]–[24]. CSA has been applied to both software [44] and hardware [52]. Following applications have benefited from CSA concepts (some of them under other terms such as adaptivity, autonomy, and goal-oriented systems): mobile applications [53], object tracking with smart cameras [24], [54], artificial intelligence [55], cloud computing [56], networks [57], operating systems [58], web [59], adaptive and dynamic compilation environment [60], Multi-Processor System-on-Chip (MPSoC) resource management [61], [62], (cyber-physical) SoC [52], mobile robots [63], industrial systems [64], [65], health monitoring [22] as well as single and multi-user active music environments [66].

The different aspects of self-awareness — like *self-monitoring*, *situation-awareness*, and *attention* — have been shown to be essential for efficient embedded CPSs [15], [52], [67]–[71]. Self-monitoring is the activity of sampling system properties (e.g., chip temperature [71]) as well as transforming and filtering sampled data in a system-specific way (see the self-aware functionality *abstraction* in Section VI-A). Situation-awareness assesses the observations and gives significance to data. On the other side, attention balances the competing tasks of data collection, processing, and responses under tight resource constraints by dynamically prioritizing goals and tasks. The overall system performance is monitored in a dynamically changing environment by means of self-awareness.

It has already been shown that self-awareness can help solve many problems of CPSs and SoCs. Furthermore, different aspects of self-awareness are used to make CPSs smarter [32], [33], [72]. However, the development of self-aware systems and related methods is still a difficult and tedious process. Moreover, efforts are fragmented among different communities because of the lack of a common framework to explore self-awareness and its properties. We propose a framework, RoSA, to overcome that obstacle. RoSA is based on principles and methods that have been published in literature but have not been combined before. The next few paragraphs overview various works that are related to RoSA.

C. REFERENCE ARCHITECTURES FOR SELF-AWARENESS

There exist several reference architectures which concern systems related to CSA [51]. One of them is the MAPE-K loop (an autonomic control loop coming from the AC field [42], [44])), which stands for *Monitor*, *Analyze*, *Plan*, *Execute*, and *Knowledge*. Information is collected from sensors in the monitor phase, and the gathered information is analyzed in the analyze phase. Subsequently, the plan- and execute phases are about planning and executing actions in order to fulfill goals or solve problems [51]. All these four phases share one common aspect: knowledge about the context, the execution environment and the hardware infrastructure. The MAPE-K loop is very similar to the Observe-Decide-Act (ODA) loop we have implemented (Section IV-B3).

The *Learn, Reason, Act* architecture is a model-based learning and reasoning loop (LRA-M loop) [73]. The architecture describes a self-aware computing system that is driven by its goals and its observations collected as empirical data. The collected data are used in an ongoing learning process that abstracts observations into models. The learned models provide a basis for the reasoning process, which might trigger actions affecting the system itself and possibly its environment. The LRA-M loop is a model-based formulation of the ODA loop.

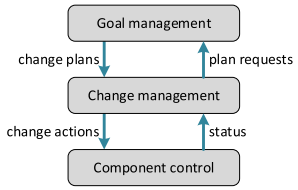


FIGURE 2. The reference architecture for self-managed systems from [74].

Another related architecture is the *Reference Architecture for Self-managed Systems* from Kramer and Magee [74]. Figure 2 shows this architecture which consists of three different layers with different tasks. The *Goal management* (the top layer of the architecture) is there for the planning. This is where plans are initiated to meet the requirements of the applications and to achieve their goals. Such plans may be required by new goals from the user or by requirements of the layer below. The *Goal management layer usually has some awareness models to be able to reflect on the layer below and address it properly [51].* This underlying layer contains *Change management*. This is where the various plans are stored, which shall be processed. The best plan for the respective current situation is selected in order to adapt the layer below. The *Change management* layer is also reflective and has typically an awareness model of the layer below; the lowest layer [51]. This layer, the layer on the bottom of the architecture, is called *Component control*. Here, the actual functionalities of the application are implemented

and accordingly adjusted by the instructions (based on various plans) from the layer above (*Change management* layer). The *Component control* layer is pre-reflective, and it sends up status reports to the layer above. If the *Component control* layer reports an inability to meet the given application goals, the *Change management* layer adapts it in a way it can achieve them in the current (environmental) situation [51]. Besides the usage of various awareness models, the hierarchical structure of this approach matches the RoSA architecture IV-B2.

The *Reference Architecture for Models@run.time Systems* is proposed in [75], and its main characteristic is that there is an explicit distinction between two systems often called managing system and managed system, where the first one manages the second one [51]. The managed system can be divided again into the (actual) managed system and its environment. The managing system often has three layers accordingly to the above-mentioned *Reference Architecture for Self-managed Systems*, where the lowest layer has an interface to the managed system. While the top layer is very similar to the previous architecture, the bottom two layers are formulated much more precisely. The bottom layer contains *configuration models* (reflecting the current state of the managed system), *plan models* (controlling the managed system), *capability models* (covering the managed system's capabilities) and *context models* (focusing on the managed system's environment). The middle layer consists of a *learner* synchronizing all models of the lowest level with the managed system, a *reasoner* making decisions based on the models of the lowest level, and an *analyzer* abstracts the information provided by models of the base layer in order to enable a hierarchical decomposition.

The ''reference architecture for self-awareness'' from Lewis *et al.* [24] describes a psychology-inspired conceptual framework of self-awareness. The architecture defines a number of different units that can be used to describe a system with self-aware and self-expressive capabilities. The components are sensor and actuator units, self-expression unit, self-awareness unit, and meta-self-awareness unit. The meta-self-awareness unit assesses the desirability of maintaining a level of awareness. The self-awareness unit consists of several subsystems for certain types of awareness:

- **stimulus awareness** is the knowledge about stimuli that act on the system and the ability to respond to them;
- **interaction awareness** is the knowledge about the interaction between the system and its environment;
- **time awareness** is the knowledge about past states and future phenomena;
- **goal awareness** is the knowledge about objectives, preferences, and constraints as well as the ability to reason about them or manipulate them;
- **meta-self-awareness** is the knowledge about possible levels of awareness and the way they are executed.

The recommended use of the reference architecture is described in a handbook [22]. A case study about implementing a service selection application in the reference architecture is available in [24].

Besides these reference architectures, a suitable modeling method, which is similar to our work, is proposed in [76]. However, that model uses a vague definition of agents as design abstraction, while RoSA provides facilities for the definition of agents based on self-aware functionalities.

D. FRAMEWORKS FOR SELF-AWARENESS

There are frameworks that focus on particular self-* properties. SAPERE [77] and ACOSO [78] are middlewares that support self-organization of autonomic nodes in distributed environments. Though they build on an agent-based model like RoSA, they are focused on self-organization (a self-awareness property that is not covered in RoSA yet) and so provide complementary features to the current set of self-aware functionalities of RoSA. The following examples provide complementary features as well. BIONETS [79] is based on similar concepts and supports self-adaptation of autonomic nodes in distributed environments. The *Collective Adaptive Systems* approach of the ALLOW Ensembles project [80] supports collaborative self-adaptation of agents within groups called *ensembles*. SEEC [61] is a framework for self-aware resource allocation based on the concept of application heartbeats, which allows monitoring and adjusting program performance. We did not base our work on any of these frameworks because (i) SAPERE and ACOSO are implemented on top of JADE, which does not fit most ESs (Section III-F); (ii) the BIONETS concepts are implemented only in simulation models, which limits its deployability in real systems; (iii) the ALLOW Ensembles approach is demonstrated by a case study in DeMOCAS [81], which is a simulation framework implemented in Java and hence has limited deployability; and (iv) the implementation of SEEC does not match the agent-based architecture, which we selected for flexibility and scalability (Section III-F).

Although these works offer more or less specific design proposals for various self-aware systems, they do not constitute a complete modeling framework.

E. DECENTRALIZED ARCHITECTURES

Decentralized architectures have already been proposed in the early days of Artificial Intelligence (AI) [51]. A decentralized system in this context consists of several agents (independent modules) which may interact with each other and work in parallel on their different tasks. According to [82], designing and building rational agents is fundamental for AI. Russell *et al.* further state that agents are rational entities that take the best possible action according to the information and capabilities they have at their disposal [82].

In [83], Wooldridge *et al.* define agents as software pieces that are autonomous (can autonomously operate without human intervention), social (can communicate with other agents or humans), reactive (can respond to changes in

TABLE 1. Multi-Agent modeling Systems.

Framework	License	Multi-Agent Simulation	Deployable Actor System	Implementation / Runtime	CPS / Embedded System	modeling Language	Native Interface
Akka [87]	Free	Possible**	Yes	Java/Scala	No	Java/Scala	JNI
CAF [88]	Free	Possible**	Yes	Native C++	Partially****	C++	C/C++
GAMA-Platform [89]	Free	Yes	No	Java	-	GUI / GAML*****	-
JADE [90]	Free	Possible**	Yes	Java	No	Java	JNI
Mobile-C [91]	Partially free*	Possible**	Yes	Native C / C++ / Ch***	Yes	Ch	Ch Binary Interface
Repast for HPC [92]	Free	Yes	No	Native C++	-	C++	-
Repast Simphony [93]	Free	Yes	No	Java	-	GUI / Groovy / Java	-
RoSA	Free	Yes	Yes	Native C++	Yes	C++	C/C++

* Proprietary dependency: Embedded Ch, free for non-commercial use on ARM-based systems.
** Not designed for simulation but might be configured for the purpose with considerable effort.
*** Ch is a scripting language with C/C++ syntax.
**** CAF poses a relatively large footprint because of its extensive non-configurable set of features.
***** GAML is the GAMA Modeling Language.

the environment), and pro-active (can take the initiative instead of just reacting). A Multi-agent System (MAS), in further consequence, is a system consisting of multiple agents that work together to fulfill one or more common goals [45].

An agent-based architecture (e.g., a MAS) implements the actor model [84], which is a programming paradigm known for scalable parallel and distributed computing. To better handle complex applications, it is usually advantageous to divide them into different tasks. Often these can be divided into different levels to cover the big picture as well as small details in particular. Accordingly, it can be helpful to have the possibility of a hierarchical structure. This is similar to the nature-inspired hierarchical system of the AC initiative from IBM [85]. Applying a *hierarchical agent-based* approach to self-aware systems has been studied in the literature [86]. An agent-based framework that facilitates self-awareness, however, has been an open issue.

F. AGENT-BASED FRAMEWORKS

Some existing self-aware frameworks are built on agents (see Section III-D). There are general agent-based frameworks, which are ignorant of the internal workings of agents. These are summarized in Table 1 and discussed in this section.

The two main use cases of the agent-based frameworks are *multi-agent simulation* and *deployable actor system*. Java-based frameworks have a high resource requirement beyond the typical capacity of ESs. The large-scale *multi-agent simulation* systems are not suitable for *ESs* for similar reasons, and they have limited capabilities for interfacing real hardware. *Deployable actor systems* with *native implementation* (Mobile-C and CAF) can support execution on ES hardware and are detailed as follows. *Mobile-C* is a small-footprint distributed actor system. However, it has a proprietary dependency and a custom native API, which limits its applicability.

CAF is an open-source distributed actor system with standard C++ implementation and with the aim of working on a wide spectrum of hardware platforms. Its extensive non-configurable feature set, however, makes it less suitable for ESs. A stripped-down version for resource-constrained systems remains a promise to date.

IV. THE RoSA FRAMEWORK

RoSA combines the agent-based actor model with self-aware properties in an ES-compatible fashion and is fully open-source. In this section, we discuss the general facilities of the RoSA framework, which are the agent-based architecture and details of its implementation. Actual *functionalities* are presented in Section VI, and the implementation of self-aware applications is showcased in Section VII by case studies.

A. TERMINOLOGY

Here, we define the following terms with the meaning we use in the context of RoSA and the rest of this paper.

1) **Agents** are design abstractions that help decompose a system into independent components. A classic definition of agents comes from the field of artificial intelligence [82]: "an agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators." RoSA agents

comply with that definition. Section IV-B describes their inner workings and interactions.

2) **Data manipulation** is the processing activity that is done by any agent: observing its environment via input, maintaining its internal state, and optionally generating output to affect its environment. Individual RoSA agents may realize different ways of data manipulation, which is described in terms of *functionalities*.
3) **Functionalities** encapsulate self-awareness concepts in reusable components. They are the basic tools that can be put together to realize desired ways of data manipulation in agents. An agent is designed by a careful selection of functionalities for the required data manipulation. The self-aware functionalities that we have already implemented are elaborated in Section VI.

B. RoSA ARCHITECTURE

The architecture of the RoSA framework is outlined in this subsection, accompanied with a discussion on some design decisions.

1) SCOPE

The RoSA architecture supports modeling self-aware applications, whose relevance is motivated in Section I and Section II. The framework is intended to be a tool for modeling and evaluating novel ideas in self-aware applications. The application model (i.e., a hierarchical agent-based system with functionalities within agents) is flexible enough to incorporate variations in different aspects of design and implementation. Those aspects are mostly related to the functionalities: (i) what functionalities are there, (ii) how they are implemented and interconnected, and (iii) how applications are decomposed. The architecture provides a structured and modular way for defining self-aware applications: applications are decomposed into agents, which are defined by functionalities. Agents and functionalities are reusable components in RoSA.

2) HIERARCHICAL AGENT-BASED MODEL

An agent communicates with its environment (i.e., other agents of the application) by message passing via input and output channels. Semantics can be informally given as: the agent receives messages on its input channels; manipulates data (i.e., the received messages and its internal state), and may send messages on its output channels.

Agents are organized into a hierarchical structure (e.g., Figure 3). Agents on different levels of the hierarchy process data on different levels of abstraction: the system obtains fine- and coarse-grained knowledge according to hierarchy levels. Such a detailed representation of knowledge helps self-adaptive systems to operate more efficiently and meet their goals [32].

Connected agents are in a master-slave relation. An agent (e.g., Agent 2 in Figure 3) receives messages from its slaves (Agents 5 and 6) and sends messages to its master (Agent 1). That is, an agent acts as slave towards its only master and as master towards its potentially multiple slaves.

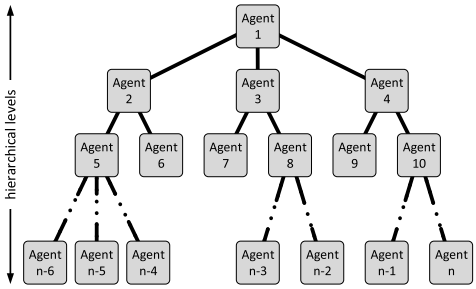


FIGURE 3. A hierarchical agent-based model.

A slave sends messages to its master regularly according to its configuration. A master may control the configuration of its slaves by sending control messages to them whenever appropriate.

A real-world application interacts with its environment via sensors and actuators, which are modeled as agents in RoSA. An agent that wraps a sensor is a data source (i.e., has no slaves) and sends sensor input to its master. Dually, an agent that wraps an actuator is a data sink (i.e., has no master). An actuator is activated (''controlled'') by slave-to-master data messages — rather than master-to-slave control messages.

3) OBSERVE-DECIDE-ACT LOOPS

AC systems consist of autonomic elements implementing a control loop [36]. Thus, self-aware applications in RoSA operate in an iterative manner implementing ODA loops [52]. ODA is our architecture of choice, however, other architectures could be chosen and implemented as well. An ODA loop (Figure 4) represents the way reactive systems operate: the system monitors the behavior of itself and/or its environment, decides about certain actions, and acts accordingly.

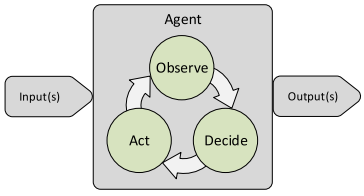


FIGURE 4. An agent implements an Observe-Decide-Act loop.

As shown in Figure 5, each RoSA agent operates in an ODA loop: receives input messages, does data manipulation, and optionally sends output messages. The composition of individual ODA loops results in a behavior that can be described as a compound ODA loop on the application level.

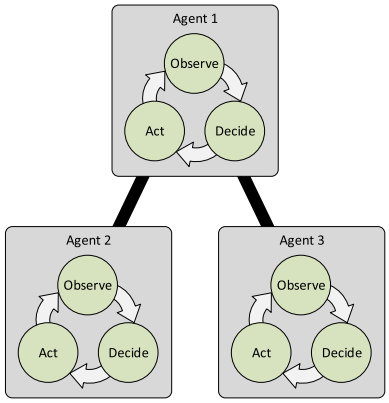


FIGURE 5. An agent system based on individual Observe-Decide-Act loops.

The RoSA architecture provides a way to implement ODA-loop-based applications that are decomposed into interacting ODA loops of lower complexity.

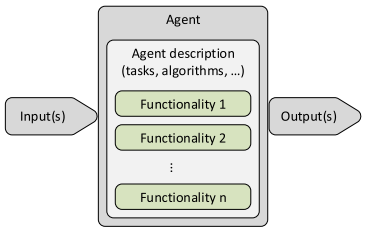


FIGURE 6. The behavior of an agent is defined by self-aware functionalities.

4) FUNCTIONALITIES

An agent is defined by functionalities (Figure 6). What functionalities an agent utilizes depends on its role in the application. RoSA provides a library of pre-defined functionalities (Section VI) and allows developers to implement new ones either based on existing ones or from scratch.

As shown earlier, RoSA agents conceptually operate in ODA loops. The functionalities that constitute an agent contribute to different characteristics of observation and decision making in the loop. For example, *abstraction* (Section VI-A) improves the outcome of observation, and *data reliability* (Section VI-B) helps decision making by providing meta-information. Our approach is inspired by the hierarchical agent-based model of Guang *et al.* [76]. While their model uses a vague definition of agents as design abstraction, RoSA agents are described as ODA loops that are based on functionalities.

C. SOFTWARE IMPLEMENTATION

We have implemented the RoSA architecture as a software framework. RoSA has a fully open implementation in standard C++ and can readily interface existing native software components. The main characteristics of the software implementation are (i) providing a high-level but safe modeling interface for application developers, (ii) allowing the same application code to be used for *simulation* and *deployment*, and (iii) realizing small-footprint software that can be deployed in resource-constrained ESs. We have done our case studies (Section VII) in simulation on a desktop computer. That is, input and output of sensor and actuator agents are fed to the system via stored files, and RoSA allows for other input-output interfaces as well. Runtime support for deploying on embedded devices requires further development to complete.

V. USE CASES OF THE FRAMEWORK

The section discusses how RoSA, the framework as a whole and its features separately, can be used in different scenarios.

A. MODELING A NEW APPLICATION IN RoSA

Modeling an application using the RoSA Architecture follows a general flow shown in Figure 7. That is,

Specify requirements: The most abstract description of an application defines input and output (sensors and actuators, respectively) and the data manipulation to be done. It can be seen as an extreme agent system with all sensors and actuators connected to the only agent that represents the entire application.

Model agent system: The monolithic application-agent is decomposed into a set of agents organized in a hierarchy. Agents enclose specific kinds of data manipulation and serve as a unit of reusability — within and between applications. Identifying agent patterns can help efficient decomposition.

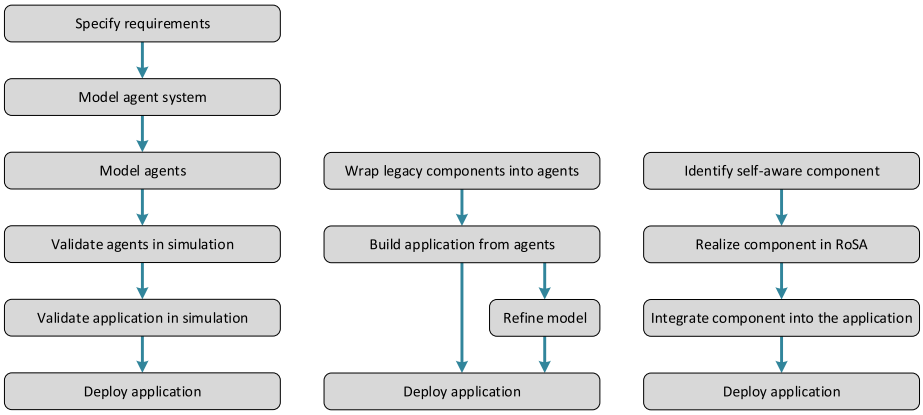
Model agents: Each agent is modeled, i.e., prescribed data manipulation is realized by available functionalities and custom code (Figure 8). Functionalities provide a level of reusability below agents. RoSA provides a set of functionalities, which is expected to grow over time.

Validate agents in simulation: *Unit testing* of agents is done by validating their input-output behavior in simulation mode: a single-agent system is evaluated with predefined input and expected output.

Validate application in simulation: Agents are put together according to the *system model*. *Integration testing* of the application is done by validating the input-output behavior of the system in simulation mode.

Deploy application: The application is deployed in an embedded device.

Though the RoSA methodology is presented as a sequential flow, the model of an application (i.e., the system model with corresponding agent models) may be refined in an iterative manner.



(a) Modeling a new application in RoSA (b) Migrating a legacy application to RoSA (c) Adding RoSA as component to a legacy application

FIGURE 7. Scenarios of using the RoSA framework.

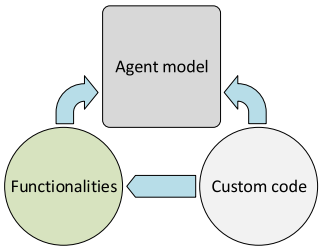


FIGURE 8. An agent is modeled based on available *functionalities* and *custom code*; reusable pieces of *custom code* are gradually promoted to *functionalities* in a generalized form.

B. MIGRATING AN APPLICATION TO RoSA

RoSA can also be used to add self-awareness to existing applications. An entire legacy application can be migrated to RoSA in a few steps shown in Figure 7(b). That is,

Wrap legacy components into agents: Each component, whose input-output behavior fits message-passing semantics, is wrapped into a RoSA agent. Legacy components may be grouped, if necessary. Existing legacy code implements data manipulation within agents.

Build application from agents: Agents are put together in an agent system according to the connections between corresponding components in the legacy system.

Refine model: The system and agent models may be refined iteratively, as in the general case.

Deploy application: The agent system is deployed as a RoSA application.

This approach turns a legacy system into a RoSA application and enables utilizing all RoSA features for further development.

C. ADDING RoSA AS A SELF-AWARE COMPONENT

It is also possible to add a RoSA agent system to an existing application as a self-aware component (Figure 7(c)). This scenario might be applied as a gradual migration path.

Identify self-aware component: The requirements are specified either as a new component of the application or based on an existing component to be replaced.

Realize component in RoSA: The component is realized as a RoSA agent system following the general RoSA methodology (Figure 7).

Integrate component into the application: The component is integrated into the existing application via input and output streams that are associated with its sensor and actuator agents, respectively.

Deploy application: The application is deployed with the RoSA system as one of its components.

This approach limits the development effort to one component of the application — in contrast to migrating the whole application. However, additional development and runtime complexity is posed by the need to integrate RoSA as a component of the existing application. Whether to take the full migration or the component approach depends on the size of the application and how much the application needs self-awareness and can benefit from RoSA.

D. USING SELF-AWARE FUNCTIONALITIES FROM RoSA

RoSA supports reusability on two levels: agents in the system model and functionalities in the agent model. The realizations

of these two levels are independent, and functionalities may be used without using agents. Should working with a RoSA agent system be uneconomic, functionalities that are defined in RoSA (Section VI) may be used in custom codes directly — without involving other parts of the RoSA framework.

E. IMPLEMENTING A GENERAL AGENT-BASED APPLICATION WITH RoSA

While the main aim of RoSA is to facilitate developing applications and concepts related to self-awareness, the applicability of the framework is not limited to that. The agent system that constitutes the base of the architecture can be used for any other application that may benefit from such an architecture (e.g., component-based systems). One can ignore self-aware functionalities and implement an agent-based application with all data processing in agents defined by custom application-specific code only.

VI. SELF-AWARE FUNCTIONALITIES

Each RoSA agent receives messages from its input channels and may send messages on its output channels. The data processing that the agent does to maintain its state based on input messages and generate output messages can be defined with full flexibility (i.e., custom application code). Nevertheless, RoSA provides predefined functionalities to be used as components when defining agents, with minimal glue code that connects them. It is also possible to mix functionalities and custom code freely within agents. The modularity enables application developers to define self-aware agents fast and efficiently by reusing existing functionalities and also customize data processing whenever needed.

The functionalities are based on self-aware properties [94], [95]. RoSA provides reference implementations of the functionalities that have been used in our case studies (Section VII): abstraction, data reliability, confidence, and history. We expect the set of self-aware properties and corresponding functionalities to grow as well as their implementation to improve — contributions from the community are welcome.

A. ABSTRACTION

Abstraction is ''an appropriate selection of the representation of the information in order to obtain compact knowledge relevant to a particular purpose'' [94]. It is a transformation of data from one domain to another. Raw input data may be abstracted into a semantic domain that the self-aware system understands [52], and the abstraction may be done at any level of a hierarchical system. It could also be done top-down instead of bottom-up [94]. An abstraction needs to be meaningful and efficient in the system's context and to have a well-defined structure.

1) ABSTRACTION FUNCTIONALITIES AVAILABLE IN ROSA

The broad definition of abstraction allows for a wide variety of approaches. RoSA currently provides the following abstraction functionalities:

1) **Lookup table** maps an input datum to a symbol (e.g., number, character, string).
2) **Overlapping lookup table** maps an input datum to potentially multiple symbols; in case the boundaries between symbols cannot be clearly defined (e.g., insufficient knowledge about the environment). Selecting one symbol in a later processing step may be a confidence-based decision (Section VI-C). In contrast, a standard lookup table maps an input value directly to one symbol.
3) **Threshold-based signal state detector** abstracts steady states from a signal waveform, that is a sequence of input values. In other words, it recognizes stable phases in a signal. These steady states of a signal are identified concerning a threshold of distance among the signal's sample values. A signal state is stored as an average value of all input samples belonging to it. A simple learning algorithm is utilized internally for state detection. Detailed discussion is available in [29], [30].
4) **Confidence-based signal state detector** also abstracts steady states from a signal waveform, that is a sequence of input values. These steady states of an input signal are identified concerning the relative distance among the signal's sample values. That is, in contrast to the *Threshold-based signal state detector*, the *Confidence-based signal state detector* makes all decisions based on a confidence assessment (Section VI-C). This assessment is not only based on a simple average, but on the most recent signal samples stored in a sliding window history. A detailed discussion of the learning algorithm behind this functionality is available in [31].
5) **System state detector** abstracts a system state from signals of an observed system. The current implementation works with stateless systems only (i.e., identifying states of a system whose output can be expressed as a function of its input).

B. DATA RELIABILITY

Data reliability is ''the extent to which a measuring procedure yields the same results on repeated trials'' [94]. The trustworthiness of data is determined by accuracy, precision, and truthfulness. The accuracy and precision are given by systematic and random error of measurement, respectively. Data can be accurate and precise but still not truthful [28], for instance, if a sensor is working outside of its operating conditions (e.g., a temperature sensor detached from the test object).

Data reliability is a piece of meta-data about the trustworthiness of the input data stream. Further actions may be taken

according to the reliability of data. The following measures may be used to assess trustworthiness:

1) **Plausibility** tells whether data is within its expected domain (i.e., range). If a variable exceeds the realistic limits of its represented quantity (e.g., human body temperature over 100°C), data might be unreliable.
2) **Consistency** tells whether data varies according to its expected variability (i.e., maximum difference between samples). If a variable changes too fast (e.g., position of a robotic arm), data might be not reliable. Checking consistency requires historical information (Section VI-D) about the input signal.
3) **Cross-validity** tells whether one piece of data correlates with other pieces as expected. If two dependent variables (e.g., two interdependent vital signs) do not follow each other, data might be not reliable.

1) DATA RELIABILITY FUNCTIONALITIES AVAILABLE IN ROSA

RoSA provides functionalities for assessing each of these three measures of trustworthiness either in a binary or in a fuzzy way (a total of 6 variants). Binary assessment makes a binary decision about reliability according to a threshold. Fuzzy assessment determines the level of data reliability in the [0, 1] range and can be configured with a custom function.

The individual assessments may be combined (e.g., considering both plausibility and consistency of a variable at the same time). RoSA provides a set of predefined methods (average and multiplication of fuzzy assessments; conjunction and disjunction for both binary and fuzzy assessments) for the combination, which may be done as custom application code as well.

C. CONFIDENCE

Confidence is ''the extent to which a procedure may yield the same results on repeated trials'' and has significant similarities to data reliability [94]. Confidence is a piece of meta-data about the trustworthiness of the data processing performed by a (sub-)system or function. It tells how well a calculated result corresponds to the expected output. Assessing confidence assumes error-free input — which may be assessed by data reliability (Section VI-B).

1) CONFIDENCE FUNCTIONALITIES AVAILABLE IN ROSA

RoSA defines an interface for assessing confidence, but the actual assessment logic needs to be provided as a custom function. The lack of predefined assessment functions is because no general confidence measures have been identified yet. The assessment of confidence varies much on a case-by-case basis in our experience.

Besides this interface, RoSA offers a confidence-based abstraction method, which is an *overlapping lookup table* (Section VI-A). This method is based on fuzzy membership functions [96], and Figure 9 shows an example of it. The input data is mapped to three symbols (A, B, and C) so that two symbols are associated for the overlapping ranges

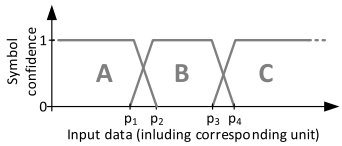


FIGURE 9. A confidence-based abstraction method to abstract data into one or more symbols with a corresponding confidence.

(i.e., (A, B) and (B, C) for $[p_1, p_2]$ and $[p_3, p_4]$, respectively). The abstracted symbols are assigned with a confidence value according to their corresponding fuzzy membership functions (i.e., *full* confidence outside of the overlapping ranges and lower confidences inside them). The membership functions can be adjusted dynamically via control feedback in the agent hierarchy whenever a higher level agent recognizes a systematic error.

Cross-validity confidence tells whether one piece of data correlates with other pieces. It is similar to cross-validity reliability in that respect. However, it calculates historical correlation information based on active monitoring, unlike the a priori expectations of cross-validity reliability. This assessment can be used to tune confidence-based abstraction in lower levels of the hierarchy.

Individual confidence assessments may be combined, similar to combining individual reliability assessments. RoSA provides predefined combination methods and the possibility of handling combination by custom application code. The reliability of the output of an agent can be assessed by combining the reliability assessment of its input and the confidence assessment of its data processing.

D. HISTORY

History is ''recording and studying a series of past events connected to an entity'' and enables extracting knowledge from the recorded time series [94]. Identifying trends in the past, understanding time-dependent aspects of the current state, and predicting future conditions [28] may all be supported by utilizing history.

1) HISTORY FUNCTIONALITIES AVAILABLE IN ROSA

RoSA includes limited support for history (only the features that we needed to implement other functionalities). A short description is still included because the history functionality can be used in the application code directly.

History functionality enables storing a sequence of data values. Its capacity can be configured for a balanced memory usage. History supports two strategies for redeeming memory once its capacity is reached: (i) stop strategy, when a full history does not accept further data values, and (ii) FIFO strategy, when history behaves like a sliding window. History functionality allows access to the individual stored values and also provides statistical properties (e.g., average) about the stored sequence.

VII. CASE STUDIES

Applications with different levels of self-awareness have been implemented in RoSA. We present two case studies in this section, which demonstrates how RoSA can be used for quick application development.

A. SELF-AWARE EARLY WARNING SCORE SYSTEM

The first case study is presented in detail for a smooth introduction of implementation details. The application — whose different variants are developed over the case study — is a health status assessment system.

1) BACKGROUND AND PROBLEM STATEMENT

Here, both, the calculation of the Early Warning Score (EWS) and the traditional EWS system are briefly described, before the next subsections deal with its extensions with various self-awareness properties.

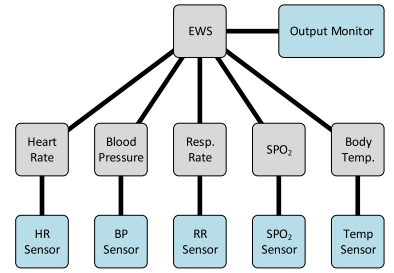
A patient's health status can be assessed based on their vital signs. Research on cardiac arrests shows that certain symptoms can be observed long before the situation turns into a case of emergency; symptoms may appear even 24 hours before actual health deterioration [97]. EWS is a standard manual tool for assessing patients' health status and predicting health deterioration. Healthcare professionals periodically monitor patients' vital signs (heart rate, respiratory rate, body temperature, blood pressure, and blood's oxygen saturation) and assess their health status by a criticality level defined as EWS [98]. For this reason, each vital value is assessed in the form of a score. A score of 0 indicates an ideal health condition of a vital sign, while score 3 corresponds to the worst. The EWS is the aggregate value of all the individual vital sign scores. The higher the score, the higher the criticality.

This manual procedure has been applied to hospitalized patients. A portable device that automates the procedure would allow high-risk patients to pursue their daily lives with a much higher chance of survival. Robustness and fault tolerance is of major importance for such a device. Autonomous monitoring of patients in a non-hospital environment needs to deal with faulty measurements: sensors can be attached incorrectly, become detached, or break down. Incorrect measurements result in incorrect EWS, which might lead to false positive or — even worse — false negative assessments.

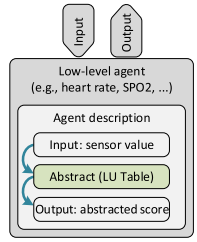
We developed several self-aware variants of the EWS application, which are able to deal with different kinds of faults [26]–[28]. Those variants and their results are discussed in detail in the referred papers. Here we motivate their high-level design and present their implementation in RoSA.

2) THE CONVENTIONAL EWS SYSTEM

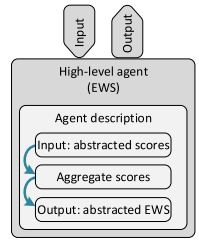
For starters, we implement an application that calculates the EWS in the conventional way (Figure 10). Five agents constitute the low level of the hierarchy, each connected to a sensor (modeled as a special agent) and assessing the corresponding vital sign. One agent in the higher level is connected to the low-level agents to make the aggregate assessment, whose result is recorded by a monitoring agent. The actual implementation is outlined in Listing 1. Even though RoSA and the applications are implemented in C++, the included listings use a C++-based pseudo-code for brevity. The sometimes verbose syntax of C++ is hidden, but the complexity of the application code is presented truly.



(a) Architecture of the EWS system



(b) Agent description of a low-level agent in an conventional EWS system



(c) Agent description of the high-level agent in an conventional EWS system

FIGURE 10. The conventional EWS system.

The implementation (Listing 1) starts with creating a RoSA *Application* (Line 1). Agents can be created and managed in the context of the Application.

For each vital sign (demonstrated by heart rate), a sensor (Line 2) and a low-level agent (Lines 4 to 21) are created. The low-level agent (Figure 10(b)) performs the EWS assessment by applying a lookup table abstraction (Section VI-A). The connection between the sensor and the low-level agent is established in a separate step (Line 23).

The high-level agent ((Figure 10(c)) aggregates vital sign assessments by summing them into a final EWS (Lines 26 to 31). Each low-level agent is connected to the high-level agent (Line 33).

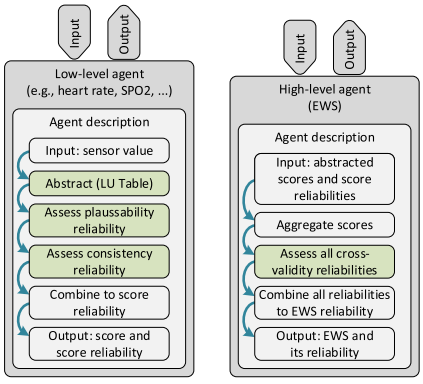
The final EWS is logged to the console by a dedicated agent (Lines 36 to 40), which is connected to the high-level EWS agent (Line 42).

3) A SELF-AWARE EWS SYSTEM WITH RELIABILITY FUNCTIONALITIES

The conventional EWS system does not tolerate faults. Thus we make the system more robust by utilizing additional

```
App = Application::create("EWS");
HRSensor = App->createSensor("HR_Sensor");

HRAbstraction = RangeAbstraction(
  {
    // upper bounds are exclusive
    {{  0,  40}, Emergency},
    {{ 40,  51}, High},
    {{ 51,  60}, Low},
    {{ 60, 100}, No},
    {{100, 110}, Low},
    {{110, 129}, High},
    {{129,  40}, Emergency},
  }, Emergency);  //default

HRAgent = App->createAgent(
  fun (updated, value) HR → Optional(Score) {
    if(HR.updated)
      return HRAbstraction(HR.value);
    return None;
  });

App->connectSensor(HRAgent, HRSensor);
// repeat for all vital signs

BodyAgent = App->createAgent(
  fun {(updated, score)} Scores → EWS {
    if(all Scores.updated)
      return sum(all Scores.score);
    return None;
  });

App->connectAgents(BodyAgent, HRAgent);
// repeat for all vital signs

LoggerAgent = App->createAgent(
  fun (updated, value) Score → None {
    if(Score.updated)
        Write(Score);
  });

App->connectAgents(LoggerAgent, BodyAgent);
```

LISTING 1. RoSA implementation of a conventional EWS system.



(a) Description of a low-level EWS agent equipped with reliability assessment

(b) Description of the high-level EWS agent equipped with reliability assessment

FIGURE 11. Agent descriptions for the EWS system with reliability assessment.

```
HRAgent = App->createAgent(
  // process new sensor values
  fun (updated, value) HR → Optional(S, C) {
    if(HR.updated)
      return (HRAbstraction(HR.value),
          ↪ HRReliability(HR.value));
    return None;
  },


  // process control feedback from master
  fun (updated, values) Scores → None {
    if(Scores.updated)
      HRAbstraction.update(Scores.values);
  });
```

LISTING 2. RoSA implementation of the self-aware heart rate agent.

self-aware functionalities within the agents (Figure 11). The agent hierarchy remains unchanged (Figure 10). Setting up the RoSA application and agent hierarchy is done similarly to the conventional EWS system (Listing 1).

The low-level agents (Figure 11(a)) assess the reliability of the abstracted vital signs by checking plausibility and consistency in combination (Section VI-B). The agent implementation is adapted by specifying the output as a pair of values (i.e., abstracted value and reliability assessment) rather than a single value and by utilizing reliability functionality in data processing (Listing 2 Lines 3 to 7).

The high-level agent (Figure 11(b)) assesses cross-validity reliability ((Section VI-B)) of the vital signs and combines all reliability assessments for the final EWS. The agent implementation is adapted similarly to low-level agents (i.e., adjust input and output types and utilize reliability functionality).

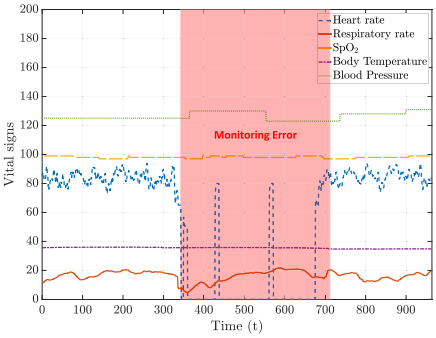
The reliability functionalities may be configured in different ways for the agents. Experiments have been performed both with binary [26] and with fuzzy [27] assessments.

Consider an experiment with the chest strap, which measures heartbeat, being loosely fastened. The measurement is not stable and provides unreliable readings during some time (e.g., 350s – 670s in Figure 12), while other sensors provide reliable data. Though the EWS is calculated according to the standard rules (i.e., results in false positives), the assessed reliability drops to 0 during measurement errors. The low reliability indicates an issue with the system's input(s).

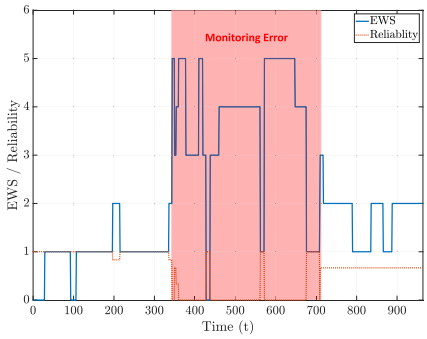
4) A SELF-AWARE EWS SYSTEM WITH CONFIDENCE FUNCTIONALITY

While the previous version calculated the EWS without any modifications, the final version adjusts the EWS in case of a low reliability [28].

The master agent (Figure 13(b)) additionally assesses cross-validity confidence of the vital signs (Section VI-C). The confidence assessment is based on personalized data and is — combined with the cross-validity reliability — used as control feedback for the low-level agents to adjust their score abstraction process. The implementation, using predefined functionalities, is still only a few lines (Listing 3).



(a) The monitored vital signs, where the heart rate signal often breaks down due to the faulty measurement



(b) The calculated EWS and its reliability

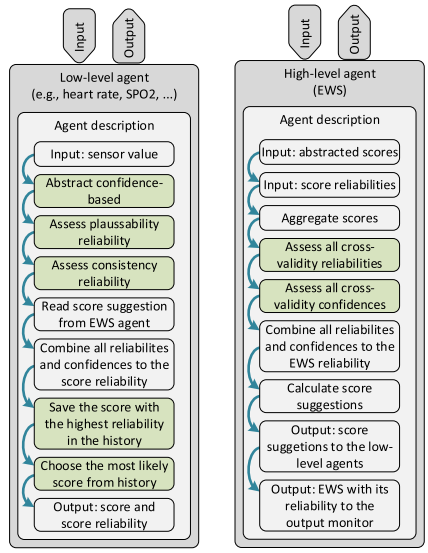
FIGURE 12. Results of an experiment in which the heartbeat sensor was not attached properly and therefore incorrect measurements were made.

```
1  BodyAgent = App->createAgent(
2    fun {(updated, score, rel)} Scores → ((EWS,
       ↪ rel), {vital-confidence}) {
3      if(all Scores.updated)
4        return (
5          // result
6          (sum(all Scores.score), EWSReliability
               ↪ (all Scores.rel))
7          // control feedback
8          {all EWSConfidence(all Scores.score)}
9        );
10     return None;
11   });
```

LISTING 3. RoSA implementation of the self-aware EWS master agent.

The agent generates its output as (i) a pair of calculated EWS and assessed reliability (Line 6) and (ii) a list of confidence feedback for each low-level agent (Line 8).

Low-level agents (Figure 13(a)) perform confidence-based abstraction (Section VI-C and Figure 14), which takes historical information into account (Section VI-D) about feedback from the high-level EWS agent. The calculated EWS is



(a) Agent description of a low-level agent in the EWS system equipped with reliability, confidence, and history

(b) Agent description of the high-level agent in the EWS system equipped with reliability, confidence, and history

FIGURE 13. Agent descriptions in the EWS system equipped with reliability, confidence, and history.

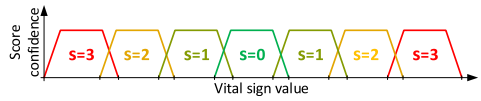
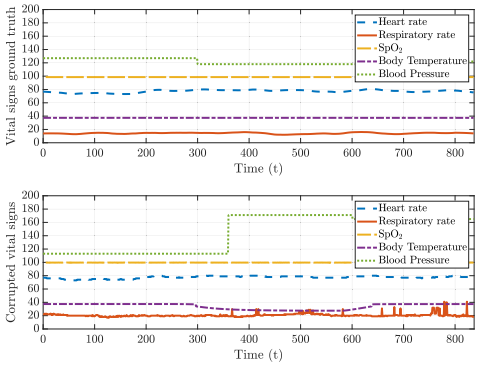


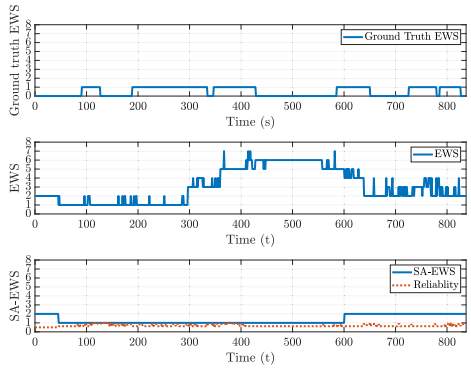
FIGURE 14. A confidence-based abstraction method to abstract a vital sign in one of four different scores (0 to 1).

adjusted in that way. The actual implementation (Listing 2) is divided into two functions: (i) one for processing input, like before, and (ii) one for processing control feedback. Sensory input is processed like before (Lines 3 to 7) except for abstraction being configured to operate based on confidence (Section VI-C). Processing control feedback (Lines 11 to 14) passes data from the high-level agent to the local confidence-based abstraction. The feedback is stored by the abstraction functionality internally with a history functionality and is utilized when processing future sensory input.

Consider an experiment when participants are monitored with both working and faulty sensors (upper and lower part of Figure 15(a), respectively). The experiment results (Figure 15(b)) show that our self-aware EWS application performs much better (even if not perfectly) than the conventional system in the presence of sensory errors. The self-aware EWS system has almost 80 times less false alarms than the conventional system in our experiments [28].



(a) The vital signs in the top are free from errors; those in the bottom were measured by faulty sensors and show significant errors



(b) The top diagram shows the correct EWS based on the actual vital signs; the two lower diagrams show EWS values calculated by the conventional and our self-aware EWS systems with faulty sensory input

FIGURE 15. Results of an experiment when participants were monitored with both working and faulty sets of sensors.

5) SUMMARY

The detailed case study followed the development of our EWS application through four versions. That process demonstrates that implementing an agent-based application with self-aware properties takes only a few steps in RoSA. Defining the agent system at the beginning was a lightweight task by using the existing agent interfaces of RoSA. Additionally, thanks to the modular design and reusable functionalities of RoSA, moving from one version to the next (i.e., including more sophisticated self-aware properties) needed only local modifications of agents and functionality configurations.

We were, of course, experimenting with different implementation alternatives during development. In the end, however, we packed the various functional components of data processing into functionalities, which are reusable modules. New applications can use those functionalities while they might also need to implement novel data processing approaches in custom application code. Those pieces of custom code, once matured, should be turned into functionalities for modularity and reusability. That is a way for sustainable development in the long run, and it is facilitated by RoSA.

B. CONTEXT-AWARE CONDITION MONITORING

This case study presents a monitoring system that assesses the working state and the health condition of another system or device; hereafter System under Observation (SuO). We limit the discussion to the modeling level (i.e., source-level implementation is ignored); the first case study provides insight into implementation.

1) BACKGROUND AND PROBLEM STATEMENT

Industry, particularly automated production plants, has an interest in reliable monitoring systems that are able to raise an alarm in case of malfunctions of the SuO [99]. Such a monitoring and warning system enables optimization of maintenance work and minimizes downtimes.

Implementing tailor-made monitoring systems for each SuO is an expensive endeavor. A reliable self-adaptive monitoring system can reduce cost and time. We present such a system, which is able to assess the health status of any SuO without detailed a priori knowledge but by observing its input and output. The system assumes that the SuO meets two requirements: (i) the SuO works as a bijective function between its input and output, and (ii) the SuO operates in steady states. Requirement (i) allows the monitoring system to uniquely identify input-output pairs of normal operation. Dissociation of input and output signals is then considered a symptom of fault. Requirement (ii) is a consequence of the fact that the monitoring system discards unstable and transient signals.

The monitoring system adapts to any SuO based on contextual information only (see context-awareness). We have performed experiments with two variants of the system: (i) Context-aware Health Monitoring (CAH) [29], [30] applies a threshold-based decision-making process and (ii) Confidence-based Context-Aware condition Monitoring (CCAM) [31] makes decisions based on confidence.

Compared to similar monitoring solutions (e.g., deep learning and data mining), our system has a considerably smaller runtime footprint and can be applied to resource-constrained applications.

2) MODELING THE MONITORING SYSTEM

The application has a hierarchical structure (Figure 16). The low-level agents are connected to the sensors and can perform pre-processing of sensory input if necessary as well as incorporate a signal state detector (Section VI-A).

The detected signal states are combined into a system state by a system state detector (Section VI-A) in the high-level agent. Its output (i.e., system state and health condition) is processed (e.g., logging or trigging a warning in case of malfunction) by a dedicated agent.

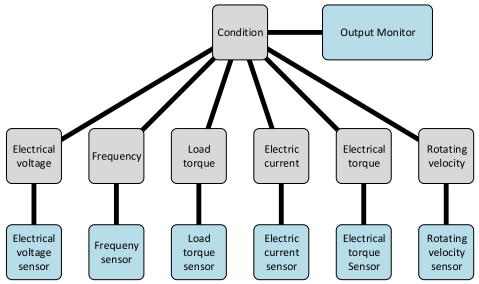


FIGURE 16. Architecture of CAH/CCAM for monitoring an AC motor.

The signal and system state detector functionalities keep historical information about their input to identify steady states as well as recognize state changes and drifting signals (see [29]–[31]). Each new signal sample is compared with historical information to detect if the signal changed state. The historical information consists of an average value in CAH and a sliding window history in CCAM. If the new sample is in close proximity to the saved data of a recorded state, it belongs to that state. Whether a signal is stable or drifting is extracted from the course of the historical data. The state of the system is then the composition of the individual signal states. For further details, we refer to our corresponding works.

The state and health condition of the observed signal/system are outputs of the functionalities. The difference between CAH and CCAM is in the configuration of signal and system state detection: they make binary threshold-based and confidence-based decisions, respectively. CCAM provides better results than CAH.

Adjusting CAH/CCAM for a different SuO takes only two simple steps: (i) defining a low-level agent with a signal state detector for each input and output signal of the SuO and (ii) connecting each low-level agent to the high-level one and associating each signal to the system state detector either as input signal or output signal. The system is easily scalable, but the system state detector could become a bottleneck in case of an extremely high number of signals. This potential scalability issue is not caused by RoSA but by the architecture of the implemented application. In the case of such a complex SuO with a massive number of signals, the problem could be scaled out by replacing the central system state detector with a corresponding hierarchy of those. In other words, the SuO would be split up in various subsystems that have their own system state detectors, which may be combined in a hierarchical structure. This approach shows the powerful implementation of RoSA and its self-aware functionalities. Furthermore, this approach would overcome not only the issue of a bottleneck but also enables highly systematic monitoring of the SuO.

3) SUMMARY

Context-aware detection of different signal and system states is now enabled in RoSA by corresponding functionalities. However, no state detector was implemented when we started to develop the application. In the experimental phase, we implemented state detection as custom code in combination with existing RoSA functionalities (abstraction, confidence, and history). Reusing functionalities in a modular way, facilitated our efforts to implement complex data processing for context-aware state detection. We turned the validated implementations of signal and system state detectors into functionalities, which can be reused and configured by application-specific rules.

VIII. DISCUSSION

Before concluding, we enumerate the lessons learned from developing RoSA and the open issues already identified. We organize the discussion in three themes: modeling self-awareness in Section VIII-A, the software implementation in Section VIII-B, and implementing on ES hardware in Section VIII-C.

A. MODELING SELF-AWARENESS

An important lesson was realizing how much application-dependent self-aware functionalities are. While a self-awareness property has some fundamental characteristic, a corresponding functionality may be implemented in different ways. For example, while confidence is a measure of how trustworthy the work of a task, part of the system, or the entire system is (Section VI-C), it may be calculated in many different ways [100] (see for example [31], [55], [101]). What interface to use for a self-awareness property depends on the actual usage. Therefore, each functionality must have a sophisticated interface to support modularity. This allows using functionalities directly or in combination with other functionalities to express more complex concepts. Whenever a new concept that cannot be built from the existing functionalities is to be developed, devising a modular interface for the new functionality is challenging but essential for reusability.

While the interfaces of functionalities are instrumental for reusability, details of their internal implementations can affect performance significantly. We provide a set of functionalities in RoSA; however, there might be better-working implementations. Hence, users of RoSA are not discouraged from adjusting and optimizing the implementations to their specific end-use. The modular design makes it possible to experiment with alternative implementations at will. In addition, users are encouraged to develop other functionalities whenever they have new ideas or specific needs.

We realize there is room for improving the modeling capabilities of RoSA. One limiting factor is the small set of implemented self-aware functionalities. Our research effort in self-awareness properties and functionality implementations will continue. We foresee exciting challenges in the area

TABLE 2. Static and dynamic characteristics of the case studies (Section VII) executed on different ES platforms with different ARM cores (Cortex-A7 and Cortex-A15 implement the 32-bit ARMv7-A architecture, Cortex-A53 implements the 64-bit ARMv8-A architecture) shows that each application can work in real-time on ES hardware; note that numbers of different applications are not to be compared as they implement independent algorithms.

Metrics	EWS			SA-EWS 1			SA-EWS 2			CAH/CCAM		
Cortex-	A7	A15	A53	A7	A15	A53	A7	A15	A53	A7	A15	A53
Executable Binary Size (kB)	798	798	817	911	911	933	1079	1079	1105	662	662	678
Maximum Allocated Memory* (kB)	3508	3508	3272	3584	3584	3440	3656	3656	3624	3348	3348	3404
Average Sample Processing Time** (ms)	0.46	0.22	0.39	0.74	0.35	0.64	1.64	0.77	1.50	7.41	4.37	7.21
Real-Time Sampling Period*** (ms)	1000	1000	1000	1000	1000	1000	1000	1000	1000	33	33	33

* Memory size includes code and all data during execution.
** The average is based on the total processing time including initialization of the application and reading sensor input files and writing output file, in single-threaded execution.
*** Maximum acceptable sample processing time for real-time execution based on application requirements.

and hope for the community's contribution in tackling them together to make RoSA a powerful common framework.

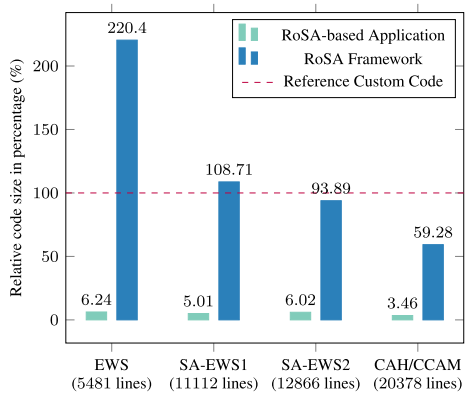


FIGURE 17. Ratio of non-comment codelines of RoSA-based application code (*Application*) and that of the RoSA framework itself (*Framework*) relative to the number of non-comment codelines of corresponding custom-written applications in our case studies. The base-line (actual number of codelines) for each application is indicated below the application names.

B. SOFTWARE IMPLEMENTATION

We made RoSA to accelerate and simplify research on self-awareness through a reusable software framework. Comparing the number of non-comment codelines (as an indicator of development effort) of our original custom-written applications and that of the RoSA-based implementation provides a quantitative measure of how much the development effort is simplified by using RoSA. This comparison for the presented case studies is shown in Figure 17, where the custom code of the corresponding application is the reference, meaning 100%. The *RoSA-based Application* sizes relative to the corresponding custom code show that RoSA-based implementations stay relatively small (3.46%–6.24%), independently from the size of the custom-written applications. Those implementations are small because they depend on the framework. The *RoSA Framework* size relative to the custom applications (the 100% references) shows that the overhead posed by the framework reduces (from an overhead of 120.4% for EWS to −40.72% for CAH/CCAM) as the size of the application increases (from 5481 to 20378 lines of custom application code for EWS and CAH/CCAM, respectively). The negative framework overhead indicates that even a custom implementation might be sub-optimal in case of complex applications. The quality of maintained framework code improves over time, while that does not typically happen with custom implementations developed in one go.

Averaging the sizes of all cases, we observe that on average, a RoSA-based implementation consists of only 5.18% lines of code relative to the custom implementation. The framework code has on average 20.75% more codelines than a custom application. The framework is, however, to be implemented only once and reused any number of times. Implementing a framework pays off when used for several applications. Particularly, implementing the framework and the four presented case studies in RoSA needed in total only 29.01% of the total number of codelines of our four original custom-written applications together; in other words, in total we needed 70.99% less codelines for implementing all four applications with RoSA. These figures confirm our initial hypothesis: an appropriate framework reduces the modeling and development efforts in self-aware systems.

Lastly, we realize that the capabilities and usability of RoSA as a software framework can be enhanced. For example, a graphical interface could help non-programmers to interact with models. Studying typical model patterns on both the agent system and the agent levels could help application developers in making better designs and utilizing available features efficiently.

C. IMPLEMENTING ON EMBEDDED SYSTEM HARDWARE

RoSA is a standalone actor framework with an open-source standard native implementation, to which CAF is the most similar from the existing agent-based frameworks (discussed

in Section III-F). For deploying RoSA in ESs, we limited the implemented features to the essentials for our case studies without limiting the generality of the agent system. The binary size of the RoSA libraries on a x86-64 linux machine is 300 kB, while that of the CAF core library (version 0.17.5) is 7248 kB. This significant (24 times) difference in favor of RoSA indicates that our framework is applicable to considerably smaller systems than CAF.

As a preliminary confirmation of this hypothesis (suitability for ESs), we ran our case studies on the *ODROID XU4* [102] and *Raspberry Pi 3* [103] systems. The former has an eight-core *big.LITTLE* [104] configuration with *Cortex-A7* and *Cortex-A15* cores and the latter has a quad-core configuration with *Cortex-A53* cores. The applications posed a moderate memory footprint well below 4 MB, which fits typical ESs, and processed samples several times faster than required for real-time execution. In Table 2, we have summarized the characteristics of each application implemented on each platform, where the real-time requirements and actual average processing times can be found. It has to be noted that the real-time sampling period depends on the nature of the corresponding application and that the table is not meant to compare the different applications. We plan to extend the evaluation of our software implementation by performing further extensive and vigorous tests by deploying RoSA on other real ES hardware in the future.

IX. CONCLUSION

Self-awareness is a hot topic, but related research and development efforts are fragmented among different fields and communities. Self-aware systems are developed from scratch in many cases. Such method of development, on long term, is redundant, inefficient and uneconomic. A major reason behind this fragmentation is the lack of a common framework that would facilitate development, cooperation, and reuse of existing results.

In this paper, we presented RoSA, a framework that aims to help researchers and engineers to explore the novel design space of self-awareness. RoSA supports modeling of self-aware applications as *agent systems* and modeling of agents based on self-aware *functionalities*. We presented the design principles of the RoSA architecture as well as use cases of RoSA-based modeling for different scenarios. The description of self-aware functionalities offered by RoSA and detailed case studies about applications implemented in RoSA demonstrate the modeling power and applicability of the framework.

Using RoSA relieves application developers from taking care of handling agents and message passing. Predefined functionalities serve as reusable components for defining individual agents. Data processing within agents can be defined as an arbitrary combination of custom application code and existing functionalities. Application code can thus be limited to the important aspects: (i) data processing within agents and (ii) the agent hierarchy of the application.

We promote RoSA as a vehicle for researchers to study various concepts that are related to self-awareness and the relation among them; and also for engineers to prototype and evaluate self-aware features in their designs with ease.

ABBREVIATIONS

AC	Autonomic Computing.
AI	Artificial Intelligence.
CAF	C++ Actor Framework.
CAH	Context-aware Health Monitoring.
CCAM	Confidence-based Context-Aware condition Monitoring.
CPS	Cyber-Physical System.
CSA	Computational Self-Awareness.
DARPA	Defense Advanced Research Projects Agency.
ES	Embedded System.
EWS	Early Warning Score.
IBM	International Business Machines Corporation.
IT	Information Technology.
MAS	Multi-agent System.
MPSoC	Multi-Processor System-on-Chip.
NASA	National Aeronautics and Space Administration.
ODA	Observe-Decide-Act.
RoSA	Research on Self-Awareness.
SoC	System on Chip.
SuO	System under Observation.

ACKNOWLEDGMENT

(Maximilian Götzinger and Dávid Juhász contributed equally to this work.) The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme.

REFERENCES

[1] J. Rivera and R. van der Meulen. (Nov. 2014). *Gartner Says 4.9 Billion Connected 'Things' Will be in use in 2015*. [Online]. Available: http://www.gartner.com/newsroom/id/2905717

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128610001568

[3] M. Yaqoob, S. R. Qaisrani, M. Waqas, Y. Ayaz, S. Iqbal, and S. Nisar, "Control of robotic arm manipulator with haptic feedback using programmable system on chip," in *Proc. Int. Conf. Robot. Emerg. Allied Technol. Eng. (iCREATE)*, Apr. 2014, pp. 300–305.

[4] D. Genius, E. Faure, and N. Pouillon, "Mapping a telecommunication application on a multiprocessor system-on-chip," in *Algorithm-Architecture Matching for Signal and Image Processing*. Dordrecht, The Netherlands: Springer, 2011, pp. 53–77.

[5] S. Mukhopadhyay, M. Heddes, M. Ravasi, and M. S. Yeo, "System-on-a-chip and multi-chip systems supporting advanced telecommunication functions," U.S. Patent 12 342 625, Jun. 24, 2010.

[6] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "Computer architectures for autonomous driving," *Computer*, vol. 50, no. 8, pp. 18–25, 2017.

[7] X. Chen, X. Jiang, and L. Wang, "Development on ARM9 System-on-chip embedded sensor node for urban intelligent transportation system," in *Proc. IEEE Int. Symp. Ind. Electron.*, vol. 4, Jul. 2006, pp. 3270–3275.

[8] N. Moreira, J. Lazaro, U. Bidarte, J. Jimenez, and A. Astarloa, "On the utilization of system-on-chip platforms to achieve nanosecond synchronization accuracies in substation automation systems," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1932–1942, Jul. 2017.

[9] B. Massot, C. Gehin, R. Nocua, A. Dittmar, and E. McAdams, "A wearable, low-power, health-monitoring instrumentation based on a programmable system-on-chipTM," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Sep. 2009, pp. 4852–4855.

[10] L. Gurgen, O. Gunalp, Y. Benazzouz, and M. Galissot, "Self-aware cyber-physical systems and applications in smart buildings and cities," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2013, pp. 1149–1154.

[11] A. Jantsch, A. Anzanpour, H. Kholerdi, I. Azimi, L. C. Siafara, A. M. Rahmani, N. TaheriNejad, P. Liljeberg, and N. Dutt, "Hierarchical dynamic goal management for IoT systems," in *Proc. 19th Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2018, pp. 370–375.

[12] M. Shafique, D. Gnad, S. Garg, and J. Henkel, "Variability-aware dark silicon management in on-chip many-core systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib.* San Jose, CA, USA: EDA Consortium, Mar. 2015, pp. 387–392.

[13] W. Huang, M. R. Stant, K. Sankaranarayanan, R. J. Ribando, and K. Skadron, "Many-core design from a thermal perspective," in *Proc. 45th Annu. Conf. Design Autom. (DAC)*, Jun. 2008, pp. 746–749.

[14] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: Survey of current and emerging trends," in *Proc. 50th Annu. Design Autom. Conf. (DAC)*. New York, NY, USA: ACM, 2013, pp. 1:1–1:10

[15] N. Dutt, A. Jantsch, and S. Sarma, "Self-aware cyber-physical systems-on-chip," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, Nov. 2015, pp. 46–50.

[16] P. R. Lewis, M. Platzner, B. Rinner, J. Tørresen, and X. Yao, *Self-Aware Computing Systems: An Engineering Approach* (Natural Computing Series), 1st ed. Cham, Switzerland: Springer, 2016, doi: 10.1007/978-3-319-39675-0.

[17] N. Dutt and N. TaheriNejad, "Self-awareness in cyber-physical systems," in *Proc. 29th Int. Conf. VLSI Design 15th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2016, pp. 5–6.

[18] K. Bellman, N. Dutt, L. Esterle, A. Herkersdorf, A. Jantsch, C. Landauer, P. R. Lewis, M. Platzner, N. TaheriNejad, and K. Tammemäe, "Self-aware cyber-physical systems," *ACM Trans. Cyber-Phys. Syst.*, vol. 4, no. 4, pp. 1–24, 2020.

[19] J.-S. Preden, K. Tammemäe, A. Jantsch, M. Leier, A. Riid, and E. Calis, "The benefits of self-awareness and attention in fog and mist computing," *Computer*, vol. 48, no. 7, pp. 37–45, Jul. 2015.

[20] F. Forooghifar, A. Aminifar, and D. Atienza, "Resource-aware distributed epilepsy monitoring using self-awareness from edge to cloud," *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 6, pp. 1338–1350, Dec. 2019.

[21] S. Kounev, X. Zhu, J. O. Kephart, and M. Kwiatkowska, "Model-driven algorithms and architectures for self-aware computing systems (Dagstuhl seminar 15041)," *Dagstuhl Rep.*, vol. 5, no. 1, pp. 164–196, 2015.

[22] T. Chen, F. Faniyi, R. Bahsoon, P. R. Lewis, X. Yao, L. L. Minku, and L. Esterle, "The handbook of engineering self-aware and self-expressive systems," *CoRR*, vol. abs/1409.1793, pp. 1–81, Sep. 2014. [Online]. Available: http://arxiv.org/abs/1409.1793

[23] H. Psaier and S. Dustdar, "A survey on self-healing systems: Approaches and systems," *Computing*, vol. 91, no. 1, pp. 43–73, Jan. 2011.

[24] P. R. Lewis, A. Chandra, F. Faniyi, K. Glette, T. Chen, R. Bahsoon, J. Torresen, and X. Yao, "Architectural aspects of self-aware and self-expressive computing systems: From psychology to engineering," *Computer*, vol. 48, no. 8, pp. 62–70, Aug. 2015.

[25] T. Bures, D. Weyns, B. Schmerl, J. Fitzgerald, A. Aniculaesei, C. Berger, J. Cambeiro, J. Carlson, S. A. Chowdhury, M. Daun, and N. Li, "Software engineering for smart cyber-physical systems (SEsCPS 2018)-workshop report," *ACM SIGSOFT Softw. Eng. Notes*, vol. 44, no. 4, pp. 11–13, 2019.

[26] M. Götzinger, N. Taherinejad, A. M. Rahmani, P. Liljeberg, A. Jantsch, and H. Tenhunen, "Enhancing the early warning score system using data confidence," in *Proc. Int. Conf. Wireless Mobile Commun. Healthcare*. Cham, Switzerland: Springer, 2016, pp. 91–99.

[27] M. Götzinger, A. Anzanpour, I. Azimi, N. Taherinejad, and A. M. Rahmani, "Enhancing the self-aware early warning score system through fuzzified data reliability assessment," in *Proc. Int. Conf. Wireless Mobile Commun. Healthcare*. Cham, Switzerland: Springer, 2017, pp. 3–11.

[28] M. Götzinger, A. Anzanpour, I. Azimi, N. TaheriNejad, A. Jantsch, A. M. Rahmani, and P. Liljeberg, "Confidence-enhanced early warning score based on fuzzy logic," *Mobile Netw. Appl.*, vol. 8, pp. 1–18, Aug. 2019.

[29] M. Gotzinger, N. TaheriNejad, H. A. Kholerdi, and A. Jantsch, "On the design of context-aware health monitoring without *a priori* knowledge; an AC-motor case-study," in *Proc. IEEE 30th Can. Conf. Electr. Comput. Eng. (CCECE)*, Apr. 2017, pp. 1–5.

[30] M. Gotzinger, E. Willegger, N. TaheriNejad, A. Jantsch, T. Sauter, T. Glatzl, and P. Lilieberg, "Applicability of context-aware health monitoring to hydraulic circuits," in *Proc. IECON-44th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2018, pp. 4712–4719.

[31] M. Götzinger, N. TaheriNejad, H. A. Kholerdi, A. Jantsch, E. Willegger, T. Glatzl, A. M. Rahmani, T. Sauter, and P. Liljeberg, "Model-free condition monitoring with confidence," *Int. J. Comput. Integr. Manuf.*, vol. 32, nos. 4–5, pp. 466–481, May 2019.

[32] F. Faniyi, P. R. Lewis, R. Bahsoon, and X. Yao, "Architecting self-aware software systems," in *Proc. IEEE/IFIP Conf. Softw. Archit.*, Apr. 2014, pp. 91–94.

[33] L. Guang, E. Nigussie, J. Plosila, J. Isoaho, and H. Tenhunen, "Survey of self-adaptive NoCs with energy-efficiency and dependability," *Int. J. Embedded Real-Time Commun. Syst.*, vol. 3, no. 2, pp. 1–22, Apr. 2012.

[34] J. Schlingensiepen, F. Nemtanu, R. Mehmood, and L. McCluskey, "Autonomic transport management systems—Enabler for smart cities, personalized medicine, participation and industry grid/industry 4.0," in *Intelligent Transportation Systems–Problems and Perspectives*. Cham, Switzerland: Springer, 2016, pp. 3–35.

[35] D. B. Abeywickrama and E. Ovaska, "A survey of autonomic computing methods in digital service ecosystems," *Service Oriented Comput. Appl.*, vol. 11, no. 1, pp. 1–31, Mar. 2017.

[36] M. Parashar and S. Hariri, "Autonomic computing: An overview," in *Unconventional Programming Paradigms*, J.-P. Banâtre, P. Fradet, J.-L. Giavitto, and O. Michel, Eds. Berlin, Germany: Springer, 2005, pp. 257–269.

[37] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing—Degrees, models, and applications," *ACM Comput. Surv.*, vol. 40, no. 3, pp. 1–28, Aug. 2008, doi: 10.1145/1380584.1380585.

[38] A. L. Randall and R. C. Walter, "Overview of the small unit operations situational awareness system," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, vol. 1, Oct. 2003, pp. 169–173.

[39] M. Rahman, R. Ranjan, R. Buyya, and B. Benatallah, "A taxonomy and survey on autonomic management of applications in grid computing environments," *Concurrency Comput., Pract. Exp.*, vol. 23, no. 16, pp. 1990–2019, Nov. 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.1734

[40] J. O. Kephart, "Research challenges of autonomic computing," in *Proc. 27th Int. Conf. Softw. Eng. (ICSE)*, 2005, pp. 15–22.

[41] A. G. Ganek and T. A. Corbi, "The dawning of the autonomic computing era," *IBM Syst. J.*, vol. 42, no. 1, pp. 5–18, 2003.

[42] D. Sinreich, "An architectural blueprint for autonomic computing," IBM Corp., Armonk, NY, USA, White paper, 2006. [Online]. Available: https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf

[43] P. Lalanda, J. A. McCann, and A. Diaconescu, *Autonomic Computing: Principles, Design and Implementation*. London, U.K.: Springer, 2013.

[44] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[45] J. Cámara, K. L. Bellman, J. O. Kephart, M. Autili, N. Bencomo, A. Diaconescu, H. Giese, S. Götz, P. Inverardi, S. Kounev, and M. Tivoli, *Self-aware Computing Systems: Related Concepts and Research Areas*. Cham, Switzerland: Springer, 2017, pp. 17–49, doi: 10.1007/978-3-319-47474-8_2.

[46] D. F. Bantz, C. Bisdikian, D. Challener, J. P. Karidis, S. Mastrianni, A. Mohindra, D. G. Shea, and M. Vanover, "Autonomic personal computing," *IBM Syst. J.*, vol. 42, no. 1, pp. 165–176, 2003.

[47] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Trans. Auto. Adapt. Syst.*, vol. 4, no. 2, p. 14, 2009.

[48] C. Landauer and K. L. Bellman, "An architecture for self-awareness experiments," in *Proc. IEEE Int. Conf. Autonomic Comput. (ICAC)*, Jul. 2017, pp. 255–262.

[49] K. L. Bellman, "An approach to integrating and creating flexible software environments supporting the design of complex systems," in *Proc. Winter Simul. Conf.*, 1991, pp. 1101–1105.
[50] M. Salehie and L. Tahvildari, "Autonomic computing: Emerging trends and open problems," *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–7, 2005.
[51] H. Giese, T. Vogel, A. Diaconescu, S. Götz, N. Bencomo, K. Geihs, S. Kounev, and K. L. Bellman, *State of the Art in Architectures for Self-Aware Computing Systems*. Cham, Switzerland: Springer, 2017, pp. 237–275, doi: 10.1007/978-3-319-47474-8_8.
[52] N. Dutt, A. Jantsch, and S. Sarma, "Toward smart embedded systems: A self-aware system-on-chip (SOC) perspective," *ACM Trans. Embed. Comput. Syst.*, vol. 15, no. 2, pp. 22:1–22:27, Feb. 2016.
[53] P. Mercati, A. Bartolini, F. Paterna, T. S. Rosing, and L. Benini, "A linux-governor based dynamic reliability manager for Android mobile devices," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2014, pp. 1–4.
[54] B. Rinner, L. Esterle, J. Simonjan, G. Nebehay, R. Pflugfelder, G. F. Dominguez, and P. R. Lewis, "Self-aware and self-expressive camera networks," *Computer*, vol. 48, no. 7, pp. 21–28, Jul. 2015.
[55] F. Forooghifar, A. Aminifar, and D. A. Alonso, "Self-aware wearable systems in epileptic seizure detection," in *Proc. 21st Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2018, pp. 426–432.
[56] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *J. Netw. Syst. Manage.*, vol. 23, no. 3, pp. 567–619, Jul. 2015, doi: 10.1007/s10922-014-9307-7.
[57] P. Spathis and M. D. D. Bicudo, "Ana: Autonomic network architecture," in *Autonomic Network Management Principles: From Concepts to Applications*. Oxford, U.K.: Academic, 2011, p. 49.
[58] L. Wanner, S. Elmalaki, L. Lai, P. Gupta, and M. Srivastava, "VarEMU: An emulation testbed for variability-aware software," in *Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, Sep. 2013, pp. 1–10.
[59] J. Strassner, S.-S. Kim, and J. W.-K. Hong, "The design of an autonomic communication element to manage future Internet services," in *Management Enabling the Future Internet for Changing Business and New Computing Services*. Berlin, Germany: Springer, 2009, pp. 122–132.
[60] W. Baek and T. M. Chilimbi, "Green: A framework for supporting energy-conscious programming using controlled approximation," in *Proc. ACM SIGPLAN Notices*, vol. 45, no. 6, 2010, pp. 198–209.
[61] H. Hoffmann, M. Maggio, M. D. Santambrogio, A. Leva, and A. Agarwal, "SEEC: A framework for self-aware computing," MIT, Cambrige, MA, USA, Tech. Rep. MIT-CSAIL-TR-2010-049, Oct. 2010.
[62] E. Shamsa, A. Kanduri, N. TaheriNejad, A. Probstl, S. Chakraborty, A. M. Rahmani, and P. Liljeberg, "User-centric resource management for embedded multi-core processors," in *Proc. 33rd Int. Conf. VLSI Design 19th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2020, pp. 1–6.
[63] A. Akbar and P. R. Lewis, "Self-adaptive and self-aware mobile-cloud hybrid robotics," in *Proc. 5th Int. Conf. Internet Things, Syst., Manage. Secur.*, Oct. 2018, pp. 262–267.
[64] L. C. Siafara, H. A. Kholerdi, A. Bratukhin, N. TaheriNejad, A. Wendt, A. Jantsch, A. Treytl, and T. Sauter, "SAMBA: A self-aware health monitoring architecture for distributed industrial systems," in *Proc. IECON-43rd Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2017, pp. 3512–3517.
[65] L. C. Siafara, H. Kholerdi, A. Bratukhin, N. Taherinejad, and A. Jantsch, "SAMBA–an architecture for adaptive cognitive control of distributed cyber-physical production systems based on its self-awareness," *e i Elektrotechnik und Informationstechnik*, vol. 135, no. 3, pp. 270–277, Jun. 2018, doi: 10.1007/s00502-018-0614-7.
[66] K. Nymoen, A. Chandra, and J. Torresen, "Self-awareness in active music systems," *Self-Aware Computing Systems*. Cham, Switzerland: Springer, 2016, pp. 279–296, doi: 10.1007/978-3-319-39675-0_14.
[67] J. Teich, J. Henkel, A. Herkersdorf, D. Schmitt-Landsiedel, W. Schröder-Preikschat, and G. Snelting, "Invasive computing: An overview," in *Multiprocessor System-on-Chip: Hardware Design and Tool Integration*, M. Hübner and J. Becker, Eds. Berlin, Germany: Springer, 2011, pp. 241–268.
[68] A. Bouajila, J. Zeppenfeld, W. Stechele, A. Bernauer, O. Bringmann, W. Rosenstiel, and A. Herkersdorf, "Autonomic system on chip platform," in *Organic Computing—A Paradigm Shift for Complex Systems* (Autonomic Systems), C. Müller-Schloer, H. Schmeck, and T. Ungerer, Eds. Basel, Switzerland: Birkhäuser, 2011, ch. 4.7, pp. 413–425.
[69] A. Bouajila, J. Zeppenfeld, W. Stechele, A. Herkersdorf, A. Bernauer, O. Bringmann, and W. Rosenstiel, "Organic computing at the system on chip level," in *Proc. IFIP Int. Conf. Very Large Scale Integr.*, Oct. 2006, pp. 338–341.
[70] G. Kornaros and D. Pnevmatikatos, "A survey and taxonomy of on-chip monitoring of multicore systems-on-chip," *ACM Trans. Design Autom. Electron. Syst.*, vol. 18, no. 2, pp. 1–38, Mar. 2013.
[71] N. T. Nejad, M. A. Shami, and P. D. S. Manoj, "Self-aware sensing and attention-based data collection in multi-processor system-on-chips," in *Proc. 15th IEEE Int. New Circuits Syst. Conf. (NEWCAS)*, Jun. 2017, pp. 81–84.
[72] S. M. Jafri, L. Guang, A. Jantsch, K. Paul, A. Hemani, and H. Tenhunen, "Self-adaptive NoC power management with dual-level agents-architecture and implementation," in *Proc. PECCS*, 2012, pp. 450–458.
[73] S. Kounev, P. Lewis, K. Bellman, N. Bencomo, J. Camara, A. Diaconescu, L. Esterle, K. Geihs, H. Giese, S. Götz, P. Inverardi, J. Kephart, and A. Zisman, "The notion of self-aware computing," in *Self-Aware Computing Systems*, S. Kounev, J. O. Kephart, A. Milenkoski, and X. Zhu, Eds. Cham, Switzerland: Springer, 2017, pp. 3–16.
[74] J. Kramer and J. Magee, "Self-managed systems: An architectural challenge," in *Proc. Future Softw. Eng. (FOSE)*, May 2007, pp. 259–268.
[75] U. Aßmann, S. Götz, J.-M. Jézéquel, B. Morin, and M. Trapp, *A Reference Architecture and Roadmap for Models Run.Time Systems*. Cham, Switzerland: Springer, 2014, pp. 1–18, doi: 10.1007/978-3-319-08915-7_1.
[76] L. Guang, E. Nigussie, J. Isoaho, P. Rantala, and H. Tenhunen, "Interconnection alternatives for hierarchical monitoring communication in parallel SoCs," *Microprocessors Microsyst.*, vol. 34, no. 5, pp. 118–128, Aug. 2010.
[77] M. Viroli, D. Pianini, S. Montagna, and G. Stevenson, "Pervasive ecosystems: A coordination model based on semantic chemistry," in *Proc. 27th Annu. ACM Symp. Appl. Comput.* New York, NY, USA: ACM, 2012, pp. 295–302.
[78] C. Savaglio, G. Fortino, and M. Zhou, "Towards interoperable, cognitive and autonomic IoT systems: An agent-based approach," in *Proc. IEEE 3rd World Forum Internet Things (WF-IoT)*, Dec. 2016, pp. 58–63.
[79] I. Carreras, I. Chlamtac, F. De Pellegrini, and D. Miorandi, "BIONETS: Bio-inspired networking for pervasive communication environments," *IEEE Trans. Veh. Technol.*, vol. 56, no. 1, pp. 218–229, Jan. 2007.
[80] A. Bucchiarone, "Collective adaptation through multi-agents ensembles: The case of smart urban mobility," *ACM Trans. Auto. Adapt. Syst.*, vol. 14, no. 2, pp. 1–28, Dec. 2019.
[81] A. Bucchiarone, M. De Sanctis, A. Marconi, and A. Martinelli, "DeMOCAS: Domain objects for service-based collective adaptive systems," in *Service-Oriented Computing—ICSOC 2016 Workshops*. Cham, Switzerland: Springer, 2017, pp. 174–178. [Online]. Available: http://link.springer.com/10.1007/978-3-319-68136-8_19
[82] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. London, U.K.: Pearson, 2010.
[83] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowl. Eng. Rev.*, vol. 10, no. 2, pp. 115–152, Jun. 1995.
[84] C. Hewitt, "Actor model of computation for scalable robust information systems," in *Proc. Symp. Logic Collaboration Intell. Appl.*, 2017, pp. 1–91.
[85] A. Sadighi, B. Donyanavard, T. Kadeed, K. Moazzemi, T. Muck, A. Nassar, A. M. Rahmani, T. Wild, N. Dutt, R. Ernst, A. Herkersdorf, and F. Kurdahi, "Design methodologies for enabling self-awareness in autonomous systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1532–1537.
[86] L. Guang, "Hierarchical agent-based adaptation for self-aware embedded computing systems," Ph.D. dissertation, Dept. Inf. Technol., Univ. Turku, Turku, Finland, 2012.
[87] J. Hunt, *Introduction to Akka Actors*. Springer, 2014, pp. 383–398.
[88] D. Charousset, R. Hiesgen, and T. C. Schmidt, "Revisiting actor programming in C++," *Comput. Lang., Syst. Struct.*, vol. 45, pp. 105–131, Apr. 2016.
[89] P. Taillandier, B. Gaudou, A. Grignard, Q.-N. Huynh, N. Marilleau, P. Caillou, D. Philippon, and A. Drogoul, "Building, composing and experimenting complex spatial models with the GAMA platform," *GeoInformatica*, vol. 23, no. 2, pp. 299–322, Apr. 2019.

[90] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi, *Jade—A Java Agent Development Framework*. Boston, MA, USA: Springer, 2005, pp. 125–147.
[91] B. Chen, H. H. Cheng, and J. Palen, "Mobile-C: A mobile agent platform for mobile C/C++ agents," *Softw. Pract. Exper.*, vol. 36, no. 15, pp. 1711–1733, 2006.
[92] N. Collier and M. North, "Parallel agent-based simulation with repast for high performance computing," *Simulation*, vol. 89, no. 10, pp. 1215–1235, Oct. 2013.
[93] M. J. North, N. T. Collier, J. Ozik, E. R. Tatara, C. M. Macal, M. Bragen, and P. Sydelko, "Complex adaptive systems modeling with repast simphony," *Complex Adapt. Syst. Model.*, vol. 1, no. 1, p. 3, Dec. 2013.
[94] N. TaheriNejad, A. Jantsch, and D. Pollreisz, "Comprehensive observation and its role in self-awareness; an emotion recognition system example," in *Proc. Position Papers Federated Conf. Comput. Sci. Inf. Syst.*, Gdansk, Poland, Oct. 2016, pp. 117–124.
[95] A. Jantsch and K. Tammemäe, "A framework of awareness for artificial subjects," in *Proc. 2014 Int. Conf. Hardw./Softw. Codesign Syst. Synth.* New York, NY, USA: ACM, 2014, pp. 20:1–20:3.
[96] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, Jun. 1965.
[97] J. McGaughey, F. Alderdice, R. Fowler, A. Kapila, A. Mayhew, and M. Moutray, "Outreach and early warning systems (EWS) for the prevention of intensive care admission and death of critically ill adult patients on general hospital wards," *Cochrane Library*, vol. 2007, no. 3, pp. CD005529:1–CD005529:24, Jul. 2007.
[98] R. J. Morgan, F. Williams, and M. M. Wright, "An early warning scoring system for detecting developing critical illness," *Clin. Intensive Care*, vol. 8, no. 2, p. 100, 1997.
[99] W. Thomson and R. Gilmore, "Motor current signature analysis to detect faults in induction motor drives–fundamentals, data interpretation, and industrial case histories," in *Proc. 32nd Turbomachinery Symp.*, Sep. 2003, pp. 145–156.
[100] N. TaheriNejad and A. Jantsch, "Improved machine learning using confidence," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2019, pp. 1–5.
[101] H. A. Kholerdi, N. TaheriNejad, and A. Jantsch, "Enhancement of classification of small data sets using self-awareness—An iris flower case-study," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.
[102] HADRKERNEL. (2017). *ODROID-XU4 Manual*. [Online]. Available: https://magazine.odroid.com/wp-content/uploads/odroid-xu4-user-manual.pdf
[103] Raspberry Pi (Trading) Ltd. (2019). *Raspberry Pi Compute Module 3+ Datasheet*. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM3plus_1p0.pdf
[104] LITTLE Technology. (2013). *The Future of Mobile*. [Online]. Available: https://www.arm.com/files/pdf/big_LITTLE_Technology_the_Futue_of_Mobile.pdf

MAXIMILIAN GÖTZINGER (Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering and information technology from TU Wien (formerly known as the Vienna University of Technology as well), Vienna, Austria, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree in computer science with the Department of Future Technologies, University of Turku.

He is also with the Institute of Computer Technology, TU Wien, as a Project Assistant and a Teacher. He has a keen and serious interest in computer science and engineering, as well as teaching. His research interest includes computational self-awareness, for which he is conducting many case studies, such as health and system monitoring. He has published ten peer-reviewed papers, for one of which he received the Best Paper Award. In 2019, he received the one Best Teacher Award and the one Best Lecturer Award for the course digital systems.

DÁVID JUHÁSZ received the B.Sc. and M.Sc. degrees in computer science from Eötvös Loránd University, Budapest, Hungary, in 2010 and 2012, respectively. He is currently pursuing the Ph.D. degree with the Institute of Computer Technology, TU Wien, Vienna, Austria.

He is an Early Stage Researcher of the oCPS Marie Curie ITN Project at TU Wien. He is also a Lead Software Architect at Imsys AB, Stockholm, Sweden. His research interests include development methodologies and runtime systems that enable efficient utilization of complex hardware solutions via a high-level software environment. His current research interests include self-aware systems and execution issues of the state-of-the-art hardware platforms focusing on non-functional requirements. He had contributed to software development on different levels of abstraction as well as design and implementation questions of programming languages, runtime systems, and instruction set architectures.

NIMA TAHERINEJAD (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from The University of British Columbia, Vancouver, Canada, in 2015.

He is currently a "Universitätsassistant" at TU Wien (formerly known also as the Vienna University of Technology), Vienna, Austria, where his areas of work include self-awareness in resource-constrained cyber-physical systems, embedded systems, systems on chip, health-care, memristor-based circuit and systems, and robotics. He has published two books and more than 45 peer-reviewed articles. He received several awards and scholarships from universities, conferences, and workshops he has attended. He has also served as a reviewer, an editor, an organizer, and the chair for various journals, conferences, and workshops.

EDWIN WILLEGGER received the B.Sc. degree in electrical engineering from TU Wien, Vienna, Austria, in 2018, where he is currently pursuing the master's degree in microelectronics and photonics.

From 2014 to 2015, he was with Siemens Austria and was doing research on preventive maintenance of complex mechanical systems. Since 2015, he has been a Research Assistant at TU Wien. His research interest includes the development of self-aware hardware and software systems.

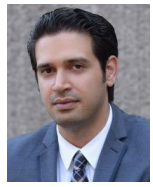
BENEDIKT TUTZER received the B.S. degree in computer engineering from TU Wien, Vienna, Austria, in 2018, where he is currently pursuing the master's degree in embedded systems.

In 2017 and 2018, he was with the Interactive Media Systems Group, TU Wien, researching the applications of virtual-reality headsets as a seeing aid for visually impaired patients at the Vienna General Hospital. Since 2018, he has been with the Institute of Computer Technology, TU Wien, focusing on electronic design automation.

PASI LILJEBERG (Member, IEEE) received the M.Sc. and Ph.D. degrees in information and communication technology from the University of Turku, Turku, Finland, in 1999 and 2005, respectively. He received an Adjunct Professorship in embedded computing architectures, in 2010. He is currently a Full Professor with the Digital Health Technology, University of Turku. He has authored more than 300 peer-reviewed publications. His current research interests include biomedical engineering, the Internet of Things, fog computing, approximate and adaptive computing, wearable sensor, e-health technology, and health data analytics. In that context, he has established and leading the Internet-of-Things for Healthcare (IoT4Health) Research Group.

AXEL JANTSCH (Senior Member, IEEE) received the Dipl.Ing. and Ph.D. degrees in computer science from TU Wien, Vienna, Austria, in 1987 and 1992, respectively.

From 1997 to 2002, he was an Associate Professor at the KTH Royal Institute of Technology, Stockholm, where he was a full Professor in electronic systems design, from 2002 to 2014. Since 2014, he has been a Professor of systems on chips at the Institute of Computer Technology, TU Wien. He has published five books as an Editor and one as an Author, over 300 peer-reviewed contributions in journals, books, and conference proceedings. He has given over 100 invited presentations at conferences, universities, and companies. His current research interests include systems on chips, self-aware cyber-physical systems, and embedded machine learning.



AMIR M. RAHMANI (Senior Member, IEEE) is currently an Assistant Professor of computer science and nursing (joint appointment) at UCI and is also a Life-Time Adjunct Professor (Docent) at the Department of Future Technologies, University of Turku, Turku, Finland. He is the Founder of the Health SciTech Group, University of California at Irvine (UCI), and the Co-Founder of the Internet-of-Things for Healthcare Group (IoT4Health), University of Turku (UTU). He has coauthored more than 200 peer-reviewed publications. His research interests include the Internet of Things (IoT), e-health, wearable sensor design, bio-signal processing, health informatics, and big health data analytics. He is especially excited about novel sensing, computation/analytics, communication, and networking paradigms, applied to healthcare/medical and well-being applications. He is the Associate Editor-in-Chief of the *ACM Transactions on Computing for Healthcare*.

Maximilian Götzinger, Arman Anzanpour, Iman Azimi, Nima TaheriNejad, Axel Jantsch, Amir M. Rahmani, and Pasi Liljeberg

II

Confidence-Enhanced Early Warning Score Based on Fuzzy Logic

Mobile Networks and Applications, 2019; 8: 1–18

Mobile Networks and Applications
https://doi.org/10.1007/s11036-019-01324-5

Confidence-Enhanced Early Warning Score Based on Fuzzy Logic

Maximilian Götzinger[1,2] · Arman Anzanpour[1] · Iman Azimi[1] · Nima TaheriNejad[2] · Axel Jantsch[2] · Amir M. Rahmani[3,4] · Pasi Liljeberg[1]

Abstract

Cardiovascular diseases are one of the world's major causes of loss of life. The vital signs of a patient can indicate this up to 24 hours before such an incident happens. Healthcare professionals use Early Warning Score (EWS) as a common tool in healthcare facilities to indicate the health status of a patient. However, the chance of survival of an outpatient could be increased if a mobile EWS system would monitor them during their daily activities to be able to alert in case of danger. Because of limited healthcare professional supervision of this health condition assessment, a mobile EWS system needs to have an acceptable level of reliability - even if errors occur in the monitoring setup such as noisy signals and detached sensors. In earlier works, a data reliability validation technique has been presented that gives information about the trustfulness of the calculated EWS. In this paper, we propose an EWS system enhanced with the self-aware property confidence, which is based on fuzzy logic. In our experiments, we demonstrate that - under adverse monitoring circumstances (such as noisy signals, detached sensors, and non-nominal monitoring conditions) - our proposed Self-Aware Early Warning Score (SA-EWS) system provides a more reliable EWS than an EWS system without self-aware properties.

Keywords Early warning score · Self-awareness · Data reliability · Consistency · Plausibility · Confidence · Fuzzy logic · Hierarchical agent-based system

1 Introduction

Cardiovascular diseases are worldwide considered as one of the major causes of death [1]. The vital signs of a patient reflect the patient's health condition, and monitoring these vital signs establishes a basis for predicting a possible deterioration of the health condition. Even up to 24 hours before a sudden health deterioration occurs, specific symptoms are visible in the vital signs of a patient [2]. The assessment of the EWS of a patient's health condition is a common practice in hospitals and manually done by healthcare professionals. The EWS constitutes a number which indicates the level of criticality [3].

The availability of an autonomous mobile EWS system that constantly monitors patients' vital signs to calculate the EWS could increase the life expectancy of outpatients. High-risk patients could wear such a system which monitors them during their daily life activities and alert in case of an emergency. Besides a much higher survival rate, a mobile EWS system could also decrease costs related to healthcare and reduce the duration of hospitalization periods.

Internet of Things (IoT) - with its small devices and wearable technologies - is a key enabler to provide autonomous health monitoring for a mobile EWS system in a cost-efficient manner [4–7]. Such a system cannot be supervised continuously by healthcare professionals, but its reliability and the accuracy of the calculated EWS are of utter importance. The manual monitoring of a patient who is admitted and is lying in a hospital bed, done by healthcare professionals, faces much fewer problems than automated monitoring of a patient who is at home carrying out daily tasks [8]. One of the widely acknowledged and intrinsic challenges for wearable devices is the movement artifact [9]. Moreover, incorrectly attached or detached sensors, broken sensors, and noise can affect the calculation of the EWS that could lead to a false or - even worse - a missing alarm with all its consequences [10].

Self-awareness has various properties which help to make computer systems more autonomous, smarter, and reliable [11, 12]. Therefore, it can also be an enabler to make the monitoring of patients and the calculation of EWS more robust as well as reliable. In one of our previous works [13],

✉ Maximilian Götzinger
maxgot@utu.fi

Extended author information available on the last page of the article.

Published online: 29 August 2019

we already presented a data reliability assessment technique based on fuzzy logic, which gives information about the trustfulness of the calculated EWS. However, although the proposed system outputs a reliability value which correlates with the correctness of the monitored vital signs, the system can only provide an unmodified EWS, which is incorrect when the input data is corrupted. To improve the decision-making ability of a system, another self-aware property can be utilized, namely, *confidence*. In other words, data reliability and confidence are two self-aware properties that can enhance the conventional EWS system. Both reliability and confidence are metadata. Reliability is metadata of the given input data and provides information on to what degree the data is reliable; in this case, the system can trust its sensors. Besides, the system can make its decisions based on confidence, a meta-data for decisions, which have been motivated by observations of various pieces of information, and other metadata.

In this paper, we propose a self-aware EWS system which validates reliability and bases all decisions on a confidence assessment. These validations and assessments are techniques based on fuzzy logic. To show the effectiveness of these two mentioned self-aware properties, we recorded vital signs of a set of persons with high-quality and low-quality sensors. In our experiments, we demonstrate Self-Aware Early Warning Score (SA-EWS) system calculates the EWS correctly or with a small error close the to the value it should have even if the monitoring circumstances are adverse. The results show that our proposed SA-EWS system is more reliable than an EWS system without self-awareness. In other words, we prove that self-awareness is a good foundation for a reliable EWS system that trustfully classifies the EWS even if there is some faulty sensory data. Our main contributions are:

1. We propose a fuzzy logic based confidence metric for the quality assessment of the calculated EWS,
2. we show how a fuzzy logic based reliability metric gives information about the correctness of the input data,
3. we introduce a method for combing the input data reliability and the confidence of the system to calculate output data reliability based on both factors, and
4. using extended experiments, we demonstrate that our proposed system gives equally good or better results than a similar system that does not use reliability and confidence metrics.

After reviewing relevant related work in Section 2, we explain self-awareness properties reliability and confidence in Section 3. Section 4 shows system architecture as well as the implementation of our proposed system. While Section 5 explains the experimental setup and presents the results, finally, Section 6 concludes the paper.

2 Background and related work

In 1997, Morgan et al. proposed a medical method called EWS that is currently widely used in hospitals helping to determine the degree of patients' health deterioration. The patient's vital signs, such as respiration rate, heart rate, systolic blood pressure, body temperature, blood oxygen saturation (SpO_2), and the level of consciousness are manually collected in a regular routine and classified in different scores. These scores, ranging from 0 to 3, are determined according to the observations and predefined ranges of the vital signs. Table 1 indicates an EWS chart used for obtaining the various scores. In this chart, score 0 is allocated to a vital sign that is in perfect condition; e.g., heart rate in a range between 60 and 100. If the value of a vital sign is a bit worse than this (a bit too low or too high), the corresponding score is 1.[1] If the value of a vital sign is in even a worse condition (still higher or lower), the vital sign is classified to be score 2. Any value worse (depending on the case, higher or lower in absolute value) than the above ranges is classified as score 3.

The EWS is a simple aggregate of the scores that are abstracted from the patient's vital signs. The lower the calculated EWS, the better the patient's condition. A high EWS corresponds to a high risk of death or critical medical conditions [15]. Therefore, this likelihood reveals early signs of health deterioration and can be used to trigger a rapid response team to evaluate the patient. Similarly, an approach to predict potential sudden patient death have recently received FDA approval [16].

The EWS itself can be classified into three different risk levels: low (EWS: 0-3), medium (EWS: 4-6), and high (EWS: 7 or higher). A low-risk level demands a nurse to assess the patient periodically. A medium-risk level requires to inform medical team urgently. In contrast, a high-risk level should trigger an urgent clinical response as the patient's condition is critical [17–19].

There are, nevertheless, various restrictions and issues such as latency and inaccuracy in this manual data acquisition. Furthermore, this system is merely restricted to hospital settings where patients are stationary. In this regard, an IoT-based health monitoring system is proposed to monitor the vital signs autonomously and deliver the EWS score to healthcare providers [20]. Estimations suggest that the ratio between the world's population and IoT devices will be one to four [21]. These small IoT devices and wearables form a good basis for a well-structured EWS system which autonomously monitors a patient in a cost-efficient way while decreasing the mortality rate [4–7].

[1] As Table 1 shows, not every score corresponds to a value of all vital signs.

Table 1 A conventional Early Warning Score (EWS) chart [14]

Vital sign score	3	2	1	0	1	2	3
Heart rate (beats per minute)	0 – 39	40 – 50	51 – 59	60 – 100	101 –110	111 – 129	≥ 130
Systolic blood pressure (mmHg)	0 – 69	70 – 80	81 – 100	101 –149	150 – 169	170 – 179	≥ 180
Respiratory rate (breaths per minute)		0 – 8		9 – 14	15 –20	21 – 29	≥ 30
Body temperature (°C)		≤ 35		35.1 – 38		38.1 – 39.5	≥ 39.6
Blood oxygen saturation (%)	0 – 84	85 - 89	90 – 94	95 – 100			
AVPU score[a]				A	V	P	U

[a]AVPU (the level of consciousness): A = alert, V = reacting to voice, P = reacting to pain and U = unresponsive

Despite IoT provides a potential solution for monitoring human's vital signs, the conventional EWS system is still not applicable for out-of-hospital monitoring since daily activities, and the environments influence the vital signs and subsequently the decision making. Usually, a person has a higher heart rate, blood pressure, respiratory rate, and body temperature when making physical effort (e.g., running and riding a bicycle) compared to more relaxed activities such as sitting or sleeping. Using the same score classification ranges, such as those in Table 1), would lead to a high EWS during physically demanding activities although there is no emergency. Towards this end, a modified EWS system has been proposed for everyday settings, providing a self-aware decision (i.e., the score) according to the context information and five[2] vital signs [22].

Autonomous mobile EWS system still faces problems that have to be solved for being able to offer a reliable EWS calculation. Incorrectly attached or detached sensors, broken sensors, or a noisy signal affect the EWS calculation. If the calculated value is still close to the truth, it may not be a problem. In contrast, an EWS that deviates more from the truth could lead to a false or - even worse - a missing alarm with all its consequences. Self-awareness is a promising solution to tackle this problem. Self-awareness is the ability of the system to monitor itself and its environment regarding the state, behavior, performance, and goals. This is often accompanied by an adjustment of some of the components and parameters which lead to achieving or approaching to the goals of the system [23]. This process has been modeled different ways by various groups, among which some of the more well-known ones are Observe-Decide-Act (ODA) [24] and Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K) [25]. Several works have been done in order to implement self-awareness in various systems, and take advantage of its properties [12, 23, 24, 26–28]. However, most of these works are more focused on the smart decision-making process, while paying little attention to the observation (monitoring) part of the process. In 2016, TaheriNejad et al. published a paper [29] which highlighted this aspect and elaborated on different elements of observation and their potential effect on self-awareness and the overall performance of the system. Since then, several publications have appeared in the literature which demonstrated this effect in various applications [13, 26–28, 30–33].

Our previous works utilize various self-awareness properties to overcome different issues. Anzanpour et al. exploited the self-awareness in IoT-based EWS systems. In this work, *situation awareness* was utilized to improve the specificity of the EWS values, considering the impact of the user's physical activities in the calculation. *Attention* as another self-awareness property was also used to enable a self-organized system, dynamically adjusting the system's configuration for power consumption reduction [26]. Such a dynamic behavior can increase system battery life, but it could decrease the reliability of the EWS in the case of low-quality signals. In another work [13], the proposed system assess the reliability of the calculated EWS. The fuzzified reliability validation tackles the fact that the knowledge about the vital signs as well as their interactions is not complete. With this technique, it was possible to recognize erroneous vital signs caused by various measurement artifacts such as detached sensors, loose sensors, and other interferences.

Our results show that self-awareness can tackle various issues that affect the reliability of a mobile EWS system. Although the proposed system of [13] provides information about the trustworthiness of the calculated EWS, the EWS itself is still incorrectly calculated if the input data is corrupted. Enhancing the decision-making mechanism of the EWS system is a way to solve this problem and improve reliability.

3 Self-awareness properties

In this work, we study two aspects of self-awareness, namely confidence and data reliability, and the interplay

[2]The level of consciousness is excluded because it is not applicable in out-of-hospital monitoring.

between the two as well as their effect on the overall performance of the system. Moreover, we have tried to formalize these concepts, which were initially described in [29] only conceptually, in order to establish a more uniform understanding of these concepts.

3.1 Data reliability

Data Reliability describes the trustworthiness of a set of data at hand, which can be divided into accuracy, precision, and truthfulness. A sensor may be accurate and precise. However, if it is used outside its assumed working conditions, it does not provide reliable data; i.e., it does not provide truthful data. Moreover, even though accuracy and precision provide general measures on the overall quality of a data set (or performance of a sensor), they do not provide an explicit meta-data on each data point. A (resource constrained) self-aware system such as ours, however, sometimes needs to make decisions based on single or few data points. Therefore, accuracy and precision do not provide enough situational information for such cases, and the system needs to estimate and be aware of the overall reliability of those data points based on which it makes a decision.

3.1.1 Formal definition

As mentioned before, data reliability can be broken to accuracy, precision, and truthfulness. Accuracy, $A(X')$, is the systematic bias of the data set at hand, i.e., $X' = \langle x'_0, ..., x'_n \rangle$, compared to the ground truth values, $X = \langle x_0, ..., x_n \rangle$. As a measure of statistical bias it can be defined as

$$A(X') = \frac{1}{n} \sum_{i=0}^{n} x_i - x'_i. \tag{1}$$

Precision presents the random errors in the data (for a measurement, it would be the random errors of repeated measurements under the same conditions). Since precision is a measure of statistical variability, it can be defined as:

$$P(X') = \sigma' = \sqrt{\frac{1}{n} \sum_{i=0}^{n} (x'_i - \mu')^2} \tag{2}$$

where $\mu' = \frac{1}{n} \sum_{i=0}^{n} x'_i$.

Truthfulness, t, is the distance of each value at hand, x'_i, with the corresponding ground truth value x_i:

$$t(x'_i) = |x'_i - x_i| \tag{3}$$

The overall truthfulness, $T(X')$, of a set of values can be defined as

$$T(X') = \frac{1}{n} \sum_{i=0}^{n} t(x'_i). \tag{4}$$

Accuracy and precision are defined on one or more data sets, X' and X, and hence are a property of a set,[3] whereas truthfulness is defined on each data sample, x'_i. Therefore, even though A, P, and t (and consequently T) are correlated, a closed-form formula describing their dependency often cannot be established. Moreover, in many cases the ground truth value, x_i, is not available which makes the calculation of t impossible. In consequence, often an estimation of t, namely t', is devised which may or may not include the effect of accuracy and precision.

In summary, given a sequence of sampled data points X', the data reliability R of X' is given as (the same can be defined for each value)

$$R_f(X') = f(A(X'), P(X'), T(X')) \tag{5}$$

where f determines the role of each parameter and thus how well would R fit its purpose. For example, the reliability of $x'_i \in X'$ could be calculated as

$$r_f(x'_i) = f_{x'_i}(A, P, t) = c_1 A(X') + c_2 P(X') + c_3 t(x'_i) \tag{6}$$

with constants c_1, c_2 and c_3 defining the relative weights given to the three components of the data reliability. Ideally, the reliability is defined such that the mapping domain is between one and zero:

$$R, r : X' \rightarrow [0, 1] \in \Re \tag{7}$$

In a cyber-physical system, A and P are usually provided by the producers of the sensors (even though that is not always the case), and the t and f are to be calculated or estimated by the system using the sensor. In the absence of these values, the designer needs to estimate r or R by r' and R', respectively, using custom methods. In this work, we present our proposed method to calculate r' and R', which we use as our measure of data reliability.

In the following, we present three measures which can provide an insight into the reliability of the data at hand. That is consistency, plausibility, and correlation of data. An important feature of these measures is that they could be applied to low-level data (obtained directly from sensors) or higher-level data (obtained from processes and algorithms within a system).

3.1.2 Plausibility

Data sets can often be associated with a membership function, specifically in the case of cyber-physical systems, that translates into how plausible is the existence of a data with a certain value in the data set. For example, the oxygen saturation can be only in the range of 0-100%; any other value reported is a sign of malfunction and unreliability of the data. The same could be said for a heart-rate of 300

[3] In a cyber-physical system that would be a property of a measurement device.

beats per minute for an adult person. By tagging such data as less reliable or unreliable, a self-aware system could react accordingly (e.g., look for further sources of information or dismiss the data).

3.1.3 Consistency

A certain consistency is often observed within the members of a data set. This is particularly valid in the case of data sets representing natural phenomena, i.e., data collected by a sensor from the real world. Such signals often experience limited changes from one sample to the next. Therefore, the history of a signal and its consistency can provide some information on how reliable is that source of data. For example, it is established that the body temperature cannot change several degrees per minute [34]. Hence, if a larger rate of change occurs in a data set, a self-aware system should tag such an observation (which may be caused by a sensor detachment or a fault/failure in the sensor) as unreliable (regardless of its cause) and react accordingly.

3.1.4 Cross-validity

In some cases, there exists a correlation between the values of two data sets (or such correlation can be established). In such cases, this correlation can be exploited to evaluate the probability or possibility of the coexistence of two or more values. If their coexistence is not possible (e.g., a living patient with valid heart rate and respiratory rate but a negative body temperature) then one or some of those data could be tagged as an unreliable (in this example body temperature). If their coexistence is possible but not very probable (e.g., a body temperature around 30^oC with typical values for other biological signals), the reliability of the data could be reduced, signaling the system a need for further analysis. In the use-case of this work, there have been several works trying to establish such correlations between vital signals of the body [35–37]. Although they do not always provide a conclusive insight, they help us to enhance the robustness of our system by enabling additional data reliability assessments.

3.2 Confidence

Confidence is a measure of the reliability of an algorithm or a process in the system[4] [29]. Conceptually, we can say that confidence provides the system with a measure on how the results of an algorithm or a process can be relied upon. In other words, how close the output of this algorithm or process would be to the ideal output. All that with the assumption that the system has received flawless input data. Although, more often than not, the input data collected by the sensors are unideal (which we discussed in the data reliability subsection). Therefore, the reliability of the output of a system depends on both its confidence and the data reliability of its inputs.

The importance of confidence is in its ability to improve the decision-making processes [12] and allow a self-aware system to question certain abstracted data it has processed, and make more reliable decisions based on the reliability of its sub-processes and sub-algorithms. An important application of this concept for the decision-making unit is to enable it to switch between different algorithms based on their confidence, the usefulness of which has been shown in [28].

3.2.1 Formal definition

If I is an ideal function defined over $X = \langle x_0, \ldots, x_n \rangle$ and g is the unideal function at hand, also defined over X, then the confidence of $g(x_i)$ (defined for each member of X) can be defined as a function Δ of $g(x_i)$ and $I(x_i)$. Δ represents the "distance" between f and g based on some application specific metric for distance, normalized such that $0 \leq \Delta \leq 1$. Thus, for the confidence, c, we have:

$$c\left(g(x_i)\right) = 1 - \Delta\left(I(x_i), g(x_i)\right). \tag{8}$$

Overall confidence of g (as opposed to confidence at each point), represented by C, is the average confidence of g over X:

$$C(g) = \frac{1}{n}\sum_{i=0}^{n} c(g(x_i)). \tag{9}$$

We note that $0 \leq c(g), C(g) \leq 1$ and $c(I) = C(I) = 1$. How to calculate c (and consequently C), however, is case specific. Often the ground truth (I) is not available and the aforementioned distance cannot be calculated. Therefore, a Δ' function is used instead to estimate Δ, which is what we do in the rest of this work too. That is, we propose an estimation of Δ (i.e., Δ'). In other words, all the confidence (c) functions hereafter refer to Δ', which is an estimation of Δ.

3.3 Combination of data reliability and confidence

In this section, we already discussed the concepts of data reliability (as a property of a data set) and confidence (as a property of a process or algorithm) independently. However, in a real-world system, these two often are tightly intertwined. Processes consume data and produce data. Assuming an ideal input, the data reliability of the output data of a process could be associated with its confidence

[4]Therefore, confidence is a property of an algorithm, process or system, as opposed to Data Reliability which is a property of the data at hand.

(although not always in a straight forward or in a simple manner). However, most often, the input data are unideal and subject to a data reliability below one. Therefore, the data reliability of the output data of a process is a function (ϕ) of the input data reliability and the confidence of the process. Calculating the output data reliability of a process (which in turn could be the input data reliability of another process) is particularly more difficult when data reliability or confidence are obtained using estimation functions. In this work, we explore this realm and try to propose a method which shows a good promise in the estimation of the output data reliability of different processes in our system based on respective input data reliability and confidence of that process. More details on our practical implementation are found in Section 4.3.

3.3.1 Formal definition

If $X' = \langle x'_0, ..., x'_n \rangle$, is the data set at hand (i.e., the unideal values), corresponding to the ground truth values $X = \langle x_0, ..., x_n \rangle$, we have:

$$R_g(x'_i) = \phi\left(r_f(x'_i), c\left(g(x)\right)\right). \tag{10}$$

Since, as mentioned before $\forall x; c\left(g(x)\right) \leq c\left(I(x)\right)$ and $R_f(x'_i) \leq R_f(x_i)$ we can conclude that

$$R_g(x'_i) \leq R_I(x'_i) \leq R_I(x_i). \tag{11}$$

3.4 History

History enables access to time-dependent information in a system. For example, whether the performance of a (sub)system has been improving or degrading. The historical data can provide meta-data on the current status of the system and its environment. They also help in predicting the (near) future status of the system and its environment. Given that most systems have memory limitation, choosing the type and mode of storing historical value, and a smart usage of it are important points to be considered when designing a self-aware system using history for enhancing its performance.

3.4.1 Formal definition

There are several methods to track the past values in a sequence. Given the sequence of values or symbols $X = \langle x_0, \cdots, x_n \rangle$, $H = \langle h_0, \cdots, h_m \rangle$ is a subsequence of X, in which $m \leq n$. If $m = n$, the system is memorizing everything which is undesirable. Therefore, most often $m < n$ and preferably $m \ll n$. We note that history function is a specific form of abstraction which concerns time, i.e., the sequence length of X. As of such we can define it as

$$H = \mathfrak{H}_y(X) = \langle h_0, \cdots, h_i, \cdots, h_m \rangle \tag{12}$$

where at the sequence point of x_s,

$$h_i = y(h|_{j=0}^{r}, x|_{k=0}^{s}), \quad r \leq (i-1) \ \ \& \ \ i \leq s, \tag{13}$$

where the function y determines how exactly the history H is extracted from X. An interpretation or abstraction of X (such as the average of certain number of data points), or a direct storage of the values themselves could be some examples of y.

4 System architecture and implementation

A hierarchical agent-based architecture (as shown in Fig. 1) consists of independent modules which can communicate with each other and may be in different hierarchical levels. The possibility of hierarchically structuring the agents enables to process data on different levels of abstraction [38].

The EWS is the aggregate of various scores abstracted from different vital signs. The task of abstraction is the same for each vital sign, but the ranges vary from vital sign to vital sign. The assessment of reliability and confidence-based decisions are done on different levels of abstraction. As an example, a part of the reliability assessment is based on the absolute value and the slope of the signal of a vital sign (principle of plausibility and consistency in Sections 3.1.2 and 3.1.3). To analyze whether a signal is plausible and consistent, the raw data is of interest. In contrast, for making a statement about the correlation between vital signs (principle of cross-validity in Section 3.1.4) already abstracted information is needed. Because of these differences horizontal direction (different vital signs) and vertical directions (different levels of abstraction), a hierarchical agent-based model (Fig. 1) constitutes an appropriate practical architecture for this purpose.

Because an ODA loop is an appropriate approach to implement self-awareness, our system is also based on this concept [23, 29, 39]. Each agent acts like an ODA loop, which means that it monitors its inputs (sensor or agent), decides what to do, and acts accordingly. Furthermore, this approach allows implementing a highly modular model easily.

While the abstraction from the raw sensor value to the vital sign score (with the help of Table 1) takes place in the lower hierarchical level, the agent on top aggregates the five scores to the overall score, the EWS. In other words, each low-level agent abstracts the actual samples obtained from its dedicated sensor and sends the result to the high-level agent, which sums up all these scores. Both, the reliability assessment, as well as the confidence-based decision-making, takes place in the lower and in the higher hierarchical level. However, the implementations of these processes are different in the two hierarchical levels. In

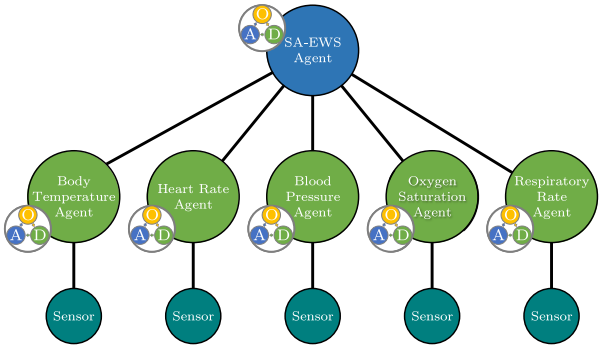


Fig. 1 Hierarchical agent-based system architecture

the next two sections, we explain the reliability assessment and the confidence-based decision-making process, before Section 4.3 shows the workflow of the proposed system in detail.

4.1 Fuzzified reliability assessment

Due to the lack of complete knowledge of all functions of a patient's body, it is very challenging to determine whether a vital sign is monitored correctly or incorrectly. Therefore, in contrast to one of our previous works [32], we use fuzzy logic instead of simple boolean logic to assess the reliability value. The usage of fuzzy logic enables the coverage of the unsharp ranges in which a patient's vital sign is not tagged merely as correct or incorrect, but rather somewhere on the spectrum of reliability. Hence, the data reliability of a vital sign is assigned a value in the range between 0 and 1.

The reliability of a patient's vital sign, vs_i, is composed out of two different reliability assessments: the reliability of the signal's absolute value $r'_{abs,i}$ and the reliability of the signal's slope $r'_{slo,i}$. This corresponds to the plausibility and consistency of data, as described in Section 3.

The reliability for being plausible, $r'_{abs,i}$, is the output of a fuzzy membership function (Fig. 2) defined by four points and three intervals. If the absolute value is in the interval of $[p_b, p_c]$, it is certainly reliable. If it falls in one of the intervals of $[p_a, p_b]$ or $[p_c, p_d]$ - depending on the absolute value, it is more or less reliable. Otherwise, it is certainly unreliable. The reliability $r'_{abs,i}$ and its counterpart (the estimated unreliability $u'_{abs,i}$) of the actual absolute value $v_{a,i}$ are calculated by

$$r'_{abs,i} = \begin{cases} \frac{v_{a,i}-p_a}{p_b-p_a} & \text{if } p_a < v_{a,i} < p_b \\ 1 & \text{if } p_b \leq v_{a,i} \leq p_c \\ \frac{v_{a,i}-p_d}{p_d-p_c} & \text{if } p_c < v_{a,i} < p_d \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

and

$$u'_{abs,i} = 1 - r'_{abs,i} \tag{15}$$

where the points p_a, p_b, p_c, and p_d respectively the intervals between them are configured in a way to match the characteristic of the assigned vital sign.

Similar to that, the reliability for being consistent, $r'_{slo,i}$, and its counterpart (the unreliability, $u'_{slo,i}$), a fuzzy membership function of the same shape exists (Fig. 2). Again, these functions are defined by for points and three intervals between them. These are as follows

$$r'_{slo,i} = \begin{cases} \frac{g_i-p_a}{p_b-p_a} & \text{if } p_a < g_i < p_b \\ 1 & \text{if } p_b \leq v_{a,i} \leq p_c \\ \frac{g_i-p_d}{p_d-p_c} & \text{if } p_c < g_i < p_d \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

and

$$u'_{slo,i} = 1 - r'_{slo,i} \tag{17}$$

where g is the gradient between the actual value, $v_{a,i}$, to the previous one, $v_{p,i}$.

This gradient is calculated by

$$g_i = \frac{v_{p,i} - v_{a,i}}{t} \tag{18}$$

where t constitutes the time between the samples.

Depending on which of the two reliabilities shall be assessed, the abscissa constitutes the absolute value or the

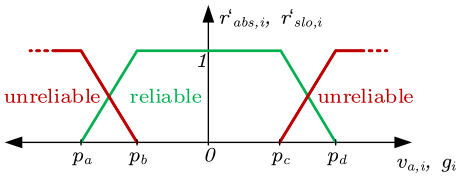


Fig. 2 Example for a fuzzy membership function to assess the reliability of the absolute value or the slope of a vital sign

slope of a vital sign. The ordinate of the fuzzy membership function constitutes then the reliability corresponding to it. While the abscissa gives space for all values (from $-\infty$ to $+\infty$), the reliability values on the ordinate are limited between 0 and 1.

After the assessment of $r'_{abs,i}$ and $r'_{slo,i}$, the input reliability, $r'_{in,i}$, can be calculated in many different ways such as conjunction ($\wedge$), disjunction ($\vee$), or multiplication of different inputs as well as if-then-rules and other methods. We decided to use the conjunction operator because a vital sign is reliable when its absolute value and its slope are reliable. Therefore the input reliability $r'_{in,i}$ of a vital sign is given by

$$r'_{in,i} = r'_{abs,i} \wedge r'_{slo,i} \tag{19}$$

where the fuzzy conjunction is equal to a minimum function [40].

Because the input reliability, $r'_{in,i}$, depends only on the raw sensor data (absolute value and gradient of the signal), it is calculated in the low-level agents which are also responsible for the abstraction of the vital signs. This input reliability is calculated for every vital sign and provides information on whether it is reliable or unreliable considered separately. In other words, the reliability of one vital sign omits the condition of other vital signs.

Since vital signs impact each other, and therefore, one vital sign, vs_i, usually does not have a terrible score while others have a perfect score, a cross-validation reliability value is needed. For this purpose, the cross-validation reliability, $r'_{cro,i,j}$, for the vital signs vs_i and vs_j is calculated by

$$r'_{cro,i,j} = \begin{cases} 1 & \text{if } s_i = s_j \\ \frac{1}{p_{cro,i,j}|s_i - s_j|} & \text{if } s_i \neq s_j \end{cases} \tag{20}$$

where $p_{cro,i,j} \in (0, \infty)$ denotes a coefficient of the strength of the correlation[5] between vital signs vs_i and vs_j, and s_i, as well as s_j, are the abstracted scores of these two vital signs.

Because the cross-validity reliability, $r'_{cro,i,j}$, already makes use of the abstracted information (the various vital sign scores), it is calculated in the high-level agent which is responsible for the calculation of the EWS.

4.2 Fuzzified confidence-based decisions

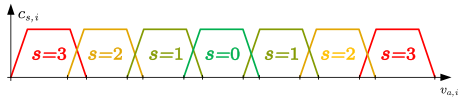
As already stated in Section 3.1, data reliability describes the trustworthiness of a set of data at hand, which can be divided into accuracy, precision, and truthfulness. For the case, a sample (a sensor value) is not very accurate, two different possibilities exist. If the real vital sign value (the ground truth) is somewhere in the middle of a score range of Table 1 and the sensor's inaccuracy is not very high, the abstracted score will most likely be equal to the ground truth. In contrast, a wrong score abstraction could result out of a ground truth value very close to a boundary of such a range or a highly inaccurate sensor.

To overcome this issue, the abstraction process in the lower hierarchical level is not merely based on a simple lookup table as in Table 1, the boundaries of the different score ranges are intersecting which means that the score ranges are partly overlapping. Figure 3a shows an example for the vital sign abstraction, with four different fuzzy membership functions; for each score, one fuzzy membership function.

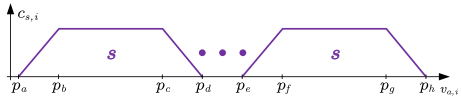
In similar fashion to the reliability fuzzy functions, various intervals describe the confidence fuzzy membership functions. Because the fuzzy membership functions are an extension of Table 1, the intervals can vary between different vital signs. While heart rate and systolic blood pressure are symmetrical in a way that each score higher than 0 is available for the vital sign's value is either too low or too high. In contrast, respiratory rate, body temperature, and blood oxygen saturation are unsymmetrical; some scores are missing on one side or both sides. In the case of a symmetrical segmented vital sign (Fig. 3b), the confidence functions of abstracting the actual value of a vital sign to a score s_i, $c_{s,i}$ are calculated by

$$c_{s,i} = \begin{cases} \frac{v_{a,i} - p_a}{p_b - p_a} & \text{if } p_a < v_{a,i} < p_b \\ 1 & \text{if } p_b \leq v_{a,i} \leq p_c \\ \frac{v_{a,i} - p_d}{p_d - p_c} & \text{if } p_c < v_{a,i} < p_d \\ \frac{v_{a,i} - p_e}{p_f - p_e} & \text{if } p_e < v_{a,i} < p_f \\ 1 & \text{if } p_f \leq v_{a,i} \leq p_g \\ \frac{v_{a,i} - p_h}{p_h - p_g} & \text{if } p_g < v_{a,i} < p_h \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

where $s \in \{1, 2, 3\}$ is one of three possible scores the actual value, $v_{a,i}$, of the vital sign can have.



(a) All three possible scores to which the vital sign, vs_i, can be abstracted.



(b) One of the scores to which the vital sign, vs_i, can be abstracted.

Fig. 3 Example for fuzzy membership functions to assess the confidence of the abstraction of a vital sign

[5] The reliability module in our implementation limits the cross-validity reliability, $r'_{cro,i,j}$, to a value between 0 to 1, although theoretically, a coefficient less than 1 can lead to an $r'_{cro,i,j}$ higher than 1. The standard value of $p_{cro,i,j}$ is 1.

Because score 0 of each vital sign has only one range in Table 1, the confidence function of abstracting a vital sign's actual value to score 0, $c_{0,i}$ is calculated by

$$c_{0,i} = \begin{cases} \frac{v_{a,i}-p_{0,a}}{p_{0,b}-p_{0,a}} & \text{if } p_{0,a} < v_{a,i} < p_{0,b} \\ 1 & \text{if } p_{0,b} \leq v_{a,i} \leq p_{0,c} \\ \frac{v_{a,i}-p_{0,d}}{p_{0,d}-p_{0,c}} & \text{if } p_{0,c} < v_{a,i} < p_{0,d} \\ 0 & \text{otherwise.} \end{cases} \quad (22)$$

In the configuration of the proposed system, the interval of the ramp of a fuzzy membership function is congruent with the interval of the ramp of the next fuzzy membership function (e.g., p_c and p_d of $c_{0,i}$ are equal to p_a respectively p_b of $c_{1,i}$). This approach leads to the possibility of abstracting a vital sign value to two different scores with certain confidences. As an example, let us assume that the actual value of a vital sign, $v_{a,i}$, is the interval between $p_{0,c}$ and $p_{0,d}$ (which is equal to the interval $p_{1,e}$ and $p_{1,f}$). In this case, the vital sign will be abstracted to score 0 with $c_{0,i} = \frac{v_{a,i}-p_{0,d}}{p_{0,d}-p_{0,c}}$ and to score 1 with $c_{1,i} = \frac{v_{a,i}-p_{1,e}}{p_{1,f}-p_{1,e}}$.

However, the high-level agent evaluates various confidences. Similar to Eq. 20, cross-validity confidence, $c_{cro,i,j}$, is calculated based on a patient's individual correlations of the various vital signs; e.g., Eq. 20 does not reflect the truth if a patient - in normal health condition - has tachypnea, hypertension, or another vital sign which leads to a score higher than the scores of the other vital signs. Based on the frequency of various occurring score differences, $SD_{i,j}$, between the two vital signs vs_i and vs_j, a patient profile is established which gives information about the likelihood of a score difference between two different vital signs. For this purpose, the patient (situated in normal condition) is monitored for the period T (the time of n samples). After n samples have been recorded, four different quantities, $q_{SD_{i,j}}$ for all four possible score differences $SD_{i,j} \in \{0, 1, 2, 3\}$, are known. With the knowledge of these quantities, the cross-validity confidence between the two vitals signs sv_i and sv_j, $c_{cro,i,j}$ is calculated by

$$c_{cro,i,j} = \frac{q_{SD_{i,j}}}{n}. \quad (23)$$

4.3 Functional description of the system

Figure 1 shows the system architecture we propose in this work. At the bottom are five sensors which monitor the five different vital signs (Table 1) and transmit the raw data to their dedicated agents in the lower hierarchical level. Figure 4 shows a simple schematic of the whole procedure for one low-level agent; the others are just faded out. The functional principle of both, the agents of the lower and the higher hierarchical, is explained in detail in the following.

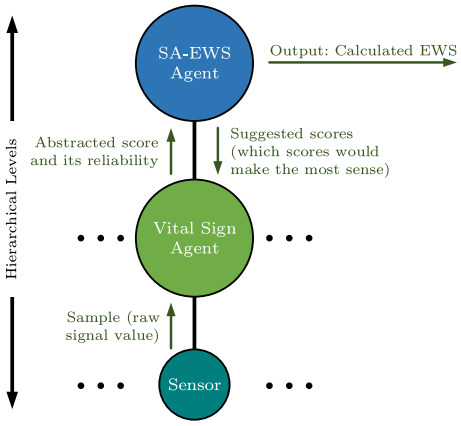


Fig. 4 Functional description explained for one vital sign

4.3.1 Lower hierarchical level of computation

Each of these five low-level agents abstracts the actual value (got from its dedicated sensor) by calculating the confidences for every possible score, $c_{s,i}$ for $s = 0, 1, 2, 3$, by Eqs. 22 and 21. As shown in Fig. 4, each low-level agent also receives score suggestions from the high-level agent. Each of these suggestions consists of a score and its reliability, $r'_{sug,s,i}$. Their calculation is based on Eqs. 20 and 23 but Section 4.3.2 will show the exact procedure of generating these suggestions. Additionally, the input reliability, $r'_{in,i}$ of the corresponding vital sign is calculated by Eq. 19.

In a next step, the output reliability of each possible score, $r_{out,s,i}$ is calculated by

$$r'_{out,s,i} = r_{in,i} \wedge c_{s,i} \wedge r'_{sug,s,i} \quad (24)$$

because the score is reliable if the vital sign value is reliable, the abstraction is done with high confidence, and if it correlates with the other vital signs (based on the suggested scores).

After every possible score has been calculated, the low-level agent chooses the one with the highest output reliability and saves it in a history if the reliability is higher than a certain threshold.[6] In the next step, the low-level agent sends the last saved score and its output reliability to the high-level agent. In other words, if the reliability of the actual score is higher than the set threshold it is sent to the high-level agent; otherwise, the previous score is sent.

[6] If the history is empty (e.g., right after the EWS system has been started), the score and its reliability are saved in the history regardless.

4.3.2 Higher hierarchical level of computation

The high-level agent calculates the EWS and its overall reliability, r'. For this purpose, the agent reads all low-level scores and their output reliabilities, $r'_{out,i,s}$. However, these reliabilities are - from the perspective of the high-level agent - input reliabilities, and therefore, they are called $r'_{in,i}$. The EWS is just the sum of all five vital sign scores, and thus, calculated by

$$EWS = \sum_{i=1}^{5} s_i. \tag{25}$$

With all five input reliabilities, $r'_{in,i}$, the combined input reliability, r_{in} is calculated by

$$r'_{in} = r'_{in,1} \wedge \cdots \wedge r'_{in,5} = \bigwedge_{i=1}^{5} r'_{in,i}. \tag{26}$$

For two vital sign scores, the cross-validity reliability is calculated by Eq. 20, and the personalized cross-validity confidence by Eq. 23. After the calculation of both of these metrics, the personalized cross-validity reliability, $r'_{per,cro,i,j}$, can be calculated in different ways. We decided to use the disjunction ($\vee$) operator because the correlation is plausible if it is according to our general rule (20) or matches the personalized body functions of the patient (23). Therefore the personalized cross-validity reliability, $r'_{per,cro,i,j}$, is given by

$$r'_{per,cro,i,j} = r'_{cro,i,j} \vee c_{cro,i,j} \tag{27}$$

where the fuzzy disjunction is equal to a maximum function [40].

The overall reliability of the calculated EWS is composed of all input reliabilities and all personalized cross-validity reliabilities, $r'_{per,cro,i,j}$. All $r'_{per,cro,i,j}$ for this purpose are combined together to the combined cross-validity reliability, $r'_{per,cro}$, by

$$r'_{per,cro} = \bigwedge_{i=1}^{5} (\bigwedge_{j=1}^{5} r'_{per,cro,i,j}) \tag{28}$$

where the cross-validity reliabilities for $i = j$ are not calculated because they will be 1 one for sure (20).

In further consequence, the overall reliability, r', is given by

$$r' = r'_{in} \wedge r'_{per,cro} \tag{29}$$

and constitutes, besides the EWS (25), the output of our proposed system.

As mentioned in Section 4.3.1, the high-level agent makes also score suggestions which are sent to each low-level agent. For this purpose, theoretically personalized cross-validity reliabilities are calculated for each possible score ($s \in 0, 1, 2, 3$) with that a vital sign could be classified. In particular, the four theoretically possible scores of one agent are calculated by Eq. 27, whereas the score difference is based on the comparisons with the real scores from the other four low-level agents. The reliability of the theoretically possible score (the suggested score) is calculated by

$$r'_{sug,s,i} = \bigwedge_{j=1}^{5} (r'_{cro,i,j} \vee c_{cro,i,j}) \tag{30}$$

for each possible score $s \in 0, 1, 2, 3$. Whereas the comparison of one vital sign with itself is not performed.

This procedure is repeated for all of the five vital signs, and the results (the four possible scores and their theoretical cross-validity reliability) is sent to the dedicated low-level agent.

5 Experimental results

In this section, we describe our experimental setup as well as the validation method of our proposed system. We also discuss the experimental results in detail.

5.1 Experimental data

The data collection was performed on eight different participants aged from 23 to 37 (see Table 2). Half of the participants were male, and the other half were female.

As listed in Table 3 and shown in Fig. 5, we recorded and abstracted the vital signs with different sensors respectively in different ways. A set of sensors provides a high-accuracy source, and another set of sensors provides a low-accuracy source for normal and fault-emulated signals. As the high accuracy sensor set, we use (i) a chest strap heart rate monitor for recording Electrocardiogram (ECG) signal, (ii) a sensitive temperature sensor attached to the subject's nose for recording the airflow signal, (iii) an accurate

Table 2 Participants who participated in our experiments

Person	Sex	Age	Test scenario(s)
P1	Male	37	S1, S2, S3, S4
P2	Female	23	S1 (twice)
P3	Male	29	S1, S2, S3, S4
P4	Male	25	S1 (twice)
P5	Female	23	S1 (twice)
P6	Male	30	S1 (twice)
P7	Female	23	S1 (twice)
P8	Female	28	S1 (twice)

Table 3 Details of the sensors used for data collection

Vital sign	Reference vital sign / source		Test vital sign 1 / source		Test vital sign 2 / source	
Heart rate	HR_r	Chest strap (Polar T31C)	HR_{t1}	PPG sensor (MAX30100) at 24mA	HR_{t2}	PPG sensor (MAX30102) at 3.5mA
Respiration rate	RR_r	Temperature sensor used as airflow sensor (MCP9808)	RR_{t1} at 24mA	PPG sensor (MAX30100)	RR_{t2}	PPG sensor (MAX30102) at 3.5mA
Blood Oxygen saturation	$SpO_{2,r}$	PPG sensor(MAX30100) at 24mA	$SpO_{2,t1}$	PPG sensor (MAX30102) at 3.5mA		
Skin temperature	ST_r	Temperature sensor (MCP9808)	ST_{t1}	Temperature sensor (TMP102)		
Blood pressure	BP_r	Arm-type blood pressure monitor (iHealth BP7)	BP_{t1}	Wrist-type blood pressure monitor (Beurer BC32)		

temperature sensor attached to the armpit (axilla),[7] (iv) an upper arm blood pressure monitor, and (v) a high-fidelity Photoplethysmogram (PPG) sensor for recording infrared and red PPG signals.[8]

The low-accuracy sensor set consists of (i) another PPG sensor which consumes less power and records PPG signal with lower Signal-to-Noise Ratio (SNR, (ii) a temperature sensor with lower sensitivity is attached to armpit (axilla) measures skin temperature, and (iii) a wrist-type blood pressure monitor measuring an estimation of blood pressure. Table 3 shows the details of the sensors in each set. All continuously recording sensors[9] were connected to an ATMEGA328P microcontroller which reads the sensors values with a sampling frequency of 50 Hz. Finally, an Android phone, connected to this microcontroller via a USB-to-Serial converter, recorded the data.

In the next step, these recorded signals were analyzed to extract the vital signs. As listed in Table 3, we use two sets of PPG signals to obtain two sources of heart rate, respiration rate, and SpO_2 values (i.e., low-accuracy and high-accuracy values). First, a filter-based method is used to extract respiratory and heartbeat signals. In this method, the cut-off frequencies are selected based on Power Spectral Density (PSD) of the PPG signals [42–44]. Note that an acceptable SNR is needed in this method, as high noise level influences the PSD of the signal and subsequently interrupts cut-off frequency selection. Next, the respiration rate and heart rate values are determined via a peak detection method. Moreover, the SpO_2 value is calculated from the PPG signals using two light sources with different wavelengths (i.e., red which has 660 nm and infrared which has 880 nm) [45, 46]. In addition to the PPG signals, another high-accuracy heart rate and respiration rate values are determined by using the two other sources (i.e., ECG and airflow signals). Similarly, we use peak detection methods for the detection of these two vital signs. In total, we extracted three heart rate, three respiration rate, two SpO_2, two skin temperature, and two blood pressure signals.

5.2 Validation of the EWS systems

Table 4 shows the different scenarios in which the participants were monitored. In Scenario S1, the participants were sitting without performing any physical activity. P1 and P3 were also monitored during three additional scenarios in that errors were induced in some of the low accuracy sensor setup (Scenarios S2, S3, and S4). Six participants were monitored two times, and the other two participants four times (Table 2), resulting in 20 measurements in total.

As mentioned in Section 5.1, we recorded and abstracted the vital signs of each scenario (listed in Table 4) with

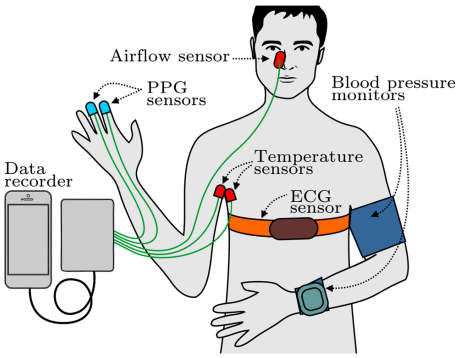


Fig. 5 Data collection sensors

[7] Because Table 1 shows the body core temperature, the measured skin temperature had to be converted to an estimated core temperature. This was done as Richmond et al. state it in [41].

[8] As shown in Table 3, the MAX30100 PPG sensor was used as accurate source for monitoring SPO_2 and as one of the inaccurate sources for monitoring heart rate and respiratory rate.

[9] The two blood pressure devices were manually operated and were not continuous.

Table 4 Scenarios of measurements

Scenario	Scenario description
S1	The person was sitting and no additional error was induced during the measurement.
S2	The person was sitting and the temperature sensor was temporarily detached.
S3	The person was sitting and contracted his/her biceps for a period of the measurement.
S4	The person was sitting and the temperature sensor was temporarily detached. In addition, the person contracted his/her biceps for a period of the measurement.

different sensors, respectively, in different ways (Table 3). All various combinations of the twelve vital signs (varying in quality) reveal a total number of 72 different signal setups for each measurement. The 20 measurements and the 72 different ways of monitoring/abstracting the vital signs lead to 1440 different experiments.

All these data sets were then processed with both, the conventional EWS system without any self-awareness properties and our proposed SA-EWS system. The output of these systems is the EWS signal of the same length as the experimental data sets (one EWS value for each vital sign sample set). To have a common benchmark for comparing both systems, a ground truth for each of the 20 measurement[10] is needed. Due to the lack of a real ground truth, we took the data set of each experiment, which matches the ground truth the most. These Ground Truth Datasets (GTDSs) consists of the vital signs HR_r, RR_r, $SPO_{2,r}$, ST_r, and BP_r of Table 3. To ensure that the GTDSs are as close as possible to the real ground truth, all of these signals were additionally filtered[11] to remove noise. Due to corrupted measurements of the vital signs of participant P5, no valid ground truth could be established. Therefore, this participant was excluded from our analysis. This exclusion leads to a reduction of the number of measurements from 20 to 18, and in further consequence, reduced the number experiments: 1296 instead of 1440.

The EWS Ground Truth Dataset (EGTDS) was then created with the GTDSs processed by the conventional EWS system. The EWS system is used for this purpose because it does not - in contrast to the SA-EWS system - manipulate the output leveraging the self-aware properties. However, because the conventional EWS system generated the EGTDSs, it is possible that, if the vital signs of the GTDSs still contain some noise or errors, the SA-EWS system assessment is tagged as erroneous whereas, in reality, the error is in the EGTDS.

We use various metrics to compare these two systems. The Root-Mean-Square Deviation (RMSD) calculation, which indicates how close two different signals are to each other is given by:

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n}(EWS_{GT,i} - EWS_i)^2}{n}} \tag{31}$$

where $EWS_{GT,i}$ is the i^{th} EWS value of the EGTDS and EWS_i the i^{th} outputted EWS value of the system that is compared with the EGTDS.

However, the RMSD is not the best way to compare the two systems. A signal that deviates slightly (e.g., a deviation of only one score) for a long period may have a worse RMSD than a signal that shows a much larger deviation but only for a short time. While the former signal will most likely not result in a false or missing alarm, the latter signal will raise problems. Another metric, namely the maximum absolute error (ε_{max}) which gives information about the highest deviation that occurs in signal compared to the ground truth, is more relevant in this context. It is calculated by:

$$\varepsilon_{max} = max(|EWS_{GT,i} - EWS_i| : i = 1, \ldots, n) \tag{32}$$

where $EWS_{GT,i}$ is the i^{th} EWS value of the EGTDS and EWS_i the i^{th} outputted EWS value of the system that is compared with the EGTDS.

The last metric is the number of false and missing alarms. As mentioned in Section 2, the calculated EWS shows low-, medium-, or high-medical risk of a patient. If the classification of the calculated EWS deviates from the classification of the ground truth EWS, a false or missing alarm is indicated. For example, if the ground truth EWS has a value which belongs to the low or medium risk class but the EWS of the system is in one of the higher classes, a false alarm is raised. In contrast, a calculated EWS in a lower class than the ground truth EWS leads to a missing alarm, which means an alarm should be raised, but it was missed. As a third option, both, the ground truth, as well as the calculated EWS, are in the same class. In this case, there is neither a false nor a missing alarm.

5.3 Results

Table 5 shows the vital signs which are corrupted (×) and which are uncorrupted (✓) in various experiments. To evaluate which of these signals are either correct or are containing errors, the output of the conventional EWS system processing an experiment was compared with the EGTDS of the same experiment. If a vital sign score abstracted from the vital sign (e.g., RR_{I1}) deviates, at any

[10] 20 measurements is the sum of all test scenarios in that the participants were monitored (Table 4).

[11] A Savitzky-Golay filter with the window size of 53 samples and a polynomial order of 3 was used.

Table 5 Signals with errors are marked with a ×, while signals that are correct are marked with ✓

Participant	Experiment	HR_r	HR_{t1}	HR_{t2}	RR_r	RR_{t1}	RR_{t2}	$SPO_{2,r}$	$SPO_{2,t1}$	ST_r	ST_{t1}	BP_r	BP_{t1}
P1	E1	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓
P1	E2	✓	×	✓	✓	×	×	✓	✓	✓	×	✓	✓
P1	E3	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	×
P1	E4	✓	×	✓	✓	×	×	✓	✓	✓	×	✓	×
P2	E1	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓
P2	E2	✓	✓	✓	✓	×	×	✓	×	✓	✓	✓	×
P3	E1	✓	×	×	✓	×	×	✓	×	✓	✓	✓	×
P3	E2	✓	✓	×	✓	×	×	✓	✓	✓	×	✓	✓
P3	E3	✓	✓	×	✓	×	×	✓	✓	✓	✓	✓	×
P3	E4	✓	✓	×	✓	×	×	✓	✓	✓	×	✓	×
P4	E1	✓	✓	×	✓	×	×	✓	✓	✓	✓	✓	✓
P4	E2	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓
P5	E1	Due to invalid ground truth, these experiments were excluded.											
P5	E2	Due to invalid ground truth, these experiments were excluded.											
P6	E1	✓	×	×	✓	×	×	✓	✓	✓	✓	✓	✓
P6	E2	✓	✓	×	✓	×	×	✓	✓	✓	✓	✓	✓
P7	E1	✓	×	×	✓	×	×	✓	✓	✓	✓	✓	✓
P7	E2	✓	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓
P8	E1	✓	✓	×	✓	×	×	✓	×	✓	✓	✓	×
P8	E2	✓	✓	×	✓	×	×	✓	✓	✓	✓	✓	×

point during the measurement, from the value, it should have according to the EGTDS, this vital sign (in the considered experiment) is classified as erroneous.

Based on the number of different vital signs, 72 different combinations (setups) of vital sign sets are possible. Such a setup can now contain some correct and some erroneous vital signs. An important factor is how many vital signs are showing an error at the same time for an experiment. The second column of Table 6 shows this number, which ranges from 0 to 4 errors at the same time. For this purpose, all 1296 experiments (18 measurements with 72 different setups) have been processed by the conventional EWS system, and the results were compared to their dedicated EGTDS. Based on the number of simultaneous vital sign errors, the EWS and the SA-EWS system are compared for each participant. In other words, all experiments performed on each person with different vital sign setups were separated in groups regarding the number of vital sign errors that occurred at the same time. Each row in Table 6 shows the performance of the two compared systems in the form of the minimum, average, and maximum RMSD of all calculated EWS values which are in the same group of the number of vital sign errors. Additionally, and more importantly, the maximum absolute error, ε_{max}, is shown for each group.

As it can be seen, in most of the cases, our proposed system performed equally good or considerably better than a conventional EWS system without self-awareness. For a better understanding of the table, here, we discuss the results using participant P1 as an example. In the experiments where no vital sign showed any error, both systems produced an output in that the calculated EWS did not deviate any single time ($\varepsilon_{max} = 0$). In these setups, the calculated EWS signal was exactly identical to the ground truth EWS signal ($RMSD = 0$). In these experiments, both system performances were equal.

In contrast, the conventional EWS system performed much worse than our proposed system when setups were used in which three of the vital signs contained errors at the same time. One of these experiments is shown in Fig. 6. Whereas Fig. 6a shows the ground truth vital signs and the corrupted signals, Fig. 6b presents the ground truth EWS as well as the outputs of both systems. As it can be seen, the difference between the EWS of the conventional system with the ground truth is large (up to 7 scores), whereas the SA-EWS shows only absolute errors of 1 or 2 in the worst case.

The RMSD values of all considered experimental results show that the output of the SA-EWS system was significantly closer to the ground truth. However, the maximum error shows the real importance of an intelligent EWS system. Participant P5 was excluded from these experiments because of corrupted measurements, which led to an invalid ground truth.

In the four cases of P3, P7, and P8 in Table 6, the conventional EWS system performed slightly better. Some of the participants were sometimes slightly uneasy, which

Table 6 The minimum, average, and maximum RMSD as well as the maximum error of both systems compared on the base of the various participants and the number of vital sign errors occurring at the same time

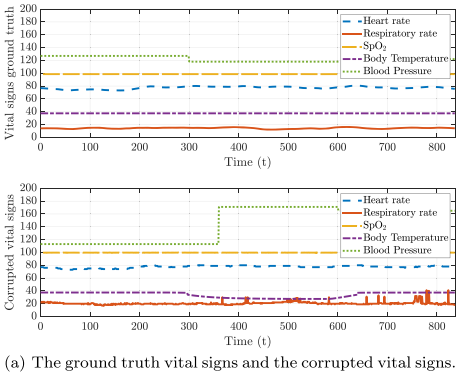
		EWS System				SA-EWS System			
Participants	Number of simultaneous vital sign errors	Min. RMSD	Avg. RMSD	Max. RMSD	ε_{max}	Min. RMSD	Avg. RMSD	Max. RMSD	ε_{max}
P1	0	0.00	0.00	0.00	0	0.00	0.00	0.00	0
	1	0.10	0.94	1.66	3	0.00	0.38	0.81	2
	2	0.68	1.90	2.63	5	0.21	0.71	1.08	2
	3	1.72	2.85	3.25	7	0.37	0.87	1.08	2
P2	0	0.00	0.00	0.00	0	0.00	0.00	0.00	0
	1	0.09	0.28	0.44	2	0.08	0.25	0.40	1
P3	0	0.00	0.00	0.00	0	0.00	0.09	0.20	1
	1	0.07	0.84	2.95	3	0.00	0.59	3.01	4
	2	0.60	1.41	3.28	5	0.23	1.05	3.01	5
	3	0.71	2.27	3.38	6	0.58	1.80	3.01	5
	4	3.35	3.35	3.35	5	2.93	2.93	2.93	5
P4	0	0.00	0.00	0.00	0	0.00	0.00	0.00	0
	1	0.31	0.67	0.74	2	0.08	0.65	0.74	2
	2	0.68	0.70	0.72	2	0.65	0.69	0.72	1
P5		Due to invalid ground truth, these experiments were included.							
P6	0	0.00	0.00	0.00	0	0.00	0.00	0.00	0
	1	0.12	0.70	1.14	3	0.12	0.64	0.97	1
	2	0.92	1.07	1.20	3	0.87	0.96	0.99	2
P7	0	0.00	0.00	0.00	0	0.40	0.53	0.58	1
	1	0.38	0.57	0.70	2	0.44	0.60	0.70	2
	2	0.53	0.63	0.73	2	0.57	0.65	0.73	2
P8	0	0.00	0.00	0.00	0	0.00	0.10	0.30	1
	1	0.17	0.46	1.01	3	0.10	0.42	0.84	2
	2	0.49	0.85	1.26	5	0.52	0.71	0.88	2
	3	1.26	1.38	1.44	6	0.68	0.74	0.77	3

Green color highlights the system with better performance

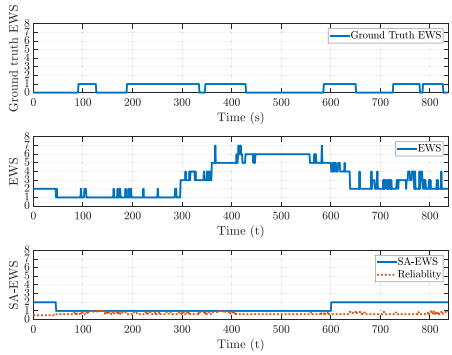
led to temporally irregular breathing. As mentioned, we removed the majority of such noise from the GTDS. However, if there were some noise left, the EWS system may have an advantage over the SA-EWS system because the conventional EWS system generated the EGTDSs.

When comparing the RMSD and the maximum error, the SA-EWS system performed in eleven cases better than the conventional EWS system. In nine cases, the performance was equal, and only in four cases, the conventional EWS system performed slightly better. However, in the latter cases, the performance difference between the two systems was very small and did not lead to any additional false or missed alarms. As a matter of fact, the number of false alarms or missed alarms was always equal or less in the SA-EWS system. Table 7 shows how often both systems missed to raise an alarm or raised a false alarm. As mentioned, the EWS itself can be classified into three different classes, namely low-, medium-, and high risk. If the class of the system's outputted EWS deviates from the class of the ground truth EWS, it causes a false or missing alarm. In all cases, our proposed system performed better (marked in green in Table 7) or equal to the EWS system.

The wrong and missed alarms were counted based on the number of samples which deviate from the ground truth and based on the number of times (events) an alarm was incorrectly raised (false positive) or incorrectly not raised and was missed (false negative). Event-based means when two or more samples of the same event (samples in a row) deviate from the ground truth, the wrong/missed alarm is counted only once. In the example of participant P8, four



(a) The ground truth vital signs and the corrupted vital signs.



(b) The ground truth EWS (based on the ground truth vital signs) as well as the output of the EWS and SA-EWS system (based on the corrupted vital signs).

Fig. 6 A experiment of participant P1 in scenario S4 with a vital sign setup in that three vital sign errors simultaneously occur

and eight false alarms were raised by the SA-EWS system. However, each of these false alarms had the length of only one sample. That is why the number in both rows (samples or events) are the same. We can argue that if a doctor monitors a patient's vital signs and obtains an unrealistic result, he/she tries to redo the measurement. A logical consequence of this could be ignoring alarms of a length of only one or few sample(s), which corresponds to one second in time. However, this is out of the scope of this paper and serves only as an additional note. Therefore, we did not discount any alarms, even if they were very short.

Figure 7 shows the occurrence frequency of absolute error in different sizes for all experiments combined. Both systems have almost the same number of absolute errors in the size of 0 and 1. Overall, except for having an error of 1 score, the proposed system is always better (including when the system has made no false recognition, i.e., 0 on Fig. 7). In particular, the SA-EWS system less often produces larger errors compared to the EWS system. We can see that the SA-EWS system never produce absolute errors larger than 5 (whereas the conventional EWS system experiences them more than a thousand times) and it produces significantly (approximately one order of magnitude) fewer errors in sizes of 4 and 5. This is particularly important since larger errors imply a deviation from the ground truth risk class, which is more important with regard to false or missing alarms. Altogether, Fig. 7 indicates that the proposed SA-EWS system is more reliable (less error-prone) than its conventional counterpart.

6 Conclusion and future work

Self-awareness has proven to be advantageous in many applications, and here we show its benefits for wearable medical devices. In particular, we showed how using basic observation elements such as history, data reliability and confidence can lead to reliable results without incurring massive processing loads that conventional Artificial Intelligence (AI) algorithms impose on systems. From the application point of view, we demonstrated that - even using less reliable, low-quality sensors (which are cheaper) - our system is able to calculate the EWS properly and comparable to a system with highly reliable, high-quality sensors (which are more expensive). We also showed that our proposed system shows good resilience against intentionally introduced measurement errors.

In summary, our contributions are; (a) formalizing data reliability, confidence, and history, (b) proposing function for aforementioned self-awareness properties, in particular for combining data reliability and confidence, (c) performing extended experiments with a large number of sensors and test scenarios, and (d) improving reliability of EWS assessment using cheaper sensors and despite adversities in real life measurements.

We note that many of the proposed functions are designed heuristically. Therefore, other functions could be proposed and studied, which lead to further improved results. We leave that for future works. Moreover, in some cases, we have tried alternative parameter settings and chose the better ones; however, these studies were not systematic or comprehensive. Mainly due to the extensive time that it takes to process all combination of sensors and errors using single setup values. That is, therefore, another future work.

Table 7 The number of missing and false alarms of both systems compared on the base of the various participants and the number of vital sign errors occurring at the same time

		EWS system				SA-EWS system			
Participants	Number of simultaneous vital sign errors	Missing alarm (samples)	Missing alarm (events)	False alarm (samples)	False alarm (events)	Missing alarm (samples)	Missing alarm (events)	False alarm (samples)	False alarm (events)
P1	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0
	2	0	0	10884	1452	0	0	0	0
	3	0	0	4124	242	0	0	0	0
P2	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0
	1	0	0	6510	132	0	0	1676	2
	2	0	0	22034	1386	0	0	10060	16
	3	0	0	23708	924	0	0	16800	32
	4	0	0	1662	16	0	0	1676	2
P4	0	0	0	0	0	0	0	0	0
	1	0	0	144	16	0	0	0	0
	2	0	0	56	8	0	0	0	0
P5		Due to invalid ground truth, these experiments were included.							
P6	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0
P7	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0
P8	0	0	0	0	0	0	0	0	0
	1	0	0	140	98	0	0	0	0
	2	0	0	380	228	0	0	4	4
	3	0	0	802	532	0	0	8	8

Green color highlights the system with better performance

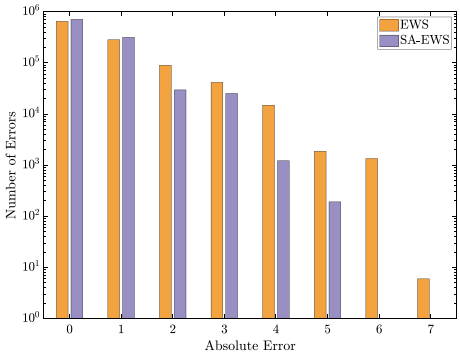


Fig. 7 Occurrence frequency of absolute errors of different sizes

Funding Information Open access funding provided by University of Turku (UTU) including Turku University Central Hospital.

References

1. WHO Chronic diseases and health promotion, Retr. on June 2017. http://www.who.int/chp/en/
2. McGaughey J, Alderdice F, Fowler R, Kapila A, Mayhew A, Moutray M (2007) Outreach and early warning systems (ews) for

the prevention of intensive care admission and death of critically ill adult patients on general hospital wards. The Cochrane Library
3. Morgan et al (1997) An early warning scoring system for detecting developing critical illness. Clin Intensive Care 8(2):100
4. Dohr A, Modre-Opsrian R, Drobics M, Hayn D, Schreier G (2010) The internet of things for ambient assisted living. In: 2010 seventh international conference on information technology: new generations. IEEE, pp 804–809
5. Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw 54(15):2787–2805
6. Anzanpour A, Rahmani A-M, Liljeberg P, Tenhunen H (2015) Context-aware early warning system for in-home healthcare using internet-of-things. In: International internet of things summit. Springer, pp 517–522
7. Miorandi D, Sicari S, De Pellegrini F, Chlamtac I (2012) Internet of things vision, applications and research challenges. Ad hoc Netw 10(7):1497–1516
8. TaheriNejad N (2019) Wearable medical devices: challenges and self-aware solutions. In: IEEE Life sciences newsletter, pp 5–6
9. Pollreisz D, Taherinejad N (2019) Detection and removal of motion artifacts in ppg signals. Mobile Networks and Applications, to appear
10. Parego P, Rahmani AM, Taherinejad N (2017) Wireless communication and mobile healthcare. Lecture notes of the institute for computer sciences. Social informatics and telecommunications engineering. Springer International Publishing
11. Dutt N, TaheriNejad N (2016) Self-awareness in cyber-physical systems. In: 2016 29th International conference on VLSI design and 2016 15th international conference on embedded systems (VLSID), pp 5–6
12. TaheriNejad N, Jantsch A (2019) Improved machine learning using confidence. In: The Annual IEEE Canadian conference on electrical and computer engineering, volume to appear, pp 1–5
13. Götzinger M, Anzanpour A, Azimi I, TaheriNejad N, Rahmani AM (2017) Enhancing the self-aware early warning score system through fuzzified data reliability assessment. In: International conference on wireless mobile communication and healthcare. Springer, pp 3–11
14. Urban RW et al (2015) Modified early warning system as a predictor for hospital admissions and previous visits in emergency departments. Adv Emerg Nurs J 37(4):281–289
15. Royal College of Physicians (2017) National Early Warning Score (NEWS) 2: standardising the assessment of acute-illness severity in the NHS. Updated report of a working party, London, RCP
16. Excel medical's WAVE clinical platform receives FDA clearance, Retrieved on February 2018. http://excel-medical.com/wp-content/uploads/RELEASE_FDA_Clearance_AchievedbyExcelMedical_1-8-18_RevMB6A.pdf
17. Kyriacos U et al (2014) Monitoring vital signs: development of a modified early warning scoring (mews) system for general wards in a developing country. PLoS One 9(1):e87073
18. Holbery N, Newcombe P (2016) Emergency nursing at a glance. Wiley
19. National Clinical Effectiveness Committee et al (2013) National early warning score national clinical guideline no 1
20. Anzanpour A et al (2015) Internet of things enabled in-home health monitoring system using early warning score. In: Proc. of MobiHealth
21. Hung M (2017) Leading the IoT, gartner insights on how to lead in a connected world. Gartner Research, 1–29
22. Azimi I et al (2017) Self-aware early warning score system for IoT-based personalized healthcare. In: Lecture notes of the institute for computer sciences, social informatics and telecommunications engineering, vol 181
23. Dutt N et al (2016) Towards smart embedded systems: a self-aware system-on-chip perspective. ACM TECS, Special Issue on Innovative Design Methods for Smart Embedded Systems 15(2):22–27
24. Hoffmann H et al (2013) A generalized software framework for accurate and efficient management of performance goals. In: Proceedings of the international conference on embedded software, pp 1–10
25. IBM Corporation An architectural blueprint for autonomic computing, 2006. IBM White Paper
26. Anzanpour A, Azimi I, Götzinger M, Rahmani AM, TaheriNejad N, Liljeberg P, Jantsch A, Dutt N (2017) Self-awareness in remote health monitoring systems using wearable electronics. In: Proceedings of design and test Europe conference (DATE), Lausanne
27. TaheriNejad N, Shami MA, SMPD (2017) Self-aware sensing and attention-based data collection in multi-processor system-on-chips. In: 2017 15th IEEE International new circuits and systems conference (NEWCAS), pp 81–84
28. Kholerdi HA, TaheriNejad N, Jantsch A (2018) Enhancement of classification of small data sets using self-awareness - an iris flower case-study. In: To be published in the proceedings of the international symposium on circuit and systems (ISCAS), Florence
29. TaheriNejad N, Jantsch A, Pollreisz D (2016) Comprehensive observation and its role in self-awareness; an emotion recognition system example. In: FedCSIS position papers, pp 117–124
30. Götzinger M, TaheriNejad N, Kholerdi HA, Jantsch A (2017) On the design of context-aware health monitoring without a priori knowledge; an ac-motor case-study. In: 2017 IEEE 30th Canadian conference on electrical and computer engineering (CCECE). IEEE, pp 1–5
31. Gotzinger M, Willegger E, TaheriNejad N, Jantsch A, Sauter T, Glatzl T, Lilieberg P (2018) Applicability of context-aware health monitoring to hydraulic circuits. In: IECON 2018-44th annual conference of the IEEE industrial electronics society. IEEE, pp 4712–4719
32. Götzinger M, Taherinejad N, Rahmani AM, Liljeberg P, Jantsch A, Tenhunen H (2017) Enhancing the early warning score system using data confidence. Springer International Publishing, Cham, pp 91–99
33. Götzinger M, TaheriNejad N, Kholerdi HA, Jantsch A, Willegger E, Glatzl T, Rahmani AM, Sauter T, Liljeberg P (2019) Model-free condition monitoring with confidence. Int J Comput Integr Manuf 32(4-5):466–481
34. Pasquier M, Moix P-A, Delay D, Hugli O (2015) Cooling rate of 9.4 °C in an hour in an avalanche victim. Resuscitation 93:e17–e18
35. Reule S, Drawz P (2012) Heart rate and blood pressure: any possible implications for management of hypertension? Curr Hypertens Rep 14(6):478–84
36. Davies P, Maconochie I (2009) The relationship between body temperature, heart rate and respiratory rate in children. Emerg Med J 26(9):641–3
37. Zila I, Calkovska A (2011) Effects of elevated body temperature on control of breathing. Acta Medica Martiniana
38. Guang L, Nigussie E, Rantala P, Isoaho J, Tenhunen H (2010) Hierarchical agent monitoring design approach towards self-aware parallel systems-on-chip. ACM Trans Embedded Comput Syst (TECS) 9(3):25
39. Hoffmann H, Maggio M, Santambrogio MD, Leva A, Agarwal A (2010) Seec: a framework for self-aware computing
40. Ross TJ (2009) Fuzzy logic with engineering applications. Wiley
41. Richmond VL, Wilkinson DM, Blacker SD, Horner FE, Carter J, Havenith G, Rayson MP (2013) Insulated skin temperature as a measure of core body temperature for individuals wearing cbrn protective clothing. Physiol Measur 34(11):1531

42. Garde A et al (2014) Estimating respiratory and heart rates from the correntropy spectral density of the photoplethysmogram. PLoS One, 9(1)
43. Pimentel MAF et al (2015) Probabilistic estimation of respiratory rate from wearable sensors. Springer, pp 241–62
44. Amiri D et al (2018) Edge-assisted sensor control in healthcare IoT. In: IEEE Global communications conference: selected areas in communications: E-Health
45. Sinex JE (1999) Pulse oximetry: principles and limitations. Am J Emerg Med 17(1):59–66
46. Maxim Integrated (Accessed Sept. 2018). https://www.maximintegrated.com/en/products/sensors/MAX30102.html

Affiliations

Maximilian Götzinger[1,2] · Arman Anzanpour[1] · Iman Azimi[1] · Nima TaheriNejad[2] · Axel Jantsch[2] · Amir M. Rahmani[3,4] · Pasi Liljeberg[1]

Arman Anzanpour
armanz@utu.fi

Iman Azimi
imaazi@utu.fi

Nima TaheriNejad
nima.taherinejad@tuwien.ac.at

Axel Jantsch
axel.jantsch@tuwien.ac.at

Amir M. Rahmani
a.rahmani@uci.edu

Pasi Liljeberg
pasi.liljeberg@utu.fi

1 Department of Future Technologies, University of Turku, Turku, Finland
2 Institute of Computer Technology, TU Wien, Vienna, Austria
3 Department of Computer Science, University of California Irvine, Irvine, CA, USA
4 School of Nursing, University of California Irvine, Irvine, CA, USA

Maximilian Götzinger, Nima TaheriNejad, Hedyeh A. Kholerdi, Axel Jantsch, Edwin Willegger, Thomas Glatzl, Amir M. Rahmani, Thilo Sauter, and Pasi Liljeberg

III

Model-Free Condition Monitoring with Confidence

International Journal of Computer Integrated Manufacturing, 2019; 32(4-5): 466–481

INTERNATIONAL JOURNAL OF COMPUTER INTEGRATED MANUFACTURING
2019, VOL. 32, NOS. 4–5, 466–481
https://doi.org/10.1080/0951192X.2019.1605201

Taylor & Francis
Taylor & Francis Group

ARTICLE

Model-free condition monitoring with confidence

M. Götzinger[a], N. TaheriNejad[b], H. A. Kholerdi[b], A. Jantsch[b], E. Willegger[b], T. Glatzl[c], A. M. Rahmani[b,d], T. Sauter[b,c] and P. Liljeberg[a]

[a]Department of Future Technologies, University of Turku, Turku, Finland; [b]Institute of Computer Technology, TU Wien, Vienna, Austria; [c]Department for Integrated Sensor Systems, Danube University Krems, Wiener Neustadt, Austria; [d]Department of Computer Science, University of California, Irvine, CA, USA

ABSTRACT

Computational Self-awareness can improve performance, robustness, and adaptivity of a system. As a key element of self-awareness, observation quality is critical to gain a correct and comprehensive understanding of the system, its own state, and the environment. This is of more importance in systems, where contextual information plays a crucial role in the functional operation of the system. In this paper, the authors introduce *confidence* as a quality metric of observation and leverage it to improve the correct identification of states of a system. To evaluate the impact of this factor on the context-aware monitoring system at hand and to show the generality of the approach, we conduct a series of tests with and without confidence for condition monitoring of an industrial AC motor and an experimental water pipe system. Our experiments show that *confidence* not only improves the quality of system performance but also simplifies the system architecture and enhances its robustness. These findings support the recent initiatives of paying more attention to observation as an important factor in self-awareness and, consequently, the performance of systems. The proposed system facilitates condition monitoring of various industrial systems and is easily deployable as it does not require a deep domain knowledge.

ARTICLE HISTORY
Received 2 June 2018
Accepted 13 March 2019

KEYWORDS
Industry 4.0; monitoring; model-free; context-awareness; self-awareness; confidence

1. Introduction

In industrial systems, manual maintenance and adjustment are often undesirable due to the high costs or the limited time available for adaptations. Increasingly, self-adaptiveness and self-awareness are desirable characteristics of many embedded and cyber-physical systems. They need to adapt to dynamic applications' phasic behaviour, changing environments and their own changing state due to phenomena such as ageing as well as the occurrence of faults. In many application domains, assessing the status of a system is commonly called *condition monitoring*, which serves diverse purposes such as early detection of faults, operation optimization, or planning of preventive maintenance.

In the early 2000's IBM formulated a vision of autonomic systems which called for self-adaptation and self-awareness (Kephart and Chess 2003). At the same time, Intel put forward a vision for proactive computing asking humans to get 'out of the loop' (Tennenhouse 2000), meaning that embedded systems should become more independent, autonomous and self-adaptive. A prerequisite for these ambitious goals is a detailed assessment of the state of a system and its environment. This assessment depends heavily on measuring relevant physical properties with sensors which is a part of Industry 4.0 (Alexopoulos et al. 2016). Many works on autonomous and adaptive systems assume the availability of correct measurements and their accurate interpretation. However, the quality of measurements and observations is not always easy to guarantee or assess. Hence, recently this challenge has received more attention (TaheriNejad, Jantsch, and Pollreisz 2016; Götzinger et al. 2017a; Anzanpour et al. 2017; Götzinger et al. 2016).

In this work, the authors focus on one of the observation aspects, namely *confidence*, and show how it can smoothen decision-making functions. Our concept of *confidence* is based on the work by TaheriNejad, Jantsch, and Pollreisz (2016) but differs from the *confidence levels assignment* used in other works, e.g. by Liu et al. (2008). To demonstrate this effect, we use this concept to enhance a black box monitor which we first introduced in (Götzinger et al. 2017b). This monitor distinguishes three different states (healthy, drifting, and broken) of the system it monitors. We demonstrate our approach by means of two case studies to illustrate its generality and application to different fields: an AC motor as in Götzinger et al. (2017b) and a water pipe system used as a model for condition monitoring in Heating, Ventilation and Air Conditioning (HVAC) systems (Glatzl et al. 2016). Extending our previous work (Götzinger et al. 2017b), we introduce the confidence metric and use fuzzy logic for its computation and for the decision-making process. We demonstrate that our proposed method improves the correct identification of motor states (i.e. increases robustness), reduces the dependency on system parameters such as threshold values (i.e. reduces sensitivity and increases robustness) and reduces the need for pre-processing of the sensory data (i.e. simplifies the architecture).

It is noteworthy that, with the growing prevalence of Industry 4.0 concepts and the increase in the number of cyber-physical systems and their associated sensors and actuators, monitoring has gained a growing importance while becoming more challenging. Model-free monitoring helps tackling this challenge from different angles. It can be applied to many

CONTACT M. Götzinger maxgot@utu.fi

different systems with substantial differences in their nature (e.g. as presented in this paper to condition monitoring of motors and water pipes) without requiring the deployment engineer to have a deep knowledge of the application field. Moreover, thanks to the fuzzy confidence evaluation proposed in this paper, it requires minimum or no adjustment of parameters, which saves a significant amount of time and resources otherwise necessary for implementing model-based monitors for a given application.

We introduce Confidence-based Context-Aware condition Monitoring (CCAM) with the following main contributions:

(1) We propose a fuzzy logic-based confidence metric for the quality assessment of systems under monitoring,
(2) we demonstrate the feasibility of the proposed monitor using two case studies: an industrial motor and a water pipe system,
(3) we demonstrate that CCAM gives equally good or better results than the similar system (Götzinger et al. 2017b), which does not use a confidence metric, and
(4) we demonstrate the robustness of the system by providing a sensitivity analysis of parameter settings and by showing that CCAM with confidence is better in identifying the correct system health status compared with a system without confidence. This is particularly notable when detecting drift situations in both the AC motor and the water pipe case studies.

After reviewing relevant related work in Section 2, we introduce confidence and describe our use of fuzzy logic in Section 3. In Section 4 we describe our proposed model-free condition monitoring approach that uses confidence to identify system states correctly. Section 5 shows and discusses results from our case studies, and Section 6 concludes the paper.

2. Background and related work

Measurements and processing of sensory data are relevant in a wide range of domains, like industrial processes, environmental monitoring, and medical applications. In all these areas the quality of measurements and observations are critical, however, less studied. Recently this aspect has come into the focus of research by characterizing various aspects (TaheriNejad, Jantsch, and Pollreisz 2016) and by demonstrating concrete benefits of using a nuanced and more realistic approach to observation (Götzinger et al. 2016; Anzanpour et al. 2017; TaheriNejad, Shami, and Manoj 2017). Götzinger et al. (2016) proposed data confidence to improve the reliability of patient monitoring in medical applications and Götzinger et al. (2017a) show that the usage of the plausibility of sensor values and value changes improves the robustness of the assessment of the health condition of a person.

AC motors are widely used in various industrial applications, and monitoring their health status is of interest to the industrial sector. A wear-out or other malfunctions in a motor may result in severe cracks or breaks in the rotor, stator or bearings, and can finally lead to reduced performance or – even worse – a failure (Ballal et al. 2007). Given the associated cost of these problems, non-invasive fault detection and preventative maintenance are important concerns in the industry (Gao, Cecati, and Ding 2015). It is known that during normal operation, the nominal range of outputs (e.g. current and speed) follows the nominal range of inputs (e.g. frequency or voltage). However, when the motor is free running in the presence of wearing out phenomena, some output signals deviate from the expected nominal values while the inputs remain unchanged. This can be used to detect wear-outs and certain other faults and failures. Various methods have been applied to the analysis of motor signals to detect faults (Kande et al. 2017). These monitoring systems mostly use methods such as current analysis (Féki, Clerc, and Velex 2013), temperature monitoring (Gao, Habetler, and Harley 2005), and vibration and noise analysis (Bellini et al. 2001); and they apply techniques such as hidden Markov modelling (Hatzipantelis and Penman 1993), thresholds (Mehala 2010), pattern recognition and neural networks (Bazan et al. 2017).

Condition monitoring in HVAC systems serves not only for fault detection, but also for optimizing control in order to reduce energy consumption (Massieh 2010). More generally, for the monitoring of water pipes, a wide variety of mostly distributed sensing principles is in use depending on the spatial extension of the network (Sadeghioon et al. 2018). For economic reasons, however, many monitoring systems rely on pressure and flow monitoring (Mounce et al. 2015), an approach we also apply in our case study in that we monitor the water flow.

With respect to data analysis, different methods have been proposed used for automated fault detection and diagnostic (FDD). These methods can be subdivided into quantitative model-based, qualitative model-based and process history-based methods (Katipamula and Brambley 2005). Process history methods are further differentiated in black-box and grey-box methods, and within the black-box methods, we distinguish between statistical, artificial neural network and other pattern recognition techniques. For example, Hyvrinen and Kärki (1996) present a fuzzy model for fault detection in HVAC systems. However, most of these methods require application-specific assumptions or models. Shun and Wen (2014) use Principal Component Analysis (PCA) and combine it with a Pattern Matching method to correctly identify the behaviour of the system without any additional knowledge of the system. While their work uses PCA's as well as distance similarity factors, we use confidence only based on fuzzy functions to detect the behaviour of the system correctly.

One generic method without detailed assumptions or a model of the system under observation is Context-aware Health Monitoring (CAH) (Götzinger et al. 2017b), which has been tested for monitoring industrial AC motors. CAH is a monitoring system that recognizes normal state changes as well as misbehaviour of the observed system. It accomplishes this task without a priori knowledge, and only through contextual information. However, two assumptions exist: first, the observed black box is in a steady state. Second, it is a *bijective function*, meaning that a unique input data set corresponds to one and only one output data set – and vice-versa. Thus, a change of the input inevitably is reflected in a change of the output and vice-versa. Otherwise, the observed system works incorrectly. We use CAH as the basis of our design and improve it using the concept of confidence to offset some of its major drawbacks, which are as follows: (i) A state-

full system, where current outputs depend on current inputs and the internal state of the system, cannot be reliably assessed; (ii) The signals of the observed system have to be smoothed through a filter; (iii) Despite filtering, the system performance is very sensitive to changes in signal values, e.g. due to normal transitions, noise, or instabilities; (iv) The thresholds used in CAH have to be set meticulously and accurately, otherwise, the system may not perform as expected, and hence, for each application tedious adjustment and tuning are necessary. Our proposed system addresses shortcomings (ii), (iii), and (iv), but shares the limitation (i) with CAH, as CCAM also can reliably assess only stateless systems.

3. Confidence

3.1. Definition of confidence

Confidence is a measure of the reliability of a system, a function, an analysis, or a process. Confidence can be defined as 'the extent to which a procedure may yield the same results on repeated trials' (TaheriNejad, Jantsch, and Pollreisz 2016). Hence, confidence can improve the self-awareness of a system regarding its subsystems and functions, and to what extent it can rely on the result each of them produces (Götzinger et al. 2016; Kholerdi, TaheriNejad, and Jantsch 2018).

To create a better and universal understanding of this concept, we formalize this definition in the following: Let us assume that f is an ideal function defined over $X = \langle x_0, \ldots, x_n \rangle$ and g is the unideal function at hand, also defined over X. We define the confidence of $g(x_i), x_i \in X$ as the inverse of a distance function $\Delta(g(x.), f(.))$, which captures the distance[1] between f and g based on some application specific distance metric:

$$c(g(x_i)) = \frac{1}{\Delta(f(x_i), g(x_i))}. \tag{1}$$

The overall confidence of g (i.e. confidence of the function/system in general, as opposed to its confidence at each point) is hence calculated as the average

$$C(g) = \frac{1}{n}\sum_{i=0}^{n} c(g(x_i)). \tag{2}$$

We note that $0 \leq c(g), C(g) \leq 1 = c(f) = C(f)$.

The implementation of a method to calculate c, however, is case dependent. In many cases, the ground truth (f) is not available and therefore, the distance cannot be calculated. In consequence, a function is often devised to estimate Δ. In the rest of this paper, we define and use heuristics as confidence functions without further reference to a distance metric.

3.2. Association assessment and confidence

As described in Section 2, the CAH system assumes to monitor a bijective function and hence, the relationship between the input and output data sets. Therefore, one of the main tasks is to assess the relationship of new data with previously observed data. For example, whether a new sample fits any of the recorded data in the history of the system or not. Thus, it is critical to make correct decisions as they affect the system performance, and it is important to know the confidence with which such decisions are made.

Let i be a variable of interest, for example, the voltage from a sensor or an abstracted observation after preprocessing. Further, let $v_{h_{i,j}}$ be the j^{th} value in the history of that variable h_i (i.e. j constitutes the position of the value in the history), and let $v_{i,new}$ be a new incoming value of this variable. Our task now is to assess whether the new sample belongs to the same group as the old sample(s) in the variable's history, and to provide a confidence value for our assessment. We use relative distance[2]

$$d_{i,j} = \left| \frac{v_{i,new} - v_{h_{i,j}}}{v_{i,new}} \right| \tag{3}$$

as a metric for this assessment and for computing the following confidence values.

Two different properties define how well a new value $v_{i,new}$ matches an existing data set: how many values of the existing data set are close to $v_{i,new}$, and how close they are. For the latter, inspired by fuzzy logic, we define four points and three intervals[3] for determining the membership of the new value (see Figure 1(a)). If the new value $v_{i,new}$ is in the interval $[d_b, d_c]$, it belongs to the same set of data as $v_{h_{i,j}}$ with certainty. If it falls in one of the intervals $[d_a, d_b]$ or $[d_c, d_d]$, it may or may not belong to that group; otherwise, it does not. Then, we define c_{sv} ('similar value') as the confidence of $v_{i,new}$ belonging to the same data set (group of values) as $v_{h_{i,j}}$, and we compute it as follows

$$c_{sv,i,j} = \begin{cases} \frac{d_a - d_{i,j}}{d_b - d_a} & \text{if } d_a < d_{i,j} < d_b \\ 1 & \text{if } d_b \leq d_{i,j} \leq d_c \\ \frac{d_d - d_{i,j}}{d_d - d_c} & \text{if } d_c < d_{i,j} < d_d \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

where the relative distance is calculated by Equation (3).

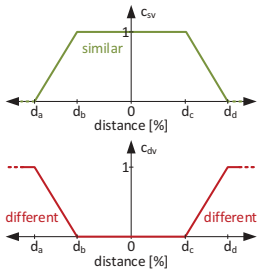
The counterpart of c_{sv} is c_{dv} ("different value"), the confidence of not belonging to the same data set, and we compute it as follows

$$c_{dv,i,j} = \begin{cases} \frac{d_b - d_{i,j}}{d_b - d_a} & \text{if } d_a < d_{i,j} < d_b \\ 0 & \text{if } d_b \leq d_{i,j} \leq d_c \\ \frac{d_{i,j} - d_c}{d_d - d_c} & \text{if } d_c < d_{i,j} < d_d \\ 1 & \text{otherwise.} \end{cases} \tag{5}$$

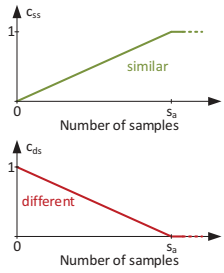
where the relative distance $d_{i,j}$ is also calculated by Equation (3).

If the intervals of the functions c_{sv} and c_{dv} are identical, the computation of c_{dv} can be simplified to $c_{dv,i,j} = 1 - c_{sv,i,j}$.

The other factor, which determines whether a new value $v_{i,new}$ matches an existing data set, is the number of values in the existing data set that are close to the new value. The more samples in close proximity, the likelier the new data fits the existing data set. Therefore, there are fuzzy functions such as the ones shown in Figure 1(b), in which, if a new sample $v_{i,new}$ does not match any of the existing data, the confidence $c_{ss,i}$ of $v_{i,new}$ being a member of the existing data set is 0, and the confidence $c_{ds,i}$ of $v_{i,new}$ not being a member of the existing data set is 1. If there are at least s_a samples in the vicinity of the new data, the new sample certainly fits the existing data set ($c_{ss} = 1$ and $c_{ds} = 0$). The function



(a) Confidences of a new sample being similar (c_{sv}) and being different (c_{dv}) to an existing datum, based on the of distance between them.



(b) Confidences of a new sample being similar (c_{ss}) and being different (c_{ds}) to an existing data set, based on the number of samples.

Figure 1.: Fuzzy functions showing the confidence of the system for considering a new sample fitting an already existing data set.

$$c_{ss,i,k} = \begin{cases} 1 & \text{if } k \geq s_a \\ \frac{k}{s_a} & \text{if } 0 \leq k < s_a \end{cases} \quad (6)$$

estimates the confidence for a new sample belonging to the existing data set, and

$$c_{ds,i,k} = \begin{cases} 0 & \text{if } k \geq s_a \\ \frac{s_a - k}{s_a} & \text{if } 0 \leq k < s_a \end{cases} \quad (7)$$

estimates with what confidence the new sample does not belong to the existing data set.

If the intervals of the functions $c_{ss,i,k}$ and $c_{ds,i,k}$ are identical, the computation of $c_{ds,i,k}$ can be simplified to $c_{ds,i,k} = 1 - c_{ss,i,k}$.

To make a final decision on whether the new value, $v_{i,new}$, belongs to the existing data set, the two factors (the confidences $c_{sv,i,j}$ and $c_{ss,i,k}$) above have to be combined, leading to the overall confidence $c_{b,i}$. Hence, $c_{b,i}$ (for $v_{i,new}$ belonging to an existing data set) is composed of the confidences $c_{sv,i,1} \ldots c_{sv,i,k}$ (Equation (4)) calculated by the comparisons of $v_{i,new}$ with k values of the existing data set, and the confidence $c_{ss,i,k}$ (Equation (6)). For this purpose, we propose to use the conjunction ($\wedge$) operator as follows

$$c_{b,i} = (c_{sv,i,1} \wedge \cdots \wedge c_{sv,i,k}) \wedge c_{ss,i,k} = (\bigwedge_{j=1}^{k} c_{sv,i,j}) \wedge c_{ss,i,k} \quad (8)$$

because the new sample $v_{i,new}$ belongs to the existing data set only if $v_{i,new}$ is sufficiently similar to all k values (and k should be a large enough number). A conjunction in fuzzy logic is equal to a minimum function (Ross 2009). Hence, this operation results in the minimum of all k calculated $c_{sv,i,j}$'s. Since the result of all k disjunctions is disjunctioned with $c_{ss,i,k}$, which is based on the number of comparisons, the confidence cannot be larger than $c_{ss,i,k}$.

The overall confidence of $v_{i,new}$ not belonging to an existing data set ($c_{n,i}$) is composed of all k confidences, $c_{dv,i,1} \ldots c_{dv,i,k}$, and $c_{ds,i}$. For this purpose, we propose to use the disjunction ($\vee$) operator as follows

$$c_{n,i} = (c_{dv,i,1} \vee \cdots \vee c_{dv,i,k}) \vee c_{ds,i,k} = (\bigvee_{j=1}^{k} c_{dv,i,j}) \vee c_{ds,i,k} \quad (9)$$

because the new sample $v_{i,new}$ does not belong to the existing data set if $v_{i,new}$ is different to at least one of the k values or the number of values k is too low. A disjunction in fuzzy logic is equal to a maximum function (Ross 2009). The result of this computation is equal to the maximum of all k $c_{dv,i,j}$, limited by $c_{ds,i,k}$ which is based on the number of comparisons.

In a last step, to determine the chances of the new sample belonging to the existing data set, we compare the calculated confidences. Only if c_b is larger than c_n, we declare the new value as belonging to the existing data set.[4]

4. Confidence-based context-aware condition monitoring

In this section, we present the details of the proposed CCAM system, in particular, the fuzzy operations. Thanks to the benefits of the fuzzy logic-based confidence concept, in CCAM there is no need for filtering, except for one case in the motor case study. In Section 5.4, we show this in detail. Note that the task of pre-processing is application specific and may differ from case to case. Some signals are not directly usable by the CCAM system because CCAM works only with steady states. Alternating signals, such as AC current and AC voltage will never be in a steady state. Hence, the pre-processing unit of the proposed system (shown in the green frame in Figure 2) abstracts the alternating signal into a non-alternating form. In the rest of this section, we first present a brief generic description of the state handler, and then move to part showing how different states and potential malfunctions are recognized. We end this section by presenting how the overall confidence of the system regarding its recognition is assessed.

4.1. State handler

The heart of the CCAM system is the State Handler (SH) which is shown in the blue frame in Figure 2. The SH detects when the System under Observation (SuO) i) is in a steady state, ii) changes its state, or iii) is not working correctly. For this reason, the SH saves information about these SuO states in C++ objects

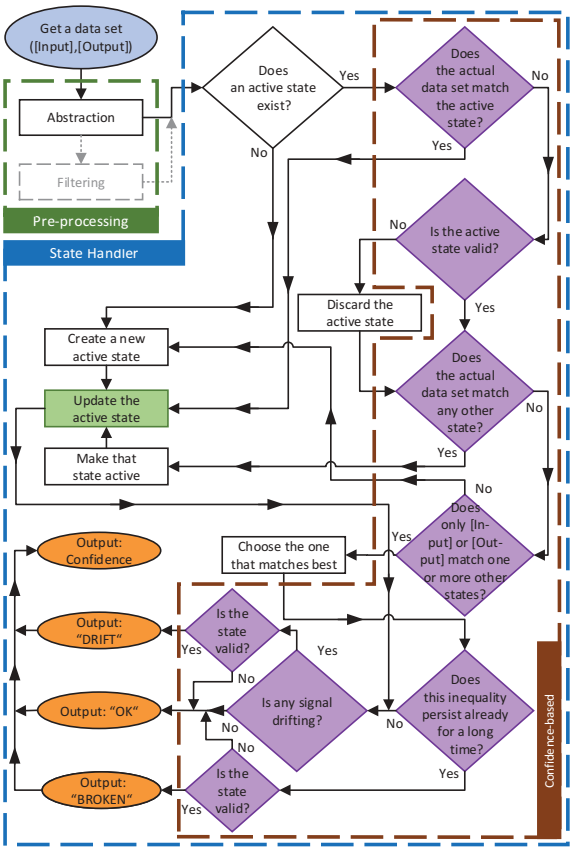


Figure 2. Flow chart of the CCAM system proposed here.

which are – for the sake of convenience – called states. These states contain a sliding history for each signal of the SuO. In the following, we describe the main tasks of this unit.

4.2. Recognizing states and detecting state changes

The SH (shown in the blue frame in Figure 2) saves information about every SuO state detected in a sliding history h. More precisely, one history exists for each variable i (input and output signals) of the SuO $\langle h_i \rangle$. Whether a state is valid or not is reflected by two confidence values: the confidence c_{val} which indicates whether a state is valid and the confidence c_{inv} when a state is invalid. Similar to the decision whether a new sample set fits a state (see Section 3.2), the less the deviations between the sample values saved in a state and the higher the number of sample sets in this state, the higher is the confidence of detecting a valid state. Conversely, large deviations between the samples or only a few samples inserted in the state indicate that the suspected state is most likely not a real state (e.g. a transient state). A state is considered as a valid state when c_{val} is higher than c_{inv}. In this case, the SH saves the respective state in the state vector. How these two confidence values are calculated is explained is detail in Section 4.4.

To observe regular state changes or an unwanted malfunction, the SH examines whether an incoming data set, consisting of the different signals $\langle v_{i,new} \rangle$ of the SuO, fits the actual state, termed active state (shaded in purple in Figure 2). Therefore, the SH calculates both, the confidence c_b (Equation (14)) for deciding that the new data set matches the active state and the confidence c_n (Equation (15)) which indicates that the new data set does not belong to the active state. Based on the confidence value, the SH decides whether the set of samples matches the active state.

How these two confidences are computed is shown in detail in Section 4.3). If c_b is higher than c_n, the data is inserted into the active state, i.e. the data is saved in the sliding window history of it. This observation is indicative of a well functioning system, unless the average value of any signal of the state is not drifting. How the SH observes a possible drifting of a signal is described in Section 4.6.

If the new data set mismatches the active state, a state change is identified. This can be either a normal state change or a malfunction. If both, input and output data sets mismatch the active state, a regular state change has happened. When the SH observes such a behaviour, the actual samples are compared to all states saved in the state vector. The state to which the samples are fitting is selected as the new active state. If the samples do not match any of the saved states, the SH creates a new active state. Regardless of whether the actual data set is compared with the active state or with any other state, the function of calculating the confidences c_b and c_n are the same. Afterwards, the actual samples are inserted into the new active state.

If exclusively one (either input or output) data set matches the actual state and the other one does not, a malfunction is detected, which is described further in Section 4.5.

4.3. Ascertaining a set of samples matches a state

To determine whether the new sample set matches a state, as a first step, a confidence value $c_{b,i}$ is computed for every new signal sample $v_{i,new}$ (e.g. voltage, current, etc.) by Equation (8). As described in Section 3.2, this confidence depends on both, the number of samples of the existing data in close proximity (see Equation (6)) of $v_{i,new}$ and how close they are (see Equation (4)). To compute the confidence whether the new sample set matches a state, the best fitting subset (size k) of history values is chosen. The new sample fits to the state if it is similar to a high number of history samples while at the same time the distance to all of the history samples is low. However, these two requirements can compete, particularly, in the case of history values with both lower and higher distances to the new sample. Furthermore, since belonging to a set is not determined by fixed thresholds, it is not trivial to decide whether the distance between the samples is close enough or how many samples have to be in each others' proximity to form a state. To find the best and – at the same time – the biggest matching subset of a variable's history, $\langle h_i \rangle$, all possible cases of comparisons are computed; from subset size 1 to n, whereas n is the size of the history. These computations, based on Equation (8), are computed as follows

$$\begin{aligned} &\text{case } 1: && c_{b_1,i} && = c_{sv,i,1} \wedge c_{ss,i,1} \\ &\text{case } 2: && c_{b_2,i} && = (c_{sv,i,1} \wedge c_{sv,i,2}) \wedge c_{ss,i,2} \\ &\qquad\vdots && && \\ &\text{case n}: && c_{b_n,i} && = (c_{sv,i,1} \wedge \cdots \wedge c_{sv,i,n}) \wedge c_{ss,i,n} \end{aligned} \tag{10}$$

where n denotes the size of the history (number of samples saved in the history), and $c_{sv,i,j} \geq c_{sv,i,k}$; $\forall j \leq k$.

Next, the SH chooses the best possible case with a subset of size k of Equation (10) (corresponding to Equation (8)) for $v_{i,new}$ belonging to the data set $\langle v_{h_i} \rangle$ as follows

$$c_{b,i}(v_{i,new} \in \langle v_{h_i} \rangle) = \bigvee_{j=1}^{n} c_{b_j,i}. \tag{11}$$

The confidence of not belonging ($c_{n,i}$) is also calculated in a similar fashion, but by using the disjunction operator for all n cases. That is,

$$\begin{aligned} &\text{case } 1: && c_{n_1,i} && = c_{dv,i,1} \vee c_{ds,i,1} \\ &\text{case } 2: && c_{n_2,i} && = (c_{dv,i,1} \vee c_{dv,i,2}) \vee c_{ds,i,2} \\ &\qquad\vdots && && \\ &\text{case n}: && c_{n_n,i} && = (c_{dv,i,1} \vee \cdots \vee c_{dv,i,n}) \vee c_{ds,i,n}. \end{aligned} \tag{12}$$

where n denotes the size of the history (number of samples saved in the history), and $c_{dv,i,j} \geq c_{dv,i,k}$; $\forall j \leq k$.

Finally, the lowest confidence with which we can consider $v_{i,new}$ not belonging to $\langle v_{h_i} \rangle$ is calculated using

$$c_{n,i}(v_{i,new} \ni \langle v_{h_i} \rangle) = \bigwedge_{j=1}^{n} c_{n_j,i}. \tag{13}$$

After the confidences $c_{b,i}$ and $c_{n,i}$ have been calculated for every variable i, the confidence of the whole sample set belonging to the considered state, c_b, is calculated by

$$c_b = \bigwedge_{i=1}^{m} c_{b,i} \tag{14}$$

because a new data set belongs to a given state only if the new samples of all variables match previous values of respective variables in that state. Its counterpart, the confidence of the whole sample set not belonging to the considered state, c_n, is calculated by

$$c_n = \bigvee_{i=1}^{m} c_{n,i} \tag{15}$$

because the data set does not belong to a state if one or more samples do not match the existing values of the variables of that state. In both equations m is the number of variables.[5]

We note that the confidence functions in Equations (6) and (7) are adaptable in a way that the boundary s_a is equal to the number of the already saved values in the state, bounded by the history length n. Without this adaptive property, due to Equations (11) and (13), the confidence c_n for each variable would be inevitably higher than c_b for any new state with few recorded values.

4.4. Validate a state

When an actual signal sample is added to a state, it is compared with all history values of this signal, saved in the state (Figure 3). From all these comparisons both, the confidences $c_{sl,i}$ (lowest $c_{sv,i,j}$) for being similar and $c_{dh,i}$ (highest $c_{dv,i,j}$) for being different are saved in the history in addition to the actual value. That is,

$$c_{sl,i} = \bigwedge_{j=1}^{n} c_{sv,i,j} \tag{16}$$

and

$$c_{dh,i} = \bigvee_{j=1}^{n} c_{dv,i,j} \tag{17}$$

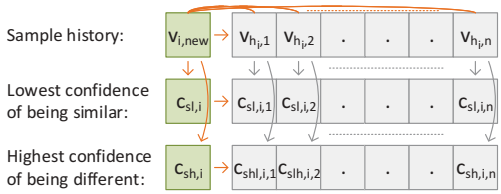


Figure 3. A graphic demonstration of a new value being compared to the values in the history. This figure shows also the storage of respective confidences of belonging and not belonging.

where n is the size of the history.

To validate the state, the confidences c_{val} of having found a valid state is calculated by

$$c_{val} = \left(\bigwedge_{i=1}^{m} c_{sl,i}\right) \wedge c_{ss,i,n} \tag{18}$$

because a state is valid if it contains many samples, and all of them are similar to each other. Its counterpart c_{inv} is calculated by

$$c_{inv} = \left(\bigvee_{i=1}^{m} c_{dh,i}\right) \vee c_{ds,i,n} \tag{19}$$

because a state is invalid if it contains only a few samples, or at least one of these samples is different. In both equations, m is the number of variables and n is the size of the history. The confidences $c_{ss,i}$ and $c_{ds,i}$ are calculated by Equation (6) and Equation (7). These confidences depend on the number of comparisons (i.e. length of the history), but it makes no difference which variable i is taken for $c_{ss,i,n}$ (Equation (18)) and $c_{ds,i,n}$ (Equation (19)) because the history of every signal has the same length. This algorithm is similar to evaluating whether a sample set belongs to a state (in Section 4.3) with the difference that the comparison is done with all of the history values and not only with a subset of them (i.e. k samples). A state is only valid if the confidence c_{val} is higher than its counterpart c_{inv}, whether both confidences are low or high.

We note that the historical information is not limited only the n values in the history. Some information of the previous values are saved in a more abstracted form. The reason being that the confidences of older history values contain information also about history values that are already out of the sliding window.

4.5. Recognizing a malfunction

Since the monitored system is treated as a bijective function, a change of only one data set (input or output exclusively) can indicate an anomaly. If the input and output data sets are or are not belonging to the active state is separately calculated through Equations (14) and (15). Since different systems show different delays to reflect a change in the output due to an input change, a small time gap between the two shall be allowed. In other words, if the other (unchanged) data set follows within a short time, the system still works correctly. Therefore, two other confidences c_{brk} and c_{ok} are calculated similarly using

$$c_{brk} = \begin{cases} 1 & \text{if } s_t \geq s_a \\ \frac{s_t}{s_a} & \text{if } 0 \leq s_t < s_a \end{cases} \tag{20}$$

and

$$c_{ok} = \begin{cases} 0 & \text{if } s_t \geq s_a \\ \frac{s_a - s_t}{s_a} & \text{if } 0 \leq s_t < s_a \end{cases} \tag{21}$$

where $s_t \in \mathbb{Z}_{\geq 0}$ is the time gap (in samples) between the change of the two data sets (input and output), thus, the time which the output needs to react on a change in the input. The more time elapses, the more likely the SuO is broken; reflected in c_{brk} is becoming higher and c_{ok} is becoming lower. In this case, CCAM signals that the SuO is broken but only if the actual state is valid. Otherwise, this discrepancy is most likely because the SuO is in a transient state. In this case, the SH just discards the active state, creates a new one and saves the actual samples in the new state.

We note that if the SuO changes back into an already known state, $c_{ss,i,n}$ of Equation (18) and $c_{ds,i,n}$ of Equation (19) denote the number of samples which were inserted into the state after re-entrance into it. This is necessary because the signals may still be unsteady after a change and cause a wrong recognition of the state. Therefore, the new active state should not be considered as valid directly after re-entrance. Once there have been enough number of samples similar to an already existing state, we can recognize that state as reactivated.

If the intervals of the functions c_{brk} and c_{ok} are identical, the computation of c_{ok} can be simplified to $c_{ok} = 1 - c_{brk}$ and the condition $c_{brk} > c_{ok}$ becomes equal to $c_{brk} > 0.5$.

4.6. Recognizing a signal drift

A system under observation can have another condition besides broken or healthy. When one or more signals are drifting (changes continuously but very slowly) characterizes another abnormal operation. In other words, a series of values of a signal belong to the same state, but the signal is gradually deviating outside its normal expected range. This very slow change is indistinguishable in the sliding history window of the state because the samples saved in the history change as the signal is drifting. Therefore, the task of inserting sample values to the active state is more complex than just saving them in the history (shaded in green in Figure 2). The SH additionally calculates Discrete Average Blocks (DABs) the average values of a certain number of sample values (DAB_{size}) inserted in the state (Figure 4). The more different the actual

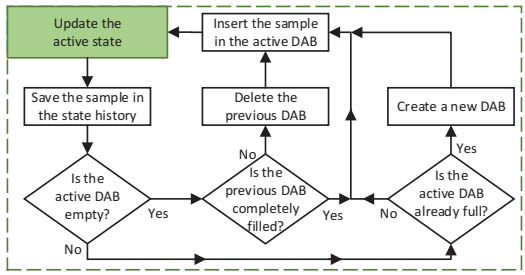


Figure 4. Block diagram of the state updating task of the proposed CCAM system. Discrete Average Blocks (DABs) are also created and kept in this procedure.

DAB to the first DAB, the more likely a signal is drifting. However, only completed DABs are compared with each other. An incomplete DAB could lead to a false result because of an outlier sample. The determination whether a signal is drifting happens after the new sample set was inserted into the active state. The confidences c_{dft} and c_{stb}, which indicates that the signal is drifting, respectively, stable, are calculated by

$$c_{dft} = \begin{cases} \frac{d_b - d_{dft}}{d_b - d_a} & \text{if } d_a < d_{dft} < d_b \\ 0 & \text{if } d_b \leq d_{i,j} \leq d_c \\ \frac{d_{dft} - d_c}{d_d - d_c} & \text{if } d_c < d_{dft} < d_d \\ 1 & \text{otherwise.} \end{cases} \tag{22}$$

and

$$c_{stb} = \begin{cases} \frac{d_{dft} - d_a}{d_b - d_a} & \text{if } d_a < d_{dft} < d_b \\ 1 & \text{if } d_b \leq d_{dft} \leq d_c \\ \frac{d_d - d_{dft}}{d_d - d_c} & \text{if } d_c < d_{dft} < d_d \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

where d_{dft} is the relative distance between the two compared DABs. That is,

$$d_{dft} = \left| \frac{v_{avg,DAB_{first}} - v_{avg,DAB_{new}}}{v_{avg,DAB_{first}}} \right| \tag{24}$$

where $v_{avg,DAB_{new}}$ is the average value of the latest (completed) DAB, and $v_{avg,DAB_{first}}$ is the average value of the state's first DAB.

If d_{dft} is higher or equal than d_{stb}, the SH of CCAM signals that the active state is drifting but only if the actual state is valid. Otherwise, this discrepancy is most likely because the SuO is in a transient state.

If the intervals of the functions c_{dft} and c_{stb} are identical, the computation of c_{stb} can be simplified to $c_{stb} = 1 - c_{dft}$ and the condition $c_{dft} \geq c_{stb}$ becomes equal to $c_{dft} \geq 0.5$.

4.7. Overall confidence of the CCAM system

The CCAM system outputs, besides the assessed health condition of the SuO, how confident it is about its assessment. This confidence c is case dependent and calculated by

$$c = \begin{cases} (c_{n,in} \vee c_{n,out}) \wedge c_{brk} \wedge c_{val} & \text{; if broken} \quad (25) \\ (c_{b,in} \wedge c_{b,out}) \wedge c_{dft} \wedge c_{val} & \text{; if drifting} \quad (26) \\ ((c_{b,in} \wedge c_{b,out}) \vee (c_{n,in} \wedge c_{n,out})) & \\ \quad \wedge c_{ok} \wedge c_{stb} \wedge c_{val} & \text{; if normal} \quad (27) \end{cases}$$

where $c_{b,in}$, $c_{b,out}$, $c_{n,in}$, and $c_{n,out}$ are calculated by Equations (14) and (15) for input and output variables, respectively.

5. Evaluation

In this section, two case studies are investigated: an AC Motor and a water pipe system. First, we describe data and experimental setup used for testing of our proposed CCAM system. It is noteworthy that the configuration of CCAM is the same for both case studies. That is, a down sampling rate of 50, $[d_a, d_b, d_c, d_d] = [-14\%, -1\%, 1\%, 14\%]$ (Equation (4)), $s_a = 10$ (Equation (6)), $DAB_{size} = 10$, $[d_a, d_b, d_c, d_d] = [-30\%, -10\%, 10\%, 30\%]$ (Equation (22)), and $s_a = 20$ (Equation (20)). Next, we discuss the results and compare them with CAH (Götzinger et al. 2017b). The results of our sensitivity analysis, where we evaluate the effect of variations in configuration parameters on the performance of the system, is also presented.

5.1. Motor case study

We have used the simulation results from (Götzinger et al. 2017b) for comparison with our simulations of CCAM. Free running, change of speed and change of load behaviours have been simulated to cover normal states and state changes. For abnormal behaviour, the drift has been modelled applying an increase to the mechanical torque. The broken state, instead, has been provided by Bessous et al. (2018) from a real experiment on a motor with broken bearings. To evaluate the system more extensively, we added longer and more complex test scenarios simulated with the same simulation tools we used in Götzinger et al. (2017b). The motor is a single squirrel-cage, three-phase, 380 V, 50 Hz, 3 kW induction motor with four poles whose parameters are based on asynchronous machine model using the SI dialogue box in MATLAB®. The input and output signals are voltage, current, torque, and speed. For the broken state, the vibration signals have been collected in addition.

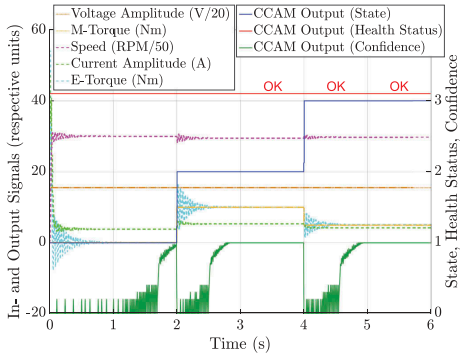
The CCAM system is modelled in C++ and fed with CSV-files containing data sets of the AC motor. The alternating signals such as AC voltage and AC current were abstracted beforehand to their amplitudes using MATLAB®. Table 1 outlines the various scenarios used in our AC motor experiments. In the following, we explain some example scenarios in more details and show how CCAM successfully classifies them.

5.1.1. Normal operation

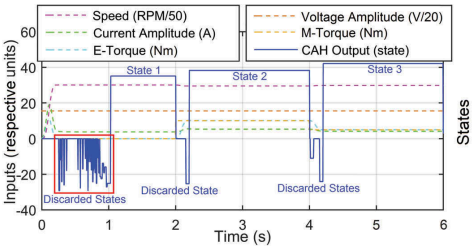
A normally working SuO can remain in a given state or change its state. We successfully tested the proposed CCAM system in various scenarios (Table 1). In the following, two of them will be discussed in detail. Figure 5(a) shows the scenario explained first (scenario 3 in Table 1). The motor was turned on, in the beginning, followed by two state changes caused by external load changes. The CCAM system continuously tries to identify the state of the system. When the motor is turned on, the output signals oscillate considerably for a relatively long time. Without detected state, it is not possible to determine whether only the input or the output data sets have changed. Because of this unsteady state, the CCAM system does not trust the system in the beginning of its first state, that is, the confidence is close to 0 ($c \approx 0$). Then, around 1.8 seconds the SuO settles to a steady state which is reflected in an increased confidence ($c \approx 1$). At 2*s*, the external load is changed from 0 to 10*Nm*. Therefore, all output signals start to change again; accompanied by an oscillating phase. Because of the relatively small change in the input signals, the period in which the SuO is unsteady is shorter. Around 0.5*s* after the external load has been changed, the CCAM system is confident about the recognition of the second state and the good health condition of the SuO. The same can be seen at 4*s* when the external load changes again. The scenario explained second (scenario 4 in Table 1) is similar but more complex. Figure 6 shows that the motor was turned on, followed by 21 times external load changes (with six different loads). CCAM detected all the six distinct states of the system and its 21 state changes.

Table 1. Various experiments (test scenarios) performed with an AC motor.

#	AC motor condition	Record length	Sampling frequency	Events in the scenario
1	OK	6 s	10 kHz	The motor was turned on in the beginning and remained without any state changes during the experiment.
2	OK	20 s	1 kHz	The motor was turned on in the beginning, followed by one state change caused by a change of input voltage and frequency.
3	OK	6 s	10 kHz	The motor was turned on in the beginning, followed by two state changes caused by external load changes. All three states differ from each other.
4	OK	60 s	1 kHz	The motor was turned on in the beginning, followed by 21 state changes caused by external load changes. This results in six different states for the motor.
5	OK	120 s	10 kHz	The motor was turned on in the beginning, followed by 51 state changes caused by external load changes. This results in seven different states for the motor.
6	OK	3600 s	1 kHz	The motor was turned on in the beginning, followed by 383 state changes caused by external load changes. This results in seven different states for the motor.
7	Drift	10 s	10 kHz	The motor was turned on in the beginning and remained without any state changes. Because of a wear-out of the motor signals are drifting very slowly.
8	Broken	10 s	12.5 kHz	The motor was abnormally running from the beginning without any state changes.



(a) Proposed CCAM system output during state changes. The legend shows the inputs of the SuO with dotted lines and the outputs with dashed lines.



(b) CAH system output during state changes (Götzinger et al. 2017b). The original plot does not reveal the numeric encoding of the state. Therefore, we labeled the states in the plot as states 1, 2, and 3. States shown as negative mean indicate that particular state was not saved during run-time.

Figure 5. Comparison of CCAM and CAH which does not consider confidence.

5.1.2. Wear-out

As shown in Figure 7(a), because of the oscillating signals (in scenario 7 of Table 1), in the beginning, it takes the CCAM system around 1.8*s* until it is confident that it has recognized a steady state. Shortly afterwards, the confidence of the CCAM is again falling because a drift is detected. Around 2*s*, the CCAM system changes status and raises the drift alarm. At around 3*s*, the system is highly confident about this decision. This circumstance lasts up to the end of the experiment.

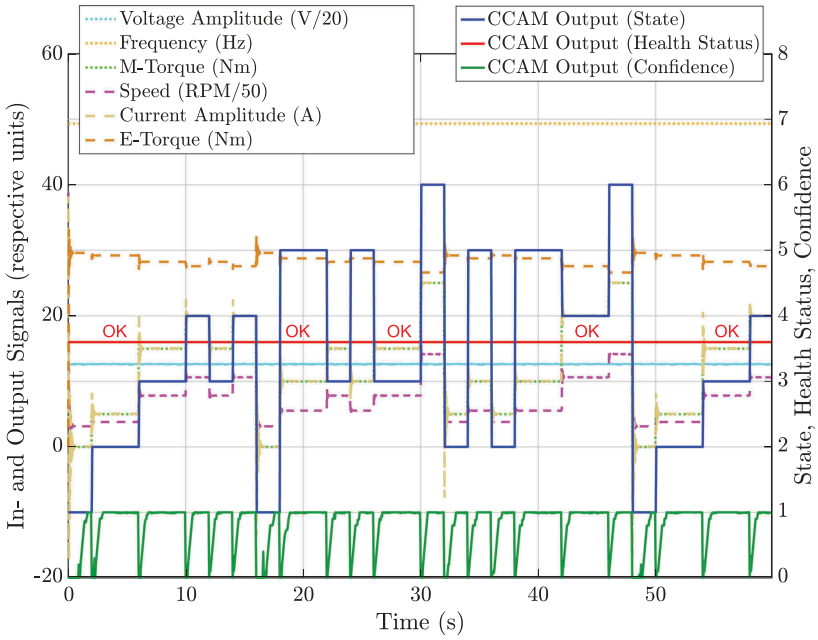
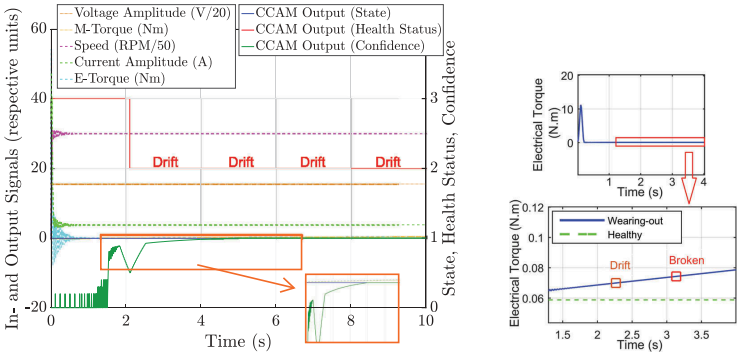


Figure 6. Proposed CCAM system output during many state changes of the AC motor. The legend shows the inputs of the SuO with dotted lines and the outputs with dashed lines.

5.1.3. Anomaly

In the case of a bearing defect, scenario 8 of Table 1 shown in Figure 8, the vibration and the current signals start to change significantly which leads to many peaks. These changes and peaks in the output signals of the SuO result in the CCAM system recognizing the SuO as broken. Due to the fact that the records of the broken motor lack data of the motor working well in the beginning, the CCAM system can find a state in this unsteady data



(a) Proposed CCAM system outputs during a drift. The legend shows the inputs of the SuO with dotted lines and the outputs with dashed lines.

(b) CAH system outputs during a drift (Götzinger et al. 2017b).

Figure 7. Monitoring drift in the AC motor case study. Comparison of CCAM and CAH.

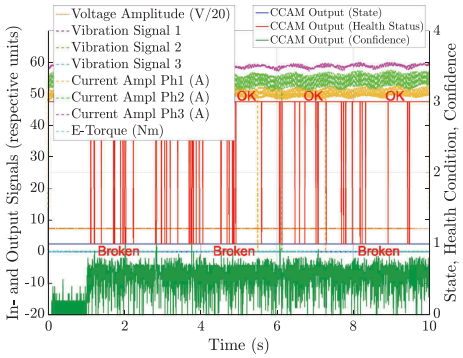


Figure 8. CCAM behaviour in the case of the bearing defect. Since the system is broken from the beginning, CCAM cannot identify a correct and stable state. Thus, it oscillates between broken and okay in its assessment and never converges. This oscillation represents the anomalous behaviour of the SuO. The legend shows the inputs of the SuO with dotted lines and the outputs with dashed lines.

(1*s*) and the mean of the confidence is rising. However, the output of CCAM changes between 'OK' and 'Broken' over and again, which signifies the broken condition. In future work, we will study if such unstable assessment could be automatically analyzed further and expressed explicitly and reliably as one broken alarm.

5.2. Water pipe system case study

In this case study, a pump is driving water through a pipe system. The system is controlled with a Raspberry Pi in combination with Arduino Uno. The desired velocities are set via a python script and all sensor values are logged. The input signal is a DAC output (normalized voltage) of the Raspberry Pi which ranges from 0.3 to 1.0, corresponding to the pump voltage range of 3 V to 10 V. The output signals are the water temperature as well as several volumetric flows. The temperature in the pipe system is measured with two temperature sensors (Pt100[6] at different positions) and the volumetric flow is measured with four flow sensors at different positions. The main part of the experimental setup is depicted in Figure 9. On the left side of this Figure, the sensors of Dynasonic and Riels are not displayed. Three different types of flow sensors are used to show that the system is able to work in a heterogeneous sensor system. The first type of sensors are in-situ ultrasonic sensors (Two Sharky FS 473 – SharkyS and SharkyB are their corresponding acronyms in the figures and in the rest of the paper). The second type is a clamp-on ultrasonic sensor (Dynasonics TFX Ultra), and the last one is another clamp-on ultrasonic sensor (Riels RIF600P). SharkyS and SharkyB are placed parallel to each other whereas the sensors Dyna and Riels are in series to each other and to the other two. The data of the Riels is not used for these experiments since the sensor was not set up correctly for this type of measurement. All measured values, actuator and sensor values are stored in CSV-files on the Raspberry Pi and later post-processed on another computer (because of the limited performance of the Raspberry Pi). Table 2 outlines several scenarios used in our experiments. In the following, we explain some of them in more details and show how CCAM classifies them correctly.

5.2.1. Normal operation

The CCAM system successfully detects different normal operation scenarios, with and without state changes. Two of these scenarios are discussed in detail in the following. Figure 10 shows scenario 3 of Table 2, where the voltage is increased two times after the water pump has been started. The temperature remains constant during the entire experiment. The valve before sensor SharkyS is closed, which means the sensors SharkyB and Dyna measure the entire water flow. The sensors SharkyS and Dyna need about 4 seconds of setup time until a steady state is reached allowing an accurate flow measurement. The CCAM system is able to detect all three states correctly. It needs 25–30 seconds to reach a high confidence in the state, and then remains high until the input is changed. As expected, the health status is also recognized as OK.

Figure 10(b) shows scenario 4 of Table 1 which is very similar to the previous scenario but the first and the last SuO states are the same. As we see in Figure 10(b) the voltage was increased first and then decreased again to its initial state. CCAM detected both state changes, recognized that the SuO

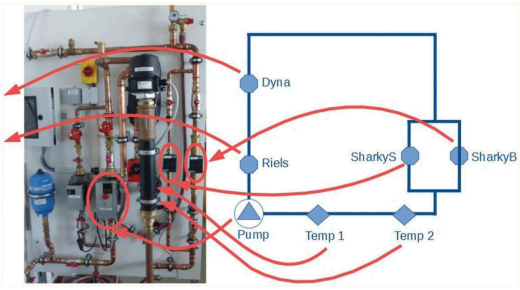


Figure 9. Parts of the water pipe system and schematics of the most important used elements. Dyna and Riels sensors are outside the picture (they would have been at the left side of the current frame of the picture) .

Table 2. Various experiments (test scenarios) using the water pipe system.

#	HVAC system condition	Record length	Sampling frequency	Events in the scenario
1	OK	649 s	30.5 Hz	The pump was started in the beginning and remained without any state changes during the experiment.
2	OK	625 s	30.5 Hz	The pump was started in the beginning, followed by one voltage increase, resulting in two different states during the experiment.
3	OK	627 s	30.5 Hz	The pump was started in the beginning, and afterwards, the voltage was increased two more times, resulting in three different states during the experiment.
4	OK	638 s	30.5 Hz	After the pump was started in the beginning, the voltage was increased once and changed back after some time. This resulted in two different states during the experiment.
5	Drift	629 s	30.5 Hz	The pump was started in the beginning and remained without any state changes during the experiment. However, signals start to drift in the middle of the experiment.
6	Broken	626 s	30.5 Hz	The pump was started in the beginning and remained without any input changes during the experiment. However, in the middle of the experiment a change happens (only) in the output, showing a broken system status.

changed again back to the first state, and classified the health status of the system correctly.

5.2.2. Wear-out

Figure 11(a) shows the input voltage of the pump is set to a constant value throughout the entire experiment. We note that the sensors SharkyS and Dyna deliver correct flow values only after about 4 seconds. After about 350 seconds the system starts to drift (mimicked by an opened valve that is not observed by CCAM) and both sensors detect a slowly rising water flow.

The CCAM system first detects correctly the stable state and then it detects the drifting behaviour of the system. The health status changes from OK to Drifting until the end of the experiment. The confidence in the state is rising again after the discovery of the deviation.

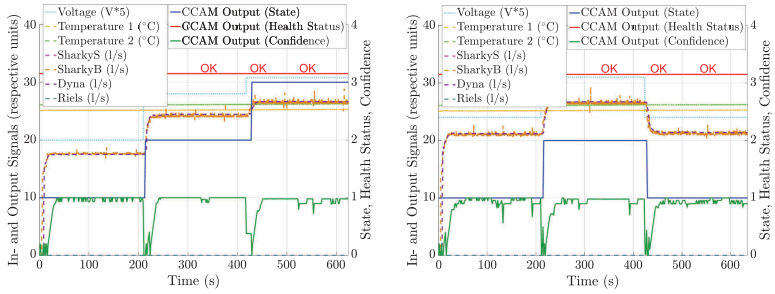
5.2.3. Anomaly

CCAM is also able to detect the break down of the system, and the results are shown in Figure 12. In this case, the pump voltage is set to a constant value for the entire experiment. At the beginning of the experiment, the entire water flow is in the pipe which is monitored from the sensor SharkyB but then the valve for this pipe is closed, and the valve of the pipe of sensor SharkyS is opened to simulate a hole in the other pipe.

In this case, the system detects the state, the confidence and the health status correctly at the beginning. After about 310 seconds it changes the health status to broken, which is the expected behaviour. The confidence drops significantly at the beginning of the broken state, and it needs about 30 seconds to return to a high confidence value.

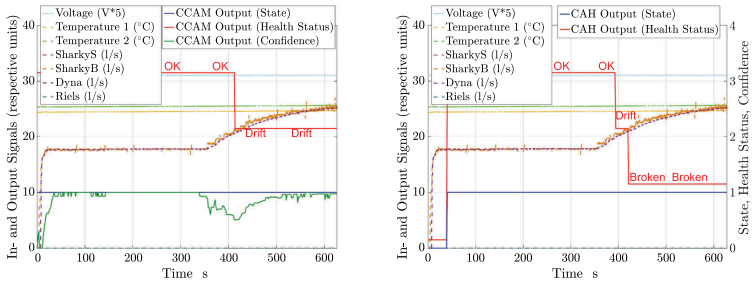
5.3. Sensitivity analysis

As mentioned, the proposed system benefits from an enhanced robustness of the recognition of states and a decreased dependence on parameter settings which reduces deployment time. To demonstrate these effects, we performed a sensitivity analysis where the impact of changing the value of certain parameters on the correct classification of the monitored system states is investigated. We studied the sensitivity of assessment results of CCAM on a set of data from the motor and water pipe system use cases. Tested parameters are the intervals of the fuzzy functions defined by Equation (4) and Equation (7), denoted by the points d_a, d_b, d_c, d_d, and s_a. In addition, various downsampling rates (DSR), DAB sizes and s_a from Equation (20) as well as d_a and d_c from Equation (22) are tested in the course of our sensitivity analysis. These parameters and their intervals are listed in Table 3. The high



(a) CCAM output during state changes. (b) CCAM output during state changes.

Figure 10. CCAM output during state changes of the water pipe system. The legend shows the inputs of the SuO with dotted lines and the outputs with dashed lines.



(a) Proposed CCAM system outputs during a drift. (b) CAH system outputs during a drift.

Figure 11. Monitoring of drifting phenomena by CCAM and CAH. The legend shows the inputs of the SuO with dotted lines and the outputs with dashed lines.

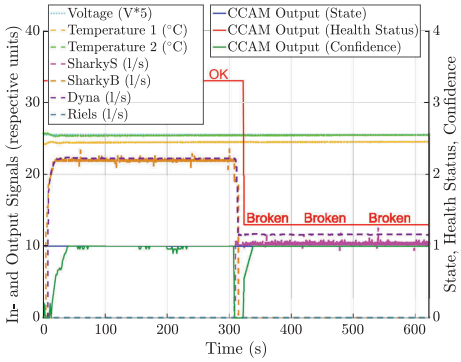


Figure 12. Proposed CCAM system outputs when observing the water pipe system showing anomalies. The legend shows the inputs of the SuO with dotted lines and the outputs with dashed lines.

number of tested values led to 2.2 million experiments for each recording which took about 20 days to run on two computers running in parallel on 32 and 24 cores.

Many configurations[7] led to results similar to those reported in the experiments above. More precisely, the classification of the well-being of SuO was correct, but the timing of the recognition and raising the alarms may differ to some extent. However, the focus was on the correct recognition rather than the time it takes before the states and their changes are detected.

Table 3. CCAM configuration parameter ranges used in the sensitivity analysis.

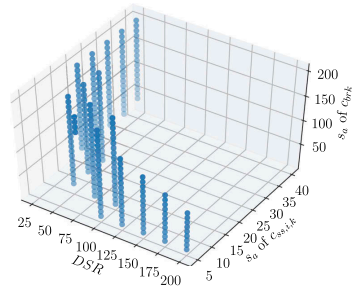
Parameter	Test Range	Test Steps	Working Range AC Motor	Working Range Water Pipe System
d_a of $c_{sv,i,j}$	[−20%, −2%]	1%	[−17%, −5%]	[−17%, −7%]
d_b of $c_{sv,i,j}$	[−10%, −1%]	1%	[−9%, −1%]	[−8%, −1%]
d_c of $c_{sv,i,j}$	[1%, 10%]	1%	[1%, 9%]	[1%, 8%]
d_d of $c_{sv,i,j}$	[2%, 20%]	1%	[5%, 17%]	[7%, 17%]
s_a of $c_{ss,j,k}$	[5, 40]	5	[5, 10]	[5, 40]
d_a of c_{dft}	[−50%, −20%]	10%	[−50%, −20%]	[−50%, −20%]
d_b of c_{dft}	10% const.	–	10%	10%
d_c of c_{dft}	10% const.	–	10%	10%
d_d of c_{dft}	[20%, 50%]	10%	[20%, 50%]	[20%, 50%]
s_a of c_{brk}	[10, 200]	10	[10, 200]	[10, 200]
DSR	[25, 200]	25	[50, 200]	[25, 200]
DAB_{size}	[5, 15]	5	[5, 15]	[5, 15]

It has to be noted that some of the parameters affect each other. Thus, not every theoretically possible combination of all these parameter values work. However, we found at least 1987 configurations which worked for all scenarios of both case studies. This includes our original configuration which was set – heuristically and with minimum effort – prior to this sensitivity analysis. In the following, we present and discuss the behavioural dependencies of various parameters based on the sensitivity analysis made for the water pipe system case study; that is, all scenarios of Table (2).

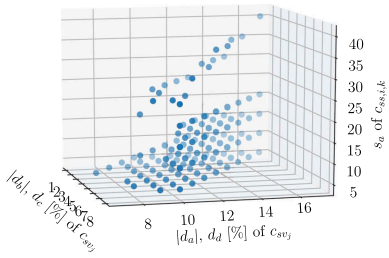
Figure 13 shows a small selection of the results of our sensitivity analysis. In these figures, each axis shows the range of the swept parameter and each dot shows the combination of those parameters which led to a correct classification (i.e. combination which did not function properly are not shown on the figure). In a closer look, in Figure 13(a) we observe that a lower Down-Sampling Rate (DSR) only works with higher s_a of $c_{ss,i,k}$, and a higher DSR needs the s_a of $c_{ss,i,k}$ to take a lower value. It can be also seen that s_a of $c_{ss,i,k}$ is rather independent of the DSR setup, whereas a higher DSR requires a lower s_a of c_{brk}.

Figure 13(b) shows that a lower s_a of $c_{ss,i,k}$ leads to a functional system with a larger set of values for the other two parameters, namely $|d_b|$, d_c of c_sv_j and $|d_a|$, d_d of c_{sv_j}. The relation between the latter two parameters is similar too; A lower $|d_b|$ and d_c of c_sv_j leads to a larger range of possible values for $|d_a|$ and d_d of c_{sv_j}. This is true the other way around too; lower $|d_a|$ and d_d of c_{sv_j} leads to a functional system with a larger set of configuration values for $|d_b|$ and d_c of c_sv_j.

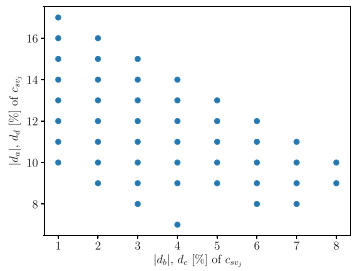
The most interesting finding, which clearly shows the advantage and importance of using fuzzy logic, can be seen in Figure 13(c). The lower $|d_b|$ and d_c of c_{sv_j}, the broader the range of acceptable $|d_a|$ and d_d of c_{sv_j}. Whereas, a higher value of $|d_b|$ and d_c of c_{sv_j} leads to the necessity of having a lower $|d_a|$ and d_d of c_{sv_j}. A low $|d_b|$ and d_c of c_{sv_j} in this context, constitutes a flat fuzzy membership function (see Figure 1(a)), whereas the opposite constitutes a steep fuzzy function. A steep fuzzy function



(a) Possible combinations of down sampling rate, s_a of $c_{ss,i,k}$, and s_a of c_{brk} , with those CCAM classifies the condition of SuO correctly.



(b) Possible combinations of $|d_a|$, $|d_b|$, d_c, and d_d of $c_{sv\,j}$ s_a of $c_{ss,i,k}$, and s_a of c_{brk} , with those CCAM classifies the condition of SuO correctly.



(c) Possible combinations of $|d_a|$, $|d_b|$, d_c, and d_d of $c_{sv\,j}$, with those CCAM classifies the condition of SuO correctly.

Figure 13. Sensitivity analysis for the different parameter combinations of CCAM. Each point on the charts represents a parameter combination which led to a correct classification of the SuO condition.

behaves very similar to a traditional threshold cut-off approach which we used in (Götzinger et al. 2017b). Therefore, the steeper is the fuzzy function (that is, more similar to a threshold function), the less flexible is the system (it functions with a smaller number of configuration values).

Based on the above analysis, we conclude that the system is in general very reliable, robust and not very sensitive to small changes of configurations with respect to those parameters studied above. In particular, the proposed method is more robust compared to CAH (Götzinger et al. 2017b) which does not use fuzzy logic or confidence.

5.4. Comparison with the context-aware health monitoring system

In this section, we compare our newly proposed CCAM system with CAH (Götzinger et al. 2017b) (which does not use confidence), under comparable conditions and using the same data.

Both, Figure 5(a) and 5(b) and 4 show the same scenarios (the same motor data) in which the external load changes. However, the curves of the motor signals appear different in the two figures. That is because Figure 5(a) shows unfiltered motor signals and Figure 5(b) shows filtered signals. This fact points to the first significant difference in the performance of the two systems. Whereas CAH needs the motor signals to be filtered in a preprocessing step, CCAM is less affected by signal instabilities and can handle unfiltered signals.[8] However, due to the unfiltered signal, CCAM requires more time than CAH to recognize the first state. While CAH recognizes the first valid state after 1*s* (Figure 5(b), CCAM needs around 1.8*s* for this. However, when CCAM analyses signals that are filtered, the first valid state is recognized after 1*s*.

The second difference is that CAH has a binary decision-making process based on fixed thresholds, whereas CCAM bases all decisions on confidences calculated with fuzzy functions. Therefore, the configuration of CCAM fuzzy parameters is less sensitive than the thresholds of the CAH system. In other words, the fuzzy functions increase the robustness. This partial independence of accurate adjustment also leads to a better recognition when the SuO drifts. We simulated 12 different wear-out scenarios with the deterioration rate of the speed signal from 0.0025 to 0.022 RPM. CAH detected the signal drift in 4 of these 12 cases (33% success) and switched after some time from drift- to broken alarm (as shown in Figure 7(b) too). In contrast, the CCAM system shows that the SuO is drifting in 100% of those cases and maintains its decision all

along (see Figure 7(a) as an example). The same unsteadiness of decision-making can be also seen when CAH monitors the water pipe system. Figure 11(b) shows that CAH detects a drift in the signals of SuO after $\approx$ 400 s. However, after again $\approx$ 25 s, CAH changes its decision and states that SuO is broken. In contrast to that, CCAM maintains its decision until the end.

The third major difference in performance is that whereas CCAM has been successfully tested for various down-sampling rates between 25 and 200, the CAH raises wrong alarms in some scenarios when the down-sampling factor is less than 50.

In summary, the results show that confidence, based on fuzzy functions, (i) simplifies the system, (ii) improves the quality of the system performance, and (iii) enhances its resilience.

6. Conclusions and future works

We present a Confidence-based Context-Aware condition Monitoring (CCAM) system which monitors the health condition of a bijective function black box system, without making further assumptions. In particular, CCAM is model-free and thus uses only contextual information to identify whether the observed system works correctly, is broken, or shows symptoms of wear-out. The advantages of such a system are the ease of deployment in a variety of applications, even those which are considerably different. The proposed system requires neither in-depth knowledge of the field, nor cumbersome effort to adjust the parameters to the given application. We are particularly pleased that CCAM – as we showed here – can monitor and assess proper working conditions of such diverse systems as a motor and water pipe system without model building or other customization.

The decision-making process of the proposed system uses confidence (which is a self-awareness property) computed using fuzzy logic. In our experiments, we ran a series of tests with and without confidence on an industrial AC motor as well as a water pipe system to show that the proposed system detects all of these behaviours correctly and more reliably while being simpler than a comparable monitor system without confidence. The proposed monitoring system is reliable as it does not depend on manually provided fixed threshold values; instead, it uses fuzzy set member functions that are robust against small variations of parameter settings.

In summary, in this paper,

(1) we introduced a fuzzy logic confidence metric for the condition monitoring of the state of a system,
(2) we demonstrated the feasibility of the proposed monitor using two case studies: an industrial motor and a water pipe system,
(3) we showed that the proposed system (CCAM) gives equally good or better results than the system without confidence (CAH) even though it does not pre-process the signals
(4) we demonstrated the robustness of our CCAM system by providing a sensitivity analysis showing that the system is robust against small variations of parameter settings.

6.1. Future work

In our experiments, we show that our proposed system is robust against small variations of parameter settings, and that it correctly detects/classifies state changes as well as the health status of the SuO. However, currently, CCAM uses a single fuzzy function for all signals of the SuO. This may lead to a wrong classification if the amplitude of the changes is significantly different among various input signals, output signals, or between input and output signals. For example, when a very small change in the input of the SuO causes a much larger change in the output. Therefore, in the future, we plan to have one fuzzy function setup for each signal of the SuO. However, that would lead to a larger number of parameters to tune. To circumvent a higher effort in setting up the parameters of CCAM, it is necessary that CCAM is able to learn from the signals themselves autonomously setup the fuzzy functions. This procedure will lead to an even easier and more effortless usage of CCAM.

In addition to that, we will study if an unstable assessment which we could see in the anomaly scenario of the AC motor could automatically be analyzed further and recognized explicitly and reliably as a single broken state. Detecting unsteady (transitory) states could be another important addition. Furthermore, CCAM should be able to detect patterns in case of a repeating sequence of state changes. A sudden change in a repeating sequence of SuO states could denote a malfunction too.

Notes

1. Distance means a difference which can be of any dimension such as performance or success rate difference, geometrical distance, time difference, difference in absolute values, and so forth.
2. We note that the relative distance, $d_{i,j}$, is not to be confused with Δ'. While $d_{i,j}$ is the distance of a sample to another sample, Δ' denotes the confidence that a membership assessment is correct.
3. There can be more than four points and three intervals. The fuzzy functions could be much more complex. However, the currently chosen shape has led to satisfactory results in our tests.
4. In fuzzy logic, the complement (negation) of a conjunction is not equal to a disjunction of complemented operands. Hence, in a generic case, $c_b \neq c_n$, although in some specific instances they may be equal.
5. We note that if input or output of the SuO is considered independently from each other, m constitutes the number of input and output variables, respectively.
6. 'Pt' stands for platinum and '100' shows the resistance at 0°C in ohms.
7. The smallest number of configurations which worked for all experiments of both case studies was 1987 sets.
8. We note that one scenario constitutes the exception: the anomaly (bearing defect). However, we believe that it would also work unfiltered if we had motor data which would show the motor working well in the beginning and before breaking. Unfortunately, we only have records of the broken motor, and therefore, the vibration signals are changing throughout the recording period. Thus, it is impossible for the SH to recognize a valid state, and the SH only raises an alarm if a valid state was found.

Acknowledgments

We gratefully acknowledge the financial support provided to us by the BMVIT/FFG under the program *Production of the Future* in the project SAVE (FFG 864883).

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Bundesministerium für Verkehr, Innovation und Technologie [FFG 864883]; Österreichische Forschungsförderungsgesellschaft [FFG 864883].

References

Alexopoulos, K., S. Makris, V. Xanthakis, K. Sipsas, and G. Chryssolouris. 2016. "A Concept for Context-Aware Computing in Manufacturing: The White Goods Case." *International Journal of Computer Integrated Manufacturing* 29 (8): 839–849. doi:10.1080/0951192X.2015.1130257.

Anzanpour, A., I. Azimi, M. Götzinger, A. M. Rahmani, N. TaheriNejad, P. Liljeberg, A. Jantsch, and N. Dutt. 2017. "Self-Awareness in Remote Health Monitoring Systems Using Wearable Electronics." *Proceedings of Design and Test Europe Conference (DATE)*, Lausanne, March.

Ballal, M. S., Z. J. Khan, H. M. Suryawanshi, and R. L. Sonolikar. 2007. "Adaptive Neural Fuzzy Inference System for the Detection of Inter-Turn Insulation and Bearing Wear Faults in Induction Motor." *IEEE Transactions on Industrial Electronics* 54 (1): 250–258. doi:10.1109/TIE.2006.888789.

Bazan, G. H., P. R. Scalassara, W. Endo, A. Goedtel, W. Fontes Godoy, and R. H. C. Palácios. 2017. "Stator Fault Analysis of Three-Phase Induction Motors Using Information Measures and Artificial Neural Networks." *Electric Power Systems Research* 143: 347–356. doi:10.1016/j.epsr.2016.09.031.

Bellini, A., F. Filippetti, G. Franceschini, C. Tassoni, and G. B. Kliman. 2001. "Quantitative Evaluation of Induction Motor Broken Bars by Means of Electrical Signature Analysis." *IEEE Transactions on Industry Applications* 37 (5): 1248–1255. doi:10.1109/28.952499.

Bessous, N., S. E. Zouzou, W. Bentrah, S. Sbaa, and M. Sahraoui. 2018. "Diagnosis Of Bearing Defects in Induction Motors Using Discrete Wavelet Transform." *International Journal Of System Assurance Engineering and Management* 9 (2): 335–343. doi:10.1007/s13198-016-0459-6.

Féki, N., G. Clerc, and P. Velex. 2013. "Gear and Motor Fault Modeling and Detection Based on Motor Current Analysis." *Electric Power Systems Research* 95: 28–37. doi:10.1016/j.epsr.2012.08.002.

Gao, Z., C. Cecati, and S. X. Ding. 2015. "A Survey of Fault Diagnosis and Fault-Tolerant Techniques Part I: Fault Diagnosis with Model-Based and Signal-Based Approaches." *IEEE Transactions on Industrial Electronics* 62 (6): 3757–3767. doi:10.1109/TIE.2015.2417501.

Gao, Z., G. Habetler, and R. G. Harley. 2005. "An Online Adaptive Stator Winding Temperature Estimator Based on a Hybrid Thermal Model for Induction Machines." *IEEE International Conference on Electric Machines and Drives*, 754–761, San Antonio, TX, USA. doi:10.1109/IEMDC.2005.195807

Glatzl, T., H. Steiner, F. Kohl, T. Sauter, and F. Keplinger. 2016. "Development of an Air Flow Sensor for Heating, Ventilating, and Air Conditioning Systems Based on Printed Circuit Board Technology." *Sensors and Actuators A: Physical* 237: 1–8. doi:10.1016/j.sna.2015.11.016.

Götzinger, M., A. Azanpour, I. Azimi, N. Taherinejad, and A. M. Rahmani. 2017a. "Enhancing the Self-Aware Early Warning Score System through Fuzzified Data Reliability Assessment." In *International Conference on Wireless Mobile Communication and Healthcare*. Vienna, Austria: Springer. 10.1007/978-3-319-98551-0_1

Götzinger, M., A. M. Nima Taherinejad, P. L. Rahmani, A. Jantsch, and H. Tenhunen. 2016. "Enhancing the Early Warning Score System Using Data Confidence." *Proceedings of the 6th International Conference on Wireless Mobile Communication and Healthcare (Mobihealth), Milano*, Italy, November.

Götzinger, M., N. TaheriNejad, H. A. Kholerdi, and A. Jantsch. 2017b. "On the Design of Context-Aware Health Monitoring without a Priori Knowledge; an AC- Motor Case-Study." In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 1–5. Windsor, ON, Canada: IEEE. doi:10.1109/CCECE.2017.7946814

Hatzipantelis, E., and J. Penman. 1993. "The Use of Hidden Markov Models for Condition Monitoring Electrical Machines." In *Electrical Machines and Drives, 1993. Sixth International Conference On, September*, 91–96. Oxford, UK: IET.

Hyvrinen, J., and S. Kärki eds. 1996. "International Energy Agency Building Optimisation and Fault Diagnosis Source Book." *IEA ECBCS Annex 25l* .

Kande, M., A. J. Isaksson, R. Thottappillil, and N. Taylor. 2017. "Rotating Electrical Machine Condition Monitoring AutomationA Review." *Machines* 5 (4). doi:10.3390/machines5040024.

Katipamula, S., and M. Brambley. 2005. "Methods for Fault Detection, Diagnostics, and Prognostics for Building Systems - A Review, Part I." *HVAC and R Research* 11 (1): 1–24.

Kephart, J. O., and D. M. Chess. 2003. "The Vision of Autonomic Computing." *Computer* 36 (1): 41–50. doi:10.1109/MC.2003.1160055.

Kholerdi, H. A., N. TaheriNejad, and A. Jantsch. 2018. "Enhancement of Classification of Small Data Sets Using Self-Awareness; an Iris Flower Case-Study." In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5, Florence, Italy. doi:10.1109/ISCAS.2018.8350992

Liu, J., T. M. McKenna, A. Gribok, B. A. Beidleman, W. J. Tharion, and J. Reifman. 2008. "A Fuzzy Logic Algorithm to Assign Confidence Levels to Heart and Respiratory Rate Time Series." *Physiological Measurement* 29 (1): 81. doi:10.1088/0967-3334/29/1/006.

Massieh, N. 2010. "Fault Detection and Diagnosis in Building HVAC Systems." PhD diss., University of California, Berkeley United States of America.

Mehala, N. 2010. "Condition Monitoring and Fault Diagnosis of Induction Motor Using Motor Current Signature Analysis." PhD diss., National Institute of Technology Kurukshetra, India.

Mounce, S. R., C. Pedraza, T. Jackson, P. Linford, and J. B. Boxall. 2015. "Cloud Based Machine Learning Approaches for Leakage Assessment and Management in Smart Water Networks." *Procedia Engineering* 119: 43–52. Computing and Control for the Water Industry (CCWI2015) Sharing the best practice in water management. http://www.sciencedirect.com/science/article/pii/S1877705815025217

Ross, T. J. 2009. *Fuzzy Logic With Engineering Applications*. Hoboken, New Jersey / United States: Wiley / John Wiley & Sons, Inc.

Sadeghioon, A. M., N. Metje, D. Chapman, and C. Anthony. 2018. "Water Pipeline Failure Detection Using Distributed Relative Pressure and Temperature Measurements and Anomaly Detection Algorithms." *Urban Water Journal* 15 (4): 287–295. doi:10.1080/1573062X2018.1424213.

Shun, L., and J. Wen. 2014. "Application of Pattern Matching Method for Detecting Faults in Air Handling Unit System." *Automation in Construction* 43: 49–58. doi:10.1016/j.autcon.2014.03.002.

TaheriNejad, N., A. Jantsch, and D. Pollreisz. 2016. "Comprehensive Observation and Its Role in Self-Awareness - an Emotion Recognition System Example." In *Proceedings of the Federated Conference on Computer Science and Information Systems, Gdansk,* 117-124. September: Poland. doi:10.15439/2016F588

TaheriNejad, N., M. A. Shami, and S. P. D. Manoj. 2017. "Self-Aware Sensing and Attention- Based Data Collection in Multi-Processor System-on-Chips." In *2017 15th IEEE International New Circuits and Systems Conference (NEWCAS)*, 81–84, Strasbourg, France: IEEE.

Tennenhouse, D. 2000. "Proactive Computing." *Communications of the ACM* 43 (5): 43–50. doi:10.1145/332833.332837.

Maximilian Götzinger, Nima TaheriNejad, Hedyeh A. Kholerdi, and Axel Jantsch

On the Design of Context-Aware Health Monitoring without A Priori Knowledge: an AC-Motor Case-Study

In 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE). IEEE, Windsor, ON, Canada 2017, pages 1—5

IV

2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)

On the Design of Context-Aware Health Monitoring Without a Priori Knowledge; an AC-Motor Case-Study

Maximilian Götzinger*†, Nima TaheriNejad*, Hedyeh A. Kholerdi* and Axel Jantsch*
*Institute of Computer Technology, TU Wien, Vienna, Austria
E-mail: {nima.taherinejad, hedyeh.kholerdi, axel.jantsch}@tuwien.ac.at
†Department of Future Technologies, University of Turku, Turku, Finland
E-mail: maxgot@utu.fi

***Abstract*—Health monitoring without a priori knowledge can save a significant amount of design and implementation time. However, for smaller devices with limited available resources, this is not feasible using most conventional methods. For small footprint sensor and actuator devices, we propose a health monitoring architecture and algorithm, which uses context-awareness to assess the health status of an "Injective-function Black-Box" without having a priori knowledge about it. The proposed algorithm can identify normal modes of operation, change of states (operation modes), deviation from a state, and abnormal functional operation. We have tested the algorithm on an AC Motor where the system was able to identify its health and changes in the operation status accordingly.**

I. Introduction

In the context of Internet of Things (IoT) and system of systems, the number of small devices and sensors is exponentially growing [1], [2]. The natural diversity of these gadgets imposes an ever increasing engineering time and effort on the design process. To reduce these efforts and extra costs associated with it, more generic methods, which can be applied to a range of devices, are desired. Deep learning, data mining, and similar methods address this issue; however, they can be applied only to larger scale systems with massive resources. We tackle this issue under tight resource constraints, which is suitable for implementation on smaller gadgets with limited computation power.

Traditional methods of control theory are often used to steer motors for a desired action. For example, moving conveyor belts and robotic arms under normal conditions, that is when all parts operate within their respective specified operational specifications [3]. When parts in the system become faulty due to a wear-out or other effects, the system has to detect and diagnose this fault and change its operation accordingly. To save engineering efforts, it is desirable that this process is fully automated, in a reliable fashion, and without requiring extensive computational resources.

Therefore, here we propose a health monitoring system with a small footprint and without a priori knowledge about the design or specifications of the system under monitoring. The proposed system is suitable for monitoring all devices which

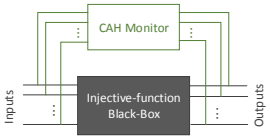


Fig. 1. Block Diagram of the proposed Context-Aware Health monitoring (CAH) system.

are considered an "injective function". That is, for the function f, we have $\forall\ a, b \in D, f(a) = f(b) \iff a = b$, where D is the domain on which f is defined. Our system uses contextual information, to find out normal modes of operation and perturbations therefrom. Since our Context-Aware Health monitoring (CAH) does not use a priori knowledge about the functionality and design of the device it monitors, it can be used for any black-box which constitutes an 'injective function'; the induction motor used as a case-study in this work included. Hence, to validate our method, we have tested it on the data for an AC induction motor, where normal modes of operation, change of state, deviation from normal mode (drift), and anomalies are detected.

The rest of this paper is organized as it follows; In Section II, we briefly review the requirements and specification of the use case, which justifies the needs and benefits of using CAH for this application. The architecture of the proposed system is presented in Section III. The set-up and result of our simulations are found in Section IV, and Section V concludes the paper.

II. Use-case Background: AC Motors

Induction motors are widely used in industry. The high costs of this equipment, its energy consumption, and the importance of avoiding downtimes highlight the necessity of continuous and reliable monitoring as well as regular maintenance [4]. Parameters, such as voltage, frequency, and mechanical torque, influence the various outputs of the motor such as its speed, and torque.

978-1-5090-5538-8/17/$31.00 ©2017 IEEE

From a high-level point of view, a motor can operate normally, deviate from such a normal state, or fail. A normal operation is when the motor is rotating at a constant speed or changes its speed due to the process plan. However, the condition of the motor and its behavior is prone to decay, and the performance may deteriorate over time because of various causes. In some cases, this deterioration may be reflected in a small deviation from a normal operation mode. In this case, one or more signals are drifting (normally very slowly) away from the normal state. Finally, the motor may get unacceptably far from expected performance or break down, which is called a failure. Some of the causes for motor failures are presented in [5].

When the motor is not connected to a speed controller (in free-running operation), the synchronous speed is proportional to the frequency of power supply and the number of poles of the motor [6]. If the motor is not deteriorated, with the nominal value of power supply the nominal speed is expected. However, sometimes the motor should change its speed. Therefore, various techniques have been introduced to force the motor to rotate at desired values. A constant voltage-frequency ratio is considered as one of the simplest methods, which changes the frequency and voltage to adjust the speed [6]. Nonetheless, when the motor wears out (due to any causes including contamination, lack of lubricant, and corrosion), its speed deviates from the nominal behavior [7].

Health monitoring and fault diagnosis in induction motors have already been studied before, most of which aim at detecting faults in the machine. Nejjari et al. [8] proposed a neural network monitoring methodology to diagnose the electrical faults of induction motors. This system can distinguish between faulty and healthy states of the motor while running at a constant speed. Blodt et al. [9] present an on-line condition monitoring system which detects the mechanical faults of induction motor drives in various load conditions, using current analysis. A variety of health monitoring techniques such as thermal monitoring, vibration and noise monitoring, as well as current analysis, have been reviewed in [10]. The used methodologies can be categorized as methods based on models [11], thresholds [12], pattern recognition and neural networks [13], [14], as well as fuzzy logic [15]. However, to the best of our knowledge, no report on utilizing multiple signals to monitor the health and operation of a motor without a priori knowledge about the motor has been published so far. Such a technique makes the task of monitoring independent from the motor specification.

III. System Architecture

Figure 2 shows the proposed CAH monitoring system consisting of three blocks that are responsible for different tasks: pre-processing, controlling stability, and handling different states. Before describing each part in this section, we present the scope of the proposed system as it follows.

A. Scope

Since the proposed system has no information on the black box it monitors (i.e., the motor), the relations of the various signals are also unknown. However, we do assume that the black-box under study is an "injective function". Therefore, any unique set of input data should correspond to a unique set of output data, and vice-versa. Thus, the working mode can be considered as normal only when a change of the output dataset is also reflected in a change of the input dataset and vice-versa. In other words, the black-box (in this case the motor) does not work well (is broken), when the output changes without being stimulated by an input, or if an input change does not lead to a changed output.

The second assumption is that the system is in a steady state. Therefore, unstable signals and states (in particular transient signals during state changes) need to be disregarded. Hence, data pre-processing steps are needed for some signals to convert the data to a format suitable for CAH.

B. Pre-Processing

The pre-processing block (shown in the red frame of Figure 2) covers both abstraction and low-pass filtering of a signal. For example, the abstraction receives the sinusoidal signal of voltage and provides its amplitude and frequency. Filtering removes some of the noise and unwanted signal values (e.g., oscillations during a transition). Pre-processing is the only case-dependent part of the system; although, the requirements on it are still generic.

C. Stability Controller

Even though filtered signals are better than the original ones, they may not be stable enough. Therefore, the Stability Controller block (shown in the green frame of Figure 2) is needed to decide whether a signal is stable or not. For this purpose, a sample history in the form of a sliding window (the size of which can be configured) saves the latest values. The Stability Controller compares an actual sensor value with the history and decides that a signal is stable if the disparity (in percentage) of the actual value to a defined number of values of the history is below a certain threshold. In other words, an actual value has to be sufficiently close to a defined number of the values saved in the history. A dataset is only stable when all the signals constituting the set are stable.

D. State Handler (SH)

The SH (shown in the blue frame of Figure 2) does the bulk of the work. This unit tries to recognize all states of normal operation, so that it can, subsequently, detect deviations therefrom.

1) Algorithm: One of the tasks of the SH is to verify whether the actual state is valid or not. For this purpose, the values inserted into this state are counted. A state is considered as valid only if enough values are already stored in it. While the SH saves valid states in the state vector, it discards invalid ones. This procedure ensures that extremely noisy data or

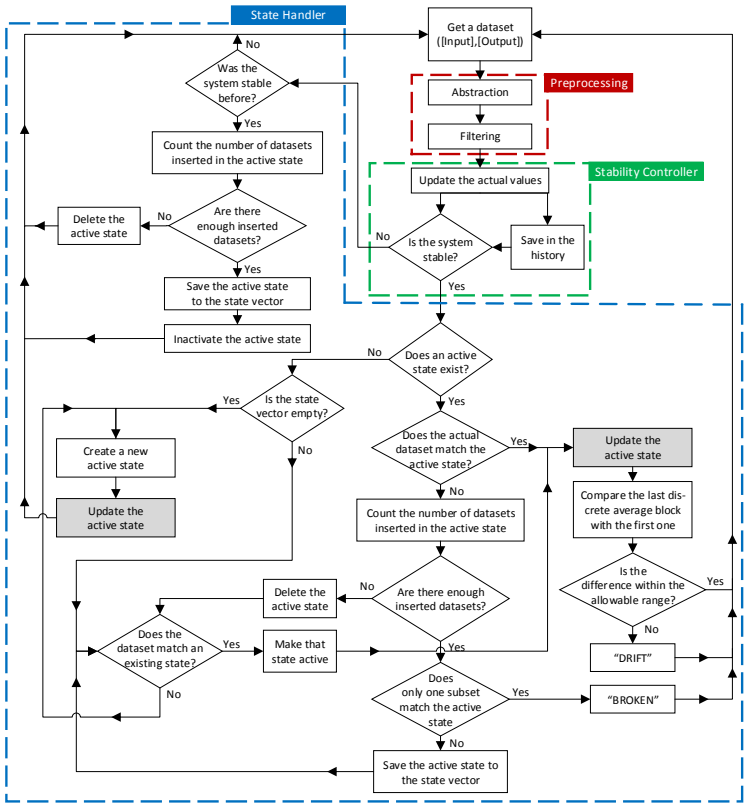


Fig. 2. Flow chart of the Context-Aware Health Monitoring system.

transition phases do not lead to creating a new state, which does not reflect the actual operation of the system.

The SH compares new values, marked by the Stability Controller as stable, with the actual state which is called active state. If deviations of both input and output datasets (measured in percentage), are not bigger than a certain threshold, the SH considers the active state still as active and updates it with the new values. Next, the discrete average of that state is updated and compared with its initial discrete average. If the two discrete averages have a difference larger than a defined acceptable threshold, a drift is observed. Discrete averages and its respective processes are described in more detail in the next subsection.

If the input or output datasets do not match the actual state, a change of state has happened, which can be normal or abnormal. Since the monitored system is treated as an injective function, the change of only one dataset (input or output exclusively) is due to an anomaly. Whereas, a change of both datasets indicates a normal state change. In the latter case, the question is whether the system changes to an already known state or a new state has to be created by the SH. Therefore, the SH goes through the entire state vector and compares all saved states with the new datasets. The SH sets an old state active if the new datasets match an old state. Otherwise, the SH creates a new active state and activates it.

2) Creating and Comparing Discrete Average Blocks (DABs): The CAH system is not meant to raise an alarm only when the system is broken, but it is meant to announce deviations from normal operations as well; that is, when some signals are drifting. In this context, drifting means a signal that changes continuously but very slowly (i.e., a change that is not reflected in continuous averaging). In other words, a series of values of a signal belong to the same state, but the signal is gradually deviating outside its normal expected range. The SH

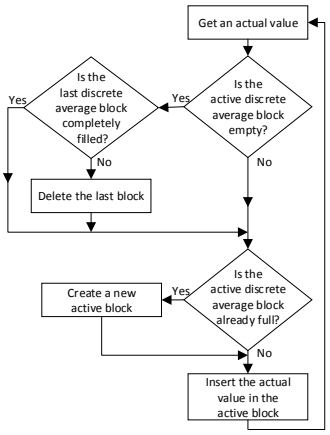


Fig. 3. Block Diagram of the state updating task of the proposed CAH system. Discrete averages are also created and kept in this procedure.

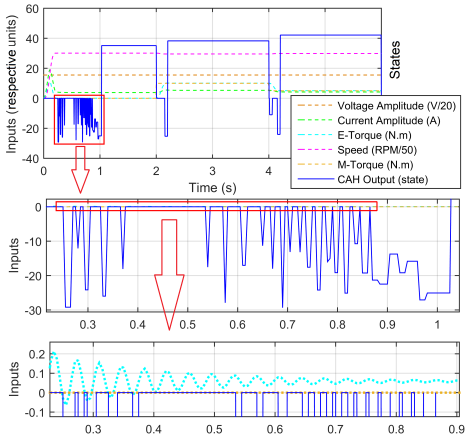


Fig. 4. Outputs of CAH system for an AC motor operating in normal mode when a load change is happening.

detects this behavior through periodically creating DABs of the signal values. To this end, the task of updating the active state (shaded in gray in Figure 2) consists of more operations than only inserting the actual datasets into a state. Figure 3 shows each step of this procedure. To avoid having semi-filled DABs, which are not reliable, the SH deletes the previous DAB, if it is incomplete and belongs to a previously active episode of that state. Afterward, the SH checks whether the active DAB is already full, in which case, the SH initiates a new active block and inserts the actual value into it. If the active DAB is not full[1], no new block is needed, and the SH inserts the actual datasets into the DAB.

A new value of a signal might be in the vicinity of the continuous average (CA) because the CA slowly changes following the drift of the signal. Thus, the comparison of the new values with the CA values does not indicate symptoms of deviation, since it is within the acceptable range of variation. In contrast to the CA, which slowly shifts due to slow changes, the difference between two DABs increases as a slow drift happens in a signal.

IV. Simulation and Results

To validate the proposed CAH system, we modeled it in C++ and simulated it on a set of data from an AC motor, operating normally, changing state, having a drift and failing.

A. Data

The data of the motor has been collected based on both simulations and real measurements. The data of normal and abnormal functional operations are based on measuring voltage, current, vibration, frequency, and torque (mechanical and electrical) signals from the sensors of a three-phase induction motor [16]. The normal state and change of states have been simulated as changes in load and operating speed. For the drift, the continuous increase of load was modeled using a gradual change in mechanical torque, to show the wear-out phenomenon. In the simulation, three-phase current and voltage, speed and load torque have been acquired and later used as inputs for the proposed health monitoring system. The motor is a squirrel-cage, three-phase, 380V, 50Hz, induction motor with 3KW power consumption and four poles [16]. The steady state model used for the motor is the model of asynchronous machine in MATLAB®.

[1]Given that no state change has happened.

B. Pre-processing

The voltage and current values had to be abstracted to extract information about their amplitudes. All output variables showed occasional unsteadiness, and therefore, they were filtered using a low-pass filter[2]. Other signals could be used without any modification. Last but not least, to avoid unnecessary extra processing, all datasets were down-sampled by a factor of 50, after which each two samples are $5ms$ apart. Here on, all the references to the numbers of samples are after down-sampling.

C. Simulations

1) Normal Operation with and without Changes: Figure 4 shows our test scenario for recognition of the normal operation of the system and respective state changes. In this scenario, the motor is started first, and then, runs monotonically; which

[2]We used an Equiripple filter, namely *fdesign.lowpass* function of MATLAB®, two times in a row, with following parameters: $Fp = 0.005, Fst = 0.1, Ap = 0.15, Ast = 0.999$. We note that a low-pass filter implementation is outside the scope of this work for which there are several light-weight methods of implementation on hardware or software.

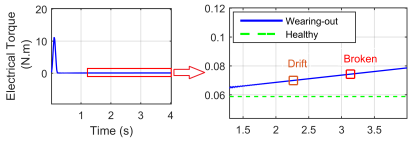


Fig. 5. CAH system outputs for a motor undergoing wear-out.

means, that no parameter changes. In the beginning, the output signals oscillate considerably. Around $0.2s$ (or the 42nd sample), for the first time, the stability-controller verifies the signals as stable. However, this changes within the next samples, and consequently the active state is discarded (shown as a negative state), and a new active state is created. This situation continues until the 196th sample ($\sim 1s$) where a new active state is created which remains stable and is updated by new samples. In this instance, the initial state collects 205 samples, until the load changes at around the 400th sample ($\sim 2s$). Since there are no big changes in any of the signals, the SH recognizes that the monitored system remains in the same state and tags the monitored system as healthy.

When the load changes, the system becomes shortly unstable. After a period of oscillating signals and instability, around the 431st sample ($\sim 2.2s$), the system settles into a new state. Once the system is stable again and recognized as such, the SH is updated again with new values. In this case, until the second load change at the 800th sample (or $\sim 4s$). The third and final state of this scenario is created at around 831st sample ($\sim 4.2s$) and remains unchanged until the end. We successfully ran a similar experiment to detect state changes due to speed changes.

2) State Drift (Wear-Out): The wear-out phenomenon describes a case where the system (here the motor) still works, but one or more signals are drifting away from the nominal value(s). In this example, shown in Figure 5, at the 188th sample ($\sim 0.9s$), a valid state is created, and after 265 samples, the CAH system recognizes a drifting signal and raises a flag. Since the drift continues, where the signal exceeds the boundary of being part of the existing state, the "drift" alarm is replaced by a "broken" alarm. This event occurs at the 628th sample ($\sim 3.1s$).

3) Anomaly: Caused by a bearing defect, the vibration signals, and the current change significantly. At the moment of failure, one of the output signals changed while the input signal (in this case the voltage) remained unchanged. Therefore, the SH raised a "broken" flag.

V. Conclusion

In this paper, we presented a small footprint health monitoring system which can track the health status of any "injective function black-box". Our system is able to achieve this without a priori knowledge about the specification or design details of the monitored system, by using only contextual information. For verifying the validity of the proposed approach, it was tested on an AC motor dataset, where it could successfully identify normal modes of operations, changes therein and deviations thereof (including drift and failure). This method reduces the engineering effort of designing health monitoring systems for various small gadgets, to a configuration setup of thresholds for an acceptable range of variation for the input and output values. In further steps, this process can be automated through a one-time application of optimization, self-awareness, or learning methods during setup or commissioning.

Acknowledgement

We gratefully acknowledge the financial support provided to us by the BMVIT/FFG under the program *ICT of the Future* in the project SAMBA (contract number: 855426).

References

[1] J. Rivera and R. van der Meulen. Gartner says 4.9 billion connected things will be in use in 2015. *Press release*, November 2014. [Online]. Available: http://www.gartner.com/newsroom/id/2905717.

[2] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128610001568.

[3] R.D. Lorenz, T.A. Lipo, and D.W. Novotny. Motion control with induction motors. *Proceedings of the IEEE*, 82(8):1215–1240, 1994.

[4] W.T. Thomson and R.J. Gilmore. Motor current signature analysis to detect faults in induction motor drives–fundamentals, data interpretation, and industrial case histories. *In Proceedings of 32nd Turbo Machinery Symposium, A&M University, Texas, USA*, pages 145–156, 2003.

[5] A. H. Bonnett. Root cause AC motor failure analysis with a focus on shaft failures. *IEEE Tran. Ind. Appl.*, 36(5):1435–1448, 2000.

[6] J.M.D. Murphy and F.G. Turnbull. *Power Electronic Control of AC Motors*. Franklin Book Company, 1988.

[7] M.S. Ballal, Z.J. Khan, H.M. Suryawanshi, and R.L. Sonolikar. Adaptive neural fuzzy inference system for the detection of inter-turn insulation and bearing wear faults in induction motor. *IEEE Transactions on Industrial Electronics*, 54(1):250–258, 2007.

[8] H. Nejjari and M.E.H. Benbouzid. Monitoring and diagnosis of induction motors electrical faults using a current park's vector pattern learning approach. *IEEE Transactions on Industry Applications*, 36(3):730–735, 2000.

[9] M. Blodt, D. Bonacci, J.érémi Regnier, Marie Chabert, and Jean Faucher. On-line monitoring of mechanical faults in variable-speed induction motor drives using the wigner distribution. *IEEE Transactions on Industrial Electronics*, 55(2):522–533, 2008.

[10] K.M. Siddiqui, K. Sahay, and V.K. Giri. Health monitoring and fault diagnosis in induction motor-a review. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 3(1):6549–6565, 2014.

[11] E. Hatzipantelis and J. Penman. The use of hidden markov models for condition monitoring electrical machines. In *Electrical Machines and Drives, 1993. Sixth International Conference on*, pages 91–96. IET, September 1993.

[12] N. Mehala. *Condition monitoring and fault diagnosis of induction motor using motor current signature analysis*. PhD thesis, National Institute of Technology Kurukshetra, India, 2010.

[13] G.H. Bazan, P.R. Scalassara, W. Endo, A. Goedtel, W. Fontes Godoy, and Rodrigo Henrique Cunha Palácios. Stator fault analysis of three-phase induction motors using information measures and artificial neural networks. *Electric Power Systems Research*, 143:347–356, 2017.

[14] R.H.C. Palácios, I.N. da Silva, A. Goedtel, and W.F. Godoy. A novel multi-agent approach to identify faults in line connected three-phase induction motors. *Applied Soft Computing*, 45:1–10, 2016.

[15] M. Dong, T. Cheang, and S. Chan. On-line fast motor fault diagnostics based on fuzzy neural networks. *Tsinghua Science & Technology*, 14(2):225–233, 2009.

[16] N. Bessous, S.E. Zouzou, W. Bentrah, S. Sbaa, and M. Sahraoui. Diagnosis of bearing defects in induction motors using discrete wavelet transform. *International Journal of System Assurance Engineering and Management*, pages 1–9, 2015.

Maximilian Götzinger, Edwin Willegger, Nima TaheriNejad, Axel Jantsch, Thilo Sauter, Thomas Glatzl, and Pasi Lilieberg

Applicability of Context-Aware Health Monitoring to Hydraulic Circuits

In *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, Washington, DC, USA
2018, pages 4712–4719

V

Applicability of Context-Aware Health Monitoring to Hydraulic Circuits

M. Götzinger*, E. Willegger†, N. TaheriNejad†, A. Jantsch†, T. Sauter†‡, T. Glatzl‡, and P. Liljeberg*
* University of Turku, Turku, Finland
E-mail: {maxgot, pasi.liljeberg}@utu.fi
† TU Wien, Vienna, Austria
E-mail: {edwin.willegger, nima.taherinejad, axel.jantsch}@tuwien.ac.at
‡ Danube University Krems, Wiener Neustadt, Austria
E-mail: {thilo.sauter, thomas.glatzl}@donau-uni.ac.at

***Abstract*—Monitoring is an important aspect of operation and maintenance in virtually every industrial system. However, the extent and methods of monitoring vastly vary in different systems, from fully automated to fully manual. One of the challenges of automated monitoring is the tediousness of, and the extent of engineering time and effort required to develop necessary models or machine learning algorithms for the units to be monitored. Model-free monitoring, on the other hand, can save resources and efforts substantially. However, more often than not they have a very limited scope and application. Such a system is needed, for example, to monitor entire Heating, Ventilation and Air Conditioning (HVAC) systems, consisting of different types of sensors such as temperature, pressure, humidity or flow sensors. Recently, we proposed the Context-Aware Health Monitoring (CAH) system for model-free monitoring of any injective-function black-box, and it was tested successfully on an AC motor. In this paper, we evaluate the CAH system for an entirely different industrial use-case, that is, a hydraulic circuit. The results show the potential for considerable benefits in monitoring HVAC systems. Moreover, in the light of applying CAH to different use-cases which may potentially need a different setup of parameters, we performed a sensitivity analysis on the values of different parameters in the system. The results show the robustness of CAH with regard to the values of these parameters.**

***Index Terms*—Monitoring, Hydraulic Circuits, Context-Awareness, Model-Free, Injective Function**

I. Introduction

The worldwide energy consumption is still increasing and, according to the Organization for Economic Cooperation and Development (OECD), industrial and residential sectors are two main consumers of electric energy [1]. It is possible to save energy significantly by improving the Heating, Ventilation and Air Conditioning (HVAC) systems [2]. In Europe, the energy efficiency directive of the European Union (EU) is one action of several efforts to improve the consumption of energy in Europe [3]. One possible measure put forward there is to improve the efficiency of HVAC systems as they are among the major energy consumers. For instance, in Austria HVAC systems were responsible for 27% to 30% of the total energy use within the years 2006 and 2016 [4]. It is possible to reduce 5 to 15 percent of the energy consumption of HVAC systems by fixing faults or with optimizing the building control system [5]. Based on that, we estimate a potential saving of 15-45 petajoules per year for Austria alone. However, that requires continuous monitoring and better maintenance of these systems, which in turn imposes certain challenges.

Manual maintenance promotes long durability, but it is costly. The goal is to perform maintenance operations as seldom as possible but as often as necessary. However, automated monitoring of a device increases the engineering effort because the rules and patterns for normal behavior and anomalies have to be derived and built into the system as trigger conditions for maintenance actions. In order to reduce design time effort and cost, generic methods are desirable that can be applied to monitor a wide variety of devices. Various methods exist for black-box monitoring; however, these methods require significant resources and computational power and are thus only applicable to large-scale systems.

Therefore, a lightweight black-box health monitoring system for inexpensive devices with limited resources would facilitate the automated monitoring of many cyber-physical and IoT devices. In an earlier work [6], we proposed the Context-Aware Health Monitoring (CAH) system, a novel method, for monitoring a black-box without prior knowledge about it. In other words, the monitoring system has no model of the monitored system which is called Device under Monitoring (DuM) in the further course of this work. CAH utilizes context-awareness for monitoring the DuM. That is, looking at the contexts under which sensory data and their changes

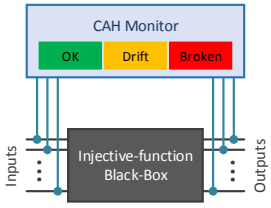


Fig. 1. Block diagram of the Context-Aware Health Monitoring (CAH) system [6].

978-1-5090-6684-1/18/$31.00 ©2018 IEEE

are observed, it discovers system states, state changes, normal behavior, and abnormalities of the DuM.

Two assumptions are made: (1) the DuM is in steady state, meaning that CAH ignores transient states; (2) the implying that one unique input dataset corresponds to exactly one output dataset and vice versa. Consequently, if only the input or the output data is changing, the DuM shows per definition an abnormal behavior. Furthermore, CAH does not only detect anomalies that signify a broken device but also slight deviations such as a drift. Drifts can happen during a wear-out, and identifying them enables maintenance of the DuM before the device condition deteriorates further. This model-free approach based on context-awareness has the advantage that no retraining, recalibration or other design modifications are required when the system structure changes or sensors are replaced.

While we demonstrated the applicability of CAH to AC motors earlier [6]; in this paper, we extend its scope to the monitoring of a hydraulic system, as it is common in HVAC facilities. We show that CAH detects normal operations, state changes, drifts as well as abnormal functions. As the characteristics of these two applications are clearly distinct, the presented cases study significantly extends the scope of CAH, importantly with no modifications of the underlying algorithms. Furthermore, we conducted a sensitivity analysis of the parameters regarding their set values. These analyses show that CAH benefits from a high degree of robustness against the exact value of these parameters and their changes.

The rest of this paper is organized as follows: In Section II, we review the literature on the HVAC systems, analyze their requirements and specification. These facts explain why the usage of CAH is beneficial for this application. We briefly present the architecture of CAH in Section III, and show and discuss the experiment results in Section IV. Finally, Section V concludes the paper.

II. Background and Related Work

During the last few years, research on Automated Fault Detection and Diagnostic (AFDD) methods for building systems have been increasing steadily [7]. AFDD methods are subdivided into three different approaches based on qualitative models, quantitative models, and models based on process history [8]. 62% of the implemented AFDD methods are process history based models [7]. This kind of models are further subdivided into black-box and gray box approaches, and within the black-box approaches, there are three different methods possible: statistical, Artificial Neural Networks (ANNs) based, and other pattern recognition techniques [8]. In the following, we describe only the advantages and disadvantages of ANN and Fuzzy logic, and two approaches are discussed in more details as they are most relevant for this work.

Among the advantages of ANN [9] is that they can model non-linear systems without detailed knowledge of the DuM. Disadvantages are that vast amounts of training data is required to model complex systems, it is challenging to gain any physical insight from the ANN, and if some relevant input is not part of the training data, the ANN may produce erroneous output. The advantages of Fuzzy logic are [9] that it can model non-linear behavior, the commission of Fuzzy Fault Detection and Diagnostic (FDD) schemes is easier, expert knowledge and knowledge learned from measured data are easy to combine, and the software implementation is not computationally intensive. Disadvantages are that the results are less precise compared to other approaches, and the used rule-based descriptions are fairly long. Several researchers have used ANN-based approaches [10]–[13], and pattern recognition techniques [14]–[17] as black-box methods. Fuzzy logic has also been used by FDD [18]. In the following, given their relevance, the approaches by Fan et al. [12] and Dexter et al. [18] are discussed in more details.

Fan et al. [12] combine Back-propagation Neural Network (BPNN) models with wavelet analysis and Elman neural networks. Their approach consists of two parts. In the first part, BPNN fault detection models are generated based on historical data under normal operating conditions. One model is based on variable correlation in the control loop and the other of sensitivity analysis. In the second part, the fault diagnosis model is created by a combination of wavelet analysis and an Elman neural network. The task of the fault diagnosis model is to determine the reason for the fault in the control loop. The diagnosis flow consists of the following five steps:

1) Both BPNN models are created.
2) New input data is analyzed by the BPNN fault detection model. If a fault is detected, step 3 is executed otherwise the FDD finishes.
3) Approximation coefficients from the historical data (including faulty and normal data) are extracted with the wavelet analysis. These coefficients are used for the Elman neural network to diagnose the sensor fault.
4) New data is entered, and if the BPNN fault detection model detects a new fault, the approximation coefficients are extracted and clustered.
5) If the input data is already known as fault data, then the Elman neural network tries to identify the fault type. If it is unknown, return to step 3 and add this data to the historical data to train the Elman neural network with it.

With his hybrid FDD strategy Fan et al. were able to detect new unknown faults in HVAC systems.

Dexter et al. [18] use a multi-step, Fuzzy model-based approach. They divide the process into two phases, the fault elimination and fault classification phase. The fault elimination phase is shown in Figure 2 and the other phase in Figure 3. Within the fault elimination phase confidence, similarity, and ambiguity of the similarities, and the strength of evidence of the possible state are calculated. Then the current evidence is combined with the evidence from previous operating points, and the values for belief in fault-free operation as well as faulty behavior are recalculated. The result of this step is:

- unambiguous belief in fault-free operation, or
- unambiguous belief in the presence of a fault, or
- it is not clear if a fault is present.

Only if two or more faults are present, the fault classification phase will be executed. The particular fault is isolated with the re-evaluated test data, but only the faults with unambiguous belief from the fault elimination phase are used. The result of this phase could be the unambiguous belief in one fault or a list of all possible faults. This approach was able to recognize a fault-free operation and detect faults in HVAC systems, such as sensor bias, a leaky valve, a fouled coil, and when the valve stuck at fully closed, midway or fully open position.Drawbacks of this solutions were that it was not able to detect drift, and Fuzzy reference models were created from the design specifications. Our proposed solution overcomes these drawbacks.

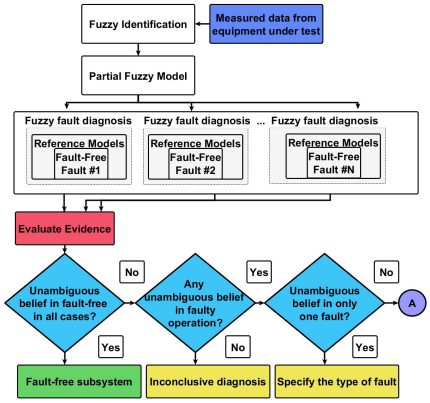


Fig. 2. The fault elimination phase. Adapted [reprinted] from [18].

III. Context-Aware Health Monitor

The CAH system consists of three function blocks (Fig. 4). Since CAH assumes that the DuM is in steady state, only stable signals are of interest. Periods in that signals are unstable (e.g., the transition during a state change), are not considered and have to be ignored. The first two functional blocks accomplish this task.

A. Pre-Processing

The Pre-processing block (shown in the red frame of Fig. 4) contains two different tasks which are case-specific: abstraction and filtering. Because datasets used in this study, namely water flow measurement data, do not need any abstraction, this unit is not used here. The second task of the pre-processing block is to filter all signals to reduce noise as well as oscillations during and shortly after a state change. For filtering, we used a MATLAB® Equiripple filter[1]

[1]The MATLAB® function *fdesign.lowpass* was used with following parameters: $F_p = 0.005$, $F_{st} = 0.1$, $A_p = 0.15$ and $A_{st} = 0.999$. We note that a low-pass filter implementation is outside the scope of this work for which there are several light-weight methods of implementation on hardware and software.

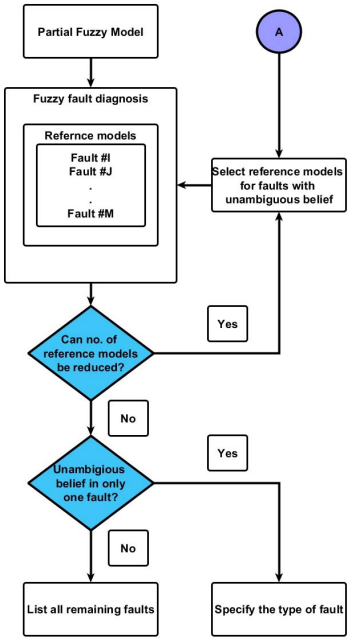


Fig. 3. The fault classification phase. Adapted [reprinted] from [18].

B. Stability Controller

Since CAH only considers steady states of the DuM, a Stability Controller (SC) is needed (shown in the green frame of Fig. 4). Whether the signals are filtered or not, this functional block is essential to discard the samples during a transition. Filtered signals may be more stable, but filtering does not replace the SC. The DuM is stable if all signals are stable. Therefore, SC provides a sliding history window for each signal. The SC saves the last samples of a signal in the history assigned to this signal, and a signal is stable if the value of the new sample is in the proximity to a certain number of the saved signal samples. In this regard, the new sample (s_{new}) is in proximity to a saved sample (s_s) when the relative distance between both samples is lower than a certain threshold (called *Threshold to be Different to Samples in the History*). In this regard, the relative distance d is given by

$$d = \frac{|s_{new} - s_s|}{|s_{new}|}. \quad (1)$$

We note that the size of the sliding window of this history, the number of samples which have to be similar (which are not different), and the threshold are adjustable. An adjustment may be necessary if the data collection sampling rate is changed.

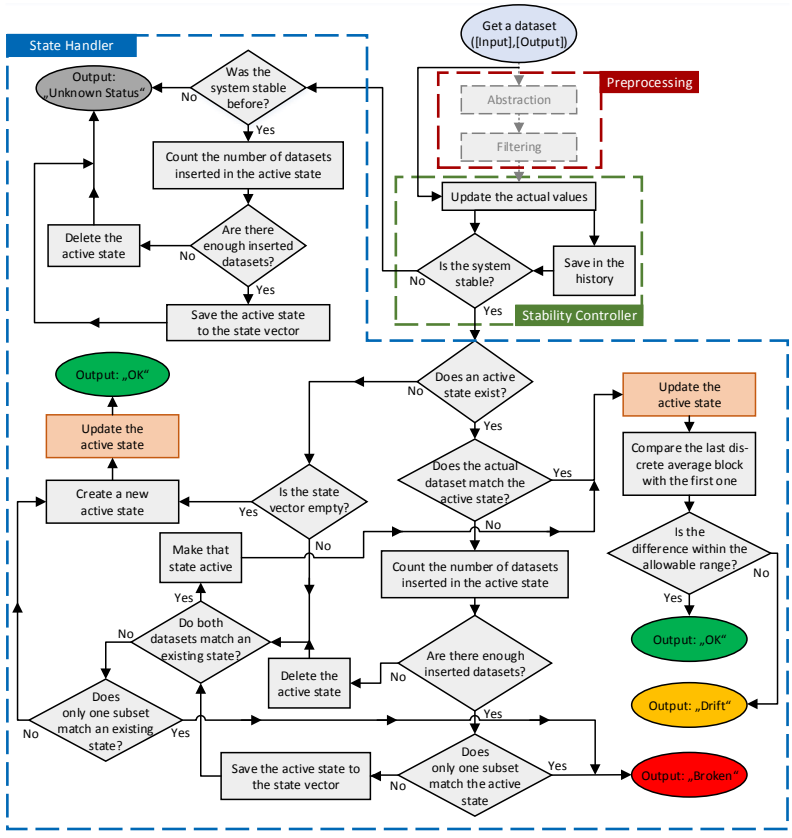


Fig. 4. Flowchart of the Context-Aware Health Monitoring system.

C. State Observation

The State Handler (SH) is the centerpiece of the CAH. It detects states, state changes, malfunctions, and drifting signals of the DuM.

1) Discovering a State: When the SH detects a state of the DuM (a DuM state), it records the data in a C++ object. For the sake of convenience, these objects are called states, and all of these states are saved in the state vector of the SH. The information in such a state is saved as an average value, in particular, an average value for each signal of the DuM. Saving this information in an average value constitutes a very abstracted form of a history. When a new sample that belongs to the same state appears, it gets inserted in the state and influences the average value. A new sample belongs to a state if its value is in proximity to the average value. In this regard, the new sample (s_{new}) is in proximity to the average value (v_{avg}) when the relative distance between them is lower than a certain threshold (called *Different-to-State Threshold*). In this regard, the relative distance d is given by

$$d = \frac{|v_{avg} - s_{new}|}{|v_{avg}|}. \tag{2}$$

In further consequence, a new sample set belongs to a state if all samples are similar (not different) to their respective average values.

2) Discovering a State Change: In the SH, the equivalent of the actual state of the DuM is called active state. If a new sample set matches the active state, the sample values are inserted into it, and it remains active. When the sample set, meaning, in particular, one or more samples of this set, are different to the active state, a state change is indicated. Such a change can be either normal or can constitute a malfunction of

the DuM. If both, input and output subsets are different from the active state, a normal DuM state chance is happening. In this case, the SH compares the actual sample set with each state saved in the state vector. If one of them matches, that state is activated, and the SH inserts the new sample set into it. If no saved state matches the actual dataset, a new active state is created, and the new sample set is inserted into it.

A state is saved to the state vector if it is valid. A high number of inserted samples characterizes a valid state. If only a few samples were inserted in the active state before a state change happens, the SH discards the active state. This procedure is necessary because in some cases, a particular combination of the sampling rate and the signal curves can make the SC consider the DuM as stable even though it is not.

In contrast to a regular DuM state change, a malfunction is identified when only one subset (input or output sample set) is different from the active state. In this case, the DuM is classified as broken, but only if the active state is valid. Otherwise, the change of only one subset has been observed most likely because the DuM was not steady.

3) Discovering a Drift: A drift of a signal is another misbehavior of the DuM. Because a drift is a very slow change in a signal, it also changes the signal's average value bit by bit. Thus, a drift cannot be discovered by comparing a new sample value with the average value. Therefore, the task of updating the active state with a new sample set (shaded in peach in Fig. 4) is a bit more complicated than just calculate a new average. The SH also saves so-called Discrete Average Blocks (DABs). A DAB is an average value of a certain number of samples. In other words, it is not a sliding window calculation over all samples inserted into a state but only over a certain number of samples. After this specific number of samples is inserted in a DAB, the average is calculated, and a new DAB is created. A drift can be detected when comparing the average values of two different DABs. If a new DAB (DAB_{new})is different to the very first DAB (DAB_1) calculated for a state, it is clear that the signal has shifted over time, even if the signal has drifted only slightly. A signal is considered as drifting if the difference between these two DABs is higher than a certain threshold (called *Drift Threshold*). In this regard, the relative distance d is given by

$$d = \frac{|DAB_1 - DAB_{new}|}{|DAB_1|}. \quad (3)$$

IV. Evaluation

A. Experimental Setup

The entire setup is illustrated in Fig. 5. It consists of a copper pipe system with four ultrasonic flow sensors (two Sharkey FS, one Dynasonics TFX Ultra and one RIELS RIF600P), two temperature sensors (Pt100), two water pumps (of which only one was used for the actual experiments) and an electric heater. A Raspberry Pi in combination with an Arduino Uno controls the entire system while logging the data of each

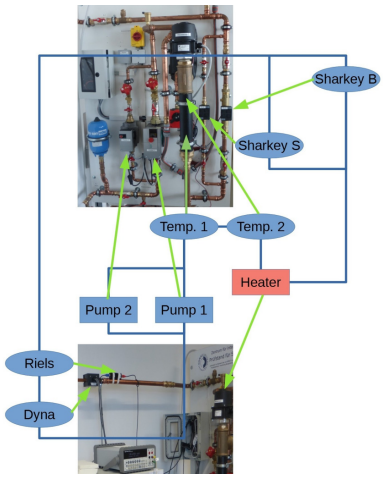


Fig. 5. Experimental setup.

sensor and actuator, realized via a Python software program. The water pump operates in a range between 3.0 to 10.0V, which corresponds to the normalized values of 0.3 to 1.0 in the gathered data. The input signal is the voltage applied to the Pump 1, and the output values are temperature and flow. The sampling frequency was 30.5Hz in average, and the gathered data are stored in CSV-files for post-processing purpose.

Table I lists the various scenarios applied to the HVAC system to validate the correctness of CAH. In the "normal" state scenarios only the voltage of the water pump was changed. The scenarios "anomaly" and "drift" were simulated through manual changes of the degree of the opening of some of the valves. For anomaly, Scenario 5, the valve before sensor SharkyB was opened completely, and the valve before SharkyS was closed at the beginning. Afterward, the SharkyB valve was closed, and the SharkyS valve was abruptly opened. This action simulated a burst of the pipe containing SharkyB. Drift, Scenario 6, was simulated with SharkyB valve half-opened initially and then it was slightly opened further every 15 seconds. This procedure reduces the flow in SharkyS which simulates the increase in sediment and gradual clogging of SharkyS.

B. Results

The output of the CAH system shows that it recognizes normal and abnormal behavior when monitoring a hydraulic system. CAH reads in the data with a down-sampling factor of 50 and classifies all of the scenarios correctly. During the experiments, it turned out that the CAH can also classify the hydraulic system correctly when its input and output signals are unfiltered. Skipping this task saves computational power and shows that CAH is robust enough to distinguish between

TABLE I
DIFFERENT TESTED SCENARIOS AND EVENTS DURING EACH OF THEM.

#	Condition of the HVAC system	Events in the scenario
1	OK	No state change during the experiment.
2	OK	One state change, resulting in two different states during the experiment.
3	OK	Two state changes, resulting in three different states during the experiment.
4	OK	Two state changes, whereas the second change leads again to state one.
5	Broken	No change in the input, but a change in the output in the middle of the experiment.
6	Drift	No state change, but a drifting signal staring in the middle of the experiment.

noise and real signal changes correctly. Some results are discussed in more detail in the following.

1) Normal Mode: All other normal operation scenarios (Table I) were tested, which led to a correct identification of the normal operation states and their changes.

As an example, Fig. 6 shows Scenario 4. Here, the pump starts operating at the beginning of the experiment, and all signals have settled approximately 40s later. At this time, the SC decides that all signals are stable, showed by the green graph raising. Further, the SH finds the first state (blue graph) and decides that the DuM is working correctly (red graph). This circumstance lasts until 210s, when a changed input voltage leads to a state change. After a ~35s long unstable period, the SC recognizes that all signals have settled again, and the SH discovers the second state. Since input and output datasets have both changed, the DuM is still considered as working normally. At ~425s the DuM changes back to its previous (initial) state. This change is again accompanied by a ~35s long unstable period. After that, CAH concludes, that the DuM is again in state 1 and working correctly. This condition continues until the end of the experiment.

2) Anomaly: Fig. 7 shows experimental results for Scenario 5, where significant changes in the output signals are observed. However, as the input voltage stays unchanged, the DuM is per definition broken. The records start with the start-up of the pump, which leads to a settled DuM at ~35s when the SC notifies the SH that it is now in a steady state, at which point the latter recognizes the first state. At 310s, the DuM suddenly changes its outputs, which is not triggered by an input change. After this change has started, it takes around 40s until the SC verifies the DuM as stable. In this 40s, the SH is just waiting to be triggered, and after that time, it recognizes that the DuM is still in the same state but shows symptoms of malfunctioning. Therefore, considering the context of these changes, the DuM is classified as broken.

3) Drift: Fig. 8 shows the results for a drifting system (Scenario 6). Once more, the pump is starting up at the beginning. After ~40s the SH recognizes the first state and the CAH concludes that the DuM works correctly. However, this changes at around 400s, when the SH recognizes a drift. After ~45s, the output signal value has drifted far enough from

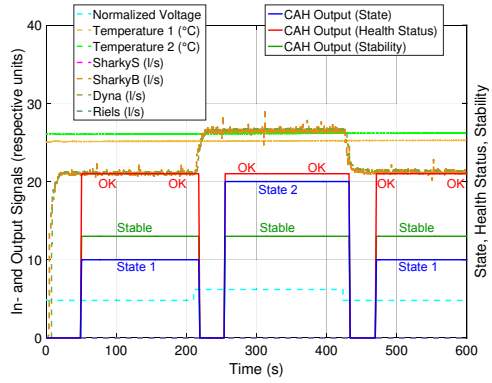


Fig. 6. Input and output of the DuM during two state changes, as well as the output of the CAH system.

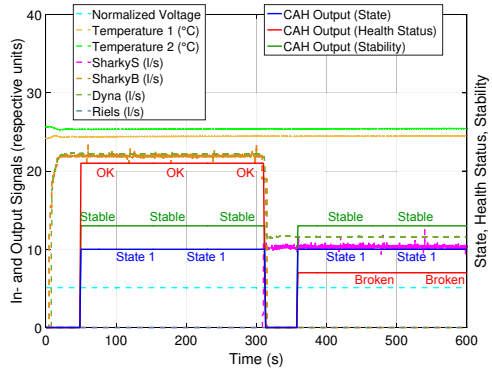


Fig. 7. Input and output of the DuM during an abnormal change leading the CAH system to classify it as broken.

the original state to signify a state change of the output signal. Since the input remains unchanged, the SH classifies the DuM as broken and replaces the drift alarm with a broken one. In the case of a slower drift, the drift alarm would last longer.

C. Sensitivity Analysis

The robustness of CAH was investigated with a sensitivity analysis in which the CAH system processed all datasets with several parameter combinations. Table II lists the ranges of the values of the parameters analyzed. After running the algorithm with these values, all CAH outputs were analyzed to see whether the health condition of the DuM is accurately classified and its states are correctly detected.

Fig. 9 shows for which set of parameters CAH classifies all scenarios correctly. In other words, those combinations of

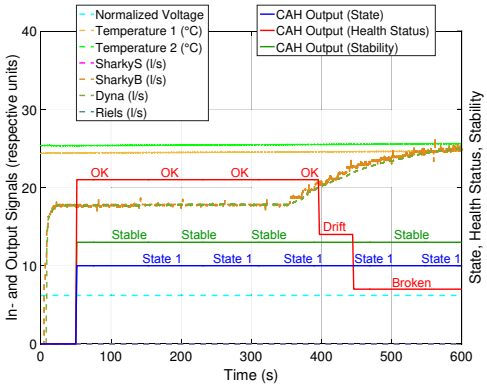


Fig. 8. The outputs of the CAH system while the DuM is drifting.

TABLE II
TEST RANGES OF THE PARAMETERS OF CAH.

Parameter	Functional Block	Difference Calculation	Test Range	Test Steps
Sliding History Window Size	Stability Controller	NA	3-10	1
Number of Similar Samples in the History	Stability Controller	NA	0-10	1
Threshold to be Different to Samples in the History	Stability Controller	Eq. 1	1%-8%	1%
Number of Samples inserted into a State to be Valid	State Handler	NA	10	NA
Different-To-State Threshold	State Handler	Eq. 2	1%-40%	1%
Drift Threshold	State Handler	Eq. 3	1%-40%	1%
Downsampling Rate		NA	50	NA

parameter values which led to an incorrect classification are not drawn in the figure.

A few points should be noted:

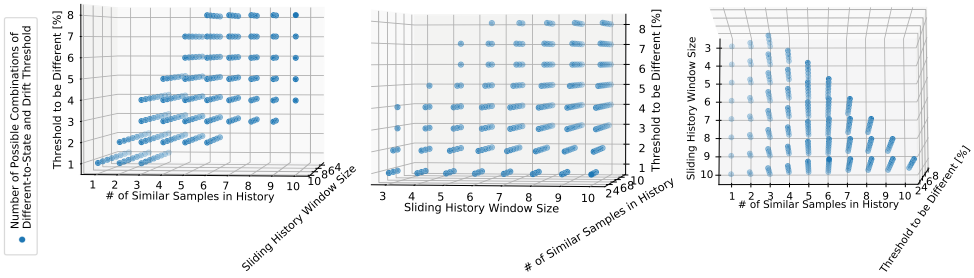
1) The window size has to be higher than the number of similar samples required for classification, Number of Similar Samples in History $\leq$ Sliding History Window Size.
2) The history window should contain at least 3 samples, Sliding History Window Size ≥ 3.
3) It is advisable to set Number of Similar Samples in History ≥ 3, as the classification is then more robust for a broader range of other parameter values.
4) The two parameters *Number of Similar Samples in History* and *Threshold to be Different to Samples in the History* should grow in tandem; if one is set higher, the other is to be set higher as well. E.g., if Number of Similar Samples in the History ≥ 7, then the *Threshold to be Different to Samples in the History* ≥ 3. This rule comes most likely from the fact that with a high threshold, the SC can classify the DuM in some cases as stable although it is not yet. A higher number of Similar Samples in the History counteracts this phenomenon because more time elapses until this number is reached.
5) For the tested datasets, we can infer that the *Drift Threshold* and the *Different-to-State Threshold* have to be between 11% and 19%.

The fact that there are no gaps between the points shows that CAH is quite robust when the parameters are in these ranges.

Fig. 10, focuses on a point in the middle of the scatter diagram of Fig. 9 (specifically, *Sliding Window History Size* of 5, *# of Similar Samples in History* of 5, and *Threshold to be Different* of 4%) and analyzes with which threshold parameters the SH classifies the states of the DuM correctly. Each point marks a correct working classification for a set of two given threshold parameters. The points missing from the figure (mostly in the upper left half) represent the combinations which did not lead to a correct classification. Therefore, for the given datasets, we can infer from Fig. 10 that in this instance, choosing larger "Different-to-State Threshold" and lower "Drift Threshold" values leads to a higher probability of correct classification.

V. CONCLUSION

CAH is a model-free, context-aware monitoring system that identifies states, normal behavior, and various anomalies, based on the context under which the Device under Monitoring (DuM) operates. Its main assumptions are (1) that the DuM resembles an injective function where input and output signals can only change together, and (2) that the DuM, as it operates, evolves from one stable state to the next. CAH has been introduced earlier with an AC industrial motor as guiding application case [6]. In this paper, we have extended the scope of CAH to the HVAC applications with good results. Considering that these two applications are clearly distinct concerning reaction times, the periodicity and regularity of signals, it is significant that the CAH setup had to be adapted only slightly. While the CAH algorithm was unchanged, the only necessary adaption concerned some thresholds due to different amplitudes of signals. Moreover, AC motors react to input changes an order of magnitude faster than hydraulic circuits. Because of these differences, at the discretion of experts conducting the measurements, the sampling rate of the measurements was also different whereas the sampling rate inside the algorithm remained the same. Apart from this adaption, CAH has shown remarkable robustness in identifying correct behavior, anomalies and drift when monitoring so diverse applications as AC motors and hydraulic circuits. We support this claim with a sensitivity analysis, which shows the wide range of values which the parameters can be set to, and have the CAH system classify every scenario correctly nevertheless. Thus, we expect that CAH is applicable to an even larger class of applications. That possibly includes medical and environmental monitoring devices, among others, which will be subject to future studies.



(a) View from the side. (b) View from the front. (c) View from the top.

Fig. 9. The 3D scatter plot shown from three different angels represents the sensitivity analysis for the three different parameters of the SC. Each point on the chart represents a combination which led to a correct classification of the DuM state.

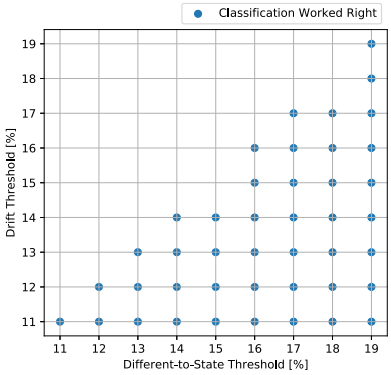


Fig. 10. Possible combinations of SH parameters with which the SH classifies the DuM correctly.

ACKNOWLEDGEMENT

We gratefully acknowledge the financial support provided to us by the BMVIT/FFG under the program *Production of the Future* in the project SAVE (FFG 864883).

REFERENCES

[1] "Electricity information: Overview," International Energy Agency, Note, 2017.

[2] X. Cao, X. Dai, and J. Liu, "Building energy-consumption status worldwide and the state-of-the-art technologies for zero-energy buildings during the past decade," *Energy and Buildings*, vol. 128, pp. 198–213, Sep. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378778816305783

[3] E. Commission, "Energy efficiency directive," Website, 2012, online at https://ec.europa.eu/energy/en/topics/energy-efficiency/energy-efficiency-directive; 06. June 2018.

[4] S. Austria, "Energiestatistik: Energiebilanzen österreich 1970 bis 2016," Website, 2017, online at https://www.statistik.at/web_de/statistiken/energie_umwelt_innovation_mobilitaet/energie_und_umwelt/energie/nutzenergieanalyse/index.html; 06. June 2018.

[5] N. Massieh, "Fault detection and diagnosis in building hvac systems," Ph.D. dissertation, University of California, Berkeley United States of America, 2010.

[6] M. Götzinger, N. TaheriNejad, H. A. Kholerdi, and A. Jantsch, "On the design of context-aware health monitoring without a priori knowledge; an AC-motor case-study," in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2017, pp. 1–5.

[7] K. Woohyun and K. Srinivas, "A review of fault detection and diagnostics methods for building systems," *Science and Technology for the Built Environment*, vol. 24, pp. 3–21, 2018.

[8] S. Katipamula and M. Brambley, "Methods for fault detection, diagnostics, and prognostics for building systems - a review, part i," *HVAC&R Research*, vol. 11, no. 1, pp. 1–24, 2005.

[9] A. Dexter and J. Pakanen, "Demonstrating automated fault detection and diagnosis methods in real buildings," 2001.

[10] Z. Du, B. Fan, X. Jin, and J. Chi, "Fault detection and diagnosis for buildings and hvac systems using combined neural networks and subtractive clustering analysis," *Building and Environment*, vol. 73, pp. 1–11, 2014.

[11] H. He, D. Menicucci, T. Caudell, and A. Mammoli, "Real-time fault detection for solar hot water systems using adaptive resonance theory neural networks," in *Energy Sustainability, 2011. Fifth International Conference on*. ASME, August 2011, pp. 1059–1065.

[12] B. Fan, Z. Du, X. Jin, X. Yang, and Y. Guo, "A hybrid fdd strategy for local system of ahu based on artificial neural network and wavelet analysis," *Building and Environment*, vol. 45, pp. 2698–2708, 2010.

[13] Z. Hou, Z. Lian, Y. Yao, and X. Yang, "Data mining based sensor fault diagnosis and validation for building air conditioning system," *Energy Conversion and Management*, vol. 47, pp. 2479–2490, 2006.

[14] Y. Guo, J. Wall, J. Li, and S. West, "Intelligent model based fault detection and diagnosis for hvac system using statistical machine learning mehtods," in *ASHRAE, 2013. Conference January Dallas*. ASHRAE, January 2013.

[15] A. Srivastav, A. Tewari, and B. Dong, "Baseline building energy modeling and localized uncertainty quantification using gaussian mixture models," *Energy and Buildings*, vol. 65, pp. 438–447, 2013.

[16] R. Sharifi and D. B., "Fault detection in lighting systems - first phase results," Philips Research North-America, Technical Note, 2011.

[17] M. Najafi, D. M. Auslander, P. Haves, and M. D. Sohn, "A statistical pattern analysis framework for rooftop unit diagnostics," *HVAC&R Research*, pp. 406–416, 2012.

[18] A. L. Dexter and D. Ngo, "Fault diagnosis in air-conditioning systems: A multi-step fuzzy model-based approach," *HAC&R Research*, vol. 7, pp. 83–102, 2001.

Arman Anzanpour, Iman Azimi, Maximilian Götzinger, Amir M. Rahmani, Nima TaheriNejad, Pasi Liljeberg, Axel Jantsch, and Nikil Dutt

VI

Self-Awareness in Remote Health Monitoring Systems Using Wearable Electronics

In *Design, Automation and Test in Europe (DATE)*. IEEE, Lausanne, Switzerland
2017, pages 1056–1061

Self-Awareness in Remote Health Monitoring Systems using Wearable Electronics

Arman Anzanpour[1], Iman Azimi[1], Maximilian Götzinger[1], Amir M. Rahmani[2,3], Nima TaheriNejad[2],
Pasi Liljeberg[1], Axel Jantsch[2], and Nikil Dutt[3]
[1]Department of Information Technology, University of Turku, Finland
[2]Institute of Computer Technology, TU Wien, Austria
[3]Department of Computer Science, University of California Irvine, USA
{armanz, imaazi, maxgot, pakrli}@utu.fi, {nima.taherinejad, axel.jantsch}@tuwien.ac.at, {amirr1, dutt}@uci.edu

***Abstract*—In healthcare, effective monitoring of patients plays a key role in detecting health deterioration early enough. Many signs of deterioration exist as early as 24 hours prior having a serious impact on the health of a person. As hospitalization times have to be minimized, in-home or remote early warning systems can fill the gap by allowing in-home care while having the potentially problematic conditions and their signs under surveillance and control. This work presents a remote monitoring and diagnostic system that provides a holistic perspective of patients and their health conditions. We discuss how the concept of self-awareness can be used in various parts of the system such as information collection through wearable sensors, confidence assessment of the sensory data, the knowledge base of the patient's health situation, and automation of reasoning about the health situation. Our approach to self-awareness provides (i) situation awareness to consider the impact of variations such as sleeping, walking, running, and resting, (ii) system personalization by reflecting parameters such as age, body mass index, and gender, and (iii) the attention property of self-awareness to improve the energy efficiency and dependability of the system via adjusting the priorities of the sensory data collection. We evaluate the proposed method using a full system demonstration.**

Keywords*—*Self-Awareness, Health Monitoring, Wearable Electronics, Situation-Awareness, Early Warning Score

I. Introduction

Vital signs reflect a patient's wellbeing status as well as the deterioration and amelioration of his or her condition. The monitored vital signs can also be the basis for predictions of a patient's health status. Research on cardiac arrests shows that certain symptoms can be observed long before the situation turns into a case of emergency; the advance apparition of symptoms can happen up to 24 hours before the actual health deterioration [1]. Early Warning Score (EWS) systems is a standard manual tool for predicting patients health deterioration which is periodically used by healthcare professionals to monitor patients' vital signs and interpret them to a level of criticality [2]. However, to support the recent trends in reducing hospitalization, there is growing demand for personalized and automated systems to enable in-home as well as mobile patient monitoring.

Internet of Things (IoT) and wearable technologies provide a competent and structured approach to improve the healthcare services in terms of social benefits and penetration as well as cost-efficiency [3], [4]. Due its ubiquitous computing nature, IoT-enabled wearables enable health monitoring systems such as EWS to continuously track and predict patients health status in an automated fashion [5], [6].

In [7], via a preliminary prototype, we presented how Internet of Things (IoT) and wearable technologies can be utilized to implement an automated EWS system. Our system deploys a wireless body area network (WBAN) – using a set of medical sensors attached to patient's body – to record physiological parameters and vital signs and send them to a cloud server for further processing and storage. Even though promising outcomes were observed, the system faced open issues which need to be addressed before it can be deployed in real field trials. Challenges such as situation-dependency, accuracy, and plausibility of input data, as well as constraints in sensor nodes call for more advanced optimization techniques to enhance the dependency of such systems. Several parameters affects the interpretation of vital signs outside the hospital (e.g., patient's activities, room temperature, barometric pressure) which need to be considered to reach a more realistic conclusion [8]. For instance, while a resting heart rate of 120 beats per minute would be an alarming sign for a patient, it can be completely normal while s(he) is exercising. Additionally, mobile and wearable sensors face disparate constraints such as energy efficiency, reliability, and computational power.

We believe self-awareness principles can be leveraged to reinforce the EWS system to tackle these open challenges. Self-awareness is defined as the ability of a system to be aware of its own state as well as the state of its surrounding environment to adapt to new situations [9]. The notion of self- and context-awareness can boost the EWS system to implement intelligent reasoning and decision making [10]. This can be realized by enhancing and personalizing the score calculation process to consider patient state parameters, to assess the confidence of the measured data and the corresponding decisions, and to optimize system-level characteristics by using the provided semantic information to adjust system knobs such as sampling and transmission rates and type of the required sensors in closed-loop manner.

In this paper, we propose a self-aware EWS system which provides personalization, self-organization, and autonomy for remote monitoring scenarios and offers intelligence in decision making process for patients in different situations. In addition, we leverage the properties of the self-awareness concept to improve the energy efficiency of the system and confidence of the calculated scores by adaptively adjusting the priorities in sensory data collection and processing w.r.t. environment changes and patient's emergency state. Moreover, we provide a proof of concept full EWS system implementation from development of cloud services to hardware-software demonstration of our prototype using a smart e-health gateway and a set of wearable and environmental sensors.

978-3-9815370-8-6/17/$31.00 ©2017 IEEE

TABLE I. EWS TABLE EXTRACTED FROM [11], [12]

Score	3	2	1	0	1	2	3
Heart rate[1]	<40	40–51	51–60	60–100	100–110	110–129	>129
Systolic BP[2]	<70	70–81	81–101	101–149	149–169	169–179	>179
Breath rate[3]		<9		9–14	14–20	20–29	>29
SPO_2 (%)	<85	85–90	90–95	>95			
Body temp.[4]	<28	28–32	32–35	35–38		38–39.5	>39.5

[1]beats per minute, [2]mmHg, [3]breaths per minute, [4] °C

II. EARLY WARNING SCORE

Several physiological signs can be used for early warning of serious illnesses and deterioration (e.g., airway, breathing, circulation, etc.). These signs are always recorded but they are not constantly recognized, even though a structured record can make them "visible". To this end, early warning systems are developed based on the conclusion of several studies suggesting that there is often a delay in the response to the deterioration of a patient's condition [13]. However, the actual work of closely monitoring the patient and taking the appropriate action is dependent on the professional competence and as such is error prone as it is mostly manually done [14]. In addition, interpreting the individual signs into a single comprehensive status information about the patient is a difficult task. In the late 1990's, Morgan et al. [2] developed a scoring technique, Early Warning Scoring (EWS), which includes the core physiological signs. It aggregates a weighted score of six signs, respiratory rate, oxygen saturation, heart rate, systolic blood pressure, body temperature and neurological status. Each of these signs will have a value between 0 and 3 based on the actual reading, either high or low, and different level of action is required, including the level of expertise of the caregiver team, for each value of the EWS. Table I shows a sample of a simple EWS.

There have been some efforts to modify EWS systems (i.e., MEWS [15]) and or standardize it (i.e., SEWS [16]), in several countries such as UK , Ireland , New Zealand, and Sweden. However, all these efforts have been conducted in a non-automated (i.e., manual) fashion and only implemented in clinical environments.

III. SELF-AWARENESS

Self-awareness is a concept which can provide systems with necessary tools to obtain many dynamically changing characteristics of interest, such as reliability, adjustability and optimality. Many of these characteristics are of particular interest for the estimated 26 billion devices expected to be connected to the Internet of Things (IoT) by 2020[1]. Therefore, using self-awareness in various applications have been explored, including mobile applications [17], cloud computing [18], networks [19], and health monitoring system [20]. This has motivated us to explore various benefits that can be obtained through a self-aware design of a remote health monitoring system which uses wearable devices.

One of the prominent architectures for self-awareness is the Observe-Decide-Act (ODA) loop [21], [22], [23]. For current application also an ODA loop has been selected as the backbone of the system architecture. As shown in Figure 1, internal and external data are first collected through the sensor network and pre-processed (Observe). Next, the situation awareness and self-awareness core further assess and process these observations in order to choose the best configuration for the system (Decide). This configuration can be seen as two separate parts; first the part that helps the system to evaluate the health of the subject correctly, despite the potential noises or misleading values [5], [24]. Second, the part that tries to improve the system operations. Here, as shown in Figure 1, the main parameter under control is the "Attention", which is set based on the requirements (observations and decisions) of the self-aware units (Act). Attention, which determines various parameters related to the activity of the sensors (e.g., sampling rate, sleeping times, or precision), is then translated to the sensor network understandable commands in the "Configuration" unit and is passed to the sensor network.

[1]www.gartner.com/newsroom/id/2636073

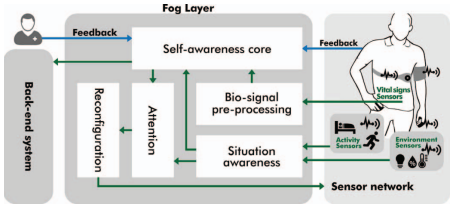


Fig. 1. Application of self-awareness concept to the remote health monitoring system

It is important to note that a crucial part of the awareness of the system is its model of itself (and the environment). A designer can try to create as comprehensive a model as possible (which comes with the disadvantages of large resource requirement), and use complementary sensory data, nevertheless, it will not provide a full image until user feedback is provided to the system. This feedback plays an important role in improving the awareness and consequently performance of the system. This feedback can be provided by the subject using the remote device or the practitioner and the support system team. Each of these completes a certain part of the image, helping the system to create a better model of itself and its environment.

IV. PROPOSED SYSTEM ARCHITECTURE

In this section, we address the challenges from both the user and system perspectives by introducing a new architecture for local computing of out-of-hospital EWS systems. The architecture incorporates the foregoing self-awareness concept in an IoT-enabled health monitoring system. As illustrated in Figure 1, the main functionalities of smart gateways [25] in the Fog tier is divided into 5 different components, all of which are included in a closed-loop system to intelligently correct EWS values as well as adjust sensor network configurations regarding the self-awareness. According to the EWS implementation, these components are specified as follows.

A. Bio-signal Pre-processing

Bio-signal pre-processing unit receives raw signals from sensor nodes (i.e., heart rate, respiration rate, oxygen saturation, body temperature and blood pressure) and converts the data to a format usable by higher level processing units. The aforementioned preparation can be divided into two parts. First, pre-processing methods such as signal filtering and normalization are implemented in this component. Second, the signals are processed in order to extract the required medical information. For example, the heart rate data is extracted by detecting RR peaks in ECG signals. Finally, the component

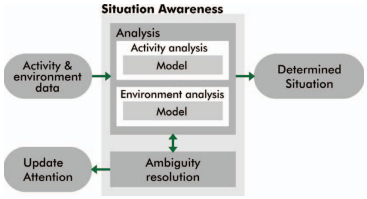


Fig. 2. Situation awareness diagram

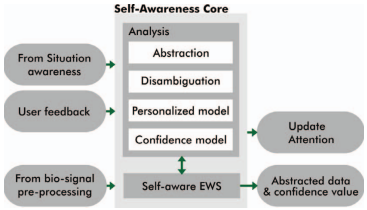


Fig. 3. Self-awareness core diagram

transmits the filtered and extracted vital signs to the self-awareness core for further analysis and decision making.

B. Situation Awareness

Situation awareness is the component that receives activity and environmental data from the sensor network and provides patient situation for the core component as well as updating *Attention* in case of ambiguity in situation determination. As demonstrated in Figure 2, it includes two main units: Analysis and Ambiguity resolution.

Analysis unit includes activity and environment analysis units, each of which determines patient situation using a decision tree [26], however, several other approaches can be also utilized to determine the status of the patient activity and the surrounding environment [27]. In the activity analysis, patient's movements (i.e., acceleration in 3 dimensions) are classified in patient postures which are sleeping, resting, walking, jogging and running. Similarly, surrounding contexts (i.e., ambient temperature, ambient humidity and ambient light) are classified into different categories, for example as indoor/outdoor, day/night, etc.

Ambiguity resolution unit updates system's setup (i.e., Attention) w.r.t the determined situations. It requests new information sources (e.g., sensor node and database) in case of ambiguity in the Analysis unit results. For example, it sends a command to turn on the light sensor if the ambient temperature sensor is insufficient for determining the indoor/outdoor situation. On the other hand, to avoid unnecessary energy dissipation, this unit will request to remove a resource if redundancy is detected in situation determination.

C. Self-awareness Core

Self-awareness core is the main analytical component of the system which is in charge of tuning system configuration (e.g., energy and bandwidth) as well as refining abstracted patient data for the back-end users. This component receives vital signs and situation values and provides an enhanced context-aware and personalized score which we call it Self-aware EWS. It also provides confidence assessment of the input data as well as correction methods to eliminate data inconsistencies. As illustrated in Figure 3, this component includes two main units: *Analysis* and *Self-aware EWS*.

Analysis unit consists of a semantic interpretation and models of activity and environment. The interpretation includes *Abstraction* and *Disambiguation* to provide meaningful information for the models and the back-end users. The Abstraction maps the medical data and the patient state to an interpretation. For instance, "low" is extracted as the emergency level for the patient with a heart rate of 140 per second while s(he) is running outdoor. Additionally, *Disambiguation* removes uncertainty in the abstracted values when the *Abstraction* encounters at least two conflicting values for the same condition.

The two data models are generated from pre-defined metadata using rule-based and decision tree classifiers. The first model, "Personalized model", is defined according to the constant patient parameters such as age, body mass index (BMI), and gender. This model is updated during the patient monitoring process with user feedback, i.e., patients and health professionals. The second model is the "Confidence model" which is defined to indicate how confident the system is. The model considers three different aspects of medical parameters to calculate the confidence value: natural ranges of parameters (e.g., a heart rate beyond 220 heartbeats per minute is not acceptable), variation ranges (e.g., body temperature increases/decreases gradually), and dependency among events (e.g., high body temperature is relative to high heart rate) [24].

Self-aware EWS is in charge of adjusting the traditional EWS value for mitigating the susceptibility of the score to the patient and environmental conditions. Using the Analysis unit's results and the determined situation and a pre-defined rule-based algorithm, the Self-aware EWS unit calculates a new method by adjusting the boundary values shown in Table I [28].

Finally, the abstracted data (i.e., adjusted EWS and the patient's condition) along with confidence values and appropriate commands regarding the obtained results are transmitted to back-end system and the *Attention* component, respectively.

It is important to note that sending confidence value along with the score and other data, can lead to a significant enhancement of the system and patient's health assessment. For example, for the patient's health assessment, that is, if for any reason (such as missing or unreliable data), the system is not confident about its assessment, respective users are informed about this factor. For example, if the health of the user is assessed to be normal, however, the confidence level is low, the physician may choose to perform certain follow ups, e.g., calling the patient to ask some extra questions. Similarly, if the score is high but the confidence of assessment is low, it may be advisable to contact the patient for follow up controls rather than dispatching immediately the emergency team (which could be the case if both score and its confidence are high). Therefore, this parameter can be significantly helpful in avoiding misinterpretations.

D. Attention

Attention is the planning component which adaptively tunes monitoring knobs to enhance system characteristics as well as the confidence and quality of the sensory data. It receives

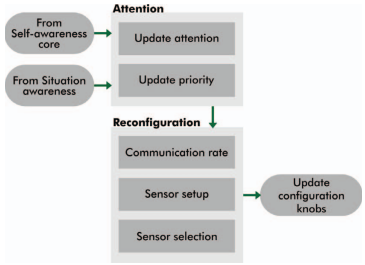


Fig. 4. Attention and Reconfiguration diagrams

information and hints regarding the state of the patient and environment from the *Self-awareness core* and the *Situation awareness* components and chooses an optimal setting for meeting the requirements while offering efficiency and reliability to the system. It then transmits proper commands to the *Configuration* component in order to update (i.e., actuate) the properties of the sensor network. Figure 4 illustrates *Attention* unit which includes two main parts; First is the attention configuration which determines which parameters (i.e., sensors) should be monitored and how often. The second part is the priority list which is used to keep track of priorities and used for conflict resolution once the attention requirements cannot be met using the available resources at the moment. In such a case, the priority list determines which requirements are of more importance and need to be honored first and foremost, and which ones can or may be omitted in the case of insufficient available resources.

In our EWS system, we prioritise the attention based on the patient emergency level, patient activity, and the environmental situation, respectively. In other words, when a patient' health state is at higher emergency levels, the Attention unit allocates more resources for monitoring the patient, and conversely when the patient is in non-emergency situation, the module considers other parameters to opportunistically enhance system characteristics such as energy-efficiency. A sample prioritization method that we used in our experiments is shown in Figure 5. We define four levels of the emergency, five states for the activity, and four situations for the environments. In this method. we define a priority score between 0 to 100 for each combination of emergency level, situation, and activity. As shown in the figure, emergency level has highest and environment has lowest effect on priority score. These priorities then are mapped to the number of actuation states available in the reconfiguration component.

E. Reconfiguration

The *Reconfiguration* component receives the priority values from the *Attention* unit and maps them to the corresponding state of the sensor network. As demonstrated in Figure 4, each state in the sensor network is determined by the communication rate, sensor configuration setup (e.g., sampling frequency), and sensor selection (e.g., activation or switching to sleep mode). This component sends the selected state as sensor-network-understandable commands to update the configuration knobs.

Fig. 5. Priority score chart

V. Demonstration and Evaluation

In this section, we describe the implementation of our self-aware solution for the remote patient monitoring system. We first present, how our system assesses the data confidence level, then show how we reconfigure and calibrate a generic EWS system in a real-time fashion by taking the patient's context into consideration, and finally demonstrate the energy efficiency gain offered by our self-aware closed loop edge controller for the body area sensor network. The system collects medical and activity data from the body area sensor network and environmental properties from another set of wireless sensors. The plausible range of data together with the rate of changes and the relation to the other parameters help enhancing the reliability of collected data. Patient activities and environmental data give an overview of the situation and help redefining the early warning score limits. The modified score reconfigures the system state to reduce energy consumption in the body area sensor network.

The body area sensor network consists of five sensors: 1) A SPO_2 finger grip sensor which provides the value of blood oxygen saturation and heart rate every second, 2) an airflow sensor to record respiration rate (we record its analog output with 100 samples per second), 3) a blood pressure sensor with arm cuff; for each measurement we continuously record the analog pressure signal for two minutes with 100 samples per second to calculate the systolic and diastolic blood pressure using the oscillometric method, 4) a 3D-accelerometer sensor to record patent's activity with 100 samples per second sampling rate, and 5) a temperature sensor to record body temperature with the same sampling rate as the heart rate and SPO_2 sensors.

The environmental sensors measure temperature and light, collecting samples once every minute. There is a micro-controller unit (MCU) in each set of sensors to collect and convert signals, send data to the RF module, and switch between states.

Adjusting EWS based on data reliability (confidence): The communication to a sensor can be faulty, or the sensor itself can be broken or detached from the patient. Therefore, in the self-aware core, we check the reliability of data and assign a degree of confidence by which the data and consequently the score assessments can be relied upon.

The algorithm consists of three different modules: checking the measured value to ensure (i) it is in a plausible range, (ii) it has plausible rates of change, and (iii) it corresponds with other vital signals (i.e., cross validity). Figure 6 shows the results of the three experiments, respectively for modules (i) to (iii). In the first experiment, a body temperature's value of $100°C$ was injected as a faulty value. Therefore, it gets tagged

as unconfident and abstains from the self-aware EWS (SA-EWS) calculation. While the SA-EWS correct shows the score *0*, the conventional EWS equals 3 at the beginning due to the faulty input and the absence of a validation system. Experiment 2 deals with the consistency of the input signals. The body temperature is initialized with a value of $36°C$ and then, after a short period of time, drops with a *rate of change* beyond the acceptable range. While the conventional EWS changes from 0 to 2, the system identifies the body temperature as unconfident and revises the SA-EWS score to 0. Finally, Experiment 3 shows the third module which works with abstracted data. The body temperature was set to a value which is equivalent to score *1* and all the other inputs where - time displaced - set to score *1*. After a while, when more than 50% of these signals have a non-zero status, the temperature is tagged as confident, and the SA-EWS becomes equal to the EWS. The details of the confidence evaluation method can be found in our previous work presented in [24].

Adjusting EWS based on the situation: In *Situation Awareness* module, we use the collected data from activity and environment sensors to find the situation of the patient. We define the environment as *day/night* and *indoor/outdoor* using temperature and light sensors and the system clock. We use the Geo-location service of the gateway smartphone together with the normal room temperature (18°C to 24°C) to indicate the indoor situation. We determine the day vs. night by comparing the time with approximate sunrise and sunset time in the local timezone. The intensity of light helps in determining is the patient indoor or outdoor. Furthermore, light intensity can be used to increase confidence level of determining sleeping of the potient. Finally, we utilize direction and amplitude changes in patient's body acceleration using a 3D-accelerometer sensor to determine the activity.

Then goal of the proposed self-aware health monitoring system is to improve the standard early warning score method by considering the fact that the patient is not in a standard clinical environment all the time. This module performs two main tasks: adjusting the scores' ranges in the EWS table based on the patient's activity and adapting the EWS calculation in the case of incorrect readings. The first task starts by calculating the normal early warning score and emergency level. Once an increase in heart rate, respiration rate, blood pressure or body temperature is observed, the system cross checks with the activity state. If this change is due to walking, jogging or running, we adjust the early warning score to avoid false alarms. The details of the self-aware modification of early warning score method can be found in our previous work discussed in [28]. Once a reliable score is obtained, we classify the score according to emergency levels. In this classification, a score 0 means a normal level, scores 1-3 indicate a low emergency level, scores 4-6 show a medium emergency level, and higher scores (> 6) represent a high emergency level.

As a case study, we use 8 hours of recorded data from a 35 years old healthy male subject whose state in practice should be detected as *Normal*. The first chart, from the top, in Fig. 8 shows the calculated scores using the original EWS table which issues several false alarms while the subject is running and jogging. The second chart in this figure shows the calibrated scores using self-aware EWS at runtime considering the state of the activity and environment which can be seen from the third and fourth charts. The results show that self-aware EWS correctly reports the normal and low emergency levels in 99% of the monitoring samples.

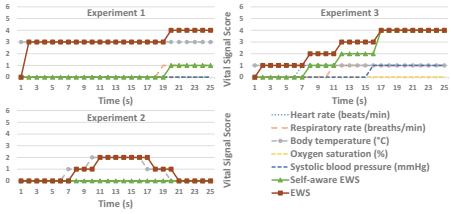


Fig. 6. Enhancing the EWS using confidence. Adjusted EWS after reliability validation of the input data, avoids elevating EWS due to faulty data.

Emergency Level:	Score:0 Normal				Score:1-3 Low				Score:4-6 Medium				Score>6 High			
	Indoor		Outdoor		Indoor		Outdoor		Indoor		Outdoor		Indoor		Outdoor	
	Day	Night	Day	Night	Day	Night	Day	Night	Day	Night	Day	Night	Day	Night	Day	Night
Sleeping	E	E	E	E	C	D	D	D	B	C	C	C	A	A	B	B
Resting	D	D	D	D	C	C	C	C	B	B	B	B	A	A	B	B
Walking	C	C	C	C	B	C	C	C	B	B	B	B	A	A	A	B
Jogging	C	C	C	C	B	B	B	C	B	B	B	B	A	A	A	B
Running	C	C	C	C	B	B	B	B	B	B	B	B	A	A	A	A

Fig. 7. States look-up table derived from attention score

In addition to provide healthcare professionals with these values, the state of the patient is used by the Attention module to fine-tune the sensor network parameters to a more efficient state.

Optimizing energy efficiency using Attention: We use HM-11 Bluetooth low energy module to transmit the signals to an Android phone acting as the gateway. We measure the power consumption of the transmission process using a power monitor device. The results show that the power consumption of this module, when operating at 3.3V is in general at one of the following levels depending on the operation mode: 1) in standby mode, when the module is on but not sending data, the Bluetooth module consumes 26.2 mW, 2) in transmission mode, when the module sends data continuously with 115200 bit/second baud rate, it consumes 29 mW, and 3) in sleep mode the module uses 1.52 mW. Considering the power consumption of Bluetooth module, we define 5 different states (A to E) for the data transmission. As the volume and resolution of the required data changes with the situation of the subject, the sampling rate of the medical and activity sensors is divided into these five states in a way that the required data is provided while maximum number of standby/sleep modes is utilized. Considering five states of transmission with different bandwidth and energy consumption requirements, we map the output of the priority list shown in Figure 5 to a slot in one of the four lookup tables shown in Figure 7. After looking up a proper state, a new configuration state is sent back to the MCUs in the sensor network to update the transmission rate and activity mode of the transmission module. Table II shows the details of the data collection orders and power consumption of the each state. As the Bluetooth low energy module takes 1235 ms to wake up from the sleep mode, we use the standby mode for states A and B, and to get the benefits of the ultra low power sleep mode, we set non-continuous parameters sampling rate to be recorded every minute in other states.

TABLE II. DESCRIPTION OF DEFINED STATES

State	Respiration Rate Activity	Blood Pressure	Heart Rate, SpO2, and Body Temp.	Transmission Power Consumption
A	Continuous	Every hour in day Disabled in night	Every sec.	29 mW
B	2 min continuous 8 min OFF	Every hour in day Disabled in night	Every sec.	26.8 mW
C	2 min continuous 3 min OFF	Every 3 hours in day Disabled in night	Every min.	12.5 mW
D	2 min continuous 8 min OFF	Every 3 hours in day Disabled in night	Every min.	7 mW
E	2 min continuous 18 min OFF	Disabled	Every min.	4.3 mW

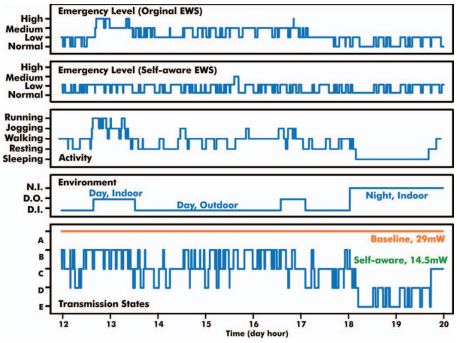


Fig. 8. Self-aware EWS system vs. conventional EWS system

The bottom chart in Figure 8 shows the achieved power saving due to closed-loop control of transmission states performed by the Attention unit for the same set of recorded data. We look up the results of self-aware EWS calculation, activity, and environmental situation using the mapped lookup tables shown in Fig.7 to adaptively adjust the transmission mode of the RF module of the sensor node. The chart shows that overall power consumption of the transmission is reduced by 50% to 14.5mW compared to a baseline non self-aware system which consumes 29mW.

VI. CONCLUSIONS

Early Warning Score (EWS) is a method to predict sudden health deterioration of patients suffering from life-threatening diseases to subsequently provide early diagnosis and treatments. Integration of health monitoring with ubiquitous IoT-based systems could enable patients to be monitored continuously not only in hospitals but also at home and at work. The traditional EWS method, however, is inappropriate for out-of-hospital patient monitoring due to challenging issues from both the user and system perspectives. In this paper, we introduced an IoT-based EWS system using the concept of self-awareness to target both perspectives. On one hand, our system offered a personalized and self-organized decision making for patients engaged in various activities in different environments. On the other hand, in this system, we proposed a self-awareness-enabled method to improve the system's energy efficiency and its confidence in its computed results, i.e. the EWS values. We demonstrated the benefits of our solution in a proof of concept full system implementation which reveals an improved level of data dependability and system energy efficiency compared to conventional open-loop systems.

REFERENCES

[1] J. McGaughey *et al.*, "Outreach and early warning systems (ews) for the prevention of intensive care admission and death of critically ill adult patients on general hospital wards," *The Cochrane Library*, 2007.

[2] Morgan *et al.*, "An early warning scoring system for detecting developing critical illness," *Clin Intensive Care*, vol. 8, no. 2, p. 100, 1997.

[3] A. Dohr *et al.*, "The internet of things for ambient assisted living," in *Proc. Int. Conf. Information Technology: New Generations*, 2010.

[4] L. Atzori *et al.*, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[5] A. Anzanpour *et al.*, "Context-aware early warning system for in-home healthcare using internet-of-things," in *Proc. of HealthyIoT*, 2015.

[6] M. Miorandi, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497 – 1516, 2012.

[7] A. Anzanpour *et al.*, "Internet of things enabled in-home health monitoring system using early warning score," in *Proc. of MobiHealth*, 2015.

[8] C. Taylor, *Fundamentals of NURSING, Art & Science of Nursing Care*. Philadelphia: LIPPINCOTT, 2005.

[9] P. R. Lewis *et al.*, "A survey of self-awareness and its application in computing systems," in *SASOW*, 2011, pp. 102–107.

[10] A. Jantsch and K. Tammemae, "A framework of awareness for artificial subjects," in *CODES+ISSS*, 2014, pp. 1–3.

[11] R. W. Urban *et al.*, "Modified early warning system as a predictor for hospital admissions and previous visits in emergency departments," *Advanced emergency nursing journal*, vol. 37, no. 4, pp. 281–289, 2015.

[12] D. J. Brown *et al.*, "Accidental hypothermia," *New England Journal of Medicine*, vol. 367, no. 20, pp. 1930–1938, 2012.

[13] D. Georgaka, M. Mparmparousi, and M. Vitos, "Early warning systems," *Hospital Chronicles*, vol. 7, no. 1, 2012.

[14] C. Johnstone, J. Rattray, and L. Myers, "Physiological risk factors, early warning scoring systems and organizational changes," *Nurs Crit Care*, vol. 12, no. 5, 2007.

[15] R. Paterson *et al.*, "Prediction of in-hospital mortality and length of stay using an early warning scoring system: clinical audit," *Clinical Medicine*, vol. 6, no. 3, pp. 281–284, 2006.

[16] G. D. Barlow *et al.*, "Standardised early warning scoring system." *Clinical Medicine (London, England)*, vol. 6, no. 4, pp. 422–3, 2005.

[17] P. Mercati *et al.*, "A linux-governor based dynamic reliability manager for android mobile devices," in *DATE*. IEEE, 2014, pp. 1–4.

[18] B. Jennings *et al.*, "Resource management in clouds: Survey and research challenges," *JNSM*, pp. 1–53, 2014.

[19] P. Spathis *et al.*, "ANA: Autonomic Network Architecture," *Autonomic Network Management Principles*, p. 49, 2011.

[20] J.-S. Preden *et al.*, "The benefits of self-awareness and attention in fog and mist computing," *IEEE Computer, Special Issue on Self-Aware/Expressive Computing Systems*, pp. 37–45, July 2015.

[21] H. Hoffmann *et al.*, "Seec: A framework for self-aware computing," Tech. Rep., 2010.

[22] N. Dutt *et al.*, "Toward smart embedded systems: A self-aware system-on-chip (soc) perspective," *TECS*, vol. 15, no. 2, p. 22, 2016.

[23] N. Taherinejad *et al.*, "Comprehensive observation and its role in self-awareness; an emotion recognition system example," in *FedCSIS Position Papers*, 2016.

[24] M. Götzinger *et al.*, "Enhancing the early warning score system using data confidence," in *Proc. of MobiHealth*, 2016.

[25] A. M. Rahmani *et al.*, "Smart e-Health Gateway: Bringing intelligence to Internet-of-Things based ubiquitous healthcare systems," in *Proc. of CCNC*, 2015.

[26] M. J. Mathie *et al.*, "Classification of basic daily movements using a triaxial accelerometer," *Medical and Biological Engineering and Computing*, vol. 42, no. 5, pp. 679–687, 2004.

[27] O. D. Lara *et al.*, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[28] I. Azimi *et al.*, "Self-aware early warning score system for IoT-based personalized healthcare," in *Proc. of IoTCare*, 2016.

Maximilian Götzinger, Nima TaheriNejad, Amir M. Rahmani, Pasi Liljeberg, Axel Jantsch, and Hannu Tenhunen

VII

Enhancing the Early Warning Score System Using Data Confidence

In *Wireless Mobile Communication and Healthcare*. Springer International Publishing, Cham. EAI, Milano, Italy 2016, pages 91–99

Enhancing the Early Warning Score System Using Data Confidence

Maximilian Götzinger[1(✉)], Nima Taherinejad[2], Amir M. Rahmani[1], Pasi Liljeberg[1], Axel Jantsch[2], and Hannu Tenhunen[1]

[1] Department of Information Technology, University of Turku, Turku, Finland
{maxgot,amirah,pakrli,hannu.tenhunen}@utu.fi
[2] Institute of Computer Technology, TU Wien, Vienna, Austria
{nima.taherinejad,axel.jantsch}@tuwien.ac.at

Abstract. Early Warning Score (EWS) systems are utilized in hospitals by health-care professionals to interpret vital signals of patients. These scores are used to measure and predict amelioration or deterioration of patients' health status to intervene in an appropriate manner when needed. Based on an earlier work presenting an automated Internet-of-Things based EWS system, we propose an architecture to analyze and enhance data reliability and consistency. In particular, we present a hierarchical agent-based data confidence evaluation system to detect erroneous or irrelevant vital signal measurements. In our extensive experiments, we demonstrate how our system offers a more robust EWS monitoring system.

Keywords: Early Warning Score · Self-awareness · Data confidence · Consistency · Plausibility · Hierarchical agent-based system

1 Introduction

Early Warning Score (EWS) systems are common practice in hospitals with the goal of detecting and predicting patients' health deterioration. In 1997, Morgen *et al.* proposed this system for the first time [1], covering vital signals such as heart rate, respiratory rate, body temperature, blood pressure, and blood's oxygen saturation. These signals are monitored and added up to derive the EWS. However, not everyone whose condition is deteriorating is already in the hospital. Therefore, portable devices and ubiquitous systems utilizing Internet-of-Things are needed for monitoring vital signals and calculating the EWS [2].

It is of key importance to provide these systems with an acceptable level of reliability. In other words, EWS systems always need to monitor vital signals accurately. Azimi *et al.* propose a system that calculates a self-aware EWS through changing the classification of the various vital signals based on the patient's activities [2]. This self-aware property is essential because the values of vital signals change when a patient is sleeping or running. Knowledge of different situations and circumstances improves the decision-making ability of the

P. Perego et al. (Eds.): MobiHealth 2016, LNICST 192, pp. 91–99, 2017.
DOI: 10.1007/978-3-319-58877-3_12

system [3]. However, they assume that the measured data is always correct and relevant. Noisy or erroneous vital signals can lead to a wrong calculation of the EWS, which can result in false or missing alarms. Hence, EWS systems need to be robust and aware of the reliability of input data.

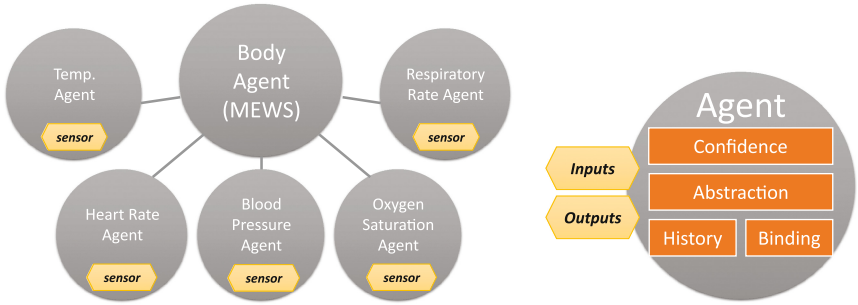
In this paper, we propose a modified EWS (MEWS) system by exploiting a customized data confidence enhancement technique. Our method is inspired by the concept of self-awareness enabling the system to - adaptively - correct the sensory data in case of faulty readings.

2 System Architecture

In an agent-based modular architecture (Fig. 1(a)), each sensor is connected to a dedicated module which we call an "Agent" [4]. It processes the sensory data and reports to a higher level agent, which is the "Body Agent". Each agent consists of an Abstraction-, a History-, a Confidence Validator-, and a Binding Module. The role of each module in an agent is as follows:

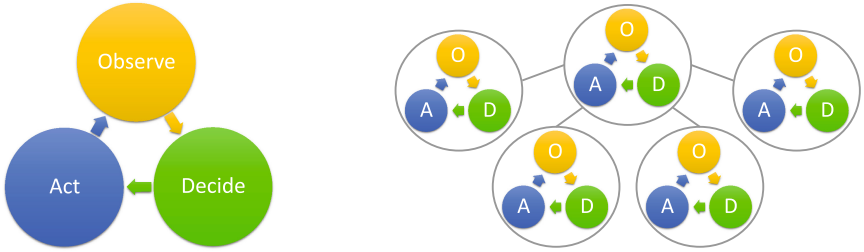
- *Abstraction*: To change the representation of the input data to the appropriate format of the output. The purpose is to provide the higher level agents with more compact and only relevant information [5].
- *History*: To save recent data, track changes, and establish a stable baseline for the data when possible. This unit also smooths the data via weighted averaging to eliminate the noise in the signal.
- *Confidence*: To assess the trustability of the input data and provide the output data with a confidence tag, that allows the higher levels to have a better understanding of the data and their validity. This topic is discussed in more details in Sect. 3.
- *Binding*: To bind several input data, relate or compare them, and perform necessary operations on them. This module is specifically useful when an agent has multiple inputs, as is the case with the Body Agent. We note that this process is more complicated than a simple mapping of the values as done in the Abstraction module.

To enhance the functionality of our system, we have incorporated some of the concepts of self-awareness. Self-awareness is a well-known concept which can be traced back to 1960s in psychology [6] and late 1990s in computing [7]. It provides several advantages to the system such as the ability to cope with changing environments [6] or changing goals [8], and to optimize resource utilization [9]. As the basis for our self-aware system design, we use an Observe-Decide-Act (ODA) loop [7,8] as illustrated in Fig. 2(a). For better modularity and simpler implementation we use a mini ODA loop inside each agent, as shown in Fig. 2(b). That is, each agent monitors its own behavior, decides about certain actions, and acts accordingly. Self-awareness covers a wide range of aspects in the system design under each of the chains of the ODA loop. All of which could provide the system with certain abilities and advantages. In this work, we specifically concentrate on the role of the confidence aspect of observation as elaborated in [5]. We then analyze its effect on the overall performance of our EWS system.



(a) System Architecture (b) Agent Architecture

Fig. 1. System Architecture; (a) Constituting agents (modules) of the system, and (b) Constituting units of the agents.



(a) ODA (b) Modular ODA

Fig. 2. Observe-Decide-Act (ODA) loops implemented in (a) overall system, and (b) each module.

3 Data Confidence Concepts

Data Confidence is meta-data and builds on Data Reliability (which consists of accuracy and precision of sensory data [5]). It provides another level of understanding regarding the validity of the data which is beyond that of the sensors. For example, in the context of the EWS, if the sensor is not attached to the body of the subject, the temperature data provided by the sensor may still be accurate and precise. However, it is not valid in the context of the application. Therefore, although the data is reliable, the system should not consider such a value. Assessing Data Confidence based on the context and the application can be very challenging [5]. Among the identified potential solutions are consistency and plausibility control as well as redundant verification [5]. Since the latter requires redundant hardware and implies additional costs and our objectives include cost as well as energy efficiency, in this work we focus on the two former aspects: consistency and plausibility.

Consistency: Anomalies are - at some level of analysis - inconsistent with the normal trend of data, which could indicate a problem[1]. Hence, Consistency is an aspect that can provide insight into how confident the system can be about its observation. In the context of our EWS system, we consider temperature continuity as an indicator for data consistency. Body temperature has very small and slow changes; a change of 0.16 °C during one minute can be normal. However, a change of several degrees per minute is inconsistent with the nature of the subject of measurements (body temperature) [10]. This may be caused by a sensor failure or a detachment of the sensor from the body. Regardless of the reason, this should not affect the warning score. After performing consistency analysis and finding an inconsistent behavior, by reducing the confidence tag of the incoming data, the EWS system knows that it should not take this number into consideration. We note that in some other parameters, such as respiratory rate, for example, some discontinuities might be acceptable and should not be marked as an inconsistency or decrease the confidence of the system in the incoming data. Hence comes forward the next aspect of confidence, which is the plausibility.

Plausibility: One aspect of plausibility which goes hand in hand with consistency is the plausibility of changes in the data, e.g. body temperature change. Another aspect is the plausibility of the absolute value. For example, a body temperature of 85 or 95 °C is not plausible and regardless of the cause, it should not be considered for score evaluation of the EWS. The same goes for negative temperatures of this magnitude, or in the case of oxygen saturation, for values outside 0 to 100.

Another aspect of plausibility is the cross-validity or co-existence plausibility. That is, whether certain data could plausibly be valid given some other (complimentary) data and given certain conditions. For example, a body temperature of few degrees is valid only if the subject does not have any other vital signals (and is practically deceased), otherwise, it shows a discrepancy and the data cannot be trusted. Therefore, by adding such logical information regarding the co-existing situations and signals, the system can perform a cross-validity check and obtain another level of holistic awareness regarding the confidence it can invest in the observed data.

4 Impact Evaluation

In this section, we explain how we have taken advantage of the concepts discussed in previous sections to enhance the reliability of our EWS. The details of our experimental set-up and acquired results are as follows.

4.1 Experiments Set-Up

EWS Table: Because human body functions have some variance from person to person, there exist several different EWS classification tables from various studies

[1] We note that the consequence of an anomaly detection should be/is decided by higher levels of the system. Regardless, the observation unit needs to alert the higher levels.

Table 1. Score classification table of a set of vital signals

Vital signal score	3	2	1	0	1	2	3
Heart rate (beats/min)	<40	40–51	51–60	60–100	100–110	110–129	>129
Systolic blood pressure (mmHg)	<70	70–81	81–101	101–149	149–169	169–179	>179
Respiratory rate (breaths/min)		<9		9–14	14–20	20–29	>29
Oxygen saturation (%)	<85	85–90	90–95	>95			
Body temperature (°C)	<28	28–32	32–35	35–38		38–39.5	>39.5

[2,11,12]. In this work, as shown in Table 1, we mainly use a similar table as in [2]. Whereas the original table showed only one possible score (= 2) for the hypothermia, Brown *et al.* introduced in their work [13] four different stages of an accidental hypothermia (called HT I to HT IV). Following that approach, we combined HT III and IV because HT III shows symptoms of weak- and HT IV of no vital signals.

Patients' Data: The vital signal data were obtained from the experiments carried out by Azimi *et al.* [2]. This dataset contained records of heart rate, systolic blood pressure, respiratory rate, and oxygen saturation of a 35 years old healthy male subject [2]. To evaluate the behavior of the system during a malfunction, instead of the measured temperature, we introduced faulty temperature data.

Analysis Environment: For the analysis of the EWS, we used our hierarchical agent-based model toolbox. It is developed in C++, and its agents can be configured in different ways based on the requirements.

4.2 Confidence Assessment and Results

Experiment 1: Absolute Bounds

The first validation step is to check if a measured value is in a plausible range. For temperature for example, according to Omics International[2], the temperature extremes in Europe are about −58°C and 48 °C. Therefore, these values can be used to define the extreme lower and upper bounds of the temperature. The measured value will be classified as valid if it is in this range. Although this boundary allows us to evaluate the behavior of system regarding this parameter, we note that more accurate values will have to be chosen when the system is used to monitor a patient's condition in real life.

Figure 3 shows the results of the confidence validation's regarding the absolute bounds. The body temperature was manually set to 100 °C which is out of the absolute bounds. Therefore, the score of the temperature is 3 for the whole time if it is not checked regarding its confidence and 0 if it is checked.

[2] http://research.omicsgroup.org/index.php/List_of_weather_records, accessed on July 2016.

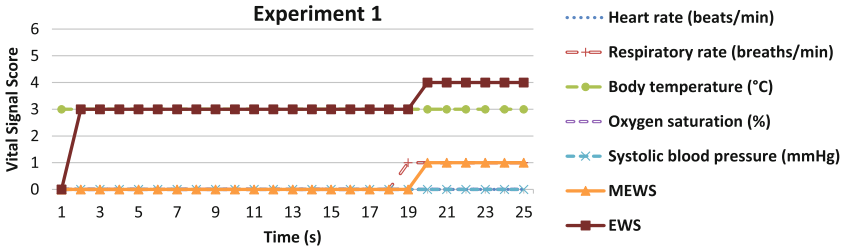


Fig. 3. Calculation of the MEWS and EWS with the same data set. Body temperature is manually set to 100 °C which is out of the absolute bounds. CCVU deactivated for showing the difference.

Experiment 2: Change Rate

Here, we concentrate on the consistency of the data based on the maximum plausible rates of change of an input signal. Regarding the body temperature, the highest cooling rates obtained from persons that got completely buried under an avalanche are between 6 °C/h to 9.4 °C/h [10,14,15]. Assuming that temperature of a human body cannot increase faster than it can decrease, we set the maximum possible rate of change to 10 °C/h (= 0.17 °C/min = 0.003 °C/s). The body temperature will be considered as unconfident if the rate of change is higher than the maximum allowed limit set. The input signal has to have approximately the same value (previous value ± allowed rates of change) it had before it was unconfident to get the confident status back[3].

Figure 4 shows the results of the confidence validation's regarding the change rate. The body temperature was manually set to 36 °C which is equivalent to score 0. After a short period, the temperature was decreased faster as the maximum allowed rate of change. We can see that in the absence of Confidence Validation Unit (CVU), we have false alarms which we do not observe in the enhanced system. If the CVU is deactivated, the body temperature score is unequal to zero when the associated input gets lower than 35 °C. On the other hand, the input signal is being considered as unconfident if the CVU is activated. Now to get the new data tagged as confident again, regardless of its change, its absolute value needs to go back to latest value tagged as confident ± allowed change. For example, we set the temperature signal to an unchanging value between the seconds 12 and 14 and although the input signal's rate of change is equal to zero, it is still classified as unconfident.

Experiment 3: Cross Confidence Validation

Humans' vital signals such as heart rate, blood pressure, and respiratory rate change with a body temperature when it is too high or too low [13,16]. A mild hypothermia can come along with symptoms such as tachycardia

[3] We remark that to ascertain a signal's rate of change, a history is needed. As a preparatory work, history has to get smoothed before calculating the rates of change, otherwise, noise could affect this measurement.

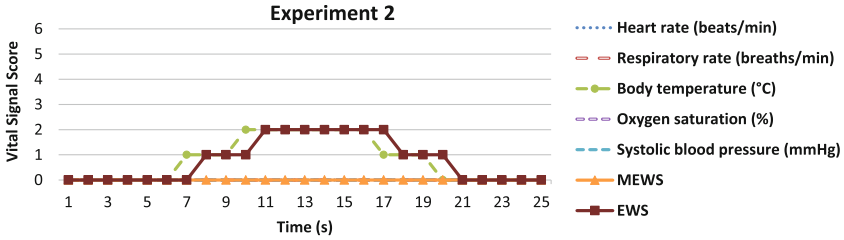


Fig. 4. Calculation of the MEWS and EWS with the same data set. Body temperature is manually set to 36 °C (score 0) and then decreased faster as the maximum allowed rates of change is set. CCVU deactivated for showing the difference.

and tachypnea, a medium hypothermia already shows signals of hypotonia and bradycardia. Henceforth, the lower the body temperature the weaker the vital signals get; until they finally stop [17]. Regarding hyperthermia, the changes of vital signals are not completely identical, but show a similar behavior; that is, general deterioration [16]. By implication, this means that body temperature cannot be injurious if all the other vital signals have a good value.

In contrast to the two steps before, the Cross Confidence Validation Unit (CCVU) needs already abstracted knowledge from different sources. Therefore, this validation is only possible at a higher hierarchical level. In our case, that is the body agent which gets the abstracted data from the different agents. The CCVU was configured to consider the measured temperature as valid if more than 50% of the vital signals (body temperature excluded) have a non-zero score in accordance to that of the temperature.

Figure 5 shows results of the confidence validation test. The body temperature was manually set to 32 °C (score 1). The other input signals are time-displaced to set their values to non-zero scores, one input after the other. It can be seen that at 17 s the MEWS is changing when more than 50% (3 out of 4) of the input variables reach a non-zero score. We can see that if the temperature sensor is included in the EWS calculation - when the CCVU is deactivated - we

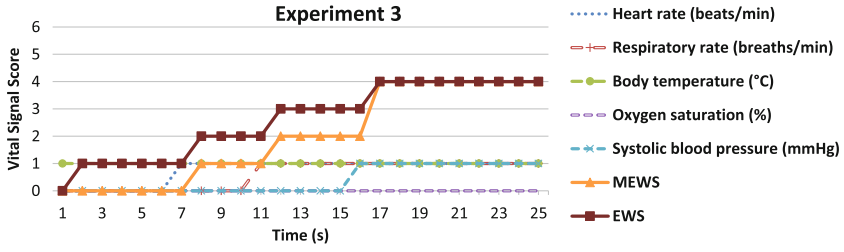


Fig. 5. Calculation of the MEWS and EWS with the same data set. Body temperature is manually set to 32 °C (score 1) and the other input signals are time-displaced (one after the other) to values where there are no score 0 present.

have a higher EWS. Such a case, nonetheless, is physiologically not possible and hence, the EWS should not be affected.

5 Conclusion

It is vital that the Early Warning Score is computed correctly at all times, in spite of potential complications in the input data stream. Otherwise, there could be false or missed alarms. In this paper, we show that it is possible to check the confidence of the input data with our modular solution based on self-awareness. Using this concept, the reliability of EWS improved in all three cases we experimented with. Thus, we demonstrated that using the data confidence validation system, the quality, and robustness of the EWS assessment can be improved.

We used a hierarchical agent-based system which allows processing both the data and their meta-data, such as the confidence assessment. Due to its modularity and a good match of the data processing flow from lower to higher abstraction levels, it is a promising architecture for EWS or similar systems.

In the future, we will extend our framework and add various features such as the ability of learning. We assume that a learning unit could help choosing better boundaries and values, based on the personalized behavior of the subject, for confidence evaluation and consequently the score calculation.

References

1. Morgan, R., Williams, F., Wright, M.: An early warning scoring system for detecting developing critical illness. Clin. Intensive Care **8**(2), 100 (1997)
2. Azimi, I., Anzanpour, A., Rahmani, A.M., Liljeberg, P., Tenhunen, H.: Self-aware early warning score system for IoT-based personalized healthcare. In: Proceedings of International Conference on IoT and Big Data Technologies for HealthCare (2016)
3. Jantsch, A., Tammemäe, K.: A framework of awareness for artificial subjects. In: 2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 1–3. IEEE (2014)
4. Götzinger, M., Rahmani, A., Pongratz, M., Liljeberg, P., Jantsch, A., Tenhunen, H.: The role of self-awareness and hierarchical agents in resource management for many-core systems. In: Many-core Systems-on-Chip (MCSoC) (2016)
5. TaheriNejad, N., Jantsch, A., Pollreisz, D.: Comprehensive observation and its role in self-awareness; an emotion recognition system example. In: The Federated Conference on Computer Science and Information Systems (FedCSIS), September 2016
6. Rinner, B., Esterle, L., Simonjan, J., Nebehay, G., Pflugfelder, R., Fernandez Dominguez, G., Lewis, P.R.: Self-aware and self-expressive camera networks. Computer **48**(7), 21–28 (2015)
7. Dutt, N., Jantsch, A., Sarma, S.: Toward smart embedded systems: a self-aware system-on-chip (SoC) perspective. ACM Trans. Embed. Comput. Syst. (TECS) **15**(2), 22 (2016)

8. Hoffmann, H., Maggio, M., Santambrogio, M.D., Leva, A., Agarwal, A.: SEEC: a framework for self-aware computing. MIT, Technical report. MIT-CSAIL-TR-2010-049, October 2010
9. Teich, J., Henkel, J., Herkersdorf, A., Schmitt-Landsiedel, D., Schröder-Preikschat, W., Snelting, G.: Invasive computing: an overview. In: Hübner, M., Becker, J. (eds.) Multiprocessor System-on-Chip, pp. 241–268. Springer, New York (2011)
10. Pasquier, M., Moix, P.-A., Delay, D., Hugli, O.: Cooling rate of 9.4 °C in an hour in an avalanche victim. Resuscitation **93**, e17–e18 (2015)
11. Urban, R.W., Mumba, M., Martin, S.D., Glowicz, J., Cipher, D.J.: Modified early warning system as a predictor for hospital admissions and previous visits in emergency departments. Adv. Emerg. Nurs. J. **37**(4), 281–289 (2015)
12. Groarke, J., Gallagher, J., Stack, J., Aftab, A., Dwyer, C., McGovern, R., Courtney, G.: Use of an admission early warning score to predict patient morbidity and mortality and treatment success. Emerg. Med. J. **25**(12), 803–806 (2008)
13. Brown, D.J., Brugger, H., Boyd, J., Paal, P.: Accidental hypothermia. N. Engl. J. Med. **367**(20), 1930–1938 (2012)
14. Putzer, G., Schmid, S., Braun, P., Brugger, H., Paal, P.: Cooling of six centigrades in an hour during avalanche burial. Resuscitation **81**, 1043–1044 (2010)
15. Oberhammer, R., Beikircher, W., Hörmann, C., Lorenz, I., Pycha, R., Adler-Kastner, L., Brugger, H.: Full recovery of an avalanche victim with profound hypothermia and prolonged cardiac arrest treated by extracorporeal re-warming. Resuscitation **76**(3), 474–480 (2008)
16. Fauci, A.S., et al.: Harrison's Principles of Internal Medicine, vol. 2. McGraw-Hill, Medical Publishing Division, New York (2008)
17. McCullough, L., Arora, S.: Diagnosis and treatment of hypothermia. Am. Fam. Phys. **70**(12), 2325–2332 (2004)

Maximilian Götzinger, Arman Anzanpour, Iman Azimi, Nima TaheriNejad, and Amir M. Rahmani

Enhancing the Self-Aware Early Warning Score System through Fuzzified Data Reliability Assessment

In *Wireless Mobile Communication and Healthcare*. Springer International Publishing, Cham. EAI, Vienna, Austria 2017, pages 3–11

VIII

Enhancing the Self-Aware Early Warning Score System Through Fuzzified Data Reliability Assessment

Maximilian Götzinger[1(✉)], Arman Anzanpour[1], Iman Azimi[1], Nima TaheriNejad[2], and Amir M. Rahmani[2,3]

[1] Department of Information Technology, University of Turku, Turku, Finland
{maxgot,armanz,imaazi}@utu.fi
[2] Institute of Computer Technology, TU Wien, Vienna, Austria
nima.taherinejad@tuwien.ac.at
[3] Department of Computer Science, University of California Irvine, Irvine, USA
amirr1@uci.edu

Abstract. Early Warning Score (EWS) systems are a common practice in hospitals. Health-care professionals use them to measure and predict amelioration or deterioration of patients' health status. However, it is desired to monitor EWS of many patients in everyday settings and outside the hospitals as well. For portable EWS devices, which monitor patients outside a hospital, it is important to have an acceptable level of reliability. In an earlier work, we presented a self-aware modified EWS system that adaptively corrects the EWS in the case of faulty or noisy input data. In this paper, we propose an enhancement of such data reliability validation through deploying a hierarchical agent-based system that classifies data reliability but using Fuzzy logic instead of conventional Boolean values. In our experiments, we demonstrate how our reliability enhancement method can offer a more accurate and more robust EWS monitoring system.

Keywords: Early Warning Score · Modified early warning score
Self-awareness · Data reliability · Consistency · Plausibility
Fuzzy logic · Hierarchical agent-based system

1 Introduction

Chronic diseases such as cardiovascular diseases are the leading cause of death in the world [1]. Such diseases put patients at the risk of sudden health deterioration, which is reflected in patient's vital signs up to 24 h in advance. Early enough health deterioration detection effectively increases the chance of patient's survival [2].

In hospitals, particularly in intensive care units, the Early Warning Score (EWS) is a prevalent manual tool, by which patient's vital signs are periodically recorded and the emergency level is interpreted [3]. To this end, a score

P. Perego et al. (Eds.): MobiHealth 2017, LNICST 247, pp. 3–11, 2018.
https://doi.org/10.1007/978-3-319-98551-0_1

(0 for a perfect condition and 3 for the worst condition) is allocated to each vital sign according to its value and the predefined limits (see Table 1). The summation of the obtained scores indicates the degree of health deterioration of the patient (the higher the EWS, the worse the patient's health condition). However, there are two major restrictions in this manual tool. First, unreliable interpretation might be made due to the presence of inaccuracy and latency in the manual data collection. Secondly, and the more important restriction from a practical point of view, this manual tool is not applicable to out-of-hospital situations where no professional caregiver is available to perform the measurements. Recent advancements in Internet of Things (IoT) technologies can mitigate these restrictions by providing 24/7 remote health monitoring. In EWS systems based on IoT devices, patients' vital signs along with context data are continuously monitored via mobile/wearable sensors, while cloud server performs data analysis and decision making algorithms for the score determination [4,5].

Data reliability of such IoT-based EWS systems in remote health monitoring is of paramount importance. In our previous work [6], we proposed an architecture which exploits self-awareness techniques to adaptively adjust the EWS in the case of faulty readings from the sensor. We indicated a binary decision-making technique to determine whether the sensory data is reliable, and if needed we accordingly adjusted the EWS. However, like many other natural phenomena, data reliability of the sensory data is a continuous value and treating it in a binary manner, although simplifying the analysis, can lead to loss of information. For example, many somewhat reliable sensory data can lead to an unreliable assessment whereas in a binary assessment they may be interpreted as reliable (since they may fall closer to a reliable value in the spectrum) and thus create a wrong assessment.

In this paper, we propose a data reliability validation technique that is based on Fuzzy logic. The usage of Fuzzy logic instead of Boolean logic to classify input data as reliable or faulty covers the unsharp (fuzzy) ranges in which vital signs can indeed be correct or incorrect. In our extensive experiments, we show how our Self-Aware Early Warning Score (SA-EWS) method can be leveraged to enhance the reliability and robustness of health monitoring systems.

Table 1. Score classification table of a set of vital signals

Vital signal score	3	2	1	0	1	2	3
Heart rate (beats/min)	<40	40–51	51–60	60–100	100–110	110–129	>129
Systolic blood pressure (mmHg)	<70	70–81	81–101	101–149	149–169	169–179	>179
Respiratory rate (breaths/min)		<9		9–14	14–20	20–29	>29
Oxygen saturation (%)	<85	85–90	90–95	>95			
Body temperature (°C)	<28	28–32	32–35	35–38		38–39.5	>39.5

2 Data Reliability Concepts

Data reliability is an additional meta-data which describes the quality of the measured data. The reliability consists of accuracy and precision of sensory data [7] and grants a higher level of comprehension on the validity of the input data. If a sensor is broken, the monitored vital sign will be most probably inaccurate and not precise. Whereas the data provided by the sensor can still be accurate and precise when the sensor is detached from the patient's body. However, in both of these cases, an EWS calculated based on their values is invalid and therefore, unreliable in the given context. Hence, determining the reliability of the input data can be very challenging, but there exist potential solutions; consistency and plausibility controls, as well as cross validation are among them [7]. While the calculation of the EWS is based on the absolute values of the vital signs, the reliability of the EWS uses additional information about slopes and inter-correlations of the vital signs.

Consistency: Signals often have some limits such as maximum rate of change, these limits can be exploited to assess the reliability of a signal. Consistency is an aspect that can provide information on whether an observed input signal is reliable or not based on its history. A signal with a physically impossible slope indicates a problem which can be evoked by a sensor failure or a detachment of the sensor from the body. Regardless of the reason, a faulty monitored vital sign affects the calculation of the EWS negatively and should be avoided. For example, a change of the body temperature of several degrees per minute is impossible [8]. Therefore, in such a case the gathered sensory data should be classified as unreliable and treated accordingly.

Plausibility and Correlation: One aspect of plausibility is the absolute value of an input signal. For example, the oxygen saturation can only be between 0% and 100%. An input data that shows values of the oxygen saturation outside of this boundary must be classified as unreliable.

Another aspect of plausibility is the cross-reliability or co-existence plausibility. Various efforts have been conducted to indicate correlations between different vital signs [9–11]. For instance, considering the possible effect of the body temperature on the heart rate value, the probability of an increase in heart rate is high in the case of elevated body temperature [10]. As a second example, we can consider that a body temperature of $-30\,^{\circ}\mathrm{C}$ is implausible in the case of a living patient, although a deceased person lying in a very cold area can have such a low body temperature.

3 Fuzzified Reliability Assessment

In contrast to our previous work [6] where the data reliability validation was based on Boolean logic, we propose here the use of Fuzzy logic. Because of the lack of complete knowledge of all body functions, determining whether a

vital sign is monitored correctly is a hard task. Fuzzy logic brings the significant advantage of covering unsharp (fuzzy) ranges in which vital signs cannot be easily tagged as correctly monitored or not. Thus, a vital sign can have a reliability value between 0 and 1 (0% and 100%), instead of just being reliable or unreliable.

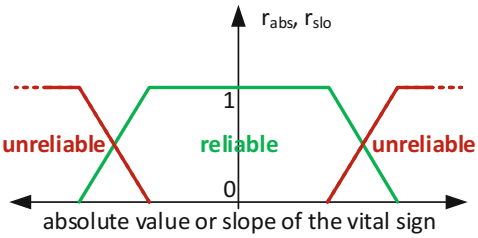


Fig. 1. Example for a fuzzy membership function.

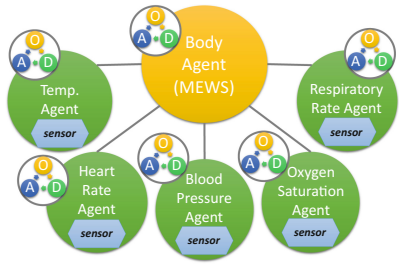


Fig. 2. System architecture.

In the proposed system, the first task of reliability module is to analyze two metrics of a vital sign, the absolute value of the signal and its slope. For this analysis, fuzzy membership functions (shown in Fig. 1) are needed, each of which is configured to match the properties of the assigned signal. The result of this analysis is given by two parameters, the reliability of the absolute value r_{abs} and that of the signal slope r_{slo}. Subsequently, the reliability of an input signal r_{sig} is calculated with

$$r_{sig} = r_{abs} \wedge r_{slo} \tag{1}$$

where the fuzzy "and" ($\wedge$) is equal to a minimum function [12]. The parameter r_{sig} gives information about the reliability of each signal considered separately and omits the correlation of the different vital signs (reviewed in Sect. 2). To consider the correlation, more highly abstracted information is needed on how one vital sign can impact another. The cross-validated reliability, r_{cro}, which exists for each pair of signals is given by

$$r_{cro} = \begin{cases} 1 & \text{if } S_{vs1} = S_{vs2} \\ \frac{1}{p_{cro}|S_{vs1} - S_{vs2}|} & \text{if } S_{vs1} \neq S_{vs2} \end{cases} \tag{2}$$

where $p_{cro} \in (0, \infty)$ denotes a coefficient of the strength of the correlation[1] between vital signs vs_1 and vs_2, and S_{vs1} as well as S_{vs2} are the abstracted scores of these two vital signs.

[1] The reliability module in our implementation limits the cross-reliability r_{cro} to a value between 0 to 1, although theoretically, a coefficient less than 1 can lead to a r_{cro} higher that 1.

When all reliability and cross-reliability values are available, the reliability of the calculated EWS is given by

$$r = (r_{sig_1} \wedge \cdots \wedge r_{sig_n}) \wedge (r_{cro_{12}} \wedge r_{cro_{13}} \wedge \cdots \wedge r_{cro_{mn}}) \tag{3}$$

where the first term conjugates all reliabilities of the various vital signs, and the second term contains the conjunction of all combinations of cross-reliabilities.

4 Experiments

4.1 Implemented System Architecture

As in our last work [6], a hierarchical agent-based model, implemented in C++, constitutes the base of the SA-EWS system (Fig. 2). Such an agent-based approach combined with the usage of mini ODA loops enable a good modularity and simple implementation. Every agent works according to an ODA loop; which means that every single agent monitors certain inputs, decides what to do, and acts accordingly.

Beside its modularity, such hierarchical agent-based architecture has another essential advantage. The input data with all its semantic content and contextual information can be abstracted in different layers [13]. As shown in Fig. 2, each agent of the lower hierarchical level is connected to a sensor. Due to the agent-based design, the scoring of vital signs and the calculation of the EWS are performed independently in different locations.

4.2 Functional Description of the System

First, each low-level agent reads the actual value of the vital sign the sensor attached to it provides. Subsequently, it abstracts the raw input data to a vital sign score S (Table 1) and validates the reliability of the signal, r_{sig} (Eq. 1). Finally, the low-level agent sends both values (score S and the signal reliability r_{sig}) to the agent of the higher hierarchical level; the "Body Agent".

Similar to the low-level agents, the body agent starts its task with reading the input values, although these are coming from the low-level agents and not from sensors. This high-level agent is responsible for the calculation of the EWS as well as the reliability of the calculated EWS. While the agent's binding module sums up all gathered scores to calculate the EWS, the reliability module calculates the cross-reliability, r_{cro}, for each pair of vital signs (Eq. 2) followed by the reliability, r, of the overall EWS (Eq. 3). As the last step and before the next data sets are read, the calculated EWS and its reliability, r, are outputted.

4.3 Experimental Data

All vital signs are collected from a 36 years old male subject with diastolic hypertension. Several sensors and devices are used for data collection. The Bioharness 3 [14] chest strap with a wearable Bluetooth sensor set is used to record the

heart rate and the respiration rate. Blood pressure and blood Oxygen saturation are recorded using iHealth BP5 [15] arm blood pressure monitor and iHealth PO3 [16] finger grip pulse oximeter which both of them are Bluetooth-enabled monitoring devices. Body temperature sensor is a DS18b20 [17] digital temperature sensor connected to ATMEGA328P [18] microcontroller and nRF51822 [19] Bluetooth low energy module. We used an Android phone to collect data from all sensors during the experiments with the rate of one sample per second.

We conditioned the data collection phase to emulate certain faults and errors. These conditions are applied in order to show how the system is able to detect the changes from normal to the abnormal condition and back from abnormal to normal condition. To this end, a change has been applied for around 5 min in the middle of a 15-min data collection. We note that the conducted experiments are proof-of-concept experiments and more extensive tests with more patients are planned for the future. The applied abnormal conditions are: (i) The temperature sensor has been detached from the body and brought to contact with an object at room temperature, (ii) The temperature sensor has been detached from the body and brought to contact with a cold object, (iii) The temperature sensor has been detached from the body and brought to contact with a hot object, (iv) A biceps contraction has happened during the blood pressure measurements, and (v) The chest strap for the heart rate and respiration rate monitor has been loosened.

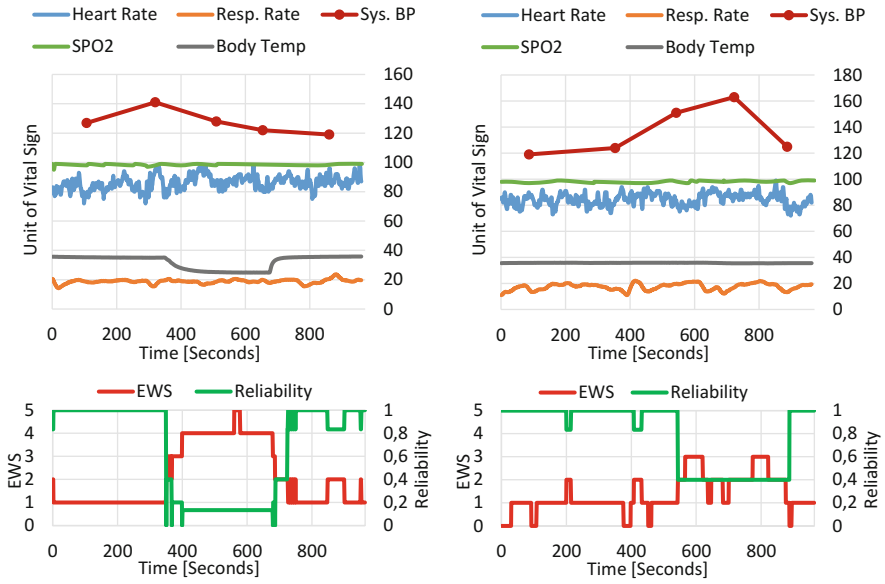
4.4 Configuration

Several factors influenced the setup of the fuzzy membership functions and the correlation coefficients. Besides the medical publications [8–11,20], expert's opinions from various physicians, the accuracy of the sensors used, and the medical condition of the patient were considered in configuring the system. To repeat the experiments with other sensors or patients, the setup should be reconfigured again to reflect such personalization. Although reconfiguration of these parameters is easy in our system, finding our the right values is a complex task which requires further research for enabling its automation.

5 Results

Our experiments show that the SA-EWS system works correctly, and the reliability of the calculated EWS coincides with the condition of the measurement setup. In other words, erroneous input data leads to a lower reliability. Due to the space limitation, only two of these cases are shown here in this section.

In the first experiment (shown in Fig. 3(a)) at around 350 s the body temperature sensor is detached and measures the room temperature until it is again attached (around 700 s). Over this period the reliability value decreases drastically. Whereas the validation of the slope causes the low reliability during the beginning and the ending phase of the period of detachment, the cross-plausibility validation does this for the rest of this period. Because of the medical condition (high respiration rate) of the test subject, the correlation between



(a) Detached temperature sensor (b) contraction of the biceps

Fig. 3. The monitored vital signs, the EWS and its reliability. (a) the body temperature sensor is detached from the patient and temporarily measures the room temperature (b) a contraction of the biceps interferes with the blood pressure measurement.

the respiration and the other vital signs was set to weak (decreased from the default value of 1.5 to 0.6). Nevertheless, during the moments when the respiration frequency reaches values greater than or equal to 20 (score 2), reliability level decreases even further.

For the second experiment shown here (Fig. 3(b)), we tampered with the measurement of the blood pressure. The gathered input data shows a high blood pressure value because the patient tensed his biceps during two of the samples (around 550 s and 700 s). Since there is a strong correlation between heart rate and blood pressure [9], the correlation coefficient p_{cro} was increased from 1.5 to 2.5. As the heart rate was more or less constant while the blood pressure was increased, the cross-reliability led to a low reliability. As in the first experiment, the temporary breathing rate with a score of 2 or higher leads to short periods of slightly reduced reliability at around 200 s and 400 s.

6 Conclusion and Future Work

In this paper, we presented an SA-EWS system with a fuzzified reliability validation which recognizes erroneous vital signs caused by various measurement artifacts such as loose sensors, detached sensors or other interferences. In our experiments, the proposed system was successful in detecting such events and

decreased the data reliability during such events. This observation shows that self-awareness techniques such as the one proposed and used here can provide more robust EWS calculations. We note that deciding the value of parameters such as possible absolute values, signal slopes, and correlations among various vital signs demands domain knowledge. As the human body is an extremely complex system, not every phenomenon is already known. Therefore, although domain knowledge can be helpful for general cases, it does not replace personalized assessment which experts can provide each patient with. For this reason, we plan to add a learning module to the SA-EWS system which should learn about the patient's body functions and its basic health condition. In addition, more metrics should be generated and used, such as the derivation or the variability of a vital sign.

Acknowledgement. The authors wish to acknowledge the financial support by the Marie Curie Actions of the European Union's H2020 Programme.

References

1. WHO: Chronic diseases and health promotion. http://www.who.int/chp/en/. Accessed June 2017
2. Kyriacos, U.: Monitoring vital signs using early warning scoring systems: a review of the literature. J. Nurs. Manag. **19**(3), 311–330 (2011)
3. Morgan, R.J.M.: An early warning scoring system for detecting developing critical illness. Clin. Intensive Care **8**(2), 100 (1997)
4. Anzanpour, A., et al.: Internet of Things enabled in-home health monitoring system using early warning score. In: Proceedings of MobiHealth (2015)
5. Anzanpour, A., et al.: Self-awareness in remote health monitoring systems using wearable electronics. In: DATE Conference (2017)
6. Götzinger, M., Taherinejad, N., Rahmani, A.M., Liljeberg, P., Jantsch, A., Tenhunen, H.: Enhancing the early warning score system using data confidence. In: Perego, P., Andreoni, G., Rizzo, G. (eds.) MobiHealth 2016. LNICST, vol. 192, pp. 91–99. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58877-3_12
7. TaheriNejad, N., et al.: Comprehensive observation and its role in self-awareness; an emotion recognition system example. In: Proceedings of FedCSIS (2016)
8. Pasquier, M., et al.: Cooling rate of 9.4 °C in an hour in an avalanche victim. Resuscitation **93**, e17–e18 (2015)
9. Reule, S.: Heart rate and blood pressure: any possible implications for management of hypertension? Curr. Hypertens. Rep. **14**(6), 478–484 (2012)
10. Davies, P.: The relationship between body temperature, heart rate and respiratory rate in children. Emerg. Med. J. **26**(9), 641–643 (2009)
11. Zila, I., Calkovska, A.: Effects of elevated body temperature on control of breathing. Acta Medica Martiniana **2011**(Supp 1), 24–30 (2011)
12. Ross, T.J.: Fuzzy Logic with Engineering Applications. Wiley, New York (2009)
13. Guang, L.: Hierarchical agent monitoring design approach towards self-aware parallel soc. ACM Trans. Embed. Comput. Syst. **9**(3), 25 (2010)
14. Zephyr: Bioharness 3. www.zephyranywhere.com. Accessed June 2017
15. iHealth: iHealth BP5. www.ihealthlabs.com/blood-pressure-monitors/feel/. Accessed June 2017

16. iHealth: iHealth PO3. www.ihealthlabs.com/fitness-devices/wireless-pulse-oximeter/. Accessed June 2017
17. Maxim Integrated: DS18b20. www.maximintegrated.com/en/products/analog/sensors-and-sensor-interface/DS18B20.html. Accessed June 2017
18. ATMEL: Atmega328p. www.atmel.com/devices/atmega328p. Accessed June 2017
19. Nordic Semiconductor: nrf51822. www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF51822. Accessed June 2017
20. Song, H.S., et al.: The effects of specific respiratory rates on heart rate and heart rate variability. Appl. Psychophysiol. Biofeedback **28**(1), 13–23 (2003)

TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

Painosalama, Turku, Finland 2021

ISBN 978-951-29-8617-0 (PRINT)
ISBN 978-951-29-8618-7 (PDF)
ISSN 2736-9390 (Print)
ISSN 2736-9684 (Online)