Hierarchical Dynamic Goal Management for IoT Systems

Axel Jantsch[1], Arman Anzanpour[2], Hedyeh Kholerdi[1], Iman Azimi[2], Lydia C. Siafara[1], Amir M. Rahmani[1,3], Nima TaheriNejad[1], Pasi Liljeberg[2], and Nikil Dutt[3]

[1] TU Wien, Vienna, Austria

[2] University of Turku, Finland

[3] University of California, Irvine, USA

***Abstract*— As the Internet of Things (IoT) penetrates ever more application domains, many IoT-based systems are increasingly becoming more complex, versatile and resource-rich, and need to serve one or more applications with diverse and changing goals. These systems face new challenges in dynamic goal management due to a combination of limited shared resources, and multiple goals that may not only conflict with each other, but which may also change dynamically. We motivate the need for hierarchical, dynamic goal management for this class of complex IoT systems and substantiate our arguments with case studies from two application domains: patient health monitoring and Cyber-Physical Production Systems (CPPSs).**

I. INTRODUCTION

Goal management may be necessary if a system has to pursue more than one goal, and it is necessary in one of two situations: either the system has several *conflicting goals*, or it has several non-conflicting goals, but *limited resources* force it to prioritize some goals over others.

A. Resource management

Goal management is distinct from, but tightly intertwined with resource management. In many IoT based systems dynamic resource management is a hard problem, because the problem space is vast even if only few resources are considered, and because time and other resources are very limited for the management task itself. Consequently, dynamic resource management is almost always handled with heuristics because optimal solutions are elusive. As a case in point Rahmani *et al.* [21] consider the mapping problem in a many-core SoC, and show that researchers have proposed a variety of different heuristics, with none of them being superior in all cases. One heuristic maximizes the overall system throughput [11], one maximizes throughput under given thermal constraints [16], and a third maximizes the system's lifetime [10]. In IoT systems the problem is even harder because an SoC is only one among many components and due to their distributed nature which makes centralized solutions infeasible. Moreover, consider the diversity of resources that have to be managed: computing devices like CPUs and DSPs, interface HW like antennas and MAC modules, special purpose engines for encryption, video, image and audio processing, memories and buffers, etc. These resources are typically allocated to a task under tight constraints depending on the availability of *liquid resources* such as energy, power, bandwidth, computing time, storage capacity, etc. Liquid resources are either given as bounds not to to be crossed, or are optimization objectives to be minimized or maximized.

Goals are, explicitly or implicitly, derived from application requirements and are orchestrated through resource management strategies. Examples of application goals are maximization of task throughput, meeting real-time requirements, or minimization of power consumption. Very often goals are combinations of the above like meeting deadlines, maximizing throughput while minimizing power consumption. These combinations can be considered attempts to reconcile conflicting goals at design time by formulating heuristics that balance these goals. This avoids dynamic goal management but may still require dynamic resource management because the system may change state dynamically, e.g., tasks commence and end dynamically, resources are depleting, etc.

B. Dynamic Goal Management

Reconciling all application requirements into one balancing goal at design time avoids the run-time burden but may cause inefficient over-design, because it cannot take into account information only available during system operation.

If application requirements are diverse and poorly predictable, dynamic goal management is often preferable. For instance Roca *et al.* [22] describe a generic, fog based infrastructure that serves multiple IoT applications. Their targets are *Ultra Large Scale Systems (ULSS)* such as smart cities [1], and autonomous terrestrial, aerial or marine vehicles [9]. In these scenarios applications dynamically come and go and different applications have different goals. Dynamic goal management usually means that general principles are applied that guide the resolution of application goals that either are in conflict or that compete for shared resources [22]. For instance, such a principle may be to support applications to meet their respective goals while minimizing energy consumption and maximizing system lifetime. To facilitate the resource arbitration process, Aazam *et al.* [2] develop a resource estimation technique, that dynamically predicts the applications' need for resources to improve the resource allocation process.

The dynamics of applications is one motivation for dynamic goal management; another reason is the change of states during the lifetime of a system. Specifically, aging, wear out and the occurrence of permanent HW faults lead to decreasing performance and reduced availability of resources. Adapting goal and resource management strategies

to the lifetime phases can greatly improve the utility. For instance Haghbayan *et al.* [10] demonstrated a doubling of system lifetime due to a sensible core allocation algorithm that models aging effects.

C. Hierarchical Goal Management

Goals can be divided into sub-goals, which can be pursued in parallel or in sequence. E.g. For instance if a system has the overall goal to meet requirements for each incoming application, this overall goal is naturally broken down into sub-goals, one for each application, with a higher level goal manager to coordinate these subgoals.

Another break down of goals into sub-goals is based on the observation that goals can be achieved in different ways. For instance to obtain the position of a node, GPS coordinates can be requested or sequences of acceleration data can be tracked. Only one of these goals needs to be achieved, but we do not know in advance, which one can deliver the necessary information with the required precision or on time, both goals may be spawned. If both goals complete successfully, their results can be combined to obtain a precision or reliability, which is beyond the ability of each individual method.

In the domain of autonomous agents goal models and strategies for planning have been extensively studied, which we briefly outline in section II. We argue that this also becomes relevant for a class of IoT systems, that

- host and serve a set of applications which are diverse and are not known at design time; or
- have a number of shared resources and constraints on their usage; or
- there are choices on how to accomplish the applications' and the system's goals.

In Section III and IV we substantiate this claim with example cases from the domains of patient health monitoring and production systems. The examples illustrate the presence of various goals which are hierarchically organized, and which in some cases conflict with each other. The conflict arises either due to limited resources (in the patient monitoring case) or due to inherently conflicting local and global goals (in the distributed production system case).

II. Background and Related Work

Goals and hierarchical goal structures are key in driving problem solving, comprehension, and learning, making them essential for autonomous and self-aware systems [7]. In this context, *Goal-Following Autonomy* (GFA) is defined as hardware/software agents that can agree to take a goal from a user (or another entity) and can automatically accomplish the goal by carrying out a sequence of actions in its environment [7]. *Goal-Driven Autonomy* (GDA) goes beyond the abilities offered by GFA and is defined as a "reflective model of goal reasoning that controls the focus of an agent's planning activities by dynamically resolving unexpected discrepancies in the world state, which frequently arise when solving tasks in complex environments" [14]. *Goal Formulation* (GF) and *Goal Selection* (GS) are the essential components of GDA. GF is defined as the ability to generate new goals considering the discrepancies and explanations, while GS decides on which goals the system will pursue next. There are two prerequisites for these functions: *Discrepancy Detection* and *Explanation*; where the former is in charge of monitoring the system and the environment for anomalies, and the latter attempts to generate explanations for detected discrepancies [14].

Research on goal formulation and management has been done in the context of artificial agents [6], [18], [25] which mainly focus on providing the capability to nominate top-level goals, and managing the nominated goals by prioritizing them. However, requirements and restrictions of resource-constrained IoT devices necessitate customized, light-weight, and minimally conflicting approaches which consider the priority, significance, objectives and requirements of each application, while holistically coupling the overlapping and/or contradicting objectives of different applications to satisfy the system constraints.

There have been also attempts in the system-on-chip (SoC) resource management domain to exploit the concept of self-awareness [15] and its embodied goal management property through presenting light-weight autonomous resource management methods. In [8], Dutt *el al.* study the concept of self-awareness in SoCs and present the key properties of self-aware systems such as desirability scale, goals, purpose, expectations, etc. The Autonomic SoC platform (ASoC) [5] is an example of attempts in the smart SoC domain which is based on the organic computing paradigm, however, in this platform, the desirability scale and the goals are implicitly coded in the rules, and an explicit dynamic goal management mechanism is not deployed. The same shortcoming applies to the other platforms in the same domain such as CyberPhysical Systems-on-Chip (CPSoC) [23] and SElf-awarE Computing framework (SEEC) [13]. In CPSoC, the goal hierarchy and goal management is in a very primitive form and the desirability scale is implicitly encoded within the goals. In SEEC, the goal formulation and management is assigned to the application.

III. IoT-based Health Monitoring

As a first example to demonstrate the significance of dynamic goal management, we present IoT-based health monitoring systems, where a certain level of Quality of Experience (QoE) is required, posing dynamic conditions under heterogeneous constraints. These monitoring systems consist of three main layers: sensing layer, fog/edge layer [19] and cloud layer (see Figure 1) [20], for which various objectives such as energy, accuracy, or bandwidth should be met. However, such objectives are contradictory in certain circumstances due to limited shared resources. In the following, two conflicting goals at the sensor layers (i.e., fog-sensor cross-layer) and two conflicting goals at fog layer (fog-cloud cross-layer) are studied.

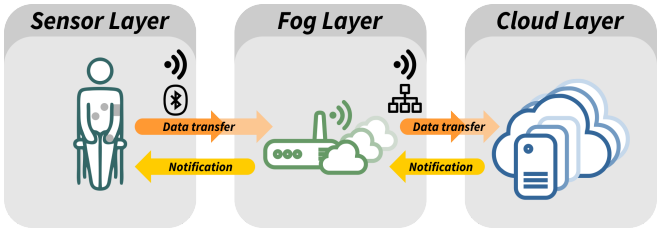

Fig. 1. IoT-based health monitoring architecture

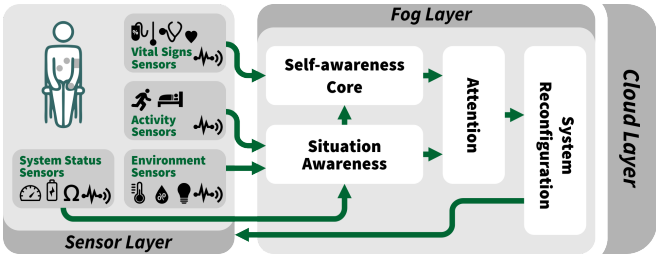

Fig. 2. Accuracy-aware IoT-based health monitoring

A. Resource Allocation at Sensing Layer (Fog-Sensor Interplay)

In remote patient monitoring systems (which are targeting mostly out-ot-hospital patients), the sensory part often consists of a wearable sensor network attached to the patient body for collecting and transmitting biomedical signals wirelessly. A gateway device in the fog layer receives patient's data and performs some local operations (e.g. filtering, fusion, compression, encryption, local storage/diagnosis/notification) and then transfers the necessary parts to the cloud server for long-term storage and processing. The limitations in size and weight in wearables necessitate these sensors to be powered by small and lightweight batteries which consequently limits the available energy at the sensor layer. In IoT-based remote patient monitoring, the continuity/frequency of monitoring, detection, and notification services depends on the available energy at the sensor layer, making sensing and transmission energy-efficiency a crucial challenge.

A prevalent solution for this problem is adjusting sensor device parameters (e.g., sampling rate, sensor selection, device sleep duration) according to patient/context/system situation. In such solutions, sensor device has several working modes where each mode has its own power consumption profile. The most power-hungry mode provides the highest accuracy and resolution for the recorded data, while other modes consume less power by lowering the sampling rate and/or putting the device into sleep mode for certain periods of time. The following are two approaches among the existing resource management approaches for managing the resources at the sensor layer:

1) Accuracy-aware Monitoring:

Problem and Objective: For out-of-hospital patients, the monitoring of health status is needed to be prioritized according to the significance of medical/activity events. The system needs to pay the highest attention to most adverse events and consider the highest priority for most efficient and accurate measurement mode in the sensor node. It should also meet the acceptable accuracy level when the patient is in the most medically stable situation.

Solution: A health monitoring system is proposed by Anzanpour *et. al.* [3] that uses a personalized model to tune system parameters to achieve a precise view of patient state. The self-awareness core is implemented in the fog layer to update an Attention core to prioritize the attention value according to patient health status and patient context/activity. In their solution, the Attention core shares more resources for highest accuracy when the patient is at higher emergency levels, and when the patient is in the safe and stable condition it updates system configuration to stay at minimum sampling frequency mode without compromising the accuracy (See Figure 2).

This system only considers the situation of the patient as a goal for enhancing the accuracy of the monitoring system, while other states of the system (e.g., current available battery/energy budget at sensor nodes) are not considered.

2) Energy/Availability-aware Monitoring:

Problem and Objective: Power failure in sensor nodes due to limited available energy resources causes a failure in the health monitoring system and possibly irreversible damages for risky patients. Several system-driven parameters should be enhanced while data is exchanged between the sensor layer and the fog layer. The system not only needs to consider the accuracy and precision of the monitoring process, but also should keep track of the available level of energy and accessibility to alternative energy sources in view. For instance, if the battery level of a sensor node is low and it is not possible to recharge/replace the battery for the next couple of hours, the system may choose to prioritize the availability of the service by compromising the frequency of data sampling and transmission (higher accuracy vs. longer availability of the service).

Solution: An energy/availability-aware monitoring solution can provide system-driven enhancements by paying more attention to system-related events and assigning a higher priority to these working modes which keep the system in the most stable state. The attention core updates system configuration for higher recording resolution and shortest system sleep time when the system has more energy at hand and closer to alternative energy sources. It also prioritizes battery endurance in the sensor node by switching the system to more low-power working modes when the battery is going to be exhausted or other energy sources are out of patient's reach.

The goal of such system would be increasing the duration of monitoring availability while the frequency/accuracy of health monitoring signals needs to be compromised.

B. Resource Allocation at Fog Layer (Fog-Cloud Interplay)

Bandwidth optimization and Quality of Service (QoS) guarantee are two fog-cloud cross-layer goals which may

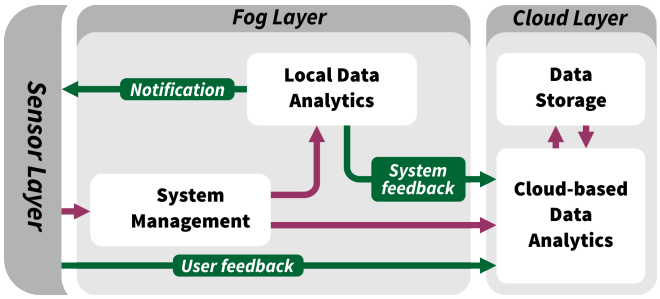

Fig. 3. Hierarchical IoT-based health monitoring system with cloud-fog bandwidth optimization

potentially conflict in certain circumstances due to limited shared resources.

1) Bandwidth Optimization:

Problem and Objective: Disconnection from cloud servers and bandwidth variations are prevalent issues in fog devices, interrupting continuous health monitoring applications. These problems cause negative impact in health monitoring applications where high communication latency may lead to irreversible damages for high-risk patients with time-sensitive demands. Moreover, transmission of raw data to the cloud burdens network resources and cloud storage capabilities. To address this issue, health analytics are proposed to be distributed into fog and cloud layers, enabling the fog layer to independently operate in case of low bandwidth in particular for mobile gateways with varying bandwidth availability. In this architecture, the fog layer needs to adaptively alleviate data transmission, while preserving accuracy and resolution of the system.

Solution: A hierarchical computing architecture for IoT-based health monitoring is proposed [4] to partition machine learning methods into fog and cloud computing resources, enabling fog-based data analytics in a standalone way (see Figure 3). Therefore, health applications operate acceptably in case of poor connection between the fog device and cloud server. Moreover, this hierarchical architecture provides a closed-loop technique to adaptively manage the data traffic based on patients conditions while the accuracy is preserved. As a result, the bandwidth is saved up to 83% by removing unnecessary data transmission.

2) Quality of Service (QoS) Guarantee:

Problem and Objective: In fog computing, certain computation tasks are offloaded to edge devices (e.g., smart gateways), providing a high-level of punctuality, reliability and availability. However, such a fog device faces issues in delivering a satisfactory QoS, when massive number of sensor nodes are assigned to them (e.g., several sensor nodes enter their coverage region due to their mobile nature). In such scenario, response time becomes highly unpredictable and decision making is significantly delayed. Therefore, there is a need for solutions to guarantee QoS in the fog layer by optimizing computational resources utilization, mitigating overloads, and subsequently minimizing response time.

Solution: An intelligent work allocation and load balancing algorithm can provide a tradeoff between fog and cloud resource utilization, considering the connected sensor nodes and data volume. As the number of connected users increases, the fog device decreases local computation activity and transfer the computations to the cloud layer to preserve an acceptable processing time. Consequently, the transmission rate is increased, transferring health analytics and decision making to the cloud servers. This algorithm guarantees a satisfactory QoS in the system as well as reducing latency and improving power characteristics in mobile fog devices.

There is a conflict between these two goals. The first solution minimizes data transmission between the fog devices and the cloud layer enabling local computation and preserving the accuracy in case of poor connection. However, the second goal guarantees QoS at the edge in case of numerous connected users, mitigating fog-based overloads by transferring data analytics to the cloud. This leads to a higher data transmission between the fog devices and the cloud server.

IV. Distributed Cyber-Physical Production Systems

The second example to highlight the need for dynamic resource management covers the use case of the Industrial Internet of Things. The growing dominance of the Internet has brought IoT to many industrial applications [17]. A CPPS, a specific type of Cyber-Physical Systems, is built upon the collaboration of software and physical components which uses data processing, information and communication technology as well as manufacturing technology to facilitate the process of production in a given industry. Distributed design solutions promise to overcome the challenges of centralized architectures and improve the robustness, flexibility and efficiency of the future CPPSs [12]. In a distributed manner, the concept of an Autonomous Cooperating Object (ACO) is defined as a computational core attached to a group of physical components, each of which has a legacy control unit. An ACO and its respective physical components altogether are called an entity which communicates with other entities in a given CPPS through the network connection [24].

Aside from enormous development in distributed CPPS, some challenges are still in progress [24]. The first challenge to name is self- and context-awareness for data acquisition and interpretation of local situation. With the increasing complexity of current and future CPPSs in both size and functionality, the diagnosis, prognosis, monitoring and self-maintenance are becoming more necessary to improve the robustness of the system and reduce downtime. Autonomous mitigation and decision making is another challenge when the complexity of the system is increasing and effective real time actions are required. The third challenge here is dynamic entity clustering and self-configuration which enables the current system to adapt to the changing environment and network and communication errors. All these challenges need to be addressed to manage the goals of the CPPS efficiently and reliably. In the rest of this section, we review some of the

example problems and how hierarchical goal management can resolve these issues optimally.

A. Resource Allocation

Entities which operate on the shop-floor interconnected with scalable IoT-based integration middleware carry information about themselves, monitor locally their environment and communicate with each other to run dynamic production processes. The large system is a community of self-configurable interacting ACOs which need to form clusters dynamically and allocate their resources pursuing their individual goals and the global goals of the system as defined by the Manufacturing Execution System (MES).

1) Autonomous anomaly mitigation:

Problem and Objective: Each individual ACO in the large system shall monitor and maintain its health status by diagnosing and mitigating anomalies. Examples of such anomalies are the wearing out of a physical component, the failure of an entity, changes in the speed of a entity, missing products, false positive product detection, and communication link failure. In case of a detected anomaly the ACO should find in real-time a policy to mitigate the problem, while minimizing the impact on the large performance of the large system. However, goals and available resources at the ACO with the anomaly may not be in-line with the goals of the overall system.

Solution: The ACO evaluates the priority of the problem periodically and decides whether it should allocate resources for solving it. The options of actions in the decision-making unit are represented hierarchically, which enables efficient problem-solving using reasoning in higher levels of abstraction. The ACO evaluates the high-level options of actions under the constraints which the global system goals set on its resource allocation. Once a compliant option is found, the ACO retrieves the necessary actions to take by looking at the lower levels of the action hierarchy.

2) Self-configuration:

Problem and Objective: For several reasons an ACO or a number of ACOs in the cluster may change their operational configurations which affects other ACOs. In this case, each ACO should be able to adapt to changes in its environment reallocating its resources in collaboration with other ACO to meet common system goals.

Solution: When an ACO plans an operational change, e.g., reducing its speed, it performs dependency check and passes this information to the other ACO of the cluster which are affected. However, this operational change may be in conflict with the individual goals of the other ACOs. In such a case, negotiation is initiated within the cluster to ensure the optimal operation (i.e., fulfilment of the goals) of the larger system through communication between the involved ACOs.

B. Conflicting Goals

Under some circumstances, the proposed CPPS is trapped between local and global conflicts in which the hierarchical goal management has been defined to improve the efficiency and robustness:

1) Decision Making:

Problem and Objective: The cognitive and decision making unit in an ACO attempts to use methods for matching existing actions to newly detected problems and their causes. Following this local goal sometimes reduces the safety of the ACO or the system if an unexpected action is taken.

Solution: In case of confusing in this situation, some action may be taken from the provided list of MES which is not in contradiction with other ACOs' operational goals. The Hierarchical Goal Management helps decision making unit to follow its local functionality goals, however it forces it to consider the global safety goals during the emergency situations.

2) Negotiation:

Problem and Objective: In case of an operational change, an ACO could independently make a decision suitable for its own efficiency as a local goal. The problem with this gaol fulfilment is the possibility of instability for other ACO(s) or the global system; for example, increase in the throughput of an entity may result in traffic jam of items in the next entity.

Solution: Hierarchical Goal Management necessitates the negotiation with other ACOs, in which a chain of negotiations from the negotiation-initializer ACO starts toward the potentially affected ACOs. Finally, the result of negotiation shows the condition in which the initializer ACO is allowed to operationally change. This type of management may reduce the efficiency of some ACOs, however it prevents the system from severe deficiency.

V. Summary, Vision and Research Directions

With the ubiquitous deployment of IoT systems across a wide range of application domains, we are increasingly frequently faced with challenges of managing multiple, and often conflicting, goals that change dynamically. Hence, contemporary approaches that addresses isolated goals are insufficient for this emerging class of IoT systems that must dynamically manage complex, competing goals while addressing heterogeneous models of IoT systems. Goal management for these systems therefore require new strategies, models, and algorithms to efficiently coordinate overlapping and conflicting goals, while meeting desired Quality of Experience (QoE) for the IoT system.

We believe a hierarchical goal management approach can manage conflicting and complementary goals to ensure a satisfactory QoE, handling IoT complexity and resources in sensing, fog and cloud layers. Such an approach performs a hierarchy of goals to plan an effective system execution, considering dynamic runtime changes for goal priorities and an optimization mechanism to evolve over time. The approach presented in the two use cases includes: 1) A predefined subset of goals and subgoals which are dynamically modified, created or removed. 2) A set of dynamic priorities indicating the importance of each goal at runtime. 3) An update period to periodically fine-tune priority of goals over time. 4) An inspection function to determine how the goals and subgoals are satisfied. 5) A goal function planning

the goals overtime with respect to inputs from inspection function, systems and users condition. The approach outlined in this paper is a step towards realizing hierarchical dynamic goal management for many complex IoT systems, and lays the foundation for active research to address this challenging problem.

ACKNOWLEDGEMENTS

We acknowledge financial support from the Marie Curie Actions of the European Union's H2020 Programme, the US National Science Foundation (NSF) through grant CNS-1702950, and that of Austrian Government through BMVIT/FFG under the action *ICT of the Future* in the project SAMBA (contract number: 855426).

REFERENCES

[1] Gerhard P. Hancke 1, Bruno de Carvalho e Silva, and Gerhard P. Hancke Jr. The role of advanced sensing in smart cities. *Sensors*, 13:393–425, 2013.

[2] Mohammad Aazam, Marx St-Hillaire, Chung-Horng Lung, Ioannis Lambaris, and Eui-Nam Huh. IoT resource estimation challenges and modeling in fog. In Amir Rahmani, Pasi Liljeberg, Jürgo-Sören Preden, and Axel Jantsch, editors, *Fog Computing in the Internet of Things*, chapter 2. Springer, 2018.

[3] A. Anzanpour, I. Azimi, M. Gotzinger, A. M. Rahmani, N. TaheriNejad, P. Liljeberg, A. Jantsch, and N. Dutt. Self-awareness in remote health monitoring systems using wearable electronics. In *Design, Automation Test in Europe Conference Exhibition, 2017*, pages 1056–1061, 2017.

[4] Iman Azimi, Arman Anzanpour, Amir M. Rahmani, Tapio Pahikkala, Marco Levorato, Pasi Liljeberg, and Nikil Dutt. HiCH: Hierarchical Fog-Assisted Computing Architecture for Healthcare IoT. *ACM Trans. Embed. Comput. Syst.*, 16(5):174:1–174:20, 2017.

[5] Abdelmajid Bouajila, Johannes Zeppenfeld, Walter Stechele, Andreas Bernauer, Oliver Bringmann, Wolfgang Rosenstiel, and Andreas Herkersdorf. Autonomic system on chip platform. In Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, editors, *Organic Computing - A Paradigm Shift for Complex Systems*, Autonomic Systems, chapter 4.7, pages 413–425. Birkhäuser, 2011.

[6] D. Choi. Reactive goal management in a cognitive architecture. *Cogn. Syst. Res.*, 12(3-4):293–308, 2011.

[7] Michael T. Cox. Goal-driven autonomy and question-based problem recognition. In *Second Annual Conference on Advances in Cognitive Systems*, pages 29–45, 2013.

[8] Nikil Dutt, Axel Jantsch, and Santanu Sarma. Toward Smart Embedded Systems: A Self-aware System-on-Chip (SoC) Perspective. *ACM Trans. Embed. Comput. Syst.*, 15(2):22:1–22:27, 2016.

[9] Mario Gerla, Eun-Kyu Lee, Giovanni Pau, and Uichin Lee. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In *IEEE World Forum on Internet of Things*, 2014.

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

[11] Mohammad-Hashem Haghbayan, Anil Kanduri, Amir-Mohammad Rahmani, Pasi Liljeberg, Axel Jantsch, and Hannu Tenhunen. MapPro: Proactive runtime mapping for dynamic workloads by quantifying ripple effect of applications on networks-on-chip. In *Proceedings of the International Symposium on Networks on Chip*, Vancouver, Canada, September 2015.

[12] M. Hermann, T. Pentek, and B. Otto. Design principles for industrie 4.0 scenarios. In *System Sciences (HICSS), 2016 49th Hawaii Int. Conf.*, pages 3928–3937. IEEE, 2016.

[13] Henry Hoffmann, Martina Maggio, Marco D Santambrogio, Alberto Leva, and Anant Agarwal. Seec: A framework for self-aware computing. Technical Report MIT-CSAIL-TR-2010-049, MIT, Cambrige, Massachusetts, October 2010.

[14] Ulit Jaidee, Héctor Muñoz Avila, and David W. Aha. Integrated learning for goal-driven autonomy. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, pages 2450–2455, 2011.

[15] Axel Jantsch, Nikil Dutt, and Amir M. Rahmani. Self-awareness in systems on chip – a survey. *IEEE Design Test*, 34(6):1–19, December 2017.

[16] Anil Kanduri, Mohammad-Hashem Haghbayan, Amir-Mohammad Rahmani, Pasi Liljeberg, Axel Jantsch, and Hannu Tenhunen. Dark silicon aware runtime mapping for many-core systems: A patterning approach. In *Proceedings of the International Conference on Computer Design (ICCD)*, pages 610–617, New York City, USA, October 2015.

[17] László Monostori. Cyber-physical production systems: Roots, expectations and R&D challenges. *Procedia CIRP*, 17:9–13, 2014.

[18] J Powell, Matt Molineaux, and David W Aha. Active and interactive discovery of goal selection knowledge. In *Proceedings of the Twenty-Fourth Florida Artificial Intelligence Research Society Conference*, 2011.

[19] A.M. Rahmani, P. Liljeberg, J. Preden, and A. Jantsch. *Fog Computing in the Internet of Things - Intelligence at the Edge*. Springer, 2017.

[20] Amir M. Rahmani, Tuan Nguyen Gia, Behailu Negash, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, and Pasi Liljeberg. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, 78(Part 2):641 – 658, 2018.

[21] Amir M. Rahmani, Axel Jantsch, and Nikil Dutt. HDGM: Hierarchical dynamic goal management for many-core resource allocation. *IEEE Embedded Systems letters*, 2017.

[22] Damian Roca, Rudolfo Milito, Mario Nemirovsky, and Mateo Valero. Tackling ultra large scale systems: Fog computing in support of hierarchical emergent behaviors. In *Fog Computing in the Internet of Things*, chapter 3. Springer, 2018.

[23] S. Sarma, N. Dutt, P. Gupta, N. Venkatasubramanian, and A. Nicolau. Cyberphysical-system-on-chip (CPSoC): A Self-aware MPSoC Paradigm with Cross-layer Virtual Sensing and Actuation. In *Proc. of the Design, Automation & Test in Europe Conference*, pages 625–628, 2015.

[24] Lydia C. Siafara, Hedyeh A. Kholerdi, Aleksey Bratukhin, Nima TaheriNejad, Alexander Wendt, Axel Jantsch, Albert Treytl, and Thilo Sauter. SAMBA: A self-aware health monitoring architecture for distributed industrial systems. In *The 43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017.

[25] M. Wilson, M. Molineaux, and D. W. Aha. Domain-independent heuristics for goal formulation. In *Proceedings of the Twenty-Sixth Artificial Intelligence Research Society Conference*, pages 160–165, 2013.