Monitoring Integration Systems and Visualization

Master of Science in Technology Thesis
University of Turku
Department of Computing
Software Engineering
2022
Simo Vuorinen

UNIVERSITY OF TURKU
Department of Computing

SIMO VUORINEN: Monitoring Integration Systems and Visualization

Master of Science in Technology Thesis , 72 p., 2 app. p.
Software Engineering
May 2022

The amount of new software and data has grown significantly in our society. In addition, new software solutions often have been built over the old ones to extend their capabilities. Also architectural solutions for software systems include multiple smaller services that have been divided all over the system. This has lead to the situation where more communication happens between the different parts of the system. Monitoring abundant traffic manually, with human resources is really challenging or even an impossible task. That is the reason why monitoring is performed with automated systems. The objective of the thesis is to develop an extension for monitoring system, that will be in charge of monitoring the integration platform. Thesis is done for the SaaS -software company BCB Medical Oy.
In the thesis solutions were sought for general monitoring challenges and integration platforms problems in monitoring. Retrieving integration platforms essential information and their visual presentation was one of these problems. In addition to these problems, the differences in monitoring between the integration part of the system and rest of the system, were evaluated. Also determining thresholds that help detecting any anomalies in the system, was considered an issue.
The research for in thesis was performed with qualitative methods, by interviewing the employees of BCB Medical Oy. Interviews were constructed with semi structural interview model and they were used to discover solutions for monitoring issues. Implementation part of the thesis was made with the same tools that the company uses for monitoring, Prometheus and Grafana.
Results of the work accomplished of determining values that are required from the integration platform. Transferring these values to the monitoring system was performed with using Prometheus exporter. Visualization of these values was done with Grafana to help discover important information and determining certain thresholds for alerts.

Keywords: software monitoring, integration monitoring, monitoring, Prometheus, Grafana, integration, anomaly detection, visualization

TURUN YLIOPISTO
Tietotekniikan laitos

Simo Vuorinen: Monitoring Integration Systems and Visualization

DiplomityöMaster of Science in Technology Thesis , 72 s., 2 liites.
Ohjelmistotekniikka
Toukokuu 2022

Uusien ohjelmistojen ja datan määrä on kasvanut yhteiskunnassamme merkittävästi. Lisäksi vanhojen sovellusten toiminnallisuutta on yritetty laajentaa uusien ratkaisujen avulla. Myös sovellusten arkkitehtuuriset ratkaisut ovat usein toteuttettu niin, että järjestelmä on jaettu useisiin osiin. Tämä on myös johtanut lisääntyneeseen kommunikointiin komponenttien välillä. Runsaasta liikenteestä järjestelmässä johtuen sovellusten manuaalinen monitorointi on ihmisresurssein erittäin haastavaa ellei jopa mahdotonta. Tästä johtuen monitorointia suoritetaan automaattisella monitorointi järjestelmällä. Työn tavoitteena on kehittää monitorointi järjestelmän osa, joka vastaa integraatioalustan monitoiroinnista. Lopputyö toteutettiin SaaS -ohjelmistoyritys BCB Medical Oy:lle.
Työssä etsittiin ratkaisuja monitoroinnin haasteisiin ja integraationalustan synnyttämiin ongelmiin monitoroinnissa. Erityisesti integraatioalustalta tarpeellisten tietojen hakemiseen ja niiden visuaaliseen esittämiseen etsittiin ratkaisua. Näiden ongelmien lisäksi koitettiin saada vastauksia integraatioalustan monitoroinnin eroavaisuuksiin tavalliseen monitorointiin verrattuna sekä miten määrittää integraatioalustan arvoille raja-arvo, jonka avulla havaitaan alustan ongelmat.
Tutkimus suoritettiin kvalitatiivisellä menetelmällä, haastattelemalla BCB Medical:n työntekijöitä. Haastattelut on muodostettu käyttäen puolistrukturoitua rakennetta ja niillä selvitettiin monitorointiin liittyviä ratkaisuja. Työosuus toteutet-tiin yrityksen käyttämillä monitoroitityökaluilla, joita olivat Prometheus ja Grafana.
Työn tuloksena saatiin määritettyä integraatioalustan monitoroinnissa välttämättömiä arvoja. Arvojen siirtäminen monitorointi alustalle tapahtui käyttämällä Prometheuksen exportteria. Sen lisäksi näiden arvojen esittäminen Grafana-ilmoitustaululla tärkeiden tietojen havaitsemisen sujuvoittamiseksi ja hälytysjärjestelmän arvojen määrittämiseksi.

Asiasanat: ohjelmistojen monitorointi, integraatioiden monitorointi, monitorointi, Prometheus, Grafana, integraatio, virheiden havaitseminen, visualisointi

Contents

1 Introduction **1**

1.1 Monitoring 1

1.2 Integration's State in Monitoring 3

1.3 Research Problem 3

1.4 Research Methods 4

1.5 Document Structure 5

2 Data Collection and Storage **6**

2.1 Monitoring Process 6

2.2 Monitoring Methods 7

2.2.1 Pull or Push 7

2.2.2 Hybrid Model 9

2.3 Monitored Values 10

2.3.1 Metrics 10

2.3.2 Events 10

2.3.3 Logs 11

2.4 Monitoring Tools 11

2.4.1 Prometheus 12

2.4.2 Riemann 13

2.5 Storing Data 14

2.5.1 Time Series Database 14
2.5.2 Search Engines 15
2.5.3 Database Management Systems 16

3 Integration **18**

3.1 Integration's Jobs and Usages 18
3.1.1 Functional Requirements 19
3.1.2 Non-Functional Requirements 20
3.2 Integration and Networks 21
3.2.1 Integration 22
3.2.2 Integration's Architecture 26
3.2.3 Health Level Seven and Integrating the Healthcare Enterprise 32
3.3 Integration Advantages 35

4 Analysis, Visualisation and Alerts **39**

4.1 Data Analytics 39
4.1.1 Analysis Techniques 41
4.2 Visualization 43
4.2.1 Visualization Tools 44
4.3 Alerting 48
4.3.1 Anomalies 49
4.3.2 Alert Types 51
4.3.3 Alerting Methods 54

5 Case Description **56**

5.1 Company's Background 56
5.2 Research Plan 58
5.2.1 Interview Plan 58
5.2.2 Implementation Plan 59

6 Results **61**

6.1 Interviews 61

6.2 Installing the Implementation 66

6.3 Application 67

6.3.1 Constructing Dashboards 67

6.3.2 Configuring Thresholds and Alerts 69

7 Conclusion **70**

7.1 Summary 70

7.2 Future Improvements 71

References **73**

Appendices

A Haastattelurunko **A-1**

B Interview Questions **B-1**

List of Figures

2.1 Prometheus's architecture, from Prometheus web page [9]. 13

3.1 Example of middleware tool for healthcare integration. 26

3.2 High level architecture model of MyKanta service 28

3.3 Figure of middleware solution describing path that the data takes after it is gathered from various information sources. 30

4.1 Graphite [47] 45

4.2 Kibana has great ability of visualizing geographical data [50]. 47

6.1 Different Grafana dashboard designs. 68

List of Tables

3.1 Table of HL7 messagetypes [35]. 32

1 Introduction

The amount of software systems has increased furiously in a last couple of decades. It leads us into a situation, where there is an excessive count of units that have to be controlled. On top of that, the structure of software and systems are much more advanced nowadays, which is making controlling them even more difficult. Simultaneously the demand of software's high quality and requirements has increased. Solution for these uprising problems is automatized monitoring, which is supporting services performance.

1.1 Monitoring

Monitoring, from technological perspective, is the tools and processes by which you administrate your information technology (IT) systems [1]. That is very simplified way to describe monitoring. Monitoring provides a translation between the metrics that your system generates and the business value. Monitoring service produces a measurable unit that can be evaluated in a similar way as user experience. Attained information can be used to detect flaws in software, systems or insufficient quality of service that is provided.

Monitoring has become a massive benefit in software maintenance. It is supporting the software development by informing about various problems. It can adapt in various different architectures and can be used differently depending on the target. Considering that the diversity might also come as a difficulty, because there is no

one right way to build a monitoring service. It should serve the business's demands and produce a usable output data that produces a value [1].

Monitoring can be completed with various manners, but all of them are really serving one idea. Use the available data to improve your service. It makes systems and services easier to understand and can sometimes find new solutions in existing problems. Data that has been gathered from a system can be anything. That is why, different data can have various use cases. Monitoring should be a trigger for actions. It is acting like a guide and pointing out directions where you should go. If monitoring system's provided information is not triggering into repairing actions, monitoring is a waste of time.

Collecting data from servers CPU, is one type of monitoring. Also keeping a track of logged users or reading the logs that the application is producing means monitoring. It is also analysing the collected metrics and trying to find solutions for occurring trivial problems. Monitoring is a complex concept with multiple subsections. It needs a lot of attention and it can have several different phases which are known as:

- collecting the data
- storing the data
- visualizing the data
- analysing the data
- alerting and recovering.

These phases can be handled differently. Either they are carried out by a single service or multiple ones [2]. Now days monitoring tends to be handled by multiple services. Distributing services are scaling well and they are easy to apply for popular applications [3].

1.2 Integration's State in Monitoring

Also use of integration has been increasing in technological evolution. That is made possible by accessibility of present application technology. Everything can be connected and used over the internet for example. The piece that connects two systems together is not so often monitored, but rather the systems themselves [4]. This might cause some problems when the systems are administered by different organizations.

The receiving system might be totally oblivious of what information the integration produces. This can lead in situations where the application is practically working, but the integration's proportion is messing up with application. Those problems are not always clearly visible in the receiving systems end. By monitoring integration, it should decrease the amount of errors that can occur in the applications. Risk of not noticing the errors in the system, that are causes of integration, is greater than detecting problems that are caused by the application.

As monitoring makes systems and software more reliable and transparent. Which means that the problems are easier to detect and handle. There is no reason why it should not be applied to the integration parts of software architecture. That should increase customer satisfaction and application's accountability.

1.3 Research Problem

This thesis is made in a collaboration with a company called BCB Medical Oy. BCB Medical is a firm that is gathering and analyzing clinical data [5]. That means wide amount of clinical data is passing through their systems. The main goal of this thesis is to build a part of the monitoring system that is supervising integration segments of the system's architecture. Present monitoring system is only capable of monitoring the applications of the company. This proportion of monitoring system is designed to solve specific problems that exist in the integration. Occurring

problems are defined from the perspective of support and integration teams. The integration team has knowledge and specific skills to comprehend these problems. Monitoring solutions should help the support team accomplish better customer satisfaction and increase performance of the integration platform. To achieve these objectives research questions have been defined as following:

- 1. How to build an effective monitoring service for integration platform?
- 2. What unique requirements integration has over normal systems for monitoring?
- 3. How to determine thresholds that will inform the integration's malfunctions?
- 4. How to visually present the state of the whole integration to users?

The main contribution of this thesis is to find integration parts that should be monitored and how to effectively bring the information about monitored anomalies to users. Monitoring is used as a tool that can solve these problems. Combining existing and new monitoring solutions will serve company's demands the best.

1.4 Research Methods

To get answers for first and second research questions literature of the subject is being viewed. It describes monitoring in multiple different ways and how the information is used and gathered. Also specific information about the integration is delivered with literature. Because the nature of monitoring is very versatile and can mean different things in different perspectives, there are a small number of scientific publications. Books that describe about designing monitoring system were used and also articles about the subject.

Third and fourth research questions are partially answered in the literature chapters, but there is also a organized a research for gathering more information. Research uses qualitative research method and there are four employees of BCB medical interviewed. Employees are chosen to be interviewed by their knowledge on the subject. Interviews' goals are to bring knowledge of how to detect problems in the system and bring the available information to the operators.

Interviews produce more efficient information of alert thresholds and visualizations, because the information that interviewees produce from the specific knowledge of problems. With qualitative methods, research gains more efficient information from less amount of interviewees.

1.5 Document Structure

The structure of this document is divided into three sections. First section includes background information and is based on the literature. This section contains Chapters 2,3 and 4. In Chapter 2, we go through the basic concepts of how the data is gathered from systems. Chapter 3 focuses on the integration's properties and their effects on that manner. Chapter 4 is going through the visualization and the detection part of the monitoring. By this section we are concluding the answer for the first research question.

The next section is about the research work of this thesis and it contains Chapters 5 and 6. It includes the interviews and their results documentation. Chapter 5 describes necessary information of the research methods and research's background. Results are analyzed in the next chapter, Chapter 6. Results are answering directly in the fourth research question and bring fundamental information for other ones.

Last section is for the conclusion. This section is answering the research questions and problems originated in the thesis. This section contains one chapter, Chapter 7.

2 Data Collection and Storage

This chapter describes the monitoring process's first phase, which is collecting data from different objects. After collecting data, it needs to be stored. This chapter is also handling the process of data storage. First section 2.1 in this chapter is focusing the monitoring in general. The second section 2.2 is introducing different methods in monitoring and data collection. Section 2.3 presents values that could be monitored from different instances. Then section 2.4 covers different tools that are used for monitoring. Last section 2.5 is focusing on giving overall picture of different data storage systems. Information is gathered from many sources, but prime one is the book "The Art Of Monitoring" by James Turnbull [1]. Additional information is brought from different books and articles about the subject.

2.1 Monitoring Process

The main task of monitoring is to get more value out of your systems or services, from the available information. This can be achieved by gathering data from applications. Gathering data seems trivial and it raises many complex questions. Decisions on, what parts of the application should be monitored or how is data transferred over to monitoring platform, are important. Nevertheless, there is not one answer for these problems, but rather there are plenty of solutions that can be used for system's advantage.

Speculation on what to monitor is a really challenging subject to face. There is

no certain rule on where to begin with this problem, but it has been learned that the progress should not start deep in the infrastructure. Rather you should start from the outside components and think what affects the users the most. One way of measuring service's efficiency is the key performance indicator (KPI). It measures how well the company is functioning on keeping the business healthy. This measure consists a lot of business related aspects. Sometimes it can mean a simple thing like customer happiness or more business related idea of how much profit the business makes with the part of the system. [2]

2.2 Monitoring Methods

We have already stated that there are multiple solutions of monitoring. Monitoring methods describe what ways there are to collect essential information from different parts of software systems. Also knowing advantages and disadvantages of these methods help constructing right monitoring model.

2.2.1 Pull or Push

Pull model and push models are two distinguished approaches for exchanging data between monitoring instance and the server. The pull model is based on request of data polling. The monitoring instance sends a data request to the server and then it handles the request and returns an answer. This practically means that monitoring service is pulling data from servers and it is always initiated by the monitoring service. The push model is converse and it is based on distributed paradigm. Monitoring service subscribes to the server and starts listening what the server pushes for it. Servers makes the push initiative and there needs to be specified rule on when the push action takes initiative. The model is basically telling us which one of the two system is making the initiative. [6]

Decision of what type of model of data acquisition you should use is trivial and totally dependent on the situation you are in. One proposal on the discussion is that the push model has more advantages than the pull model. The reasoning becomes from units that matter.

Host, services and applications work as the emitters sending the data to monitoring service. Data is distributed between these divisions and that is resulting in linear scalability. Emitters are sending the data when it is necessary [6]. Usually they are stateless and transfer happens when the data is generated. This improves the functionality of the application, because units can operate themselves rather than being forced to act in a way monitoring tools want. Also configurations of push emitters are already adapted in the application meaning the destination and messages are already usable. There might be a situation where the short-lived activity is immediately sent to the destination that could not been correctly processed with the pull method.

Additionally push methods also bring more secure way of transferring data. The emitters are more secure when they are not receiving any connections. This reduces the attack surface on our hosts, applications and services.

The main drawbacks of the push model are two things. To begin with the fact that push method can fill up the monitoring service with unwanted information [3]. Unwanted information is pushed to the monitoring tool, because of the emitter's configurations. There might be situations when the emitter is set to send monitored data in some sort of time periods. This might increase the pile of unusable data. The next problem is about transmission bandwidth. To minimizing communication and computational costs the push algorithms have been optimized to a level where they are used as little as possible.

2.2.2 Hybrid Model

Hybrid model means that application is not only utilising one of the pull and push methods. The agile way of designing software is making it more and more flexible and easier to fulfill [7]. That gives the opportunity to build a monitoring service with different ways of data transmits. With hybrid solutions you are trying to reach for the most optimal and convenient way to monitor systems. This is usually providing more stable data and cleaner outcome, but the costs of implementation are more.

Hybrid model can be used in lot of scenarios and can cover various situations. When using hybrid model the responsibility of monitoring is distributed among the system [3]. Basic ways of data collection are remaining the same. Distributed smaller services can push the wanted information upwards where the monitoring service is pulling the data.

The hybrid strategy revolves over two methods that are combined. It means that pushing the actions done are either event based or periodic based. Pulling the data is almost always periodic based. In hybrid strategy the periodic push happens in regular intervals but it has a requirement that system contains under a certain limit of operations. [3]

Alternatively, an event based is pushing the data towards the monitoring service only if there is a change is detected in particular entity. This will decrease the amount of metrics sent to the monitoring service if the values are unchanged. The adaptive push model is working in different probe that is collecting the data from the other parts of the system. It collects the data to the transportation window. The window is waiting for the updated data to be complete and then it is sent to the monitoring service. This type of data transfer is giving a better coherency between hosts and monitoring service. [3]

2.3 Monitored Values

2.3.1 Metrics

Metrics in a software are giving you a value of your software's state [2]. They can reflect different parts of the products infrastructure and give a very different amount of value. CPU (Central Processing Unit) status is usually transferred to the monitoring service. It gives the amount of application's CPU utilization. With these metrics it is possible to figure out if the application has crashed. Usually there are two types of metric data, counter and gauge. Counters are very easy to understand a growing value that can indicate amount of something [2]. It can be used to count for example usages of specific part of the application. Counters idea is to give out an ever-increasing metric. The counters do have a upper bound and usually they wrap around and start again from zero. Gauge in the other hand is point-in-time value [2]. Gauges don't tell you about previous or future values, but are giving you the value at the current time. Most of the metrics you are using are gauges. Also the weakness of the gauge of not memorizing the previous values is avoidable. Their data is usually saved in the database that works as a gauge's memory.

2.3.2 Events

Events are used to track the state of the applications environment [2]. Events are telling us about changes and occurrences in the system [8]. Events are extensively used to support debug and system integration activities. They reduce the amount of code developers must review for troubleshooting purposes. Events can provide information that provides a track on how the application is executed. This can produce information about problems or flaws that exist in the applications environment. Events are usually activated by the application so they are designed in the software. Building events afterwards is an unpleasant task.

2.3.3 Logs

Logs are a subset of events. They are most useful for fault diagnosis and problem investigation. Logs are essentially strings of text that provide us the information what's happening. They are often structured in a certain format and hold specific information. Simple logging system usually constructed by the event log file, log entry and logging mechanism. Logging mechanism is triggering the logger and it produces a log entry which is stored in the log file. Log file holds a series of entries that include data about the application's state.[8]

Usually timestamp is considered as one of the more valuable information included in the logs. Logs contain significantly more data than metrics and often need a bit of parsing before human can analyze them. Logs are great source of application's monitoring data and can often give more specific information about application's state than metrics. Logs also bring an efficient way to analyze distributed system simultaneously [8]. This might lead in a large set of logs, which means that they are harder to manually go through. It is important to structure your logs systematically for better reviews. In that way log entries can be filtered with timestamps, identifiers and log messages. That way it is easier to find errors and key values of specific problems.

2.4 Monitoring Tools

In this chapter the most functional monitoring tools are introduced. The tools introduced will focus on storing and handling data. Tools will have different features and are introduced as an example of various different monitoring tools. These tools are: Prometheus and Riemann. Prometheus is tool that is used by the company BCB Medical which is the initiator for this thesis. Riemann is alternative way and is introduced in James Turnbull's book "The art of Monitoring" [1].

2.4.1 Prometheus

Prometheus is an open-source, system's monitoring and alerting tool. Originally it is built by SoundCloud. Since 2012 it has been more and more popular and Prometheus has created a very active community around it. It is standalone project which is maintained independently by any companies. That is one of the reasons why it has an active developer and user bases. It also joined Cloud Native Computing Foundation in 2016 as the second hosted project, after Kubernetes.[9]

Prometheus basically takes a responsibility of gathering and handling metrics. Metrics are stored in a time series database and it means that metrics information are stored with a timestamp of recording time. It is established on the pull based method of gathering data from different systems. In order to get data that is needed for monitoring Prometheus needs an exporter that can be installed on various of services. Exporter is a service, that exports a small data interface, where the monitoring server can fetch different metrics [10]. Since the Prometheus community was rather large there are lots of ready made exporters [11]. These exporters are usually made for popular applications, for example MySql database, Mongo database, Linux servers and many more.

Main features of Prometheus are: multi-dimensional datamodel, which can be identified by metric's name and key pair. PromQL-language, pull based model over HTTP and a opportunity to also push time series to the server, targets discovered from static configuration or service discovery.

Occasionally there is a need to push metrics to the monitoring service. That is one of the reasons why there is a Pushgateway built in Prometheus, it is presented in the Prometheus architecture figure 2.1. Gateway allows you to push different jobs into the server and it is easy to instrument even shell scripts without client library. [11]

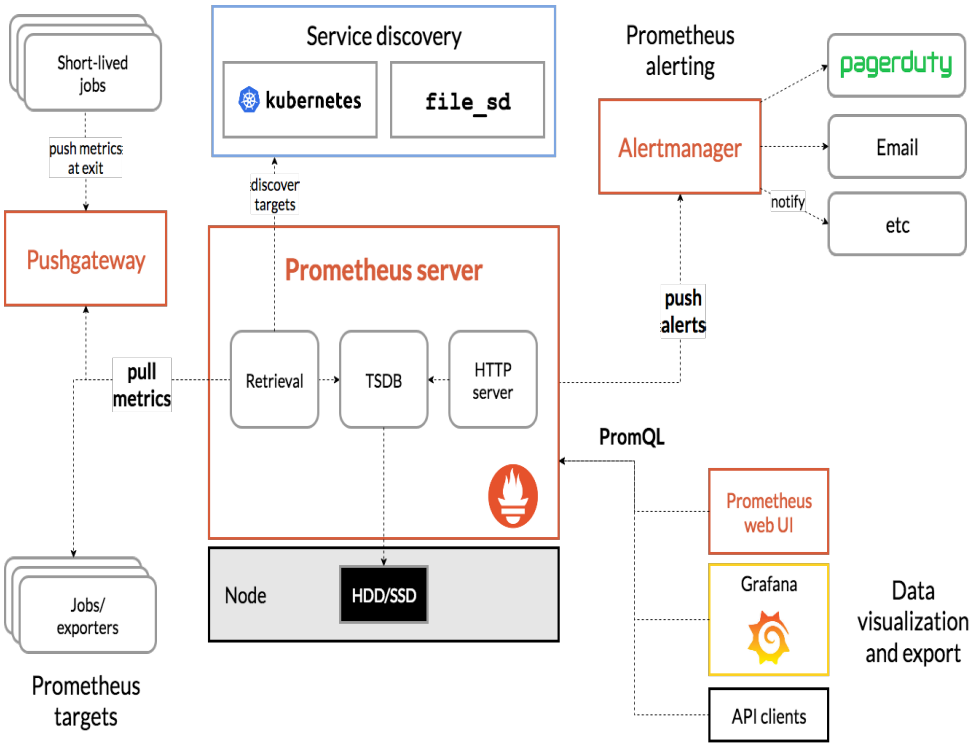


Figure 2.1: Prometheus's architecture, from Prometheus web page [9].

2.4.2 Riemann

Riemann is a monitoring tool that aggregates events from your servers and applications with powerful stream processing language. There the events can be summarized or analyzed for actions. Riemann's idea is to make measuring events and monitoring an easy norm. Riemann is an event-based monitoring tool for monitoring distributed systems. Opposite from Prometheus, Riemann works with a push model, and it is receiving the events rather than polling them from the other systems. This means that applications, services and hosts will send their event data into Riemann and then it will make decisions about those metrics or events. [12]

Riemann's benefits are that it uses streams which accept an event. Events are usually labelled as simple as ":host" and ":service", you can use built-in streams or build up your own. Streams are used for filtering altering and combining events. Its

configuration is a Clojure program and syntax is extendable, regular and concise. Configuration-as-code minimizes boilerplate and improves the flexibility of development in complex situations. It is good for trying to keep large and dynamic infrastructure running. In addition it also helps engineers who need to understand the source of errors or performance leaks in production environment.

Differences with Prometheus are mostly originated from the methodology of the application. Prometheus values reliability and is always viewing the statistics that are available even under failures. But if you need 100 percent accuracy such as billing, Riemann is more likely to be better, since Prometheus's data will not be detailed or complete enough. Traditionally monitoring systems run polling loops every detailed amount of time. Push model will activate immediately from the event and it is outperforming the polling method. You can get vision of anomalies much faster and also in the instant when they are fixed.

2.5 Storing Data

This section will describe the efficient ways of storing the data. More precisely it tells why and what to take in consideration when storing data that is used for monitoring.

2.5.1 Time Series Database

Time series database (TSDB) is a efficient way of storing data that needs to be monitored. TSDB is optimized for time series or time-stamped data. Time series data are incidents or measurements that are tracked, monitored, downsampled and aggregated in the long run [13]. This kind of data is usually meant to be analyzed and viewed. It could be server metrics applications performance, events or trades in a market [14]. TSDB is meant to be measuring change over time, that's why it

is optimized for time-stamped metrics.

TSDB is not a new thing, but there has been discovered new ways to use it. It primarily focused only on financial data. Now, when fundamental circumstances of computing have changed fiercely it has changed the need of monitoring. Everything outside and inside the company is transmitting a stream of metrics or time series data, and it is useless without using the right tools. [14] In addition TSDB was the fastest growing division of the database industry over the past year (2020-2021) [15].

TSDB have some key properties in architectural design that make them differ from other ordinary databases. These properties include data lifecycle management, summarization of data, ability to use wide time dependent scans and queries, that are aware of time series. With TSDB it is common to summarize data over different time periods. This requires going over a large set of data points to execute some computations. TSDBs are made for exactly this use and are giving milliseconds query answer times over the months of data. Also the data lifecycle differs from the normal databases. Every datapoint gets deleted after its period time is up. It means that data is aggregated and downsampled into long term trend data, so you don't lose the wanted information from longer amounts of time [14]. It rather means that the data farther in the history is a calculated mean from the values of that time.

2.5.2 Search Engines

Search-engine database is type of nonrelational database. It is dedicated for searching and data by the contents of data [16]. Search-engine database uses indexes for categorising similar characteristics in the data. This improves the search efficiency. Search engines are also optimized for dealing with the data that is not structured so well [17]. So they are capable of supporting full text search and complex expression searches, from the data that can be long and unstructured.

Popular search engine, for example like Google search, is not helping much in

monitoring application. Focusing more on the search engines that are designed for analysing logs. When maintaining large applications that are distributed across several nodes, your need of monitoring increases. Several entities emitting log data can produce tedious amount of lines of text waiting to be analysed. Search engines are good solution of handling logs from different origins.

One of the search engine tools that can be paired with Prometheus is Grafana Loki, that is built-in system in Grafana dashboards. Loki is not indexing the contents of logs, but it's labelling each log stream. It takes a unique approach by only indexing the log's metadata.[18]

2.5.3 Database Management Systems

Usually the monitored data is dropped when no longer needed or it has been aggregated. That suits well for the database models that are used. TSDB is the most efficient choice of database but we should not forget original solutions like normal databases flexibility of are used with database management systems (DBMS). DBMS term will cover a series of applications and database models. We are not trying to cover all the details that they have but rather focus on the pros and cons of them. DBMS is a tool that is used for retrieving, viewing and handling the database's data. Although there are many different applications accomplishing this goal, there are many different solutions and advantages on different setups.[19]

For example MySql is one of the popular databases. MySql is using variety of data tables, that are also modifiable [20]. It has multiple different user interfaces, lot of features and it is free of charge, but there is a possibility to upgrade into a paid version of it. It has great features and a simple structure which improves the usability [21]. On the downside you might need to do manually work on matters that are done automatically with other applications. Also there is a lack of support in XML or OLAP file types.

Another database model worth of mentioning is a MongoDB. MongoDB is also usable for free of charge and it can be used structured and unstructured data, which is not a feature many databases has. It is fast and easy to use and data is saved and accessed very quickly. Also database supports JSON file format which supports MongoDB's strengths [21]. Weakness for the database is that it is not supporting the SQL-language which is widely used in other databases. In MongoDB you use queries to retrieve data and there are also tools for helping to construct those queries.

These DBMS are providing different style of monitoring for your applications. They are essential for monitoring the data that is going through the applications. DBMS systems usually also bring information that is not straight used by the applications, but can show data like what modifications has been done and how many users are logged in. In DBMS systems you have to retrieve your data with queries and usually the wanted information is complication of the stored data. Results are flexible and can be complex but the queries are heavier and usually unique and require more work. Compared to the TSDB it is slower to search similar information, but it is more reliable.[19]

3 Integration

This chapter covers information about integration. The section 3.1 describes what integration does. Section 3.2 will present the integration in software level and it's architecture. Last section 3.3 defines advantages of integration.

3.1 Integration's Jobs and Usages

Today's businesses need to be agile. This means that changes are adapting your applications from every direction. Merges, acquisitions, new regulations, customer demands and competitive pressures [22]. Unfortunately applications are not designed to be flexible and are expensive to update. Advances in infrastructure have allowed a new breed of applications to be produced. One example of that is software as a service (SaaS). This has increased the benefits for customers and providers. Customers share same code bases and instances of the application but get more modifiable and unique product designed for them.

Integration is a key of success for any application. With integration it is possible build bridges between different parts of the application [23]. Data can be transferred and used in different applications, and for different purpose. The SaaS also brings the capability to scale [22]. It also can be available to many customers with different software solutions. This can also bring forth problems, because of the scalability the amount of data running through systems can be exponential. That amount of data requires more resources for monitoring.

Traditionally organizations implement information systems to solve internal problems. This has led into a situation with islands of information. Now with the advancement of technologies, companies see the need of integrate the legacy information in new technologies [24]. The most common form of integration might be data integration. Usually the situation is that users have access to many databases that are containing similar data, but the direct transfer of information between these databases is missing. The integration is not only a technical challenge because of the possibility that there are two different companies in commerce [25]. This issue might lead in fractions in the business side of the system also. Because of the integration's agile environment, it's also facing problems in technological side. Technological problems are usually about having availability of internal and external technologies. Also in some situations the updates in the legacy system or new technologies might affect the integration part of the application and it usually needs to be updated pretty commonly.

Most of these SaaS service subscribers have already applications deployed on their premises. This makes their company's application environment to become a hybrid model [22]. SaaS services are usually web applications which can be accessed through internet. It's also adapting normal web application's structure. It is composed by three major layers: business logic, data and user interface (UI) [25]. By other means SaaS applications work a little bit differently. Customers are usually paying by the usage of the application. Application quality of service should reach a level of Service Level Agreement (SLA) between service's provider and consumer.

3.1.1 Functional Requirements

SaaS services support certain business functions. However any business function cannot be separated from others. Therefore different services or applications of the company need to be integrated into working alongside each other [25]. The integra-

tion will take a part in three different levels of the application. User interface layer is wide but usually every application has its own users with individual interfaces and differing access control. Consumers could switch among these different user interfaces with different log in credentials. Also users coordinating through applications with one log on to the system. They use Single Sign On (SSO) and with that the application can grant all the services that are available to the user.

Then there is process part of the integration. Processes can be supported by another SaaS services or applications that work on-premises [25]. If applications want to have end to end process transactions many integration patterns can be used. Processes are capable of communicating through different instances. These linkages are formulating different patterns of which process activates certain event.

The data integration layer can be divided in to two parts, master data and transactional data. SaaS services synchronize and migrate the data from on-premise application or vice versa. The application environment should be connected into one data source that is master. The master data source should populate the data for applications and services timely. It means that the SaaS service should be synchronizing with the master data all the time. [25]

3.1.2 Non-Functional Requirements

Non-Functional Requirements (NFR) are often behind the surface [25]. Applications support similar type of usage as web services. That's how the SaaS environment can reach same benefits as the modern web applications. One good example of NFR is security and privacy aspect. In most occurrences SaaS subscribers' business data are centrally managed and stored by provider of SaaS environment. That means the data is separated over the Internet and it is hosted remotely. Business data still flows back and forth among the application and SaaS services. The integration technology should guarantee that the customers data is not open for attacks or accessed by any

other party.

Another advantage is earlier mentioned bill reporting and management. Because SaaS services charge by usage the costs of the service are easier to manage. Also different formats and deals from different packages are available. They also carry out the risks of the application.

Last one to mention of NFR is Quality of Service (QoS) reports. SLA is usually included in service's contract. SLA reports give performance indicator like information of applications availability. Most services provide these QoS reports frequently, however you should keep in mind that the reports are formulated from service providers point of view. It is possible that the customers also measure applications from their point of view and then can compare it with the provider. That's how customers can ensure that the SLA is fulfilled. [25]

3.2 Integration and Networks

This section will clarify how integration is connecting components in the system. It defines how parts of the application should or should not communicate to each other. When the components of the system are capable of transferring information to each other, there are distinct forms of the integration layer. This layer can affect the whole applications architecture and this section will describes some possible forms of the system's architecture. These examples are chosen from BCB Medical's point of view. Because they are from the same field and using similar technologies as BCB Medical, they are good examples to analyze. Healthcare in general is giving out good examples of integration since it is global, well researched and has practical problems.

3.2.1 Integration

Integration is a word that is commonly used in enterprise applications, technical papers, messages and even conversations. This word has multiple and misunderstood meanings [26]. Integration word is usually defined by the context of the matter at hand. That is why integration can mean different things, but it is defining similar capability. Generally speaking we can define integration as a technology that allows applications to pass information to each other that were not designed to work along [23]. Passing the information can also be done in many different ways and that is why there are many different types of integration possibilities. Each of them is transferring the information but there are different aspects to focus on. For information system it is also difficult to draw a line around application's properties and clarify the information's owner. As value chains extend beyond enterprises, they become a part of the other information systems [24]. They can be used a bit differently and that is why they can be distribute over multitude of heterogeneous systems.

Software integration is usually transferring information in multiple ways and can be required for multiple reasons. Common reasons that lead to integration are:

- Migrating from legacy system to a new system.
- Setting up a data warehouse between production system and the data storage system. There data goes through a process where it is extracted, transformed and loaded.
- Linking different systems to each other. They can be databases or file-based systems for example.
- Joining various standalone systems. This makes it easier to replicate processes and have uniform results. [27]

These reasons can be exploited by many different applications and processes [28]. The goal of the integration is to make one monolithic unit. This sometimes needs integration tools that can work as a middleware component between systems. It is capable of collecting and processing the information when needed [29].

Also the enterprise software systems can be categorized differently. These perspectives cover a large variety of technologies that need to be managed, but not every one of them, because the amount of different system architectures is infinite. Three different perspectives on integration that are considered are: monarchical integration, oligarchical integration and anarchical integration approaches. [28]

Reason behind using three perspectives as describing different integration conventions is two-fold. Software integration literature often seems to be divided into three categories and software integration is also concerned by agreements of the component developers, and there appears to be three different cases. The situation depends on what kind of an agreement has been made between developers. Integration between different components works usually by the rules of how the object-orient objects can be assessed. It is much secure than having components and processes directly addressing to another processes. Integration may sometimes work around these rules or there can be special assessments on how components communicate to each other or can they be sidestepped. [28]

Monarchical integration

Monarchical integration means that that the integrator has total control over the integration solution, in a same analogy as monarchical governance. That means the integrator is also the developer of the system components. Being the developer of the components, provides the integrator an uncontested option of integration solution. Integrator has to follow the rules of the platform developer, that can include things like hardware or operating system. Successful integration means that

the components can accomplish control and data transfer. This means that the other components can have access to the parts of the system they need or send data to them. Message passing has been implemented in many ways and that includes pipes, procedures, sockets etc. With processes and objects, there can be different triggers and regulations how the procedures are handled. Only thing that can affect on the execution of the processes are the developer and the platform. [28]

Oligargchical integration

In the oligargchical situation, integrators are multiple component developers. This means that two systems with different developers are explicitly considering constructing integration. This means that managing the transfer of data and control are decided upon by negotiation and agreement. When the agreement has been decided the systems can be developed to match agreed integration attributes. If connected to each other by network layer there might become issues if the other source is not available. Also in these situations the presentation of the data is important. This means that the data structure can have incompatibilities and lead to problems. This can also be fixed if not only the data is transferred but also the structure of the data. Then the integration ends have better chance of correctly interpreting the data. Data transferred like this has a potential to be affected by data compression and encryption problems. [28]

Anarchical Integration

In anarchical integration there are rarely standards that are generally agreed. Sometimes it is even good business decision to develop competing standards than adhere to the existing ones. Anarchial integration situation is well described as every man is for himself situation. The component developers have not agreed on anything and the integrator must accept the components by their rules. Components are also usu-

ally considered as independent systems in their own, before the ideas of integration existed. Sometimes businesses have a separately accessible database. In integration the databases can allow the access to data but not necessarily the functionalities of the application.

Anarchial integration solution is the unfortunate one with most of the problems. Easiest way to access the application's data or functionality is to employ interfaces to the application. With interfaces you can have a restricted or complete access to system. If the interface needs to be transformed, use of the adapters might help. These adapters can produce data format modifications for example. Some other cases for getting access to the application are scraping and manipulating applications interfaces, providing software access to it. A final choice could be modifications of the source code, to allow access to the application. [28]

For managing these functions there are a solutions like middleware tools that can operate between two systems. It is a new software layer that can transfer data to different end points. It usually performs two main functions, connection and translation. Connection to native databases is typically done with database adapters and the translations are correcting the data. With the middleware we can reach out from the situation of anarchical integration to monarchical integration, simply by passing the messages between processes. [28] The advantages of middleware are great and it is great solution for transferring information between systems. It can also have logical functions of how to handle messages and even forwarding the messages to the right destination [23]. One problem of using middleware is the loss of semantics for services. There is not too much surveillance on the middleware or the functions that it has performed.

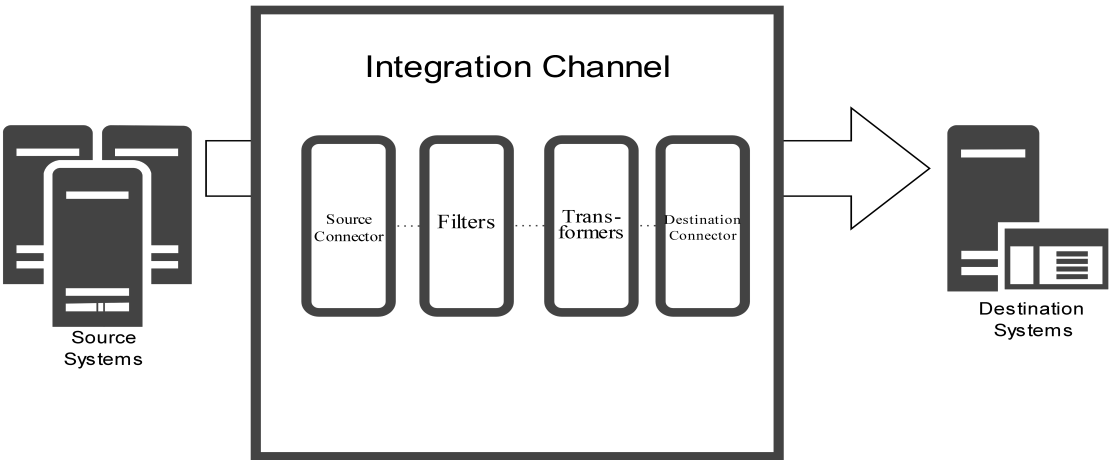


Figure 3.1: Example of middleware tool for healthcare integration.

Middleware should have some kind of anomaly detection that can detect critical errors. Because middleware systems now are more complicated and have different parts it is important to know how to recognize possible problems. It will lead into a better performing application. Also it's important to have a monitoring system because the integration part is often the link between systems, that are scaling. [26] Sudden changes in either systems might be critical from integration's point of view. Automated monitoring would solve problems where integration is set between different parties and their communication is not fluent.

3.2.2 Integration's Architecture

Integration in medical field is quite common. There are ready made infrastructural software that work, but tend to be missing some aspects. Connecting new applications is handled with middleware integration platform. Medical data is gathered from the patient and gets send to the integration platform. Each set up differs for each client, but the integration level is often much more similar. When the data has been received it usually needs some preprocessing before distributing it to the applications. That phase consists usually cleaning unwanted information and analysing the destination and the type of data integration is handling.[30]

The architectural view is dependent on what kind of level of integration has been planned and what is the purpose of the application that is utilizing. One of the integration examples in healthcare's digitization could be an platform, where patients can take a look of their prescriptions and their own personal record. It can improve the patients process to examine their own patient record and access the prescriptions with much less of a work. Only few countries have done a nationwide use of these environments, including Finland. In Finland the application is known as MyKanta (OmaKanta) and it is easily accessible with patients with smart devices [31]. The application is used by patients, but also the pharmacists are able to connect it and get the patients' prescriptions over it.

From the integration's point of view we can treat the whole MyKanta application as a one big integration platform. As how the platform works, it shares the patient's electronic health records with three different types of services, shown in Figure 3.2. [31]

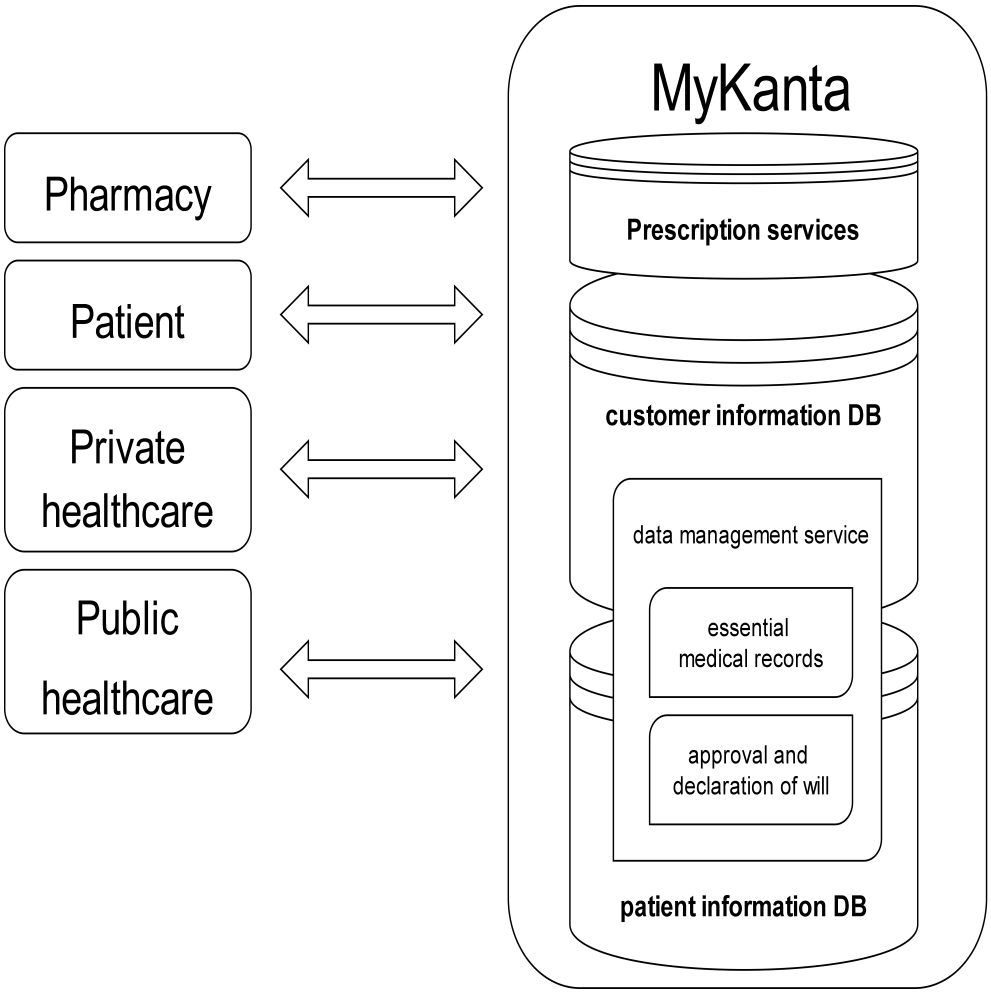


Figure 3.2: High level architecture model of MyKanta service

Healthcare institutions can share their examination results and update patients' medical records to MyKanta. With patient's approval other institutions are able to use the information that has been gathered from the patient. This helps the healthcare professionals to give the patient right diagnosis and treatments if patient is changing the facility where he has been examined. Also patients prescriptions will register to the MyKanta services. This will make the paper copies less dependent and pharmacists can access the prescribed medicines without a physical copy of the prescription. [31]

In addition the MyKanta also gives patients access for viewing their own elec-

tronic health records. Records can be categorized differently and patients can see for example their lab results, information of their visits and prescribed medicines. Service will let the patients to be more aware on their condition and improve the overall information flow. Couple of weaknesses exist in the service and they are anticipated. The oldest users that are not familiar with the digitization are usually the ones that struggle with the use of the application. Also some of the users that rarely have needed the healthcare services, have no knowledge of the application. [31]

Although integration can be usable application, in most cases it is an invisible tool between the visible application and the information systems. One already mentioned case would be when the data exists but it has not been designed to be used in a certain way. Creating new system that can utilize the data, needs a bridge between the data source and application. [23]

Automated integration can connect multiple systems and format data to application's needs. In healthcare there are systems that use the same existing data that has been saved, but for different usages. This also allows new parties to take a part in healthcare business. Hospitals do not need to redesign or manage new software, but can collaborate with third party firms that are specialized in other tasks.

Third parties need data from the main application, to produce services and products around it. Integration layer is the key for transferring information from one source to another. Transfers can happen from various events or they can be called in different time spans. Architecture depends on the applications needs and capabilities. Integration solutions that are well constructed and planned are easier to maintain. This means systematic procedures like receiving and sending information should be easily accessible. [23]

Third parties get the data from multiple sources with middleware. In the Figure 3.3 there are even mobile devices and Internet of Things (IoT) equipment where the

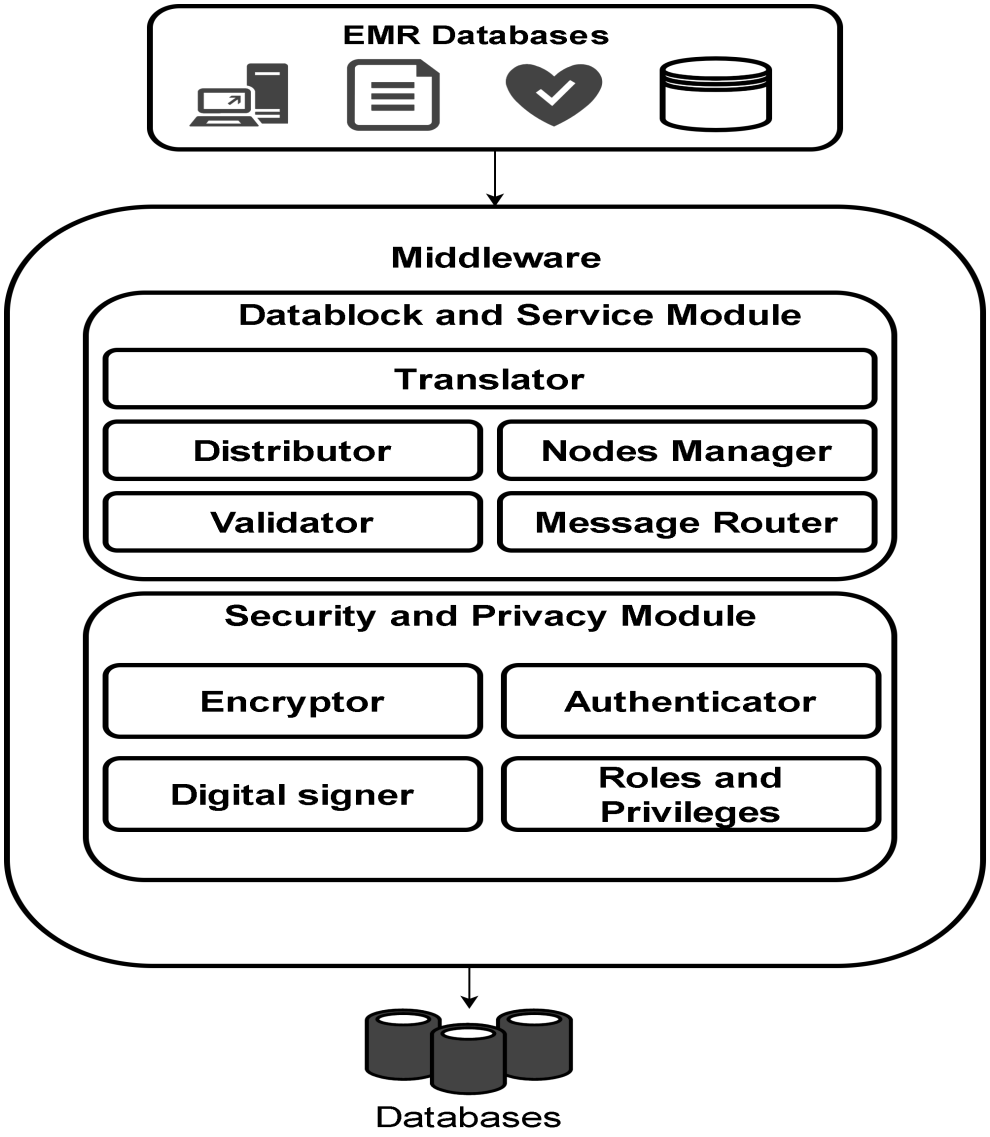


Figure 3.3: Figure of middleware solution describing path that the data takes after it is gathered from various information sources.

data is collected [32]. Then the data is transferred trough the integration layer which is usually capable of doing some sort of deduction. Other part of the integration layer that we have not talked about is the security aspect. In medical field the data is confidential and often protected very precisely.

Data could be encrypted so the external factors can't interfere with the information. Even hijacked message wouldn't give the attackers any delicate information if it is correctly encrypted [33]. Authenticators and digital signers are keeping the unwanted events away. With authentication only the people that have access should be able to use product. There are also definitions of what roles and privileges the users have [28]. This helps to limit users access and rights. Integration connects a large set of systems together and you shouldn't give all users to access and ability modify everything.

When the message has been handled by the integration's logic, it should be sent to the destination. How the messages are handled is up to the system, but usually the message is saved to the database and possibly triggering events of the application. Integration's structure is straightforward, but it should also be flexible. If the systems are inactive or crowded, there might be need to delay the messages functionalities in the integration level. This would be better resource management for the whole system.

There is set some kind of rules and regulations for the better understanding of the data that moves between applications [30]. The most frequently used standards are Health Level 7 (HL7), openEHR and ISO/EN 136063. Electronic Health Records (EHR) defines these protocols, on how to digitally store and exchange patient's data. Other types of standards are set by terminologies. They are used to establish common understanding of vocabulary and concepts [32].

3.2.3 Health Level Seven and Integrating the Healthcare Enterprise

Health Level Seven

HL7 provides a framework and standards for exchange, sharing, retrieval and integration of electronic health information. These standards determine how information is transported between different parties and applications. [30] They can define how the information is packaged or communicated. Also setting the language, structure and type of data is essential for achieving a seamless link between systems. These standards are supporting clinical practice and management and also making the information recognized and available for evaluations. [34]

HL7 consists of several message types. Each has its own unique purpose and also message contents. All of the message types are presented at the table below. Each of the message type holds a purpose where and why it should be used. The message types are often divided into subsections, describing the precise job of the message.

Message Type	Description
ADT	Admit, Discharge and Transfer
ORM	Order Entry
ORU	Observation Result
MDM	Medical Document Management
DFT	Detailed Financial Transactions
BAR	Billing Account Record
SIU	Scheduling Information Unsolicited
RDS	Pharmacy/treatment Dispense
RDE	Pharmacy/Treatment Encoded Order
ACK	Acknowledgement Message

Table 3.1: Table of HL7 messagetypes [35].

Message type's description gives us some detail on what kind of information the message consists. Although messages are handling different fields they contain much similar information and are constructed with a similar structure [36]. Messages are formed from different amount of segments. One similarity in each message is that the message starts with message header segment. This segment has data about system sending the information, receiving system and their location and necessary information of the messages metadata. For example like trigger of message and time stamps. [35]

Other segments usually contain the data of the information that is transferred between systems. This may vary very much for each message types. Messages can use common segments or have unique ones and that is totally dependent on the information. That is also a sign of a good structure of forming the messages. If there are similar objects they can use similar segments. [35] Frequently used common segment is a Patient Identification (PID) segment. That is pretty obvious when transferring medical data, where most of the information needs to be united with patient. [34]

Also these already mentioned message subsections are quite useful. They are headed with one specific task, that is describing the whole purpose of the message. Knowing that, deciding what data we want to pick from the message or how information should be parsed has been made a lot of easier.[35] Also when knowing the headline we can decide what function the message should trigger. These message subsections could be for example: Patient information update, Patient discharge and Patient registration [34]. These are classic types of Admit Discharge and Transfer (ADT) messages. ADT message type is used to communicate visit information, patient statistics and patient's current state in healthcare facility. They are often the most used and have the highest volume of all HL7 message types [34]. There is example of an ADT message below and there you can get a good view of how

message is constructed.

Listing 1 HL7 Message: ADT

```
MSH|^~\&|MESA_ADT|XYZ_ADMITTING|iFW|ZYX_HOSPITAL|||ADT^A04|103102|P|2.4||||||||
EVN||200007010800||||200007010800
PID|||583295^^^ADT1||DOE^JANE||19610615|M-||2106-3|123 MAIN STREET^^GREENSBORO^NC^27401-1020|GL|(91
NK1|1|BATES^RONALD^L|SPO|||||20011105
PV1||E||||||5101^NELL^FREDERICK^P^^DR||||||||||||V1295^^^ADT1|||||||||||||||||||||||||200007010800|
PV2|||^ABDOMINAL PAIN
OBX|1|HD|SR Instance UID||1.123456.2.2000.31.2.1||||||F||||||
AL1|1||^PENICILLIN||PRODUCES HIVES~RASH
AL1|2||^CAT DANDER
DG1|001|I9|1550|MAL NEO LIVER, PRIMARY|19880501103005|F||
PR1|2234|M11|111^CODE151|COMMON PROCEDURES|198809081123
ROL|45^RECORDER^ROLE MASTER LIST|AD|CP|KATE^SMITH^ELLEN|199505011201
GT1|1122|1519|BILL^GATES^A
IN1|001|A357|1234|BCMD|||||132987
IN2|ID1551001|SSN12345678
```

Example of ADT message [35].

Let's present some of the message segments, that can hold valuable information. Patient Visit (PV1) segment has information that is related to patient or any visit specific details. These details can be attending doctor, servicing facility and visit ID. Also similar segment Patient Visit - Additional info (PV2), exist to give more detailed information about the visit. Additional information can be such as diagnosis of the patient. This information is shared with Diagnosis Information (DG1) segment, and usually they are related to each other. Usually only one of these fields is mandatory, to get the patients diagnosis information included into message. Diagnosis information consists of diagnosis code of certain disease, some other data about the disease, patient's symptoms, abnormalities or complaints. Procedure (PR1) segment includes valuable information about numerous procedures that can be done to patient. Also Role (ROL) segment is usually used, because it includes valuable data about the personnel that occur in the event. These can be additions, updates or removals of roles in the patient record. [35]

Integrating the Healthcare Enterprise

Integrating the Healthcare Enterprise (IHE) is an initiative started by healthcare professionals and healthcare industry to improve how computers and systems transfer the information. IHE promotes the coordinated use of standards like HL7 and Digital Imaging and Communications in Medicine (DICOM). IHE takes a bit wider aspect on the integration than HL7. HL7 is more about the information's structure and how to transfer messages in the system. IHE is giving more guidelines on what information is valid and how to share that between the system, and it is more upper stage of the integration. [37]

IHE is trying to improve the integration of the healthcare. Setting some standards as how to share documents to patients or what information should be shared with patients. These kind of decisions need more healthcare professionals and researchers that can conclude boundaries for the systems. They have set certain profiles to focus specific parts of the healthcare systems and there are over ten different profiles. Each own set contains technical framework of documents in close coordination of with other IHE profiles. These cover some topics for example, cardiology, dental, patient care coordination and pharmacy. Also providing precise definitions of how these standards can be implemented to meet specific clinical needs.[37]

3.3 Integration Advantages

Integration is a sign of advance. It means that you are trying to reach a better performance or expand your service. Integration platform having a major role in the data transmissions, emphasizes the importance of supervision in integration level. With the current technologies and trends it is quite probable that the amount of data moving between applications is increasing. [33] This is fundamentally going to increase the need of monitoring the flowing data. Also the healthcare field is

bringing the challenges to the table. There are many standards and regulations that need to be reached. These are often concerned in security and the validity [27].

Integration's advantages are complex. They are the software's next level of evolution and are a necessity for data to move between ontologies. Advantage and new information that can be gained when fusing the data is real. Different data formats are bringing challenge for sharing information and integration is a way to solve that problem. Integration is taking the advantage of existing systems and it is extending systems lifetime. [29] But one of the biggest advantages of the integration is the ability to expand. New technologies are coming and the integration is a tool that can connect the new and old. Obvious facts are that the fast development of technologies is still going and nobody can know what kind of technologies there will be in the future [27].

Integration's best capabilities are to transfer information to the destination. This enables a fluent chain link between two systems. The integration platform can usually also convert the data into wanted format [29]. This would improve the applications usability by having better understanding about the incoming data. It is important to keep data's native state unchanged [29]. Although we would not need all the meta data that is available, there might be windows or usages for it in the future. That is why it is important that the integration doesn't modify the messages too much.

Although integration solves problems on how to construct the data it is an extensive and still existing problem. Problem occurs every time there is new development or new additions to the system. This means that as the product system is upgrading it might need to be managed in the integration part also. There is as many ways to construct the messages as there are developers. That means each class can be labelled differently. Since healthcare industry is quite an extensive field there are standards and regulations that are making this better [38]. It is helping when

constructing the messages and the data structure in databases.

Communicating between the system is necessary and that is what integration is made possible. There are still gaps between ontologies in different systems [29]. That means that data can be in other formats or excess the boundaries of originally planned classes. Between these entities integration is a solution that can deliver information to the same destinations. This can give access to totally new kind of implementations that were not even considered back in the days. In healthcare industry this might be seen as implementation on how the data is gathered from the patients. Patients often have some kind of measurements and surveys that can analyze their well being. Now it is more accessible and it can be combined with different data entities.

Integration is only a part of the system and should not be forgotten after it has been developed. Integration's attributes are very vulnerable for changes. This means that there are lots of values and attributes that can be changed or used differently. That leads us to the fact that integration platform must be maintained and it adds a load to the software's maintenance. [27]

Platform's management can mean different things. In this context it contains platform application-, network and communication-, environment- and data management. Integration platform stats like uptime, performance monitoring and other values can be used in the business management. There is no single tool to provide complete analysis on the integration framework and parts of it can mean many things. The manners of the integration tasks that are trying to connect two links together, means that if one of the parts is modified it might lead into necessary modifications into integration platform. [27]

That leads us in the situation where there has to be some kind of surveillance on the integration platform. Because the amount of data can be exponential, it can be objectionable or even impossible for developers to find anomalies in the platform

[26]. That is why we should try to monitor the integration part of the systems. It takes some management time from the overall work, so it should be done efficiently and by making some kind of alarms that can inform about errors. [27]

By monitoring the integration part of the system, we are trying to gain more control and knowledge about the data. Especially finding aberrations and errors in the integration. The obvious errors that can be found are the problems occurring in the integration platform. This usually means that the message is constructed wrong, or it's handled poorly. These are common type of flaws that are happening in the integration level. Monitoring would not necessarily solve the problem, but if detected, information on the problem is gained and possible predictions can be made to deny any errors in the future. [27]

4 Analysis, Visualisation and Alerts

This chapter covers the analysis, visualisation and alerting topics. These are phases that are close to user in monitoring process. In 4.1 section we are focusing on the data analysis and how the information is extracted from data. In 4.2 section visualisation process is introduced in order to produce visually functional data. In 4.3 section focus is on alerting and what kind of rules apply to alerts.

4.1 Data Analytics

Data analytics is the final stage in generating the comprehensions from data, that is gathered by system. The goals of analytics are finding useful information, making suggestions and helping in decision-making through a series of processes. Processes affecting the data can be: cleaning, transforming, inspecting or remodelling [27]. There are multiple distinct analytical techniques which can help in finding insights from data.

Monitoring data, that is gathered from software applications, tells about the state of the software. State tells what is happening in the software and can present different outcomes and processes that the software has. Exceptions in the data, are the main insights of the software bug triggers. Usually the storage of monitoring data is constructed mainly from log files generated by the system. It is great improvement if monitoring application can simultaneously analyze logs of the system and the parameters of the software defects. Also the interaction and influences between

those two is hard to measure and analyze, but essential for gaining more knowledge about system.[39]

The data itself often has some kind of aberrations. These are often detected when analysing the data. There are rules that define how data is formed. These rules can point out data anomalies or help cleaning up the data. In healthcare software system that is managed through rule-based error detection, rules have been gathered from public sources. The basic idea is that the data that is problematic is detected by rules [40]. Similar thinking could be applied when analysing the monitoring data. Errors could be found by detecting abnormal events with the monitoring system.

Rules can also be added, deleted and modified [40]. Situations where data is processed through many components, is sometimes hard to say where the problem existed. These problems and their causes can be analysed with machine learning, pattern recognition and statistics. One way to detect the errors would be with association rule. It generates associations between items in the database. With associations you can logically conclude a pattern that might lead into error. This is one type of pattern recognition technique and it searches typical patterns that are happening when error occurs. This way it is possible to detect what flaws affected to the occurring error and possibly even predict error happening when the patterns are recognized. [41]

The association rule will be more efficient if the systems are distributed, complex and heterogeneous [40]. This makes the monitoring more troubled and it is harder to recognize connections of different components and objects. It is harder to generate and express the connections, but the algorithms can detect them. These rules can work well when monitoring the whole system. Finding certain objects that will lead into problems in the integration stage is more important than solving a singular event. Also the rule can be set to find certain associations in the integration's messages and possibly detect actions that will lead in some kind of problems in the

integration. [41]

4.1.1 Analysis Techniques

There are multiple different techniques on generating useful information out of the data. These techniques can variate from traditional business intelligence to more refined solutions like time series analysis with machine learning and data mining [42]. There is also a possibility to combine these techniques with the help of IoT and cloud computing. Also the use of these techniques is dependent on the data and the results that you are expecting. Best approach would usually be using multiple techniques for allowing you data analysis to be more flexible and having more capabilities [27]. To have long term successful impact, some of the analytical capabilities should be self-learning.

Different types of data need different types of approaches from the analytics. In many occasions data is heterogeneous and need to be handled differently.

Qualitative and quantitative data analytic approaches differ from each other. Qualitative analytics is gaining an understanding of the underlying reasons. It is not statistical data so it can't be measured as easy. It is information about qualities of the data, so it can't be analysed as numbers. Qualitative research is used to get more understanding about uncovered trends and thoughts from the data. More traditional way of analysing is to use quantitative methods. It means that it can detect attributes from numerical data or data that can be transformed as statistics [27]. In quantitative approach the data is statistical and algorithmic. That means the data can be measured and compared. Most of the data analysis is quantitative data. Quantitative examination works better if there is big data set that needs to be analyzed. Qualitative needs more resources and it is better when examination needs to get more results that can't be discovered from the numbers alone. [43]

Another way of diving in the data analytics is separating analytics into three

categories: descriptive, predictive and prescriptive. These three categories form a compact analysis model. Descriptive case describes the history of accumulated data. It is used for learning about past behaviors and processes.

Predictive case tries to predict the future. Predictive thinking controls the actions what you are doing with the descriptive data. This might need of applying statistic models and machine learning algorithms to find this kind of results. This type of analysis is widely used when analysing business and financial side of the company. Also it is important for error detecting and predict failures in ongoing systems.

Prescriptive case thinks one step further than predictive case. It tries to analyze on what happens if you follow the predictive analysis. It is focusing on why these results happen and rather than only thinking the outcome, it tries to think reasoning behind it. These analytics are relatively complex and usually measure the risk of possible actions.

All of these analytic approaches for analyzing the data are giving some insights and ideas on how data should be approached. The point to be taken for these ideas is that you should build a process that is agile and versatile. The more diverse your analytic model is, easier it is to solve complex problems. Also if there is a need to extend the analysis project it would be better if ready made solutions would be flexible and allow that. Analytical techniques will give answers and improve your analysis, but building a good analytical process needs understanding of different approaches and techniques. It also means that understanding the data, and how to work with it is a major part of the process. Real world situations always have errors and flaws, so even the best analytical tools can't prevent that, but it is possible of noticing them.

4.2 Visualization

Visualizing data is very powerful analytic and interpretive technique and amazing learning tool. However metrics and their visualizations are sometimes tricky to construe. The idea of visualizing the data is that data can be clearly shown [44]. Highlighting the abnormal substances for better representation and detection. Also visuality makes improvement for recognizing meaningful patterns.[1] One way that will accomplish delivering critical information of the data with only a glance is a dashboard. Demands of providing accessibility, productivity and visibility are key factors of designing a dashboard.

Dashboard should also include reliable and relevant information, this is accomplished with conducting data filtering, sorting the relevant information and grouping data. Then it should have conducted a clear picture of information that is easily understandable and readable. Information should have some sort of triggering points that can lead into actions. Dashboards interfaces should display the information with various shapes such as diagrams, reports and visual indicators usually combining dynamic and relevant information [45]. Also there are numerous studies showing that dashboards improve learning [46].

Visualizing dashboard is not an easy task. To generate an effective visualization which considers the goal and shows characteristics of the given information, user has to be an expert. Expertise is needed in both fields, visual presentations and knowledge of data. Otherwise produced visualizations can be misguiding and lead to wrong conclusions. Good visualizations can clearly illustrate points. It should be obvious to see what value means and the purpose why it is there. Also the dashboard should be tailored for the audience and the analyzers. It means that users of the dashboard should understand how the dashboard is working and for whom it is intended. Sometimes technical observations don't fully support the business side of the company. [46] Also the presentation should be accessible for comparison and

present subjects without misleading users. [44]

4.2.1 Visualization Tools

In this subsection some of the popular open-source data visualization tools are presented. Observing applications strengths and qualities and comparing them to each other. Visualization tools have much similarities, but when choosing the right one, you should try to pick the one that serves your needs.

Graphite

Graphite is simple and fast monitoring solution which is serving two purposes. It stores numeric time-series data and renders graphs of this data on demand. Graphite does not collect data, but it is possible to send data to Graphite. It is licensed under Apache 2.0 license and it is written in Python. Graphite is an enterprise-scale monitoring tool that can run on cheap hardware. It started as a side project but now many large companies are using it. [47]

Graphite is constructed from three components: carbon, whisper and graphite web application. Carbon is daemon process that is listening for time-series data. It is the one that receives the data from connectors. Whisper is a simple database for storing time-series data. Graphite webapp is the application that can render graphs on demand. The architecture is shown in the figure

Figure 4.1: Graphite [47]

One of the advantages of Graphite is its own specialized database library whisper. It is very similar to round-robin database model, but it is customized for Graphite's needs. Graphite is more flexible with the incoming data and does not need a regular incoming data streams. It performs better when measuring uncommon requests that need to be handled with the specific time they happen. Also whispers allows inserting multiple values into database at the time. It's compacting them into single write operation and this improves performance tremendously. It is true that some of the database models have been fixing this issue, but at the time whisper was more efficient choice. Also Graphite is easily scalable. It can scale from front- and backend, meaning that simply adding more computational power will improve its throughput. Graphite performs a lot of tiny operations on lots of different files and that is happening for a reason. Graphite is storing each distinct metric received into its own database file. This sounds like it would be a very inefficient. Graphite is

handling the high volume of metrics in other way than adding more hardware. By its ability to combine the write operations it's saving a lot of needed computational power.

Grafana

Grafana is a data visualization software which started as a front-end application for Graphite. It lets users to create dynamic dashboards with multiple visualization options. It supports mixed data sources, customizable alert functions, annotations and it is possible to extend it with hundreds of plugins. Grafana's advantage is that it is versatile. It can support many different data sources and plugins. This makes the application compliant with multiple applications and it supports monitoring of multiple divided systems. Metrics can be analyzed and visualized in different dashboards or in the same one. Customization of the dashboards is made very easy and there are multiple ready made solutions for popular applications.[48]

Grafana's diverse use has also flaws. It is a bit of an overkill for creating simple visualizations. In addition, visual solutions don't offer as good customization options and it is not the best for creating visual images of certain information. Grafana is sometimes used with other monitoring tools. It is a solution for tools that lack of visualization power or alerts. Grafana has a supporting team and community behind it. That is why it is so compatible with other systems and can be used for monitoring diverse technologies. The flexibility and versatility give Grafana a cutting edge between other visualization tools. It can also be used fully on cloud, but with limited use, or it needs to be paid. [48]

Kibana

Kibana is also free and open user interface that will let you visualize your monitoring data. Kibana is a visualization tool designed for Elasticsearch and is part of

formerly known ELK stack entity. Kibana is a little bit different from two of the first visualizations tools. Because Kibana is closely connected to Elasticsearch it can handle log data in Elasticsearch clusters. Users can search the indexed of log data from Elasticsearch for specific strings or events. That way the root analysis and diagnostics can perform better. Also Kibana is capable of pretty effective data analysis and visualization. The advantage among other applications comes from effective log analysis and visualization of geographical maps. [49]

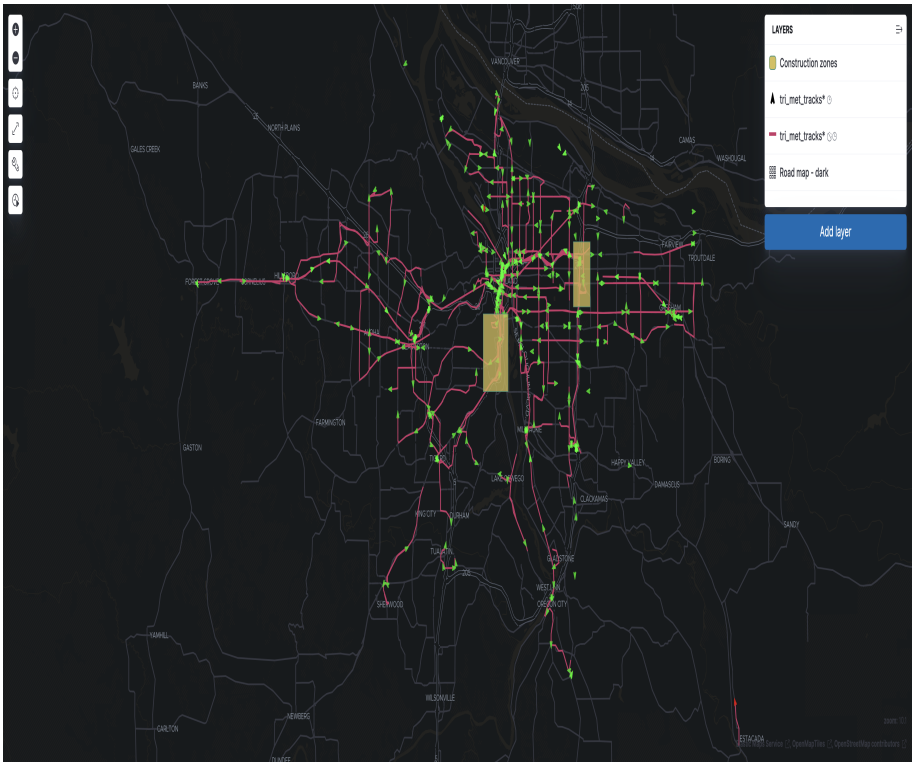


Figure 4.2: Kibana has great ability of visualizing geographical data [50].

Grafana with plugins is also capable of working with logs, but not as efficient as Kibana and Elasticsearch. Also the flaw of the tool is that limitation of one data source Elasticsearch. Although it is possible to transfer applications data into ELK stack with other applications, if wanted. That is the only way to use Kibana with other sources.

The ELK stack has a similar architecture as Graphite, where the parts of the

monitoring system are: Elasticsearch, Kibana, Logstash and Beats. Elasticsearch can save schemaless data and search it efficiently. Logstash is datapipeline that can receive and send data to other sources. Beats is a software with a single purpose of sending data to Logstash server from individual servers. Kibana is the visualization tool that can analyze and make graphs and presentations from the data in Elasticsearch. Usually ELK stack is used to analyze the system logs. Elk stack is managed by Amazon Web Services (AWS) although they allow maintenance from other sources. One reason of Kibana's success is that it is widely used by the AWS systems. It is also pleasant to use without experience in complex information systems. [50]

4.3 Alerting

Alert is a notification of unexpected information. It often signifies about danger or malfunction. Alerts are part of the monitoring process and append when there is a need of action. They are also eliminating the human operator watching the system's state continuously. Alerts signify that there is a problem in a system that needs human interaction for reverting back to normal state. Alerting process is a two phased with detecting and informing activities. Detection is a triggering effect of informing. It is determined state of malfunctions that needs to be informed. Informing is basically sending a right level of stimulation to the right destination. Managing these two activities can construct plenty of different forms of alerts. Most of the information is gathered from book "Effective Monitoring and Alerting" by Slawek Ligus [43].

4.3.1 Anomalies

In monitoring, data points are changing all the time, but some of the time they are significant to be pointed as anomalies. Anomalies can be defined as data points that have broken defined value. These values can be direction, duration, magnitude of deviation and progression. Dips and spikes differ in direction. They are usually short-lived so the values will normalize in over time, that means there is no progression. Elevation and depression also differ in direction but they take more data points to recover. That is why they might reveal different patterns of progression. With anomalies it is important to understand the failure and how it is impacting the system. Businesses can measure the cost of each failure on variety of ways for example: time, effort, quality and money. Every failure will affect negatively to the system with different impact.

Failures can be divided in 4 sections:

- Resource unavailability
- Software problems
- Misconfigurations
- Hardware defects

Each of the failure types can be a worth of an alert. These different sections can have a different administrator. That might affect on how and where the alerting will happen. Alerting the people who are managing failure's originate is as important as alert itself. Alert should be triggered for the reason that it should be handled quickly. Choosing the destination wisely for immediate actions will be more efficient.

Resource unavailability is by far the most common problem in production systems. Network outage and resource saturation causes a lot of increase in response times. In some unusual situations they can be responsible for complete denial of service. Some events that can cause this type of behavior in the system:

- Exceptionally high input rates
- Cyber attacks
- Partial system maintenance
- Wide range of operator errors

These events causes will usually lead up to saturation. The process is not immediate and with right kind of detection these flaws could be noticed and handled correctly. if these actions are not performed the problems will end up failing the system slow and hard.

Software problems are also common and have wide range of failures that can happen. Faults can vary from simple software bugs to dependency and architectural problems. Bugs are always present, but the likelihood and criticality depend on different conditions like: high complexity, premature releases, bad quality and design. Detection of bugs also varies much. Some of the bugs will be detected immediately and others show up after long running times. Usually when the bug is detected, they can be fixed in short time, dependent on the impact that the bug has. These problems will always need someone to fix them and software repairs should always be included when designing a software. Good deployment protocol would have possibilities to roll deployed software backwards and forwards.

Misconfigurations are one type of human errors. These are errors that are hard to detect programmatically. These problems usually don't get caught on testing and have unfortunate consequences. Users of the application usually detect these problems before administrators.

Hardware problems are more rare than other bugs, but they are harder to deal with. These problems have to be fixed physically. Hardware faults are usually observed less in smaller systems but in larger applications it comes mandatory.

Usually hardware problems cause complete device shutdown or delays in operations. Some of them can be detected from the logs that the system emits.

Anomalies will append in the system, but the problem is how alerting will handle them. The benefits of alerts can disappear if they are not used correctly. This means that alerts shouldn't be occurring too often. This would turn recipients numb to notifications and probably just ignore or tune them out. Every anomaly should not trigger the alerts. That is a very delicate matter to determine which type of actions will be directed into alerts. Each anomaly can be measured by its impact and system's recoverability.

Also the amount of alerts is not only definition to take seriously. Contents of the alert are just as important than the alert itself. Triggering metric itself is not enough content for alert. If we have added monitoring to our system we can expect that there are multiple things handled. This means that metric can't tell you enough critical information if you can't specify the problem. Usually bringing out the problem's context with the problem is necessary. Good rule to remember when designing the alerts is to focus on the symptoms [10]. Symptoms reveal the real problem in the system and metrics should just help to clarify it. When sending the alert you should be sending the information of metrics that are associated with the problem. Also metadata of the problem should be included like, has it happened before, how long it has occurred and everything else to make that can make alert more useful.

4.3.2 Alert Types

There are many ways to detect and process different anomalies. In "Effective Monitoring and Alerting" book [43], Slawek Ligus lists nine anomaly types that are ranked based impact to system, from the most severe to least:

1. Critical

2. Urgent

3. Intervention necessary
4. Recoverable
5. Inactionable
6. Automation Tasks
7. Early Indicators
8. Optimization Tasks
9. Nonissues

Critical events are sudden with impact so severe they are blocking users access to the application, or reduce systems usability severely. If these critical conditions emerge it increase system's downtime and it results to unrecoverable losses either in productivity or revenue.

Urgent anomalies will lead in partial loss of availability. It means that the system is partially unresponsive or portion of users is unable to use the system. These events require quick actions for minimizing the impact to system. Also stopping the anomaly before it reaches the critical point is necessary.

Intervention necessary type of events require intervention. It prevents them from escalating into critical events. Events are not immediately catastrophic but without repairing actions their impact will increase.

Recoverable event has a negative impact on the system but it is not crucial. These anomalies will need to be taken care in a certain time-frame. There is no need of alerts for these issues, but it is important to detect and keep track of them.

Inactionable events are anomalies that will have small impact for small fraction of users and they immediately disappear. High recover speed usually leads into situations where the operator doesn't respond quickly enough. That means it would

not be effective to have alerts on them. These anomalies are also important to track, for keeping an eye of the system's performance.

Automation Task anomalies will not have direct impact on the system, but have a potential to be causes of other problems.

Early Indicator events have little impact on system with partial group of the users. Momentary resource saturation or small bugs are causing these flaws. This might indicate that the system's performance decreases when there is more users and diverse usage.

Optimization Tasks events that are caused by poor architectural structure or lack software optimization. These anomalies have no effect on the systems functionality but are causing inefficiencies in resource utilization. Nonissue events that come up rarely, have no impact and are observable in any systems operations.

It's suggested that the first four groups [1.-4.] require alerting. These groups can have severe impact on the system or they could advance into that situation. Intervention is necessary by the administrator or else there occurs substantial damages. Alerts of different anomaly types are different and alerts should serve the best solution for individual events. That's why alerts can be configured in many different ways. Alerts can be defined in variety of ways, but have two main goals to achieve: accuracy of detection and speed.

The type of alert, that is triggered, is dependent on the anomaly's criticality. More critical alerts need more rough actions. That is why sacrificing accuracy over the speed of the alert is more favorable. A couple of false positive alerts is not too big of a deal for maintain system's performance. If the impact of negative events decreases so does the need of fast recovery of the system decrease. This means that the existence of abnormal actions should be recorded and alerted but with more confidence. The less severe problems system has the less is needed to make recovering actions to the system. That is why there are no needs to alert or at least

immediately inform the operator. Nevertheless it is needed to keep on track of the anomalies. Because even the small issues can transform into bigger ones overtime. Monitoring and recording the issues is good behavior in managing software systems.

4.3.3 Alerting Methods

Multiple different alerting methods exist because diversity of alerts. There are various ways to trigger alerts and to send them. Triggering happens with already configured threshold values. Monitoring systems keep track of multiple metrics and are producing data where those values can be compared and analyzed more closely. Easiest way to determine alerts is define limits that values can't cross. Occasionally that is enough of setting one static value for triggering alerts. More than often it is more situational and better habit to set threshold values over behaviour and dimensions of the system. Then it is equivalent for specific situations and can bring better information of the anomaly.

The only logic of alerts should not be threshold values of the system. There can be multiple different situations that can't be controlled by just evaluations of certain values. Sometimes it is the lack of values can tell more about the system's state. That is why it is important to have good knowledge of the monitored data so you can make efficient conclusions of it. Also because in a complex monitoring system there are multiple parts of application that have effects into each other, so it is needed to set up logic to the linking metrics as well. Effective alerting means that is if some part of the system fails and leads up to other parts to falling down too, there shouldn't be alerts on each part of the one linkage. Rather than having multiple alerts, having a one alert that can construe a reason or origin of the failure is more efficient. Logic should analyze metrics in both ways. That means of either suppressing the alerts or combining the alerts of different values.

The way system informs the operator, of each fraction of the system, can differ.

System should detect the anomaly and have logic to determine what type of problem this is, also the severity can have effect on where to send notification. With that it can be determined the message type and destination. Most used alert type is emailing, for its reliability, costs and speed. Wide distribution and unitary use of emails is an advantage. Another excellent way of reaching the operators quickly is through a phone. That's usually done with sms or a phone call. By the mobile nature of phones it is more likely to get a connection to administrators.

The evolution of technology is nowadays enabling many ways of notification. All of these alert types can be received with multiple devices. Calls might have a little bit of advantage, because there is a need of action to settle the alert. Also there are many other ways to send notifications these days. There are several applications where you can configure how the alert will go off. To apply more stimulus in the alarm you should get more quick attention to the problem. The one mentioned problem itself in alarming sounds and flashing lights is the lack of information it produces. It's good way to catch attention, but they are rarely used because there is no real need of flashy alarms. Also the alerts can be accumulative and escalate if the problem is not handled or the alert has not been acknowledged.

5 Case Description

In this chapter there are basic information concluded of the research. In Section 5.1 background information of the company is given. This section helps reader to understand on what kind of grounds the monitoring application has been built. In next Section 5.2 plan of the interviews and implementation is provided. The results of these are presented in the next Chapter 6

5.1 Company's Background

BCB is a company that brings hospitals software solutions that can help improve their treatment methods and decisions. These services are developed with SaaS technology where hospitals can use different applications over the internet. By providing the service to the customers, BCB has made agreements of level of availability. This SLA is defined with customers and different solutions have a certain limit of inactivity allowed. Company has also policies and agreements on how and what information is transferred between the systems. There are different technology solutions depending on each customer's situation.

Most of BCB Medical's applications can be constructed with three proportions: Integration Platform, Databases and Applications. These proportions work tightly together and construct services that are useful to the end users. Every application has its unique functionality and customers can have different decisions and values configured in their own application. The amount of applications and the diverse way

of using them is causing requirement on monitoring. Monitoring with only human contribution would become too harsh.

Integration Platform

The integration platform (IPla) is the part that is transferring information between applications and hospitals' patient health records. In the platform patient's treatment, procedure and analysis information is moved to right destinations depending on the data that they contain. IPla can also filter and transform information, this will help the platform produce and pick right values. IPla is also capable of transferring other information: Patient's personal details, patient's laboratory results, patient's visit details and context, patient's medication information and details about users, for example login data and job title. IPla transfers many messages daily and the total amount of messages has exceeded over several hundred million messages.

Databases

Databases are pretty self-explanatory. Databases can receive data from either applications or the integration platform. They contain all the data that we use in the applications in medical specific architecture, linking information between each other. BCB uses two database solutions: MySQL and MongoDB.

Applications

Applications provided by BCB Medical are mostly Disease-Specific Registries (DSR). These registries include diseases from different medical branches, for example cardiology, orthopedics and neurology. DSR will show information about the patient's symptoms, treatment and healing. It is a good tool for professionals to follow up results of their treatment. Also they can analyze the effectiveness and quality of treatment and analyze medical data that is being saved into DSR. DSRs are easy to

use and they can bring new high quality information about different diseases. The DSR is still not a tool that will replace the original patient records in the hospital. It's more of an updated tool on top of the old system that is being used in examining patients and produce better analytics. There are some applications that don't fit under the DSR title, but are extremely useful and supplement these applications.

BCB has already installed monitoring system for applications and databases. Monitoring and alerting are happening based on the metrics that these systems emit. The integration platform is somewhat unmonitored and it is only controlled with the middleware tool, that is used with human contribution. Building effective monitoring should cover all fractions of the systems and it's important not to forget the integration platforms monitoring.

5.2 Research Plan

This section express the plan for research. Plan is divided into interview and implementation subsections. The practical workflow is getting the needed information and description of desired monitoring application. Using that information for implementing the monitoring solution and then having a review of that solution.

5.2.1 Interview Plan

Research method that is used in this thesis is qualitative research method. Qualitative method works better than quantitative because of the small size of the subject group. Qualitative method will take more time to collect the data and analyze it, but the amount of data and its reliability are better. [51]

Collecting data is done with interviews and interviewees are chosen from the company based on their knowledge on integration platform. Also picking interviewees from different branches in the company will ensure that we are approaching the

problem with multiple perspectives. Interviews are conducted in pairs (two interviewees are interviewed simultaneously) and in semi-structured form. Semi-structured form means that interviews are generally following more of conversational flow. That means that the structure of the interview is already constructed but the conversation around the topics is free and can generate new questions [51]. Analysis of the semi-structured interview data is done with qualitative analysis methods. By pairing up the interviewees, I'm trying to gain more advantages of the interviews and decrease the amount of disadvantages. Advantages are utilizing the semi-structured form of interviews and hopefully getting better results in less controlled environment. Also it is potentially saving time when analysing the interview data and splitting the interviewing time in half. Disadvantage that should be kept in mind during the interviews is that the conversation's flow should follow the topics and questions already defined. If conversation drifts away from the subject, the advantages that we are trying to attain are lost.

Interviews include ten questions, which are located in the Appendix A. Interviews will last approximately 60 minutes and there is included the wrap up of the interview and possible new questions that have popped up. In total there are going to be two of these sessions each and together there are total of four interviewees. Interviews are documented with recordings, that are turned into notes after the interview. The reason for that is avoiding the interruption in the flow of the conversation. Interviews could be held remotely or physically. COVID-19 situation has modified the norms of physical presence at work and each one of the interviewees can have unique plan for their own presence. Each interview is chosen to be held with either of the choices.

5.2.2 Implementation Plan

BCB has already developed a working monitoring environment. As an expansion of the monitoring system, implementation of integration platform monitoring part

is done. That is produced with the same tools and technologies that the current monitoring system is using. With interview data it is possible to configure efficient solution for integration monitoring. When the implementation has been done, alerts can be set in the monitoring system. These alerts are configured based on the data provided by interviews and the literature.

Possible examination or questionnaires could be performed after the first implementation, for gaining more valuable information. It is easier to provide feedback of the application that has been implemented than a one that is only hypothetically planned. Also after the implementation it should be possible of analyzing benefits of the system.

6 Results

This chapter will show the reader results of the research and implementation work that had to be done with monitoring application's extension. Last Chapter's Section 5.1 did introduce us into BCB Medical Oy backgrounds. More detailed information about integration platform and monitoring technologies will be delivered in the implementation section. The implementation of integration platform's monitoring system is following insights of the interviews and literature guidelines.

6.1 Interviews

First interview was concluded with two participants where the other one joined remotely. Interviewees' backgrounds were somewhat similar, both had over two year experience with the integration platform and both of them have worked in the support team. Support team is handling the problems that the customers and coworkers bring up. First interviewee is now working in software engineering team and has special responsibility of integration in that team. Also he has developed and worked with the currently active monitoring system. The second interviewee has participated in development and planning of integration solutions along with the customer support work.

The second interview was also concluded with two participants in face to face contact at office. They both had experience of integration technologies for over 15 years. That includes a lot of different integration solutions and technologies. Both

of the interviewees work in the team that will develop and manage integration and its platform. Second interview had a lot more knowledge but the results of the interviews followed a similar path. Answers raised similar problems and thoughts but the differences were mostly on the aspect of how the monitoring will affect their own work.

The rest of this section will present the interview topics with titles and the answers and opinions that the interviewees had on the subject.

Integration Problems

The problems that directly affect only the integration platform are minority. These problems happen rarely, but are harder to solve. Most of the problems that reflect in the integration platform are really occurring in the application. Interviewees agreed that although integration platform does not generate the problem it is great at detecting them, monitoring the integration platform together with the other application would be the most effective choice. Monitoring the integration layer will also detect problems from the applications that are being administrated by the clients. Reasons for errors could be something like poorly constructed messages or messages piling up in the application's execution. Some of the problems can be detected by having issues in the applications execution and some of them can be found from the timestamps of messages.

Detection and Problem Solving

All of the interviewees agreed that the detection of the problems is coincidental. That means that the problems are not always noticed and have no efficient way of being observed. Monitoring system would solve the problem of detecting errors from multiple instances with one glance. Also the origin of the problems is not as important as detecting them. From the integration's point of view it can detect problems

from multiple instances and solving the problem would have to be performed by other teams. If the problems that happen are frequent, you should be able to make playbooks that describe and help solving the problems.

Consequences of the Problems

Amounts and consequences of the problems are very hard to portray with the integration platform. Integration's job is to transmit information into another instance, so if the applications work as they should there would be no errors. If the integration platform has a problem, it usually means that the data of the sent message is lost however these troubles are quite rare. If there are problems in the application that trigger because of message from integration, that might lead up to the application's unavailability.

Effective Way of Presenting Integration's Status

Classification of the errors and their detecting are the main elements of the integration that need to be monitored. Interviewees told that they are usually seeing plain numbers on the amounts of different type of messages. The problem is that there is nothing where to compare that amount. For the monitoring application it was proposed that the problems would stand out easily and the navigation would be flawless. Integration has many channels and they work little bit differently, that's why there needs to be a little bit different solutions for analysing them.

Useful Information for Visualization

The most important informative thing is to describe the amount of errors. Increasing amount of errors is indicating that there is a problem that should be corrected. Although interviewees all mentioned that some applications have unique manners and don't process the standard messages correctly, which results in situations where

errors are ignored because they have no real negative effect. Other information that could stand out is rate of the messages. Normally the growth of the message rates remains the same. Anomalies in the message flow could illustrate problems that the system won't detect itself. Situations where the integration has not received messages does not raise any errors because everything is working correctly. Although nonexistent data is a signal that part of the system is not working.

Setting Up the Alerts

Interviewees are in different position in the workplace. That is why everybody had their own preferences on how the alerts should work. Support team, which is responsible of applications state, wanted that the alerts would trigger if there were critical errors and that the information of errors would reach team quickly. Developer of the application stated that it is not the speed but rather the information that will make the alert more effective. For example playbooks that help solving the problem. Integration's managers were more interested on how the informatics will be displayed in the dashboard and if the problems are new or existing ones. When designing alerts I think all of these opinions should be taken care of.

What Kind of Problems and Notifications

The challenge in the alerts is that each of the information channel is unique. Each of the problem also can mean different things. Interviewees thought that they preferred alerts on our own products and then if there are possible faults in our client's network or missing data. There can be various of reasons why clients are not transferring the information and alerts should be set in a suitable time frame.

Setting Threshold Values

Threshold values should be also set based on individual values of channels. There was proposed that some machine learning or artificial intelligence would be used for determining the alert limits. This could be the situation in the future but currently designing a lighter version of monitoring system is the goal. But like we mentioned that the threshold values should be set based on the history of the monitored data. This way the anomalies in the monitoring system would stand out.

Overall

Overall all of the interviewees agreed with the fact that monitoring application is needed. The problematic part was that the interviews exposed the fact that some of the problems in applications, that appear in integration, are being ignored. That means our monitoring system should not alert on the messages that are not going to produce actions. Also all of the answers for interviewing questions were hinting in the same direction. On how the application should be created.

In addition to interviews the application's visibility and layouts were introduced and personnel had a clear opinion on that. They were given a set of different types of dashboard layouts and they would give me feedback on them. There was a clear view that was preferred and that was a map of all the integration channels, where there are color coded states of the channels. That gives a lot of information fast and can be used as a transition tool for navigating to right instances. After that there were no clear answers which type of layouts users want, but there was need of graphs that show the amount of messages at certain time. Capabilities of comparing that information to past data and data that is not coming from integration were desired.

6.2 Installing the Implementation

The application first developed is just a demo version. It can be installed to the production, when application is authorized as secure and working properly. Demo version of the application is using two different services. Monitoring system is using Prometheus to collect metrics from the integration platform. Prometheus is also temporarily saving the metrics in it's database. The other service, Grafana is used as a visualization tool. Grafana is capable of creating dashboards and visualising the collected data. Additionally Grafana can be used as an alerting tool. Some of visualized metrics work very well with alerts and alerts can be visually presented in the dashboard.

In the BCB's monitoring environment the whole architecture works similarly as the demo application for integration platform monitoring. The only big difference is that Prometheus is running in the Kubernetes environment. Kubernetes brings advantages like scaling, deploying, resourcing and better management for monitoring multiple instances. Prometheus gathers data and metrics from on-premise applications. This includes multiple registers, MyHealth-application and Mirth integration platform. Also the monitoring system uses a separate Alert Manager tool of Prometheus, which handles the alerts. Grafana has upgraded its alerting tools and there are multiple channels to get the notifications to the client. It is also good to get acquainted with multiple options before choosing just one. Additionally when installing the application in the production there can be compromises which technologies and techniques to use. Demo application is just a good way to imply the effects and present the dashboard without having to configure all the settings and righteous deployment.

6.3 Application

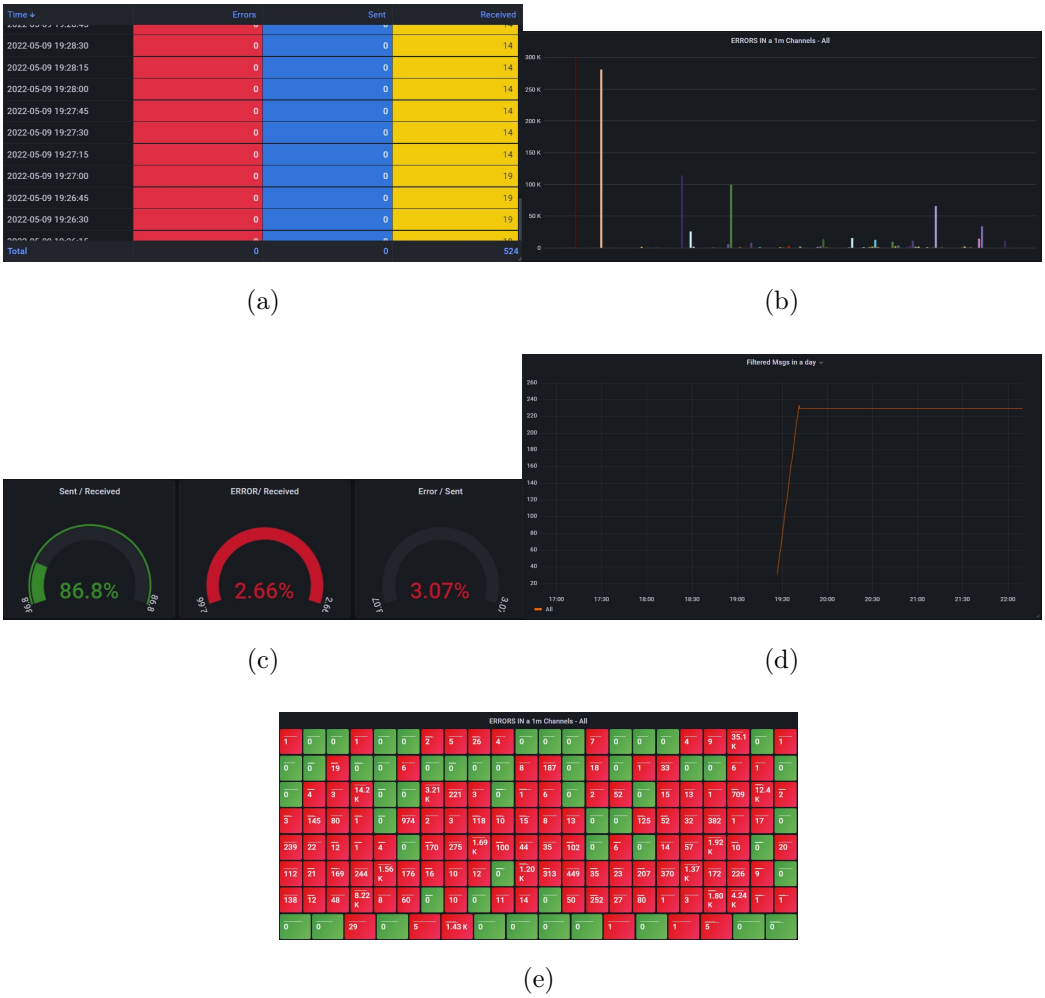
Monitoring application has similar architecture to Prometheus in Figure 2.1. It has Prometheus server that centralizes and stores the metrics that are needed. Multiple exporters that gather the data from the integration platform. Prometheus uses exporters to pull data from the application. Exporters can be installed in either own instance or to the monitored application's instance, depending on the rules and application's API. Prometheus server queries a list of data sources at specific pull frequency. Each of exporters emit the data to the end point that's queried by Prometheus. It is capable of handling the data and can aggregate data through data sources. In addition, Prometheus has also developed its own query language PromQL which is quite efficient. After gathering all metrics to Prometheus, visualization will take a place.

Prometheus doesn't provide efficient visualization tools, but has a close integration with Grafana. It's the most optimal solution to use and provides lots of different solutions. Visualization with Grafana is done with dashboards that can combine and compare different data repositories. Visualizing the right metrics is important and also this thesis's priority. Detecting the anomalies and informing about problems is also possible to do with Grafana's alerts. Alerting can be provided with multiple communication channels and setting the right amount of alerts into right destination is another puzzle to solve.

6.3.1 Constructing Dashboards

Constructing dashboards has been made pretty easy with Grafana's graphical user interface. Designing them from a scratch and modifying them has been made pretty simple. Some parts needed a bit more configurations, to get them work properly. Also Grafana has wide community and network with a great amount of already constructed dashboards. With the help of Grafana and the interviewees' visions of

the dashboard, it was not so problematic to develop.



(a) (b)

(c) (d)

(e)

Figure 6.1: Different Grafana dashboard designs.

When developing the dashboard, I asked for interviewees to give feedback on my designs. Most negative feedback were accumulated for the Figure 6.1 (a). Interviewees agreed that the timestamp and the message type values are hard to read and comprehend. That figure was only one that had negative feedback. After that the Figure 6.1 (b) and (c) were mediocre. They were liked overall but two out of five interviewees thought that they don't bring enough valuable information. They can give good overall understanding, but they are not extracting enough beneficial information. The last two Figure 6.1 (d) and (e) were overall liked. The graph in (d) was a good example of how to bring efficient information from monitoring. It

clearly defines the state of channels messages, and dashboard should have multiple different graphs from different message types. Also (e) was very effective, because the errors of the channels are visualized clearly. Red color indicates problems and that map of integration channels could be used to navigate into the right channel. That one saves a lot of time since you can navigate into problematic areas easily.

6.3.2 Configuring Thresholds and Alerts

Setting up the right thresholds and alarms was the issue in third research question. It's still an existing problem when using the application's demo version and that is result of the unique requirements of each integration channel. We approached the idea from a more practical way. Alerts and thresholds are configured in a way that they fit with most channels. After learning what kind of alerts are needed there can be more configuration and analysis on the matter, but before we are using the application in the production environment there are limits that are hard to overcome.

7 Conclusion

In conclusion chapter the summary for the research is given. Conclusion will cover all the research questions that were introduced in Chapter 1. Also the findings in the literature and interviews is analyzed. Most important findings are raised and reflected.

7.1 Summary

First research question formed a question: "How to build an effective monitoring service for integration platform?". The plan was that the answer would be conducted from literature. We formed a well rounded base knowledge about monitoring in general and the integration concept in the Chapters 2, 3 and 4. The research question is so comprehensive that solution for it can be concluded from the whole thesis. Thesis did present techniques on gathering the data from literature and the interviews where we gained valuable information on efficiency and more practical aspect of monitoring.

Second of the research questions asked "What unique requirements integration has over normal systems for monitoring?". This question's answer was given in Chapter 3. The chapter was describing integration's capabilities and qualities, which helps in determining the differences in monitoring integration platform. Thesis did handle the subject from multiple different aspects. The most important outcome was the information that would be concluded from the monitoring data. If integration

works as a piece between technologies it is important to have knowledge of the destination and sending services.

Third research question was: "How to determine thresholds that will inform the integration's malfunctions?". It was already a lot of more specified question than the first two. Thesis did mention this problem, but did not analyze it very deeply in the literature. This is also qualitative question where there can be different solutions. With interviews we managed to get vision of the right thresholds and limits. The problem included on the subject is the amount of unique metrics that have to be observed.

Fourth research question was determined to solve the visual problems and it was stated as "How to visually present the state of the whole integration to users?". Question was designed into that format, because one the existing issues in the BCB Medical's integration tool is lack of visualisation. It can detect problems and failed messages, in the integration, but how the information of that is retrieved is troublesome. Research managed to solve the question with the knowledge from interviews. Also visual layouts were noticed and different solutions considered. Then the solutions were analysed and they gained some feedback. That way the application's design was affected by users. Additionally interviews gave an answer that would take into consideration the application where the integration's problem is occurring, in that way we get more data available about the problem. The possibility of presenting multiple integration channels were implemented and that way it improves the error detection ability.

7.2 Future Improvements

For the future, clear goal would be to get the application in the production. With that we could get a lot more precise information on the last configurations that need a bit more work. The reason why application was not released in the production,

is strict security policies that our company has. In medical field the security aspect is taken seriously. Also the monitoring project has not been prioritized by the company, because it is not adding anymore revenue.

Also the application could be upgraded. Deeper analysis on the data could be achieved with methods like machine learning or artificial intelligence. This would result in possible predictions of errors and problems. This is achieved by analysing existing monitoring data and metrics that are affecting the problems. Also finding more correlations between the system would be possible. Rather than focusing the intelligent methods in monitoring, it would be more beneficial to get this version into production.

References

[1] J. Turnbull, *The art of monitoring*. 2016, OCLC: 1007536451, ISBN: 978-0-9888202-4-1. [Online]. Available: https://proquest.safaribooksonline.com/9780988820241 (visited on 05/11/2022).

[2] M. Julian, *Practical Monitoring: Effective Strategies For the Real World*. Nov. 27, 2017, ISBN: 978-1-4919-5734-9. [Online]. Available: https://learning.oreilly.com/library/view/practical-monitoring/9781491957349/ch05.html (visited on 10/21/2021).

[3] M. N. Birje, "Cloud monitoring system: Basics, phases and challenges", *IJRTE*, vol. 8, no. 3, pp. 4732–4746, Sep. 30, 2019, ISSN: 2277-3878. DOI: 10.35940/ijrte.C6857.098319. [Online]. Available: https://www.ijrte.org/wp-content/uploads/papers/v8i3/C6857098319.pdf (visited on 05/11/2022).

[4] M. S. Aktas, "Hybrid cloud computing monitoring software architecture", *Concurrency Computat Pract Exper*, vol. 30, no. 21, e4694, Nov. 10, 2018, ISSN: 15320626. DOI: 10.1002/cpe.4694. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/cpe.4694 (visited on 05/02/2022).

[5] BCBMedical Oy. (Dec. 16, 2020). "BCB medical", BCB Medical, [Online]. Available: https://bcbmedical.com/ (visited on 02/03/2022).

[6] J.-P. Martin-Flatin, "Push vs. pull in web-based network management", SWISS FEDERAL INSTITUTE OF TECHNOLOGY LAUSANNE, 1998, pp. 7–17.

[Online]. Available: https://arxiv.org/ftp/cs/papers/9811/9811027.pdf.

[7] P. Abrahamsson, O. Salo, and J. Ronkainen, "Agile software development methods: Review and analysis", p. 112, 2002.

[8] A. Pecchia, M. Cinque, G. Carrozza, and D. Cotroneo, "Industry practices and event logging: Assessment of a critical software development process", in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Florence, Italy: IEEE, May 2015, pp. 169–178, ISBN: 978-1-4799-1934-5. DOI: 10.1109/ICSE.2015.145. [Online]. Available: http://ieeexplore.ieee.org/document/7202961/ (visited on 05/11/2022).

[9] Prometheus. (Jan. 1, 2022). "Overview | prometheus", [Online]. Available: https://prometheus.io/docs/introduction/overview/ (visited on 01/11/2022).

[10] B. Brazil, *Prometheus: up & running: infrastructure and application performance monitoring*, First edition. Sebastopol, CA: O'Reilly Media, 2018, 369 pp., OCLC: on1031344954, ISBN: 978-1-4920-3414-8.

[11] Prometheus. (Jan. 1, 2022). "Pushing metrics | prometheus", [Online]. Available: https://prometheus.io/docs/instrumenting/pushing/ (visited on 01/12/2022).

[12] Riemann. (Jan. 1, 2022). "Riemann - a network monitoring system", [Online]. Available: https://riemann.io/ (visited on 01/12/2022).

[13] L. Fitzgibbons. (Feb. 1, 2020). "What is a time series database? TSDB explained", IoT Agenda, [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/time-series-database-TSDB (visited on 01/12/2022).

[14] P. Dix, "Why time series matters for metrics, real-time analytics and sensor data", Jul. 5, 2021, p. 19.

[15] InfluxDB. (Jun. 6, 2017). "Time series database (TSDB) guide | InfluxDB", InfluxData, [Online]. Available: https://www.influxdata.com/time-series-database/ (visited on 01/12/2022).

[16] K. Verheggen, H. Ræder, F. S. Berven, L. Martens, H. Barsnes, and M. Vaudel, "Anatomy and evolution of database search engines—a central component of mass spectrometry based proteomic workflows", *Mass Spec Rev*, vol. 39, no. 3, pp. 292–306, May 2020, ISSN: 0277-7037, 1098-2787. DOI: 10.1002/mas.21543. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/mas.21543 (visited on 05/02/2022).

[17] Amazon. (Feb. 4, 2012). "What is a search-engine database?", Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/nosql/search/ (visited on 01/12/2022).

[18] Grafana. (Jul. 20, 2022). "Loki", Grafana Labs, [Online]. Available: https://grafana.com/docs/grafana/latest/datasources/loki/ (visited on 01/12/2022).

[19] tutorialspoint.com, "Database management system [DBMS] tutorial", p. 80, Jan. 1, 2020.

[20] E. Vanier, B. Shah, and T. Malepati, *Advanced MySQL 8: Discover the full potential of MySQL and ensure high performance of your database*, ISBN: 9781788833790 OCLC: 1126569046, 2019.

[21] C. Arsenault. (Apr. 20, 2017). "The pros and cons of 8 popular databases", KeyCDN, [Online]. Available: https://www.keycdn.com/blog/popular-databases (visited on 01/22/2022).

[22] H. Hai and S. Seitaro, "SaaS and integration best practices", *FUJITSU Sci. Tech. J.*, vol. 45, no. 3, p. 8, 2009.

[23] Wilhelm Hasselbring, "Information system integration", COMMUNICATIONS OF THE ACM, vol. 2000, no. 43, Oct. 2000.

[24] M. NORSHIDAH, B. MAHADI1, M. SURAYA, H. HANIF, and M. Hazifuddin, "Information system integration: A review of literature and a case analysis", p. 11, Mar. 23, 2014.

[25] W. Sun, K. Zhang, S.-K. Chen, X. Zhang, and H. Liang, "Software as a service: An integration perspective", in *Service-Oriented Computing – ICSOC 2007*, B. J. Krämer, K.-J. Lin, and P. Narasimhan, Eds., red. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, and G. Weikum, vol. 4749, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 558–569, ISBN: 978-3-540-74973-8. DOI: 10.1007/978-3-540-74974-5_52. [Online]. Available: http://link.springer.com/10.1007/978-3-540-74974-5_52 (visited on 11/25/2021).

[26] T. Gulledge, "What is integration?", *Industrial Management & Data Systems*, vol. 106, no. 1, pp. 5–20, Jan. 1, 2006, ISSN: 0263-5577. DOI: 10.1108/02635570610640979. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/02635570610640979/full/html (visited on 05/02/2022).

[27] S. R. Sinha and Y. Park, *Building an Effective IoT Ecosystem for Your Business*. Cham: Springer International Publishing, 2017, ISBN: 978-3-319-57391-5. DOI: 10.1007/978-3-319-57391-5. [Online]. Available: http://link.springer.com/10.1007/978-3-319-57391-5 (visited on 03/23/2022).

[28] P. Johnson, "ENTERPRISE SOFTWARE SYSTEM INTEGRATION", Ph.D. dissertation, KTH, Stockholm, Apr. 2002. [Online]. Available: https://citeseerx.

ist.psu.edu/viewdoc/download?doi=10.1.1.108.3962&rep=rep1&type=pdf.

[29] J. Euzenat, *Ontology matching*. Berlin ; New York: Springer, 2007, 333 pp., OCLC: ocn124038270, ISBN: 978-3-540-49611-3.

[30] M. Da Silveira, C. Pruski, and R. Schneider, Eds., *Data Integration in the Life Sciences*, vol. 10649, Lecture Notes in Computer Science, Cham: Springer International Publishing, 2017, ISBN: 978-3-319-69751-2. DOI: 10.1007/978-3-319-69751-2. [Online]. Available: http://link.springer.com/10.1007/978-3-319-69751-2 (visited on 01/22/2022).

[31] E. Lämsä, J. Timonen, P. Mäntyselkä, and R. Ahonen, "Pharmacy customers' experiences with the national online service for viewing electronic prescriptions in finland", *International Journal of Medical Informatics*, vol. 97, pp. 221–228, Jan. 2017, ISSN: 13865056. DOI: 10.1016/j.ijmedinf.2016.10.014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1386505616302295 (visited on 03/23/2022).

[32] A. Roehrs, C. A. da Costa, and R. da Rosa Righi, "OmniPHR: A distributed architecture model to integrate personal health records", *Journal of Biomedical Informatics*, vol. 71, pp. 70–81, Jul. 2017, ISSN: 15320464. DOI: 10.1016/j.jbi.2017.05.012. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1532046417301089 (visited on 03/23/2022).

[33] K. Ralf-Detlef and N. Milanovic, Eds., *Model-based software and data integration: first international workshop, MBSDI 2008, Berlin, Germany, April 1-3, 2008: proceedings*, Communications in computer and information science 8, Meeting Name: MBSDI 2008, Berlin ; New York: Springer, 2008, 124 pp., ISBN: 978-3-540-78998-7.

[34] hl7. (Jan. 1, 2022). "Introduction to HL7 standards | HL7 international", [Online]. Available: http://www.hl7.org/implement/standards/index.cfm?ref=nav (visited on 02/01/2022).

[35] i. Inc. (Jan. 1, 2010). "HL7 - ADT message", [Online]. Available: https://www.interfaceware.com/hl7-adt (visited on 02/01/2022).

[36] Thomas Beale. (Jan. 18, 2020). "Architecture overview", [Online]. Available: https://specifications.openehr.org/releases/BASE/latest/architecture_overview.html (visited on 03/23/2022).

[37] IHE. (Jan. 1, 2020). "Integrating the healthcare enterprise (IHE) - IHE international", [Online]. Available: https://www.ihe.net/ (visited on 03/23/2022).

[38] A. Moumtzoglou, Ed., *Design, Development, and Integration of Reliable Electronic Healthcare Platforms:* red. by A. Moumtzoglou, Advances in Healthcare Information Systems and Administration, IGI Global, 2017, ISBN: 978-1-5225-1724-5. DOI: 10.4018/978-1-5225-1724-5. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-5225-1724-5 (visited on 05/13/2022).

[39] J. Wang, B. J. Liu, W. He, J. K. Xue, and X. Y. Han, "Research on computer application software monitoring data processing technology based on NLP", *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 1043, no. 3, p. 032 021, Jan. 1, 2021, ISSN: 1757-8981, 1757-899X. DOI: 10.1088/1757-899X/1043/3/032021. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/1043/3/032021 (visited on 05/13/2022).

[40] F. Wang Zhang, J. A. Dagtas Serhan, T. John, Baghal Ahmad, and Zozus Meredith, *Rule-Based Data Quality Assessment and Monitoring System in Healthcare Facilities.* Amsterdam, NETHERLANDS, THE: IOS Press, Incorporated, 2019, 460-467, ISBN: 978-1-61499-951-5. [Online]. Available: http:

//ebookcentral.proquest.com/lib/kutu/detail.action?docID=5775655 (visited on 03/22/2022).

[41] G. Maier, M. Schiffers, D. Kranzlmueller, and B. Gaidioz, "Association rule mining on grid monitoring data to detect error sources", *Journal of Physics: Conference Series*, vol. 219, no. 7, Apr. 2010, Place: Bristol, United Kingdom Publisher: IOP Publishing, ISSN: 17426588. DOI: http://dx.doi.org.ezproxy.utu.fi/10.1088/1742-6596/219/7/072041. [Online]. Available: http://www.proquest.com/publiccontent/docview/2579975922/abstract/2B153D7CC01D47EEPQ/1 (visited on 03/22/2022).

[42] M. Last, A. Kandel, and H. Bunke, *Data mining in time series databases*. Singapore: World Scientific, 2004, OCLC: 897095154, ISBN: 978-981-256-540-2.

[43] S. Ligus, *Effective monitoring and alerting*. Farnham: O'Reilly, 2013, 149 pp., OCLC: ocn817262597, ISBN: 978-1-4493-3352-2.

[44] E. R. Tufte and G. M. Schmieg, "The visual display of quantitative information", *American Journal of Physics*, vol. 53, no. 11, pp. 1117–1118, Nov. 1985, ISSN: 0002-9505, 1943-2909. DOI: 10.1119/1.14057. [Online]. Available: http://aapt.scitation.org/doi/10.1119/1.14057 (visited on 05/13/2022).

[45] M. Lubis, F. Dennis, R. Andreswari, and A. Ridho Lubis, "Dashboard information system development as visualization of transaction reports in the application BackInd (backpacker reservation system)", *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 801, no. 1, p. 012 145, May 1, 2020, ISSN: 1757-8981, 1757-899X. DOI: 10.1088/1757-899X/801/1/012145. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/801/1/012145 (visited on 03/28/2022).

[46] G. Sedrakyan, E. Mannens, and K. Verbert, "Guiding the choice of learning dashboard visualizations: Linking dashboard design and data visualization concepts", *Journal of Computer Languages*, vol. 50, pp. 19–38, Feb. 2019, ISSN: 25901184. DOI: 10.1016/j.jvlc.2018.11.002. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1045926X18301009 (visited on 03/28/2022).

[47] Graphite. (Jan. 1, 2020). "FAQ — graphite 1.2.0 documentation", [Online]. Available: https://graphite.readthedocs.io/en/latest/faq.html (visited on 03/28/2022).

[48] Grafana. (Jan. 1, 2022). "Grafana® features", Grafana Labs, [Online]. Available: https://grafana.com/grafana/ (visited on 03/29/2022).

[49] A. Srivastava, *Mastering Kibana 6. x: Visualize Your Elastic Stack Data with Histograms, Maps, Charts, and Graphs*. Birmingham, UNITED KINGDOM: Packt Publishing, Limited, 2018, ISBN: 978-1-78883-403-2. [Online]. Available: http://ebookcentral.proquest.com/lib/kutu/detail.action?docID=5485022 (visited on 03/28/2022).

[50] Elastic. (Jan. 1, 2020). "Kibana: Explore, visualize, discover data", Elastic, [Online]. Available: https://www.elastic.co/geospatial (visited on 03/28/2022).

[51] F. Shull, J. Singer, and D. I. K. Sjøberg, Eds., *Guide to advanced empirical software engineering*, OCLC: ocn173720742, London: Springer, 2008, 388 pp., ISBN: 978-1-84800-043-8.

Appendix A Haastattelurunko

Haastateltavien kokemus ja osaaminen.

Kysymykset:

- 1. Minkä tyyppisiä ongelmia integraatiossa esiintyy?
- 2. Havaitaanko ongelmat tai onko ongelmien selvittäminen sujuvaa?
- 3. Kuinka paljon ongelmia esiintyy?
- 4. Mitkä ovat vakavimmat ongelmat ja niiden seuraukset?
- 5. Mikä olisi tehokas tapa esittää integraatioalustan/integraation tilaa?
- 6. Mitkä tiedot olisivat hyödyllisimpiä ja millä tavalla?
- 7. Miten integraatioalustan ongelmista pitäisi hälyttää?
- 8. Mistä ongelmista halutaan hälytys?
- 9. Miten hälytysrajat pitäisi asettaa ja määritellä

Appendix B Interview Questions

Interviewee's experience and skills:

Questions:

- 1. What kind of problems exists in the integration
- 2. Do the problems get detected fluently?
- 3. How much of these problems occur?
- 4. What are the most severe problems and their consequences?
- 5. What is the best way to describe the integration's state?
- 6. What information about integration would be required and how?
- 7. How different problems in the integration platform should be displayed?
- 8. What problems should trigger the alert?
- 9. How thresholds for alerts should be determined?