

## **Bachelor's Thesis**

# **Safety as Service**

## **Service oriented Architectures for safety-critical Systems**

Submitted by: Stefan Lengauer

Registration number: 1210587029

Academic Assessor: FH-Prog.Dipl.Ing.Dr. Holger Flühr

Date of Submission: xx August 2015

# Declaration of Academic Honesty

I hereby affirm in lieu of an oath that the present bachelor's thesis entitled

**“Safety as Service - Service oriented Architectures for safety-critical Systems”**

has been written by myself without the use of any other resources than those indicated, quoted and referenced.

Graz, xx August 2015

Stefan LENGAUER,

A handwritten signature in black ink, appearing to read 'Stefan Lengauer', written in a cursive style.

# Preface

This thesis was written as result of the internship at VIRTUAL VEHICLE at Inffeldgasse, Graz from April to July 2015, which was conducted as part of the Degree Programme in Aviation at FH Joanneum in Graz, Austria.

The VIRTUAL VEHICLE research center is an international company, specialized in automotive and rails industry. In total, it has over 200 employees, which are splitted up in four main research areas. For the time of my employment, I worked as part of the Electrics/Electronics (E/E) & Software Area, ... more specifcly .. functional safety.

During my employment is was part of the EMC2 project, a European project, .... my part soa

This thesis consists mainly of two documents, which were produced during my internship. The first one is a glossary, which aims at defining certain terms and unifying the opinions from different research areas. The second document, was an extensive investigation on how the service oriented architecture paradigm can be applied in a safety-critical embedded system. In detail, with a vehicle as the system.

Although the employment was oriented towards the automotive industry, the functional safety and fault tolerance concepts, reappear in a very similar way in other engineering disciplines, like aviation. Furthermore, the service oriented approach, which is considered in terms of automotive, will most likely become also an important issue for aviation in near future.

Furthermore, the location of the company at Inffeldgasse was another big advantage, for it allowed the conduction of various courses at the Technical University of Graz, alongside the employment.

At this point I want to thank my supervisor from FH JOANNEUM, FH-Prof. Dipl.Ing. Dr. Holger Flühr, as well as my supervisor provided by the company, Dipl.Ing. Helmut Martin. ...

and also Dipl.Ing.Dr. Andrea Leitner and Mr. Mario Driussi, which helped develop the ideas and concepts featured in this thesis during numerous meetings and discussions.

# Contents

<b>Abstract</b>	<b>vi</b>
<b>Kurzfassung</b>	<b>vii</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>ix</b>
<b>List of Abbreviations</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Related instances . . . . .	4
1.1.1 ISO 262662 international standard . . . . .	4
1.1.2 MISRA . . . . .	4
1.1.3 AUTOSAR . . . . .	4
1.2 service . . . . .	4
1.3 System . . . . .	4
1.4 Component . . . . .	4
1.5 Service . . . . .	4
1.6 Architecture . . . . .	4
1.7 Service oriented architecture . . . . .	4
1.7.1 Definition . . . . .	4
1.7.2 Historic development . . . . .	4
1.7.3 Structure of SoA . . . . .	4
1.7.4 Service composition . . . . .	4
1.8 Communication in SoA . . . . .	4
1.9 Dependability . . . . .	4

1.9.1	Reliability . . . . .	4
1.9.2	Availability . . . . .	4
1.10	Functional safety . . . . .	4
1.10.1	Safety related terminology . . . . .	4
1.10.2	Definition of safety . . . . .	4
1.10.3	Fault tolerance . . . . .	4
<b>2</b>	<b>Methods</b>	<b>5</b>
2.1	SoA in Embedded Systems (Embedded SoA) . . . . .	5
2.1.1	Correlation of design philosophies . . . . .	5
2.1.2	Difference between Web SoA and Embedded SoA . . . . .	6
2.1.3	SoA in automotive . . . . .	10
2.2	Safety services . . . . .	12
2.2.1	Fault detection service (WDT) . . . . .	13
2.2.2	Error detection service . . . . .	14
2.2.3	Memory Protection Service . . . . .	14
2.2.4	Orchestration Management Service . . . . .	15
2.3	Use Case: Error Detection Service . . . . .	15
2.3.1	Service Investigation/Planning . . . . .	15
2.3.2	Service inventory analysis . . . . .	15
2.3.3	Service oriented analysis . . . . .	15
2.3.4	Service oriented design . . . . .	15
<b>3</b>	<b>Results</b>	<b>17</b>
<b>4</b>	<b>Discussion</b>	<b>18</b>
<b>5</b>	<b>Conclusion</b>	<b>19</b>

# Abstract

One of the major issues in the automotive industry is the constant growing complexity of E/E (Electrics/Electronics) systems and the thereof resulting fault propagation due to the strong interconnection of the systems. The area of *functional safety* is concerned with the prevention of non tolerable risks in event of error. This is conducted by identifying possible hazards, estimating the potential risks and developing necessary countermeasures, based on these investigations. The requirement therefore is an accurate and thorough comprehension of the observed E/E system.

The service oriented architecture is a design paradigm, which features the concept of software reuse by implementing functionalities as technology independent and loosely coupled services. Although the design paradigm is already standard for Web applications, it has not yet been applied in safety-critical embedded systems.

Thus, this Bachelor's thesis investigates the applicability of service oriented architectures for such systems, with a focus on *functional safety* and *fault tolerance* context.

The first part of the thesis is mainly a glossary, which defines certain terms, which are critical for the subsequent presented concepts. On this basis it is defined what *safety as a service* can mean in general, and how the actual implementation in a given safety-critical system may look like, in order to meet with the requirements specified in the ISO 26262 standard. This is the safety standard for safety-critical E/E systems in vehicles with a maximum gross weight of 3500 kg.

# Kurzfassung



# List of Figures

2.1	Illustration of the overall system architecture by Gleirscher and Vogelsang [1].	7
2.2	Relation of service oriented architecture to service oriented architecture in embedded systems. . . . .	9
2.3	Classification of various services into safety and security services. . . . .	12
2.4	Schematic design of a windowed watchdog timer [2]. . . . .	14
2.5	Schematic illustration of the working process of a <i>Sequenced Watchdog Timer</i> [2]. . . . .	15
2.6	Example architecture for an <i>fault detection service</i> , like a WDT. . . . .	16
2.7	Example architecture for an <i>error detection service</i> . . . . .	16

# List of Tables

# **List of Abbreviations**

# Chapter 1

## Introduction

Due to the very different comprehension of certain terms by different people from different fields of research The first part of my work during the internship was the creation of a glossary describing certain given terms ... in order to unify the understanding of these terms and create a basis for discussions and researches.

- The glossary was later taken as standard for this tasks ... - and deals with the terms: *service, system, system of systems, architecture, service-oriented architecture, configuration, static reconfiguration, dynamic reconfiguration, inter-core communication, intra-core communication* and *binding*. The most important parts of this glossary will be covered within this chapter. - furthermore a database with the used literature was created in order to provide the necessary references for the glossary and provide a ... where to find further information for the employees.

- disambiguity safety and security - what is safety and fault tolerance



## **1.1 Related instances**

### **1.1.1 ISO 262662 international standard**

### **1.1.2 MISRA**

### **1.1.3 AUTOSAR**

## **1.2 service**

## **1.3 System**

## **1.4 Component**

## **1.5 Service**

## **1.6 Architecture**

## **1.7 Service oriented architecture**

### **1.7.1 Definition**

### **1.7.2 Historic development**

### **1.7.3 Structure of SoA**

### **1.7.4 Service composition**

## **1.8 Communication in SoA**

## **1.9 Dependability**

### **1.9.1 Reliability**

### **1.9.2 Availability**

## **1.10 Functional safety**

### **1.10.1 Safety related terminology**

# Chapter 2

## Methods

This chapter is structured in the three main section, labeled *SoA in Embedded System (Embedded SoA)*, *Safety services* and *Use Case: Error Detection Service*.

The first section deals extensively with the possible application of the service oriented design paradigm in embedded system and deals with the challenges this approach has to face when approached with safety-critical requirements. Moreover, the differences to conventional SoAs, as Web application is investigated in detail. The second section presents xx services, which are relevant for *functional safety* in detail and presents their intended functionality, as well as possible implementations in terms of an architecture.

The final part consists of a simplified use case, dealing with the design of an error detection service with respect to the design phases *service investigation/planning*, *service inventory analysis*, *service oriented analysis* and *service oriented design*.

Most of the findings and concepts in this chapter is “not yet” covered in literature, but is the result of numerous meetings and discussions.

## 2.1 SoA in Embedded Systems (Embedded SoA)

### 2.1.1 Correlation of design philosophies

The CBSE and SoA are coexisting design philosophies and usually either one or the other is applied in a given system. In contrast, Gleirscher and Vogelsang present in [1] a proposal, how both paradigms can exist in parallel in a single system.

The presented system models consists of two layers with three different models. The *feature layer* contains the *feature hierarchy* model, which describes all the features a vehicle

presents to its user. The *platform layer*, in contrast, is the technical realization and consists of the *service hierarchy* and the *component model* [1].

This relation is depicted in figure 2.1.

**Feature Hierarchy** . Alongside *services* and *components* their model disposes of a third hierarchy, denoted *features*.

According to [1], a feature “is a behavioral property of a system observable (and influenceable) by its user.” AUTOSAR describes it by the following quote:

*“The term feature is commonly used in the software tool community to describe characteristics (functionality) of the software” [3].*

The *feature* has a many to many relationship to the services, since a service could offer various features, but one and the same feature, can also be offered by the very same service. For example the service *braking service* could contain a *safety braking mechanism*, which is responsible for performing an emergency brake when needed [1].

The set of features for a specific vehicle is given by its configuration in terms of hardware software, etc. A convertible car may include a feature for automatically opening and closing the top, while a sports car could allow to switch to a “sport mode.”

**Service Hierarchy** . The services are the link between the features and the technical platform, they are realized on. A service can be provided either directly by a “material component”, like a sensor or an actuator, or it can be composed out of various other services [1]. For an acceleration sensor, the corresponding service would be the *acceleration measurement*.

**Component Model** . The components are the material or immaterial, in other words hardware and software, parts of a system. Accordingly, a material part is for example a sensor and an immaterial part a communication software.

## 2.1.2 Difference between Web SoA and Embedded SoA

The service oriented design paradigm was originally designed for the application in the web, which offered, more or less, ideal prerequisites for this kind of architectural style. There is an underlying network already available, which allows interconnection, and *time constraints* are



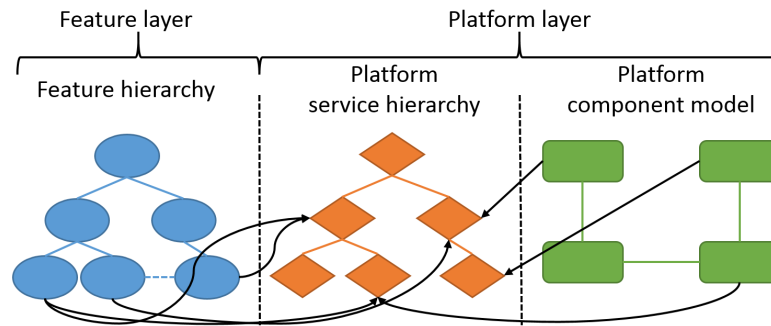


Figure 2.1: Illustration of the overall system architecture by Gleirscher and Vogelsang [1].

quite ..., because delays are unlikely to cause a disaster. Thus, it is not surprising that web services are the application area, where SoA has scored the highest market penetration [4] [5].

In contrast to web services, *embedded systems* are networks consisting of numerous interconnected nodes, with diverse measurement-, steering-, or computation capabilities. They have to face certain challenges like *limited resources*, *different complexity of hardware* [6] [7].

Some of the issues, the SoA has to face in embedded systems are described in the following.

**Limited resources** . One obvious major drawback are the very limited resources in embedded systems, which are designed for highly specialized purposes and lack computation power as well as storage size [4] [6] [7].

Also the set of components is determined at assembly time ...

**Different levels of complexity** . The complexity usually varies greatly. The applied hardware components may include very simple sensors, with low resources and capabilities as well as very advanced nodes like MPSoCs) [6] [7].

Accordingly, there are high level information systems services as well as low level, generic embedded system services. SoA has to deal with the connection and integration of them [4]. Especially when it comes to the interaction of SoAs in embedded systems with SoAs at higher levels.

**Event- and data-driven** . In contrast to web services, an embedded system disposes of (many) sensors which are interconnected in a network. Thus, the ad-hoc “request-response message pattern” which is known from web services does not work, for embedded systems are event- and data driven: For example a sensor measures some data and publishes it to all connected services. Those can then act according to the received

input, e.g. trigger actuators or other services. This is denoted as “fire and forget” scheme [7].

**Lifespan of services** . Another difference to web services is the lifespan of services. While web services are used to work only a limited number of hours (or even minutes), the services in embedded systems could have application times of multiple years, e.g. a measurement service for a sensor [5].

**Dynamic character** . This is even aggravated by the dynamic character of embedded networks - *“new nodes may enter the network, existing nodes may fail and network characteristics can change over time, especially if wireless communication media are used”* [7].

**Time constraints** . Embedded systems are time-critical, meaning that computation must be conducted within a given time window in order to allow the correct operation of the system. Especially in a safety-critical system like a vehicle, which is used to operate at high velocities, a violation of those time constraints could cause serious incidents.

These obstacles give reason to the statement that web services and safety-critical embedded systems are non-related areas [4].

Nevertheless, there are a lot of benefits in applying a service oriented architecture style, like decoupling configuration from environment, improvement of reuseability, maintainability, higher level of abstraction, interoperability, more interactive interfaces between devices and information systems.

[5].

The SoA paradigm offers a promising possibility to overcome many of embedded systems related drawbacks and implementing some of the stated advantages [5] [7]. Thus, it is not surprising that there are various different projects dedicated to investigate the applicability of SoA for embedded systems. Those include SIRENA, SOCRATES, OASiS, MORE, RUNES and  $\epsilon$ SOA. In the proceeding the  $\epsilon$ SOA project by Scholz, Sommer et al. [6] [7] [5] will be investigated in more detail.

## **eSoA**

The  $\epsilon$ SOA project from Sommer, Gaponova and Buckl from the Department of Informatics at TU München, is an approach to master the obstacles which come with applying service

oriented architectures in embedded systems.

However, the proposed architecture omits some functionality which is included in conventional SoAs, like the dynamic detection of services within the network. Moreover it features *long term service composition* and uses external tools for the reconfiguration. However, a necessary requirement is the ability to update deployed services and migrate services to new hardware without a complete system interrupt [7].

## Embedded SoA

Unfortunately, there are not many sources covering the topic of SoA in embedded systems yet. Accordingly, the term *embedded SoA* does not appear in literature, but has been defined during the EMC2 project, in order to refer to service oriented architectures in embedded systems. The relation of “embedded SoA” to “conventional SoA” has been defined as it can be seen in figure 2.2. According to these findings the *embedded SoA* operates at a lower, very hardware oriented level, with a predefined and unchanging component architecture.

The SoA is located “above” embedded SoA and connected by interfaces. With regard to the example of a vehicle, the embedded SoA operates on elements like a power train, various devices (sensors, actuators, ...) etc, while SoA level contains the vehicle as overall system, but also other vehicles, which can be referred to as the *System of Systems*.

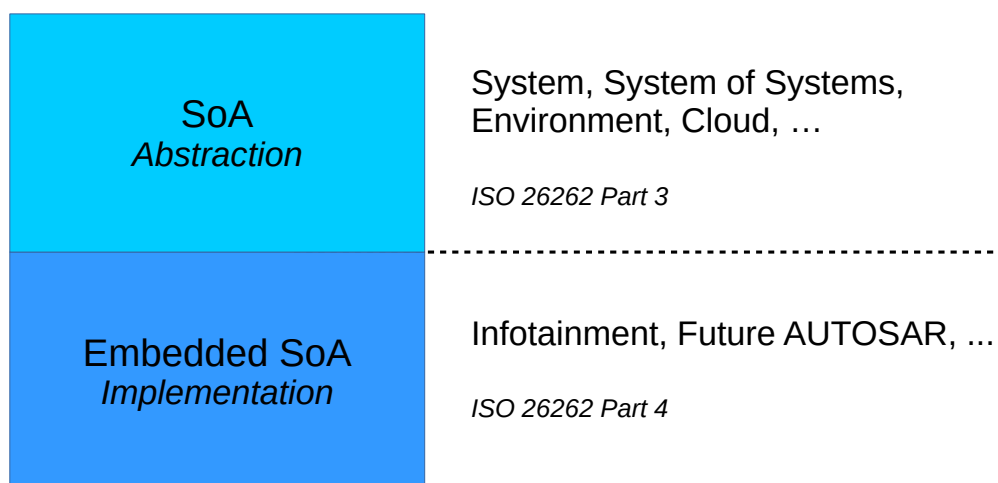


Figure 2.2: Relation of service oriented architecture to service oriented architecture in embedded systems.

### 2.1.3 SoA in automotive

An important subpart of the EMC2 project is the investigation of the applicability of the SoA concepts for the automotive industry.

In the future of automotive all vehicles will be, most likely, interconnected with each other and also their environment. This could give them opportunities like automatically detecting whether the traffic lights are green at a crossing and acting automatically. Within this document those futuristic vehicles, which are interconnected with their environment, are referred to as “connected cars”.

The opportunities and advantages of the SoA approach have been dealt with in the previous section. With respect to vehicles its major advantage would be the possibility to reduce the quantity of computation hardware, because at the present time each component disposes of his own dedicated hardware for this task. Many of those are unused for the majority of time, but nevertheless they come with a lot of extra weight. As stated in [8, p.7], “Embedded systems should exploit the available hardware architecture as much as possible.”

The SoA approach might reduce not only the weight, but also the complexity and costs of the overall vehicle. So far it is usually the case that every vendor used its own proprietary network and additional hardware, prohibiting the application of hardware components from another vendor [7].

Despite of all these advantages, right now the SoA paradigm is hardly applicable for vehicles. On the one hand, this is due to the strict regulations which apply for safety critical real-time systems like vehicles, which do not yet include any requirements with respect to SoAs. In terms of automotive the safety aspect also has to be considered, since vehicles are safety-critical systems where an error or a malfunction can easily harm people [9].

On the other hand, there are technical constraints. The AUTOSAR architecture, which is widely applied today, is not constructed for dynamic reconfiguration or binding of services, but everything has to be specified at building time. Still, there is already an approach by AUTOSAR, denoted “Future AUTOSAR”, which is dealing with this very issue.

Although, the SoA for vehicles is still a long way ahead, it might be easier applicable at other levels, like the *System of Systems* level, meaning other vehicles and the overall traffic. For self-driving vehicles there could be for example services for the communication with traffic lights. If the the traffic lights in Germany would use another service as the ones in Austria, a SoA would allow to dynamically bind the new service when the border is approached. There are already projects which are dealing with the communication at this level of implementation.

## Location of the service repository

The service repository is described in detail in section ??, but its exact location and implementation in connection with automotive remains ambiguous.

For SoAs in automotive it is likely that there exist various distributed repositories. One repository inside of every vehicle, containing all the services provided by the vehicle itself, which is also accessible if the vehicle is operating in urban regions and is not able to connect to its environment. For the SoA inside the vehicle, there is no other possible location, because otherwise it would be operable only as long as it is connected to the Internet, or whatever platform, which is used for the interconnection with the environment.

For the scope of this document these repositories will be referred to by the term “local service repository”. Examples for the services they hold are services which origin from hardware components of the vehicle like, acceleration measurement, temperature measurement, communication services and the like. But also safety critical services for fault- and error detection, which need to be available whenever the vehicle is operated.

The counterpart to the “local service repository” is denoted by “external service repository”. This could also be physically distributed and holds mostly services which are needed for interacting with the environment. For “self-driving”- or “connected cars” it could host all the services for managing the traffic. With an example: A traffic light in Graz publishes its service, “Show me the traffic light signal”, somewhere on a server, where it is detected and bound by a vehicle driving in Graz, which is then able to decide whether it has to stop at an intersection.

Other services which could be located in this repository are *update services*. If an update for an existing service in the local repository is available, the vehicle could detect this automatically and subsequently load and install the specific service.

## Service Contract

As already specified in section ??, the service contract is the complete and extensive description of the service and should contain, with regard to vehicles, Documents from AUTOSAR, the ISO 26262 standard and also documentation regarding *functional safety*. It is more or less the goal of work package 1 of the EMC2 project to extend the *service contract* by a functional safety part.

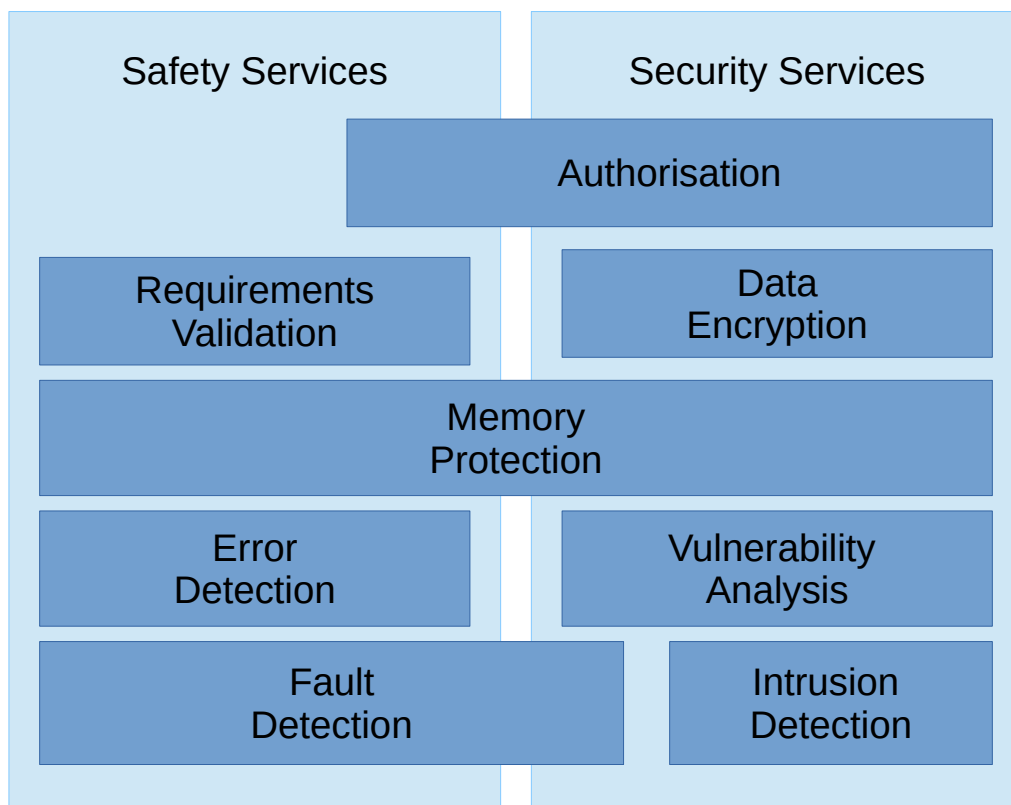
At the beginning of a service development process the contract is just an empty template

which gets filled more and more as the development advances. A template, which could be used for the development of services in the automotive industry is already provided by the ARROWHEAD project.

## 2.2 Safety services

Disambiguation of Safety and security services

Figure 2.3: Classification of various services into safety and security services.



The most frequent application of service oriented architectures is in the field of Web services, where safety does not play a major role. However, in recent times this philosophy is also emerging in safety critical embedded systems, where specific safety and fault tolerance requirements have to be met.

There are many challenges for safety and security management in SoA systems, like the distributed hardware and software structure, loosely coupled components from different vendors, etc.

### 2.2.1 Fault detection service (WDT)

The fault detection service is described by means of its most prominent “Vertreter”, the so called Watchdog Timer.

The *Watchdog Timer (WDT)* is an fault detection service, which is, dependent from its implementation, also capable of detecting *control flow errors*. Technically, the WDT is a simple timer circuit with a specified threshold time. If this limit is reached, it changes its state, which triggers further actions, like restarting a component or activating another safety service [2].

The advantages of the WDT is the simple design, which reduces the additional complexity of the overall system, as well as the costs. Concerning the functional principle, there are three slightly different designs, with individual capabilities and advantages [2].

**Standard Watchdog Timer** . With this basic setup, the service mirrored by the WDT periodically sends a simple heartbeat signal, indicating that the service is alive and active. This signal resets the timer. If a predefined amount of time elapsed, without an incoming signal, it is assumed that a fault has occurred and the WDT changes its state [2].

In terms of *control flow errors* in a service, the execution sequence of the instructions of a service is changed, which means that the heartbeat signal may, or many not, be sent too late. Thus, in the latter case, the WDT is capable of detecting the error [2]. The probability of noticing such an error is higher the closer the threshold time is to the time between the heartbeat signals.

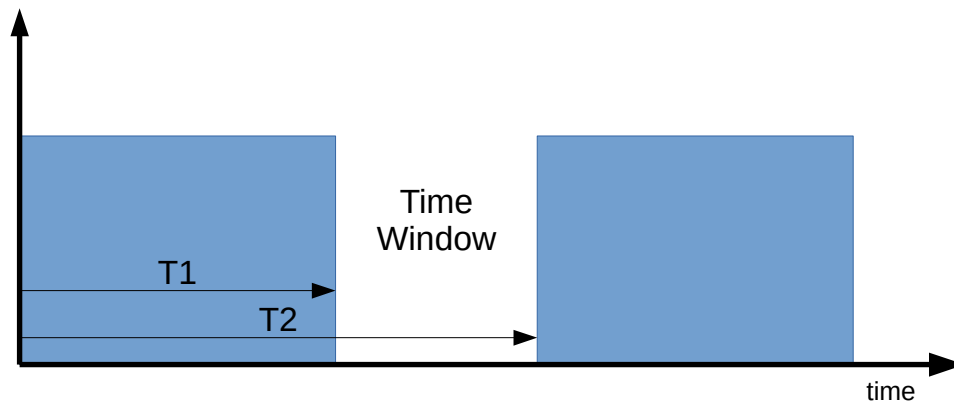
**Windowed Watchdog Timer** . The *Windowed Watchdog Timer* is an improvement of the standard version, which is capable of detecting most of the *control flow errors*. This is enabled by the application of two timers instead of one. With two timers the WDT is able to specify a time window, during which the heartbeat signal from the mirrored service must be received. The WDT is triggered if it receives a signal outside the window, or the timer reaches its threshold [2]. This is illustrated in figure 2.4 with the *time* on the x-axis and the Timers labeled  $T1$  and  $T2$ .

In case of an *control flow error* this signal is a bit ahead or past in time. The error detection coverage increases with narrowing the time window [2].

The advantage of this design is clearly that it allows the detection of more errors, but

on the other hand the implementation is slightly more complex.

Figure 2.4: Schematic design of a windowed watchdog timer [2].



**Sequenced Watchdog Timer** . This design is a further improvement of the *Windowed Watchdog Timer* and bases on the same principle. In contrast, to the other designs, the signal sent from the mirrored service carries a sequenced parameter. Only if the signal arrives in time and within the specified time window, this parameter is evaluated and compared to a parameter inside the WDT. If those match the sequence variable in the WDT is incremented and the timer reseted, starting the cycle over again [2]. The whole process is illustrated in figure 2.5.

The disadvantage of this design is the clearly higher complexity, for the *Sequenced Watchdog Timer* must be capable of holding and modifying state information as well as comparing it to received information.

## 2.2.2 Error detection service

### 2.2.3 Memory Protection Service

*Memory protection* is related to *safety* as well as *security*. It is a method for preventing processes or users from accessing memory that is not allocated to them.

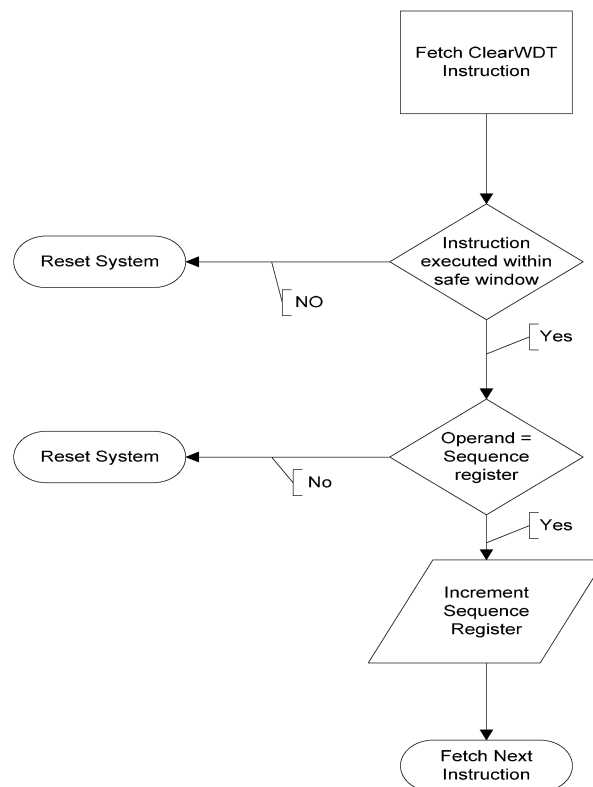
Former embedded systems, or such that are small in size, do not necessarily require memory protection mechanisms, but modern, large scale system, have several advantages ... and with the increase of computing power the overhead is no longer crucial [?].

There are several aspects that stress the need for memory protection in embedded systems:

- It can serve as fault and error detection and containment mechanism, preventing a failure of one service to propagate and infecting the whole system [?].



Figure 2.5: Schematic illustration of the working process of a *Sequenced Watchdog Timer* [2].



- It is an important aid in the development process and helps at debugging by identifying “illegal” behaviour of erroneous services [?].
- It is also crucial from a security point of view, because it prohibits unauthorised access.

It is bizarre that memory protection is nowhere mentioned in connection with service oriented architectures

## 2.2.4 Orchestration Management Service

## 2.3 Use Case: Error Detection Service

### 2.3.1 Service Investigation/Planning

### 2.3.2 Service inventory analysis

### 2.3.3 Service oriented analysis

### 2.3.4 Service oriented design

Figure 2.6: Example architecture for an *fault detection service*, like a WDT.

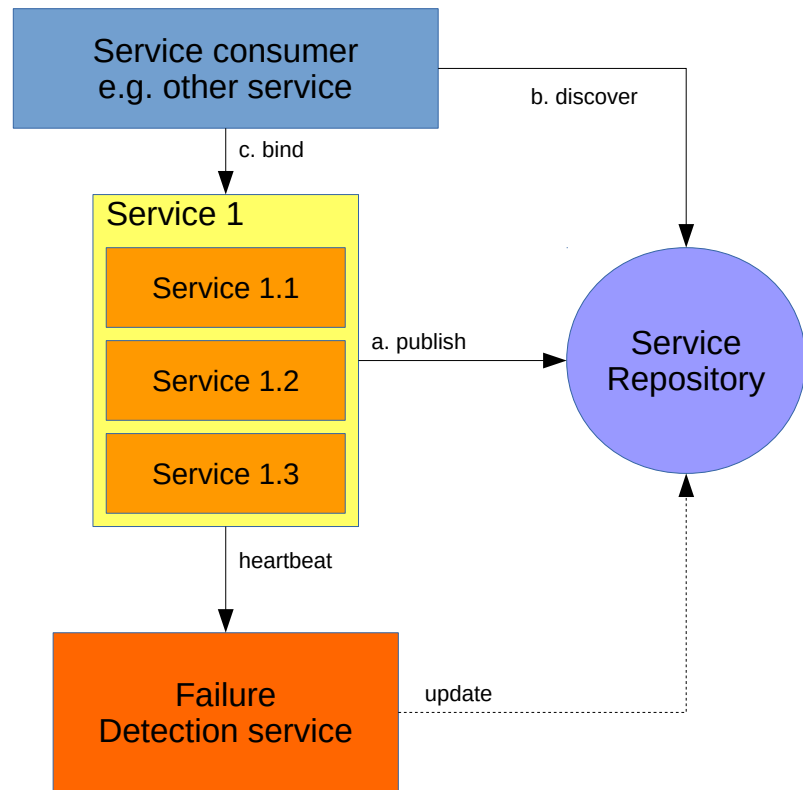
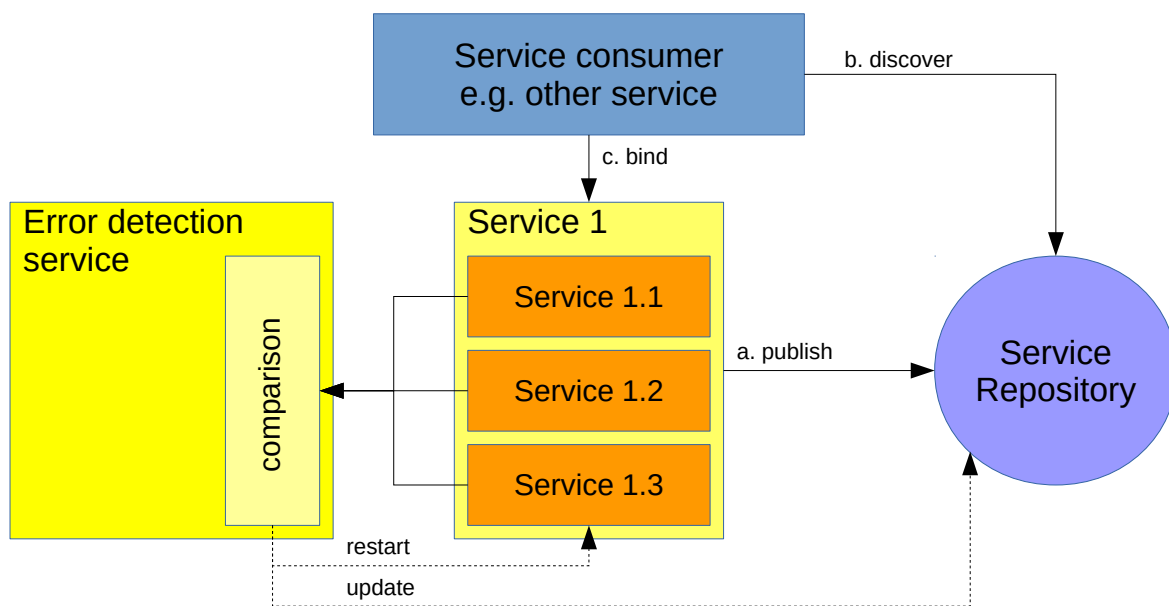


Figure 2.7: Example architecture for an *error detection service*.



## **Chapter 3**

### **Results**

## **Chapter 4**

### **Discussion**

# Chapter 5

## Conclusion

- still a long time until real application + many advantages of the soa paradigm - vermarktung von soa schwierig - ¿ was ist der vorteil von auto mit soa + wsl unvermeidbar für selbstfahrende fahrzeuge und etc. - das problem ist safety critical ...

# Bibliography

- [1] Mario Gleirscher, Andreas Vogelsang, and Steffen Fuhrmann. A model-based approach to innovation management of automotive control systems. In *Software Product Management (IWSPM), 2014 IEEE IWSPM 8th International Workshop on*, pages 1–10. IEEE, 2014.
- [2] Ashraf M El-Attar and Gamal Fahmy. An improved watchdog timer to enhance imaging system reliability in the presence of soft errors. In *Signal Processing and Information Technology, 2007 IEEE International Symposium on*, pages 1100–1104. IEEE, 2007.
- [3] AUTOSAR. Glossary, autosar release 4.2.1, 2014.
- [4] Douglas Rodrigues, Rayner15 de Melo Pires, Júlio César Estrella, Emerson Alberto Marconato, Onofre Trindade, and Kalinka Regina Lucas Jaquie Castelo Branco. Using soa in critical-embedded systems. In *Internet of Things (iThings/CPSCoM), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, pages 733–738. IEEE, 2011.
- [5] Christian Buckl, Stephan Sommer, Andreas Scholz, Alois Knoll, Alfons Kemper, Jörg Heuer, and Anton Schmitt. Services to the field: An approach for resource constrained sensor/actor networks. In *The Fourth Workshop on Service Oriented Architectures in Converging Networked Environments (SOCNE 2009) - extended version*. IEEE, 2009.
- [6] Andreas Scholz, Stephan Sommer, Christian Buckl, Gerd Kainz, Alfons Kemper, Alois Knoll, Jörg Heuer, and Anton Schmitt. Towards and adaptive execution of applications in heterogeneous embedded networks. In *Software Engineering for Sensor Network Applications (SESENA 2010)*. ACM/IEEE, 2010.
- [7] Stephan Sommer, Christian Buckl, Gerd Kainz, Andreas Scholz, Irina Gaponova, Alois Knoll, Alfons Kemper, Jörg Heuer, and Anton Schmitt. Service migration scenarios for embedded networks. In *The Fifth International Workshop on Service Oriented Architectures in Converging Networked Environments (SOCNE 2010)*. IEEE, 2010.

- [8] Peter Marwedel. *Embedded system design: Embedded systems foundations of cyber-physical systems*. Springer Science & Business Media, 2010.
- [9] Dae-Hyun Kum, Joonwoo Son, Seon-bong Lee, and Ivan Wilson. Automated testing for automotive embedded systems. In *SICE-ICASE, 2006. International Joint Conference*, pages 4414–4418. IEEE, 2006.