Safety-relevant development by adaptation of standardized safetybconcepts in AUTOSAR 4.0

M. Niklas, S Voget, J. Mottok

Safety-relevant development by adaptation of standardized safety concepts in AUTOSAR 4.0

M. Niklas[1], S. Voget[2], J. Mottok[3]

1: Continental Engineering Services GmbH
Osterhoferstr. 17, 93055 Regensburg
Michael.Niklas@conti-engineering.com
2: Continental Automotive GmbH
Siemensstr. 12, 93055 Regensburg
Stefan.Voget@continental-corporation.com
3: Laboratory for Safe and Secure Systems, LaS³
Hochschule Regensburg, Seybothstr. 2, 93053 Regensburg,
juergen.mottok@hs-regensburg.de

Abstract: *The ECUs in a vehicle network need a reliable and safe interconnection between each other. This is especially valid for new functionality, e.g. for systems like drive-by-wire. Current published standards like the software standard AUTOSAR or the functional safety standard ISO26262 ease and enable the development of safe interconnections. Furthermore, to enable fast innovation cycles, an incremental adaptation and development process is necessary. The combination of the development of new functionality, the introduction of new standards, and the reuse of existing solutions is a big challenge for the development of the vehicle E/E network. This paper presents approaches to handle these challenges. For that, a classification of communication related errors is introduced. Solutions are given with the help of error detection mechanisms and means to adapt the safe communication concepts from AUTOSAR release 4.0 to an existing solution based on a former release.*

Keywords: Functional safety, dependable communication, E2E, error detection, automotive, AUTOSAR, model based development

1. Introduction

Behind the visible parts, a modern car is an electronic network of up to 100 electronic control units (ECUs) connected via several bus systems. The realization of significant part of the functionality is distributed among several ECUs. E.g. the software, that controls the lights of the indicator functionality, is distributed over up to eight ECUs in high end vehicles. Furthermore, some of the future functionality will not be realized in a loose set of side by side ECUs but needs a large amount of interrelationships. E.g. drive-by-wire [1] will need a very close and safe interlocking of ECUs across the network.

Usually, the functionality within a car is designed as a communication chain from a sensor (e.g. light on switch) to an actuator (e.g. the light). In such communication chains, faults may occur that lead, sooner or later, to failures. One risk is a break of data integrity within such a chain. That can lead to an unpredictable, unwanted behaviour of the actuator. Data integrity from the sensor up to the actuator has to be ensured.
Depending on the kind of failure, a set of SW mechanisms exists that help either to prevent or to detect them. Some of these mechanisms were introduced in the release 4.0 of the AUTOSAR standard. In this paper, we analyze these mechanisms due to their applicability in project environments that do not fully realize the AUTOSAR release 4.0. One important use case is the usage of AUTOSAR release 3.0, 3.1 or 3.2 which do not include all of these mechanisms but will still be used for several years in already started or planned developments of ECUs.

In this paper, we will consider and categorize communication errors that may occur in the communication within or between ECUs. A set of mechanisms is identified that help to detect the errors. For a subset of these mechanisms we show how they can be adapted effectively to former AUTOSAR architectures than AUTOSAR release 4.0 without violating the former versions.

2. Related work

A model based development approach has been applied since the beginning of AUTOSAR so this chapter will introduce in the first part model based development, go on with an introduction to the AUTOSAR standard and conclude with a short overview of functional safety.

2.1 Model based development

Today, for a rapidly growing amount of computer-based systems in automotive, software has become a key factor. Model-based development relies on the use of explicit models to describe software development activities and products.

Model-driven Engineering refers to a range of development approaches that are based on the use of software modeling as a primary form of expression.

The definition and use of complex development steps that are correct by design, the generation of proof obligations for a given transformation, requirements tracing, and documentation of the process is possible on available process and product models as given in AUTOSAR.

High safety requirements demand that the creation of software-intensive systems need a systematic engineering approach as given by model based development which is also used by the methodology and meta-model of AUTOSAR.

2.2 AUTOSAR introduction

The AUTomotive Open System ARchitecture (AUTOSAR) initiative develops an open, standardized software architecture for automotive electronics [2]. The partnership is focused on managing the growing complexity in the development of automotive electric/electronic architectures, with the aim of enabling new technologies and improving development efficiency – without making compromises on quality and corporate identity.

AUTOSAR mainly concentrates on three pillars [3] in the standardization. First, a layered architecture [4] for electronic control units (ECU) is defined and the lower layers, the basic software (BSW), are standardized on module level. Second, a methodology enables the configuration of systems within a collaboration process between OEMs and suppliers. Third, on the highest architectural layer, SW-components and their application interfaces are standardized. Also in safety relevant ECUs AUTOSAR gets in the focus of development environment.

2.3 Safety Requirements

When developing safety-relevant ECUs in the automotive domain, the automotive-specific safety norm ISO26262 [5] (which is derived from the IEC 61508 [6]) has to be applied. During the early stages of the conceptual development of such a product, a hazard analysis has to be made to detect single safety-critical situations the ECU could be affected with. Safety goals are derived for the ECU that lead to more specific safety requirements.

The process until this point in the development timeline is the so called functional safety management. To satisfy the safety requirements, technical safety concepts and development processes have to be used in the design phase of the whole system to assure high dependability. The satisfaction of the safety requirements leads to the achievement of the safety goals of the ECU.

With the communication between safety-relevant ECUs and the need of dependable communicated data also the communication part of the system has to fulfil safety requirements to achieve the dedicated safety goals. The ISO26262 [5] provides an error model and technical means to detect communication errors. Also, other publications provide information regarding dependable and safe communication. A survey of the common communication errors and detection mechanisms is introduced in the following.

3. Communication basics

The next part introduces possible errors in the communication between ECUs and lists error detection mechanisms. As inputs for this chapter mainly a paper of J.H Saltzer and colleagues [6], the books of Hermann Kopetz [8] and Peter Loew, Roland Papst and Erwin Petry [9] and the ISO26262 [5] are considered. The part dealing with communication errors takes the specification of AUTOSAR E2E Library [10] also into account.

3.1 Communication Errors

The possible errors in dependable communication are described in this chapter. A summary of all mentioned errors is given in table 1.

The first relevant error is the Data Loss. It occurs when part of the data or the whole data is lost during transmission. The origin can be a various number of faults like an EMI impulse or a partially, respectively permanent, damaged wire.
A second relevant error is the Repetition: The same data information is received in successive messages. As in the data loss, the fault can be of various origins. In this case, a software defect at the sender can also be the origin of repeated identical data.
The next error is the Timeout Error or Time Delay Error. A Timeout occurs if data is not received within an expected timeslot. The Timeout Error can only occur in a system with defined timing requirements. That means that the sender and the receiver have a common understanding of “time”, e.g. if the sender sends data every 20 ms and the receiver expects data every 20 ms.

Incorrect Sequence is an error that is typical for highly distributed systems and is defined as follows: The data arrives at the receiver in another sequence than originally sent. The cause for this error is often a system with positive probability that the sequence of the data is mixed up. This can happen in case of buffered communication or communication via several ECUs (e.g. in gateways).
The Insertion of an unintended message means that an additional message or a part of it is added in the communication stream. This is an error that occurs very unlikely and has its origin in hardware faults of the internal bus systems in a vehicle like in a CAN or FlexRay.
One of the most common errors in communication is the Data Corruption. Data Corruption is the violation of the information integrity of the transmitted data. The origins of Data Corruption are in most cases random hardware faults, e.g. a bit flip caused by an Electro Magnetic Interference (EMI).
An Addressing Error occurs if data is sent to the wrong destination and treated at receiver side as correct data. The reason can be a random hardware fault or a systematic software fault. Usually a system is protecting this by assigning unique IDs to the single data elements or to the sender and receivers.
The Masquerading Error goes a step further: in this case, a unique DataId mechanism exists in the system but the data is ”disguised” and therefore accepted although the data origin is not the one it pretends to be. In summary: The receiver side accepts data that is not from the intended sender but pretends to be from it (the correct and intended sender). The cause of this error can be a corruption in the DataId that leads to a false acceptance of single data. A security-relevant origin would be that something changes the DataId with intent.
The last considered error in this document is a Constant ”over-” Transmission: It can happen that due to a fault on the hardware level, different or the same messages are sent again and again, leading to a bus overload. This frequent retransmission is blocking the bus and, therefore, other safety relevant data could be detained from being sent.

Data Loss
Repetition
Timeout
Incorrect Sequence
Insertion of unintended data
Data Corruption
Addressing Error
Masquerading
Constant ”‘over-”’transmission

Table 1: Table of errors in communication

In summary, if considering a system, there are fewer errors than faults and, therefore, the detection on error level is easier and more sufficient than on fault level. Table 1 summarizes the single errors considered in this chapter.

3.2 Error detection mechanisms in communication

This section lists a set of single error detection mechanisms and gives a short description.
The single error detection mechanisms are:

- Hardware Redundancy: Sufficient to detect most of the errors. This is part of the system design and is achieved by providing two or more independent hardware communication channels.
- Time Redundancy: The same information is transmitted twice via two different messages in different timeslots.
- Checksum: A checksum is created by an algorithm for a data block. This checksum is transmitted and recalculated at the receiver side
- Sequence Counter/Number: The sender adds a Sequence Counter to the transmitted data. This Sequence Counter is then evaluated at the receiver that has stored the last valid received Sequence Counter.
- Message ID/Data ID: Each unique message in the distributed system has its own Data Id. By check of the Data Id e.g. addressing errors can be detected.
- CRC (Cyclic Redundancy Checksum) polynomials: The whole data block is used as base for a calculation of a polynomial division carried out by a polynomial generator to create a memory-dependent signature that is sent and recalculated on receiver side.
- Including State of sender/receiver in CRC calculation: The sender and receiver have the same amount of corresponding states. These states are numbered accordingly so that the value of the sender state corresponds to the value of the receiver state. If the CRC is carried out with no error, the consistency of the states can be evaluated indirectly.
- Parity bit: One additional bit is added to the data stream. The goal is to produce in the data stream an odd or even amount of digital “0” or “1”. Which digital value is selected and if odd or even parity is used depends on the design.
- EDC (Error Detection Codes)/ECC (Error Correction Codes): They allow not only the detection of corruption errors but also the correction of these errors. There exist several different algorithms for such codes that can be described by their Hamming Distance. The Hamming Distance describes how many bit failures of the code can be detected. A Hamming code with distance d can correct d-1 errors.
- Timeout by a priori knowledge: Detection of delays using the measurement of time on the receiver side with the knowledge of expected timing.

- Time stamp: The Time Stamp works only in a system that has a globally defined time base that is synchronized in the system. A Time Stamp is explicitly transmitted and checked on the receiver side.
- Plausibility and acceptance checks: It usually compares if the received value of the data is within the upper and lower boundaries. The boundaries can be static if just accepting certain ranges or they can be dynamic if verifying the plausibility of the data. With the dynamic verification, the first derivative of the data is evaluated to be within the boundaries for the technical plausible positive and negative gradient of data change.

Errors		Deletion	Repetition	Timing Delay	Incorrect Sequence	Insertion of unintended data	Data corruption	Addressing Error	Masquerading	Constant "over" Transmission
State of the art mechanisms	Hardware Redundancy	x	x		x	x	x	x		
	Time Redundancy	x					x			
	Information Redundancy						x			
	Checksum						x			
	Sequence Counter	x	x		x	x				
	Message ID/Data ID							x		
	CRC polynomials						x			
	State in CRC calculation							x		
	Parity bit						x			
	EDC/ECC						x			
	Timeout	x		x						
	Time stamp	x		x						x
	Plausibility checks						x		x	
	Identification procedure							x	x	
	Cryptographic techniques						x		x	
	Retry mechanisms	x					x			
	Acknowledgement	x	x				x			

Table 2: State of the art mechanisms and their error detection capabilities

- Information Redundancy: The same information is included twice in one message
- Cryptographic techniques to detect unauthorized manipulation: The whole data is encrypted by an algorithm at the sender side and decrypted with an algorithm according to the sender at the receiver side to detect violations of the data.
- Identification procedure: The sender and receiver work with identification keys to check if the destination and source of the data are valid. Usually this is achieved by bidirectional exchange of identification messages.
- Retry mechanisms: In case of data loss or data corruption, the receiver sends on a not successful reception a retry message to the sender. This mechanism uses again bidirectional communication and therefore contributes to a higher busload.
- Acknowledgement: It is very similar to the Retry mechanism. But in this case the sender sends the message until it gets an acknowledge message from the receiver.

Table 2 in this chapter gives an overview of the relationship between the single mechanisms and the errors they are capable to detect (or contribute to their detection, in combination with other error detection mechanisms).

4. Specific communication solutions

For the detection of single communication errors, the AUTOSAR Standard provides in the current Release 4.0 mainly three concepts that can be used independently from each other. Which one shall be used and to which extent depends on the needs derived from the safety goal analysis and the corresponding failure mitigation strategy. Two of these concepts are allocated in the COM module [11], which is the base for data communication between different ECUs. The third concept is realized by an End-To-End protection library [10]. All three concepts can be used independently from the target bus system. The two COM module concepts are multiple communication links and PDU counter. The principles of this mechanism are described in the following part.

The multiple communication link routes the data over two or more configurable diverse bus systems which can be of different bus type (e.g. FlexRay and CAN) or the same bus type but different physical buses (e.g. two CAN buses). With this redundancy mechanism, it is possible to detect all errors with a high degree of diagnostic coverage. The PDU counter mechanism is a more simple approach to detect data losses, out of sequence errors, repetition and insertion of unintended data. The Receiver initializes its counter value on the first reception of a valid message (PDU) and compares this value to the

next received counter value. Both concepts are configurable, independent for each message, and the according behavior is realized by the COM module.

The E2E Library extends in principle a certain message with the help of several mechanisms to assure a high coverage in the detection of errors. The mechanisms are CRC calculation, double inverse data, timeout detection, message counter, data Id and counter value dependent data id. The mechanisms are implemented by three different profiles. Each of the three profiles uses another subset of these mechanisms leading to a different algorithm. The details of the algorithm and the connection to the data to be protected are modeled in the system template [12] of the AUTOSAR standard.

5. Adaptations of Safety Concepts

All three concepts have in common that the intended behavior is configurable. For the E2E library the meta-model part of the System template [12] is relevant. For the mechanisms allocated in the COM the parameter definition template needs to be adapted. Considering the fact that most development projects of ECUs use AUTOSAR release 3.0 and 3.1, an adaptation of the mechanisms to these releases is of highly interest. However in former releases, priot to 3.0, this is not considered. The adaptation has to be done on three different levels: First the model information, second the generator behavior and third the static source code.

5.1 Meta-model adaptation

Comparing the releases 3.0/3.1 and 4.0, the E2E library model is only an addition and can be applied easily. The model information is stored in xml format, so the according xml containers have to be readable and editable in the tools that handle configuration and code generation. All the information for the E2E library is concentrated in one model element that is referencing dependent model elements like e.g. the data to be protected. With the AUTOSAR release 3.2 the adaptation is applied for the E2E library in the way as described above: The AUTOSAR 3.1 meta-model is extended by the modeling containers needed for the E2E Library. Due to the fact that the E2E Library is already useful for 3.0 and 3.1 projects, the solution can also be done for releases before Release 3.2. But, this needs a vendor-specific adaptation of the used ECU configuration tooling. Such adaptations have been done, e.g. for the configuration tool CESSAR-CT provided by Continental Engineering Services.
In the case of the two COM modules' related technical safety concepts, the parameter definition file has to be extended by the model information in form of adding vendor-specific parameters. As the structure of the 3.0 and 3.1 COM configuration is similar to the one of the 4.0 COM, there is no major impact on extending the parameter definition file by using vendor-specific parameters to configure the multiple communication links and the PDU counter.

5.2 Code Generator adaptation

In a second step, the code generators for the COM module have to be extended to consider the vendor-specific parameters. With the extension of the generators, the code generation of the additional parts is also possible. There is no need to change the generators but the need to add some further functionality without modifying the old generator behavior. This also allows for a regression test strategy including only the verification of the new parts. .

5.3 Code adaptation

The same can be applied to the static source code. For the E2E library the static source code is the same as in 4.0. In case of the COM module the need arises to assure a 3.0/3.1 behavior even if the concepts are NOT used, which is realized by compiler switches that are generated depending on the content of the configuration. In case there is no element configured to be protected by one of the two means, a compiler switch can "turn off" the behavior in the static code to reduce the footprint and runtime of the model to the original 3.0/3.1 behavior.

The adaptation of the single concepts to the former release 3.0/3.1 is possible based on this approach for all relevant levels: model, generator and code as well as the corresponding tests.

6. Conclusion

The introduction in an early stage of the development i.e. in the meta-model leads to the advantage that the changes in the source code are not as extensive as otherwise but there is also the need of adaptation of the generator. The second advantage is the easy introduction of additional functionality compared to proprietary software development. The adaptation of the concept E2E has already taken place in the development and is used in current projects to contribute to the functional safety goals of the ECUs. This approach could also be used for further safety concepts like program flow monitoring in the Watchdog Manager (WdgM) [13].

7. Acknowledgement

The authors would like to thank all their colleagues for their help with the adaptation of the concepts, review sessions, and, therefore, the contribution to the results of this paper.

8. References

[1] E.A. Bretz: *"By-Wire Cars Turn the Corner*", IEE Spektrum April 2001, 2001

[2] S. Fuerst, J. Moessinger, S. Bunzel, T. Weber, F. Kirschke-Biller, P. Heitkaemper, G. Kinkelin, K. Nishikawa, K. Lange: *"AUTOSAR – A Worldwide Standard is on the Road"*, 14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden, 2009

[3] S. Fuerst: *"AUTOSAR – An open standardized software architecture for the automotive industry"*, 1st AUTOSAR Open Conference & 8th AUTOSAR Premium Conference, Detroit, 2008

[4] AUTOSAR: *"Layered Software Architecture R4.0*", AUTOSAR, 2009

[5] ISO: *"ISO 26262 – Road Vehicles – Functional Safety*", ISO, 2011

[6] IEC: *"IEC 61508 – Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems*", IEC, 2000

[7] J.H. Saltzer, D.P. Reed, D.D. Clark: *"End-To-End Arguments in System Design*", ACM Transactions on Computer Systems, 1984

[8] H. Kopetz: *"Real Time Systems – Design Principles for Distributed Embedded Applications*", Kluwer Academic Publisher, 1997

[9] P. Loew, R. Pabst, E. Petry: *"Funktionale Sicherheit in der Praxis*", dpunkt, 2010

[10] AUTOSAR: *"Specification of SW-C End-to-End Communication Protection Library R4.0*", AUTOSAR, 2009

[11] AUTOSAR: *"Specification of Communication R4.0"*, AUTOSAR, 2009

[12] AUTOSAR: *"System Template R4.0*", AUTOSAR, 2009

[13] AUTOSAR: *"Specification of Watchdog Manager R 4.0*", AUTOSAR, 2009

9. Glossary

AUTOSAR: Automotive open software architecture

COM: AUTOSAR Communication Module

CRC: Cyclic redundancy checksum

E2E: AUTOSAR End-to-End library

ECC: Error correction code

ECU: Electronic control unit

EDC: Error detection code

EMI: Electro-Magnetic Interference

IEC: International Electrotechnical Commission

ISO: International Organization for Standardization

WdgM: AUTOSAR Watchdog Manager