OCORA

Open CCS On-board Reference Architecture

Assessment Strategy and Software Development according to EN 50128 resp. EN 50657

Document ID: OCORA-TWS09-031

Version: 1.00

Date: 16.05.2022

Management Summary

This document is a contribution of DB Systemtechnik GmbH to the project OCORA. It is intended to support the elaboration of a process for the development of software for safety-related embedded systems up to SIL4 according to the standard DIN EN 50128:2012-03 (EN50128) and taking into account DIN EN 50657:2017-11 (EN50657).

This document deals with a selection of requirements of the EN50128 resp. EN50657 standard for the development of software and applies in the project OCORA.

This document is valid in the context of software development in the project OCORA.

This document is provided by DB Systemtechnik GmbH and handed over to the project OCORA for use or further processing. This includes a suitable configuration management by OCORA.

Revision history

Version	Change Description	Initial	Date of change
1.00	Version for OCORA Release 2 *(Based on DB document 68388-00n-TT.TVE42-DO-V01.01_Verification_Assessment Strategy and Software Development)*	JL	16.05.2022

Table of contents

References

Reader's note: please be aware that the numbers in square brackets, e.g. [1], as per the list of referenced documents below, is used throughout this document to indicate the references to external documents. Wherever a reference to a TSI-CCS SUBSET is used, the SUBSET is referenced directly (e.g. SUBSET-026). OCORA always reference to the latest available official version of the SUBSET, unless indicated differently.

[1] OCORA-BWS01-010 – Release Notes

[2] OCORA-BWS01-020 – Glossary

[3] OCORA-BWS01-030 – Question and Answers

[4] OCORA-BWS01-040 – Feedback Form

[5] OCORA-BWS02-010 – Executive Summary Slide Deck

[6] OCORA-BWS02-020 – Program Slide Deck

[11] OCORA-BWS03-010 – Introduction to OCORA

[13] OCORA-BWS04-010 – Problem Statements

1 Introduction

1.1 Purpose of the document

As a concept, the purpose of this document is to discuss an assessment strategy for electronic control units and various aspects related to software development according to the EN50128 resp. EN50657. The development process will be detailed in associated planning documents.

The application of processes to safeguard reliability, availability, maintainability and safety of the products resp. the application according EN50126 is required and is not part of further considerations. The development and approval of the electronic hardware according to EN50129 resp. EN50155 is also required and is only considered in this concept as part of the configuration and integration.

The document is not intended as a guide for software development or assessments neither for Generic Products or Generic Applications nor Specific Applications.

This document is addressed to experts in the CCS domain and to any other person, interested in the OCORA concepts for on-board CCS. The reader is invited to provide feedback to the OCORA collaboration and can, therefore, engage in shaping OCORA. Feedback to this document and to any other OCORA documentation can be given by using the feedback form [4].

If you are a railway undertaking, you may find useful information to compile tenders for OCORA compliant CCS building blocks, for tendering complete on-board CCS system, or also for on-board CCS replacements for functional upgrades or for life-cycle reasons.

If you are an organization interested in developing on-board CCS building blocks according to the OCORA standard, information provided in this document can be used as input for your development.

Before reading this document, it is recommended to read the Release Notes [1]. If you are interested in the context and the motivation that drives OCORA we recommend to read the Introduction to OCORA [11], and the Problem Statements [13]. The reader should also be aware of the Glossary [2] and the Question and Answers [3].

1.2 Applicability of the document

The document is currently considered informative but may become a standard at a later stage for OCORA compliant on-board CCS solutions. Subsequent releases of this document will be developed based on a modular and iterative approach, evolving within the progress of the OCORA collaboration.

1.3 Context of the document

This document is published as part of an OCORA Release, together with the documents listed in the release notes [1]. Before reading this document, it is recommended to read the Release Notes [1]. If you are interested in the context and the motivation that drives OCORA we recommend to read the Introduction to OCORA [11], and the Problem Statements [13]. The reader should also be aware of the Glossary [2] and the Question and Answers [3].

During assembly, software is installed on different controller of each automobile according to its individual configuration. During maintenance, updates can be installed on each controller. And even if a controller shall be substituted, necessary upgrades of other software or hardware are recommended. And all this on a tremendous number of configurations.

To do so in the domain of rail systems, and especially in case of widely distributed electronic control units, like the European Train Control System (ETCS), in this document a strategy for assessments as well as appropriate methods for software development will be proposed as a concept. The key element is the development of a Specific Application based on a specific configuration of a Generic Application.

The concept corresponds both to the relevant principles of information technology and to the applicable normative requirements. Even if future applications with configurations of electronic control units from different providers are in focus, aspects are discussed to improve the efficiency of the engineering process and evaluations in general and to apply them soon.

1.4 Declaration

This document is exclusively for use within the framework of the project OCORA and presupposes the standard DIN EN 50128:2012-03 resp. DIN EN 50657:2017-11 also in the sense of copyright or author's right. Reproduction of any kind, including copying of parts, is not permitted.

In the standard DIN EN 50128:2012-03 resp. DIN EN 50657:2017-11, reference is made to the possibility that various aspects of the standard may affect patent rights. This also applies to this document. Therefore, no guarantee is given that the procedures etc. described in this document are free of third-party property rights.

The authors of the standard DIN EN 50128:2012-03 resp. DIN EN 50657:2017-11 or the responsible organisation declare that they are not responsible for the identification of corresponding patent rights. This also applies to this document.

The information contained in this document has been compiled to the best of our knowledge and with care. Nevertheless, errors cannot be completely ruled out. The information contained in this document does not claim to be complete. Therefore, the information contained in this document is not subject to any obligation or guarantee of any kind. The authors of this document accept no responsibility of any kind and assume no liability whatsoever arising in any way from the use of this information or parts thereof.

Should this document use common names, trade names, product designations, etc., this does not entitle the user to assume, without special identification, that such names are to be regarded as free within the meaning of the laws on trademarks and service marks and may therefore be used by anyone.

1.5 Definitions

References, names, terms and abbreviations as well as notations are to be used consistently in the context of software development according to the standard EN50128 resp. EN50657. These are defined within the scope of the following subchapters for the scope of this document.

Names, terms and abbreviations, unless specified in this document, are taken from EN50128 resp. EN50657 and IEC 60050-351:2013. If contradictions arise, the definitions from these standards shall apply.

1.7 Relevant Standards

Reference	Source
EN50126*)	EN 50126:2017 – Series; Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety
EN50128*)	EN 50128:2011; Railway applications. Communication, signalling and processing systems. Software for railway control and protection systems
EN50657*)	EN 50657:2017; Railways Applications - Rolling stock applications - Software on Board Rolling Stock
IEC61160*)	IEC 61160:2005; Design review
IEC60050*)	IEC 60050-351:2013; International electrotechnical vocabulary - Part 351: Control technology
ISO/IEC250nn*)	ISO/IEC 250nn – Series; Systems and software engineering. Systems and software Quality Requirements and Evaluation
ISO/IEC9126	ISO/IEC 9126 – Series (withdrawn); Software Engineering – Product Quality
*) The referenced standard corresponds to the respective available complete edition. In the context of the further development of this strategy, however, corresponding additions or replacements shall also be considered in accordance with their validity.	

2 Assessment Strategy

Generic Products, Generic Applications as well as Specific Applications shall be developed according to EN50126 and in compliance with goals to be reached resp. with requirements about reliability, availability, maintainability and safety. The hardware of the electronic control units shall be developed according to EN50129 resp. EN50155. The software units shall be developed according to EN50128 resp. EN50657. According to assessed combinations of the electronic hardware units and/or software units, configurations shall be approved. This shall include application conditions as well as referring necessary algorithms, parameters and data if necessary. In general, each, Generic Products, Generic Applications as well as Specific Applications shall include assessment and approval.

Let's have a closer look at the Generic Product as well as the Specific Application and then discuss further aspects of the Generic Application. But let's not forget to enlighten the motivation.

2.1 Generic Product

According to assessed combinations, electronic hardware units as well as hardware related software units shall be approved as a Generic Product. According to assessed combinations, software units providing functions and features of operation system and/or communication, but not part of the application layer, shall be approved as part of or as a Generic Product.

2.2 Specific Application

An efficient way to develop a Specific Application is to assemble already existing parts: configurate the units according to specific requirements and add parameters as well as other data according to the configuration and the specific requirements. The related development process shall be streamlined according to Chapter 8 of the EN50657 resp. EN50128, focussing on the configuration and as far as possible safeguard only those parts of the configuration, parameters as well as data not been safeguarded before. In no other configuration ever before.

This concept presupposes that the following aspects are or will be fulfilled:

1) The Specific Application shall be based on a Generic Application.
2) As part of the Generic Application and based on a generic model, an architecture shall be provided as data.
3) Standard interfaces and functions as well as supporting applications shall pe provided.
4) The traceability shall be part of the Generic Application.
5) The specific architecture shall be the basis of the contract, assessment as well as the approval of the Specific Application.
6) The specific architecture as well as necessary artefacts for development shall be providable and be comparable as data.
7) As a conclusion a central organisational unit is highly recommended providing management support and data processing accordingly.

2.3 Generic Application

A Generic Application shall prepare Specific Applications and shall contain both: Generic Products and generic software as part of the application layer. Generic software shall be developed according to the EN50128 and considering Chapter 7.8 of the EN50657. As part of this concept, a Generic Application shall meet the aspects mentioned as part of the reflections before about a Specific Application.

2.4 Conclusions

The following tactic can be derived based on the previous reflections:

1) The Specific Application shall be based on a Generic Application.
    a) A Generic Application shall be based on Generic Products according EN50126
        i) Document structure of a Safety Case according to EN50129
        ii) Development process according to the EN50128 and considering Chapter 7.8 of the EN50657
    b) A Specific Application shall be based on a Generic Application according EN50126: Systems configured by application data or algorithms
        i) Document structure of a Safety Case according to EN50129
        ii) Development process considering Chapter 8 of the EN50128 resp. EN50657
2) As part of the Generic Application and based on a generic model an architecture shall be provided as data:
    a) Verry focussed and simple, preferably in a relational format, the architecture shall comply with the EN50128 resp. EN50657 standard.
    b) The architecture shall address functions, units, interfaces, parameters and data as well as necessary artefacts for development, like tests and results.
    c) Associated with the application conditions, a set of rules shall be provided as part of the architecture, so that a Specific Application can be derived.
    d) The architecture shall consider both: configuration and compatibility management.
3) Standard interfaces and functions as well as supporting applications shall pe provided:
    a) Setup configuration and parameters
    b) Integration and tests
    c) Administration
    d) Service
4) The traceability shall be part of the Generic Application.
5) The specific architecture shall be the basis of the contract, assessment as well as the approval of the Specific Application.
6) The specific architecture as well as necessary artefacts for development shall be providable and be comparable as data.
7) As a conclusion a central organisational unit is highly recommended providing management support and data processing accordingly.
    a) An overall data flow shall consider the development process based on a Generic Application. The distribution of necessary tasks between different manufacturers (vehicle and ETCS) and operators must be taken into account.
    b) An overall configuration and compatibility management shall be derived.

3 Application

The cases to be discussed will be derive from simple structures. A monolithic system, including the hardware, the operating system and communication software as well as the application, hereby is rejected and excluded from further considerations.

It is assumed in general that the hardware as a Generic Product is configurable. This includes associated software. Next, there are the following questions to be focused on:

a) Can the software be configurated in terms of operating system, communication and application?

b) Is there a distinction between algorithms, parameters and data?

c) And is there a distinction between generic and specific application?

An application could be recognised as Generic Application if there is a distinction between algorithms and parameters. So potentially, a Specific Application can be derived by a configuration and a set of parameters even if no specific algorithms can be added. But if there is no distinct application the dependency about hardware, operating system and communication is evident. So consequently, further reflections will be focused on the combinations as stated in [

Table 1].

Table 1

Distinct			Specific		
Application	Communication	Operating System	Algorithms	Data	Parameters
1	0	0	0	0	1
1	0	0	0	1	0
1	0	0	0	1	1
1	0	0	1	0	0
1	0	0	1	0	1
1	0	0	1	1	0
1	0	0	1	1	1
1	1	1	0	0	1
1	1	1	0	1	0
1	1	1	0	1	1
1	1	1	1	0	0
1	1	1	1	0	1
1	1	1	1	1	0
1	1	1	1	1	1

4 Integration

Compatibility should also be considered in case of substitutions, updates and upgrades. This shall include the integration of units from different providers (X-Compatibility). To reduce the complexity of such a model is highly recommended.

Let’s explore a configuration pretending a distinct application and enables specific parameters as well as specific algorithms:

- Start with the names of a function and of an interface being required from the perspective of the application.
- Next, use a simple pattern to list several units, e.g. to process the input, the state and the output as well as the required function.
- Using tables, map relations including a simple data flow and derive a simple control flow.
- Maybe picture the structures using a rectangles and arrows.
- Define an input and an output signal as well as a constant, e.g. an offset of the required function.
- Map the interface and the signals and add the threshold of the signals.

And that’s it: A Specific Application could be developed. Doing all the stuff according to the valid standards, including to refine the architecture, the application can be certified.

- But let’s substitute the constant with a parameter as well as an associated threshold and add a unit to provide the parameter. Only this unit will contain the specific part of the application.

Without this specific unit: The application, including an interface to provide the parameter, can be recognised as the Generic Application.

- Let’s add another unit but with a specific function, e.g. to attenuate the output. At least, an appropriate interface shall be provided by the Generic Application.

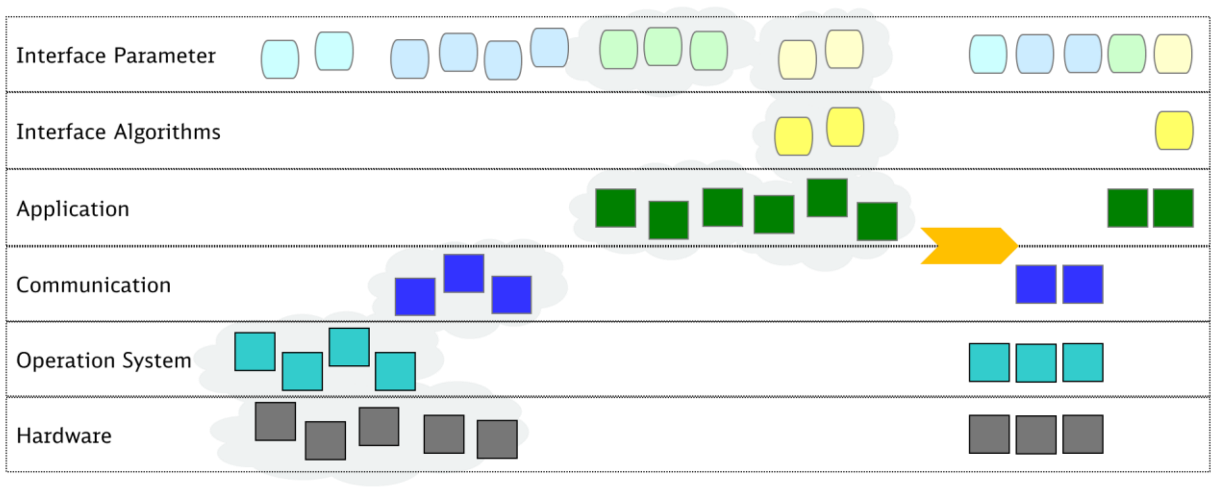
Step by step, a concept of generic and specific application was demonstrated. In this context, cases of integration, retrieval and compatibility of units from different manufacturers (X-Compatibility) as well as update and upgrade shall be analyzed. This in turn should provide the basis for the short-, medium- and long-term strategy for development and approval. It should be noted that for various reasons, practical software development does not tend towards a consistently systematic respectively formalized way of working. A high level of understanding at decision-making levels for organizationally oriented methods may also contribute to their preference.

4.1 Generic Application

When developing application software, units should be identifiable and categories such as hardware, operating system, communication and application should be assigned. In addition, identifiable interfaces for parameters are necessary and advantageous for extensions by specific algorithms.

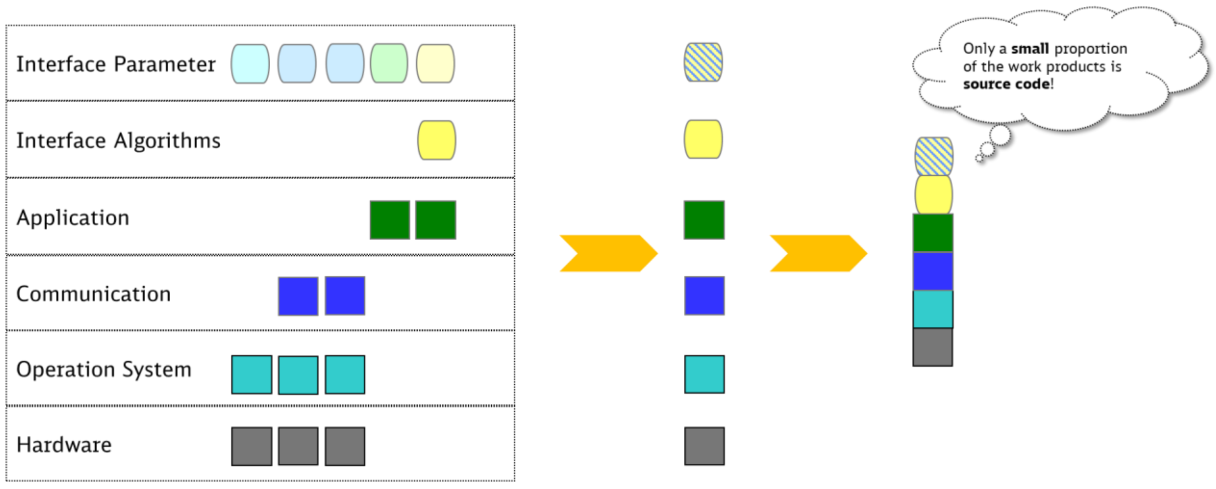
In the following figure, the units are assigned to the categories on individual levels and are also distinguished in color. The units with the rounded corners represent interfaces. Their slightly lighter coloring in turn marks the relation of the parameters to other units, such as parameters related to the hardware or the operating system.

Figure 1



Possibilities for configuring units and interfaces as well as parameterization can be limited based on sets of rules. If a configuration is subjected to a complete development cycle, a generic application is realized in this way. It should be clearly emphasized that the units and interfaces represent above all a very high proportion of design and verification documentation.

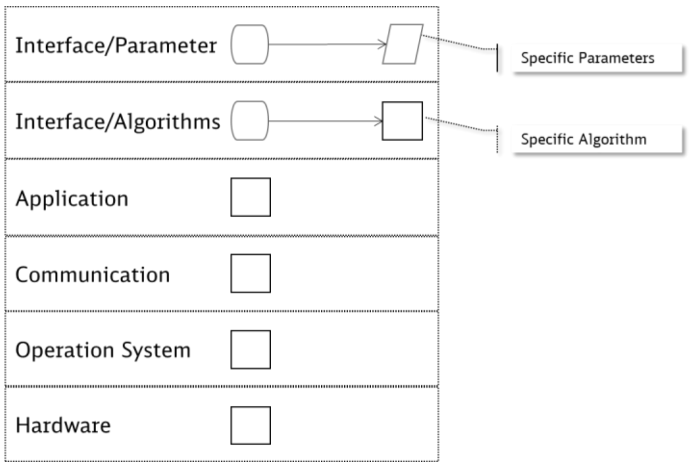
Figure 2



4.2 Specific Application

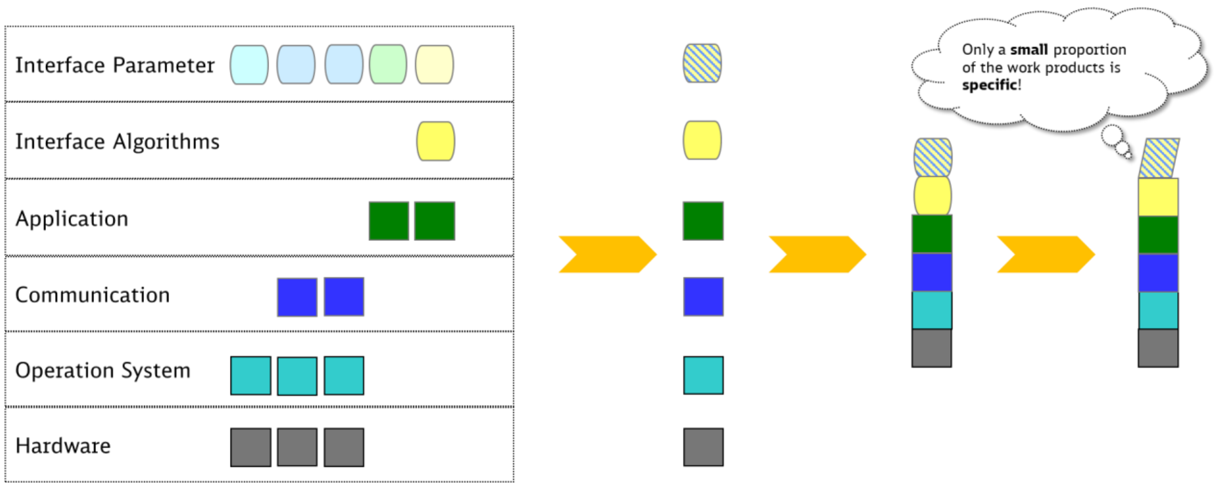
The interfaces are implemented in the context of specific applications, by concrete parameter values and, if necessary, by interface- and performance-compliant algorithms. In the context of the illustrations, this is understood by appropriate formatting of the unit.

Figure 3



Again, it should be emphasized that specific configuration, parameters and algorithms represent a very small proportion of design and verification documentation. These can essentially be anticipated by interfaces as well as regulations for configuration and parameterization.

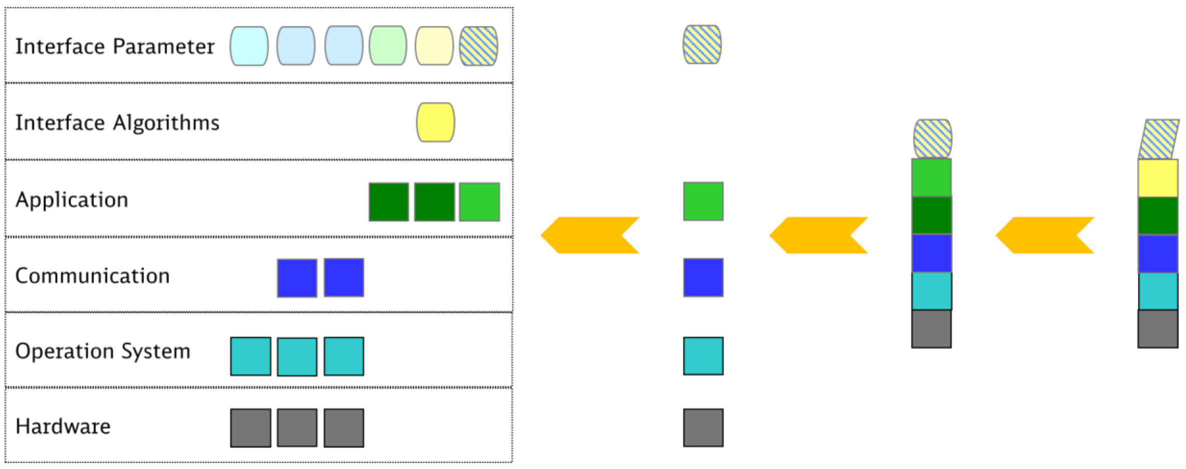
Figure 4



4.3 Retrieval

The retrieval of specific application information, with regard to concrete configurations, parameter values as well as interface and performance-compliant algorithms, in turn enriches the portfolio of the generic application.

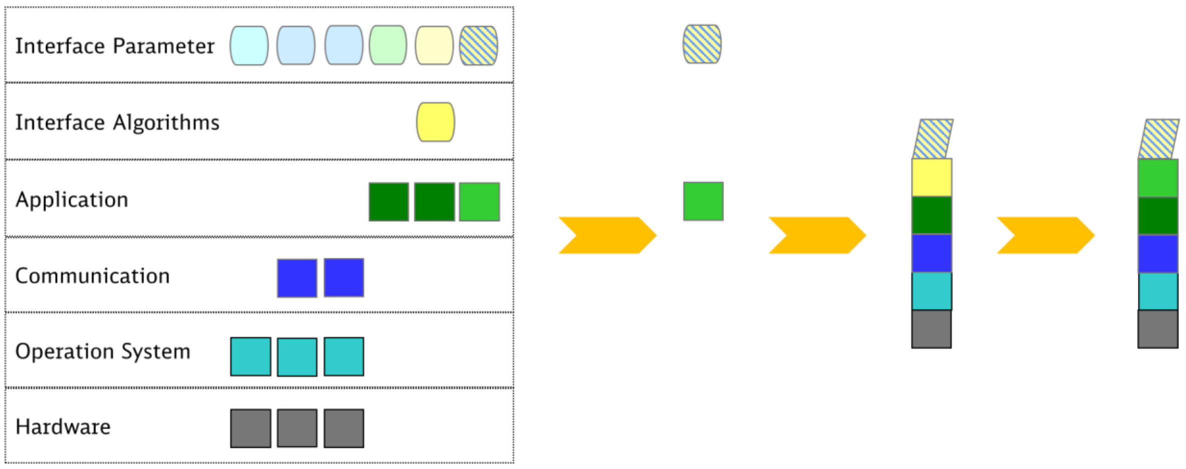
Figure 5



4.4 Update

Interface and performance conformity or the application of adequate techniques and measures also form the basis for efficient updates of specific units. Potentially, however, this principle can be applied to the entire, at least to the development of selected generic units. The fulfillment of corresponding requirements already results from the requirements for a system with SIL4: Performance Modeling.

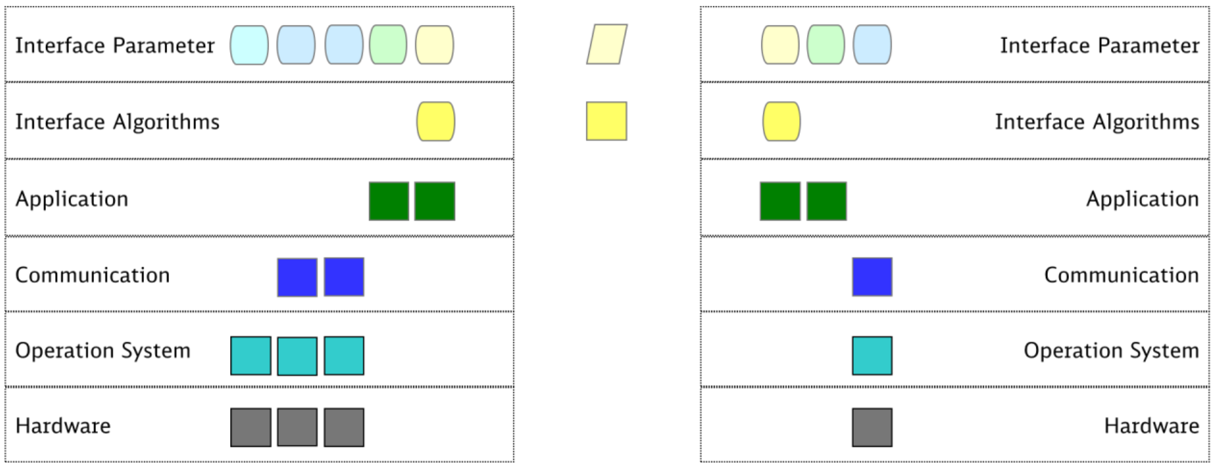
Figure 6



4.5 Compatibility

The conformity of standardized interfaces and performance requirements also forms the basis for the compatibility of units from different providers. However, this must be understood and coordinated as an integrated approach to the development of generic applications.

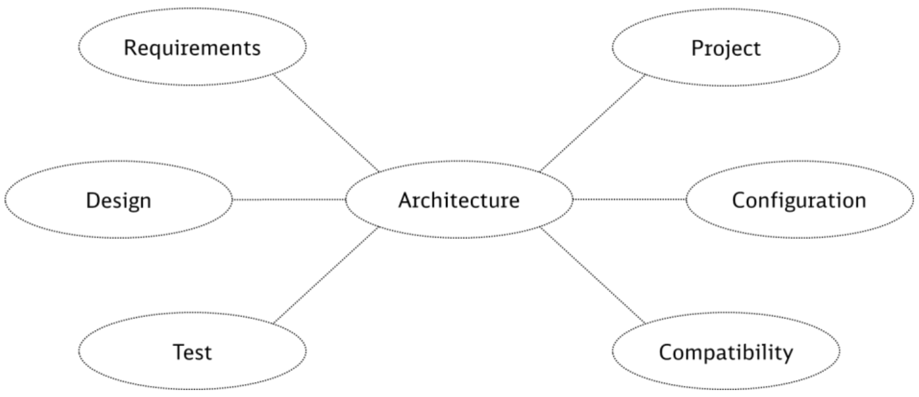
Figure 7



4.6 Significance of architecture

In connection with the previous considerations, the importance of architecture is to be underlined. Architecture is not only a central discipline of development, specification or execution of requirements, design, analysis as well as test and integration, rather, it forms the basis for project management as well as configuration and compatibility management as the basis for deployment.

Figure 8



6 Methodology

Table 1], for example to increase efficiency it is possible, to focus on already defined architectures. However, within the framework of development in accordance with EN50128 or EN50657, this also leads to techniques and measures that are to be applied anyway. This leads to the question: How efficiently are techniques and measures implemented?

With reference to the following sources as well as the standard DIN EN 50128 or DIN EN 50657, a list of questions is derived for the classification of methods.

- *Software Engineering eingebetteter Systeme; Liggesmeyer, Rombach Hrsg.;*

  2005; Spektrum Akademischer Verlag, Elsevier GmbH; dort Kapitel 11:

  Formale Entwicklungsmethoden und Analysetechniken; Santen
- *Sicherheitsgerichtete Echtzeitsysteme; Halang, Konakovsky;*

  2. bzw. 3. Auflage; 2013 bzw. 2018; Springer-Verlag GmbH Deutschland

The design of a safety-relevant embedded system requires the application of the principle of simplicity. This principle is also relevant for the application of methods. As part of the specification of requirements or an architecture, for example, decision tables or state machines can be used. The verification of completeness and consistency or the generation of suitable tests is thus possible in a simple way. As part of the implementation, these can also be used directly.

In particular, methods with formalized graphical representation, such as function block diagram, sequential schedule or diagrams according to the Unified Modeling Language (UML), support the principle of simplicity both in the application and in the representation. In the context of safety-relevant PLCs, function block diagram and sequential schedule are proposed as graphically based methods and inherently safe paradigm.

Development environments, such as SELECTRON or CODESYS, integrate methods according to the IEC 61131-3 standard, such as function block diagram, sequential schedule and the corresponding text-based language Structured Text, and supplement them, for example, with state diagrams according to UML. The development environment Enterprise Architect also integrates diagrams according to UML. The application of the mentioned graphical methods is focused on the design but can be integrated into the development process up to the area of implementation through graphical or text-based programming languages. However, such graphically formalized solutions do not have the concepts to provide formal proof of correctness.

A major challenge in practice is the number of combinations of input data or, above all, the resulting states of an embedded system. The resulting complexity of safety-relevant systems is also due to corresponding requirements in terms of reliability, availability, maintainability and safety. Unconsidered states of a system can have fatal consequences. Here, the handling of corresponding models, also by means of graphically supported tools, can reach its limits. Top-down approach, design patterns, modularization, etc. can make a significant contribution to master corresponding projects. With appropriate CAE tools, the handling of methods can be supported or different methods can be integrated on the basis of suitable interfaces.

The outputs of a system are determined by its inputs as well as its internal state, whereby the inputs can also cause changes in the internal state. Corresponding model-based specifications describe the possible states or state transitions. Formal methods are characterized by a mathematically strict semantics. This ascribes clear meaning to the elements and units of the corresponding models. The symbolic analyses of the entire state space, which were possible on this basis, supplemented common syntactic and type-based checks. This makes it possible to prove the correctness for all possible input variables. Therefore, the application of formal methods is recommended by the standard DIN EN 50128 or DIN EN 50657.

The standard DIN EN 50128 and DIN EN 50657 distinguish between formal methods for parallel and sequential systems or sequential aspects of parallel systems. Depending on whether formal models focus on the possible states of a system and their transformations or on possible state sequences, these can also be classified as static or dynamic models.

A static model describes the data that determines the internal state space. For valid states, it describes relationships that must always apply between these data. And it describes what operations can be performed under permissible conditions, including applicable preconditions. A static model describes the relationships between the outputs of the operations, output states, inputs, and target states.

A dynamic model, on the other hand, focuses on possible processes. VDM (Vienna Development Method) as well as the Z method and the B method are included in the category of static modeling or as a suitable method for sequential systems. The method CSP (Communicating Sequential Process) is assigned to the category of dynamic models or as suitable for parallel systems. In addition, the standard DIN EN 50128 and DIN EN 50657 assign the method CCS (Calculus of Communicating Systems) and LOTOS (Language of Temporary Ordering Specification) as suitable for parallel systems.

Model-based verification can be used to prove the correctness or specific properties of a suitable model. This is also a formal method. However, the application is only possible or only in a reasonable time if such a model has a limited number of states or results. A distinction can also be made between explicit and symbolic verification. Explicit verifications model all state transitions. Based on initial states, the transition relation is calculated on this basis until cycles are found. In symbolic verifications, states, trans-relations, and properties to be checked are described by logical predicates. Logical joins of these predicates represent state transitions and checking the properties.

Formal methods usually focus on a partial aspect of system or software development. The methods CSP, CCS, LOTOS, VDM and Z support the specification of functional requirements or modeling at the architectural level. Corresponding models can be checked semi-automatically against formal semantics and thus their correctness and completeness can be determined. A method that covers a large part of the development from specification to implementation is rather rare. Thus, the B-Method offers the possibility to model a system based on abstract machines and its dynamics based on events. Based on the models of the B-Method, burdens of proof can be generated automatically. These support the proof of the integrity of specification and design as well as static and dynamic properties, for example the detection of conditions as well as invariants and termination. (for more information see: methode-b.com).

Despite undeniable advantages, there is the impression that even the B-Method has not currently prevailed in widespread application. Experience and discussions suggest that there is an acceptance problem regarding formal methods that focus on the specification or verification of models. And that this may also be the reason for a general skepticism towards formal methods.

Techniques such as Floyd-Hoare calculus prove the correct implementation of a formal specification. However, these can be extremely complex if the verification task is not integrated into the design or implementation. Even if the criticism is taken up and, for example, for the B-Method its intellectual controllability, gradual refinement and step-by-step proof of correctness are brought into the field, the subordinate importance of corresponding methods is not unfounded due to potentially high complexity.

And so further, critical aspects of formal methods are to be pointed out. In order for appropriate tools to be used successfully or to be executed in finite time, the detailing of corresponding models, for example, must be restricted in an appropriate manner. This means that the real system will have properties that the model does not have and that therefore cannot be part of the proof of correctness. This must be considered both in the modelling and interpretation of the proof of correctness.

On the one hand, it would be logical to use formal methods, because they can make a significant contribution to error-free software. On the other hand, both the "astronomical" magnitude of the state spaces of real systems and the necessary expertise for the application of the methods are cited. According to the standard DIN EN 50128 or DIN EN 50657, there are various tactics in the application of formal methods to counter the "astronomical" magnitude of a state space. These include modularization and hierarchical arrangements as well as reduction of state spaces by mapping value ranges analogous to discrete variables. In addition, the application of formal methods can be focused on high abstraction as well as on critical components. The application of formal methods is usually limited to critical components of safety-relevant systems.

According to the standard DIN EN 50128 or DIN EN 50657, a scheme can be derived that integrates the specification of software requirements and architecture. Top-down approach as well as a modular and hierarchical structure shall be applied accordingly. Such a scheme is mainly intended to combine tabular representations. In the specification, states as well as inputs/outputs of the system shall be defined. Value ranges of relevant inputs and variables are preferably mapped to discrete variables. Based on the inputs and the actual state, the target state shall be mapped and, depending on this, the execution of possible operations or the determination of corresponding initial values. In addition, safety-relevant measures shall be safeguarded at the level of process automation, for example by appropriate plausibility check of the status sequence. Based on these principles, it is possible to safeguard corresponding interfaces to the basic software as well as to carry out automatic checks of the static model. The entire development process can be based on such a scheme.

Based on the previous considerations and the outlined scheme, a list of questions is derived below. The questions are intended to support a systematic classification of methods without claiming to be complete. Above all, they are intended to shed light on the suitability and mode of action of given methods and, if necessary, to assess effects on the development process.

1) Is a method suitable for solving the specific problem?

Not every method is equally suitable for solving a problem, or different forms of a solution principle can be suitable for different aspects. If necessary, for example, a method offers several model types and only a part of them is needed in the specific case. Here there is also an interrelationship with the process to be controlled. In this context, it must also be questioned whether a method is suitable for each project category.

2) At what stage of development and for what task can a method be applied?

Methods can be specialized in a specific task in the development process. However, methods must be coordinated with each other in the development process, such as specification, analysis, testing at the respective levels of abstraction.

3) Is special competence for the application of a method necessary, available or, if necessary, acquirable?

The application of a method may require specific competence, such as the application of UML or the application of a formal method. If this competence is not available, the question arises as to whether it can be acquired.

4) Is the result of a method easy to interpret?

Although the application of a method may require special competence, the result can essentially be understood intuitively, for example formalized graphical representations such as UML. Details may require additional information or competence.

5) Is the support of a method by suitable tools necessary, available or, if necessary, acquirable?

A method may be intuitive to use, but if the amount of data grows or changes shall be considered, suitable tools are usually indispensable. In addition, the integration of methods as part of the development process is much easier and more efficient to accomplish. A wide range of data evaluations are practicable.

6) Is a suitable, complementary method for verification the results necessary or available?

Specifications shall be checked, the implementation tested according to the degree of abstraction. Both the specification as well as the verification must be carried out based on appropriate, corresponding methods.

7) Does a method contribute to error prevention or error control?

The contribution of a method to the reliability, availability, maintainability and safety of the software or the system shall be classified.

8) How flexible is a method in the application or in the event of changes in the model?

The processing of data or models of a method should be easy to manage in the event of changes. However, the application of a method itself, or underlying schemes, should also be flexible.

9) To what extent does a method support an analysis based on formal models?

Formal methods can also be understood as an indication of which attributes are important for the analysis. Even if a method does not provide the means of formal methods itself, essential elements can contribute to quality. Thus, a requirement that is formulated in natural language can be interpreted and, if necessary, is not very precise. However, suitable attributes can contribute to the quantitative evaluation of quality.

10) To what extent can a method be integrated into the existing development process?

An established development process is often also associated with routines in the application of methods. Comparability with previous projects seems to be an argument, even if a technical integration is not realized. However, technical interfaces and organizational framework conditions can represent a hurdle in the establishment of methods or tools.

11) Is a method accepted in the potential field of application?

Even if a method seems suitable, acceptance can be decisive for success. Acceptance can depend on many factors. Consideration must be given to how acceptance should be established, if any, or whether an alternative should be chosen.

12) To what extent does the introduction of a method affect the efficiency of the development process?

A final assessment of the extent to which a method affects the efficiency of a development is indispensable. As with all the questions listed, there will be facts to name as well as assumptions to make. It is therefore extremely important to check them regularly.

A process model to be developed in accordance with the DIN EN 50128 or DIN EN 50657 standard or the already outlined scheme for the specification of requirements and architecture can be understood as a reference. An assessment based on the list of questions regarding the reference model can simplify the evaluation of established development processes as well as the estimation of the added value of the methods to be used.

In all considerations, it is necessary to focus not only on the constructive methods as well as the methods for analysis and testing. Rather, the generally necessary methods for quality assurance, such as configuration and compatibility management, must also be considered. For example, the corresponding effort in the context of standardization based on platforms or frameworks can grow rapidly.

Another aspect seems important: It can be observed that experts in a knowledge domain shape a specific world of terms and thus distinguish themselves by a specific language. This allows a domain to be restricted for external influences. A neutral and well-founded technical language is therefore promoted, for example in accordance with the DIN IEC 60050-351 standard. Thus, it can be possible to open up the domain consistently with the knowledge/canon of systems theory, electronics and control engineering as well as information theory and information technology.