

SYSTEMATIC USE OF MODEL-BASED SOLUTION PATTERNS USING THE EXAMPLE OF A LOAD CELL

Zirui Li; Faizan Faheem; Stephan Husung

Technische Universität Ilmenau

ABSTRACT

Complex mechatronic products are usually decomposed into several sub-systems for their development. These sub-systems are developed in parallel or even independently based on their specifications and use cases. The application of model-based solution patterns is an effective way to comprehensively and efficiently describe the available knowledge about the sub-systems. This contribution proposes an approach to support the selection and application of model-based solution patterns. The approach, based on a metamodel for solution patterns using SysML, describes the process for selecting solution patterns and aligning requirements and constraints with the as-is properties of the sub-systems. Additionally, the approach supports the design of solution patterns taking into account special knowledge from the development of the sub-systems as well as the usage of the solution patterns in different systems and contexts. As an example, an application scenario of a specific load cell within a measurement system is explained.

Index Terms – Model Based System Engineering (MBSE), solution pattern, metamodel, reuse, SysML models

1. MOTIVATION

For the development of complex mechatronic products, these products are usually decomposed into several sub-systems according to the system theory [1]. These sub-systems are developed in parallel or even independently based on their specifications and use cases. The final mechatronic product is realized by integrating the sub-systems into the overall system. In the process, top-down requirements are first decomposed to the sub-systems. These requirements include the functional and quality requirements as well as constraints. Based on the requirements, relevant sub-systems are selected (if these already exist) or developed. In many development processes, existing sub-systems are reused in order to use existing knowledge and thus reduce uncertainties, shorten development cycles and reduce development costs [2]. For the reuse the relevant requirements (top-down) have to be aligned with the as-is specifications of the sub-systems (bottom-up) [3]. For top-down and bottom-up alignment, a suitable description of the overall system and the sub-systems is needed to support the assignment and alignment process between required and as-is properties.

Using solution pattern is an appropriate way to describe the knowledge about a unified sub-system as a base for the reuse [4]. A fundamental condition for the use of solution patterns is the transfer of the solution pattern paradigm to the specific conditions in the respective product development process. This requires a common understanding of the solution pattern, but also a general pattern model for all relevant domains and a description of how a solution pattern as a sub-system interacts with the overall system [5]. The term “model” in engineering usually refers



to an representation based on an abstraction of a real-world product or process [6]. In the context of model standardization, metamodels are used to formally describe the concept of the models, including the abstraction of the properties or attributes contained in the models [7]. The metamodel supports the target-oriented modelling of the solution patterns as a basis for the subsequent selection and alignment process. Therefore, the metamodel has to be instantiated context-specifically for the definition of the model-based solution pattern.

An efficient way for model-based descriptions of systems or sub-systems is the application of Model-Based Systems Engineering (MBSE) [8, 9]. MBSE supports a clear documentation of the solution pattern through explicit descriptions of system elements and their relationships. MBSE promotes systems engineering processes that focuses on the creation and use of system models for different use cases during development [10–12]. When properly implemented, MBSE models allow for standardized representation of systems knowledge across engineering disciplines and sub-systems, and simplifies the related systems engineering tasks while minimizing the development risks [13]. System Modelling Language (SysML) is the most widely used modelling language in MBSE [14]. As a semi-formal modelling language, it facilitates analysis, verification and validation activities on the design [12]. Seen in relation to the earlier approaches, MBSE supports the systematic capture of requirements and functions [15], which provide an efficient way to compare as-is-properties of the sub-systems with the required properties [4, 9].

2. STATE OF THE ART

There is a lot of research on combining solution patterns with MBSE for the consistent reuse of sub-systems in product development. ANACKER et al. proposed a solution pattern-based knowledge management approach including a procedure model [5]. The approach describes the unified structure of solution patterns and the related multidimensional knowledge space to demonstrate its certain scope of application and interdisciplinarity. WU et al. proposed an approach that relies on the concept of patterns to realize the reuse of existing knowledge [16]. They introduced the Mining-Maturation-Implementation (MMI) method to search, mature and reuse the solution patterns.

GAO et al. developed a reusable modelling framework based on a designed metamodel for a satellite system [17]. They used the elements of the metamodel and the profile to develop a model framework that describes the relationships between the system, user model, reusable concepts, viewpoint and pattern. PFISTER and CHAPURLAT proposed that ensuring interoperability is the key to achieving model reusability [18]. They propose a design pattern metamodel within a system meta-model, which provide a method to represent the invariant knowledge and experience in design. It is legitimated by citing known application cases. The pattern provides the potential problem in known applications and the corresponding solution. SCHINDEL and PETERSON proposed a comprehensive and abstract data model “S*Metamodel” and framework, for addressing the core system science issues needed to design CPS [19]. Furthermore, based on PBSE (Pattern Based System Engineering), they developed an application of particular S*patterns: embedded intelligent patterns. These patterns provide a rapid and holistic method for the system risk management in Cyber-Physical System and fault identification, analysis and planning.

ERNADOTE introduces a method to support MBSE - Modelling Planning Process (MMP) - that combines a standard metamodel with a specialized project ontology [20]. Furthermore, the focus is on the implementation scenario of the methodology to accelerate the modelling process by addressing the problem of mapping redundancy when building reusable patterns as an extension of his work [21]. This is realized by optimizing the model management through

dynamic mapping, which enables the transfer of information and computations expressed at the stakeholder level to the system model.

However, the analysis of the state of the art shows current research mainly focuses on the application of a solution pattern. The processes of selecting different solution patterns or aligning solution patterns with the required properties and the constraints in the products have not been discussed much.

3. APPROACH

Based on the current state of art and the needs in product development, a methodology is required that supports the selection and alignment of different solution patterns, including the alignment of the as-is properties of the sub-system and the requirements (also required properties) from the overall system. The process should ensure reusability and interdisciplinarity of solution patterns based on a unified metamodel of solution patterns. This paper presents a methodology for model-based description of solution patterns using SysML. The methodology is applied to the example of a load cell in a concrete application scenario of a measurement system.

The approach has two aspects:

- The first aspect aims to describe a metamodel of the solution pattern, which contains all the elements that the solution pattern should cover and the relationships between the elements.
- The second aspect of the approach aims to describe a method for selecting and aligning a solution pattern as a sub-system of an overall system for the development of a technical product, thus achieving knowledge reuse in the product development process.

3.1 Metamodel of solution pattern

A metamodel enables to specify the necessary constituent elements of a solution pattern. Thereby, a metamodel represents the abstraction of a model. This means that the definition of a particular solution pattern requires the instantiation of model elements based on the properties of the sub-system.

In this paper, the metamodel of solution pattern was defined in the software Cameo Systems Modeler using SysML, based on the SYSMOD terminology [22] and MagicGrid [23] method, as shown in Figure 1.

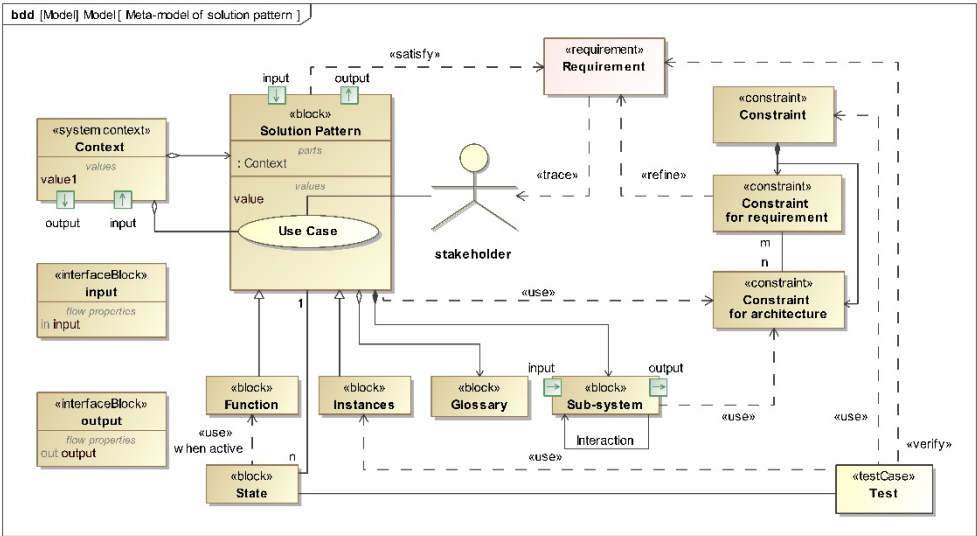


Figure 1: Metamodel of the solution pattern

In the following, the elements of the metamodel are briefly explained:

- **System context**
System context of the solution pattern describes the environment considered in the development of the solution pattern. The description can include the environment of a specific system if the solution pattern was developed in its context. Alternatively, the environment can be the expected consolidated environment of multiple systems. The environment includes the elements in the environment and the interactions with the system at its interfaces. The environmental aspects have a direct impact on the requirements for the solution pattern.
- **Requirements**
Required properties (e.g., behavior) that need to be satisfied by the solution pattern are derived as “Requirements” from the stakeholder demands and design principles. Different stakeholders are involved. One stakeholder is the overall system, but there are also stakeholders arising from the environments of the product in the different life phases. Stakeholders can also be governments or standardization organizations (e.g., ISO, DIN).
Stakeholder are tracked by requirements through “Trace” relationship, because stakeholders are where all the information on system requirements come from.
- **Constraints**
Constraints provide a restriction of semantics for the model elements of solution pattern to refine requirements or describe complex project knowledge of solution pattern architecture (e.g., the Wheatstone Bridge for the electrical resistance measurement) in the form of formulas or parameters [22], which support the understanding of the knowledge described in the solution pattern and are a base for the system analysis.
In the specific *Parameter Diagram*, the relevant constraints are linked to parameters owned by solution patterns. The “constraints for architecture” and “constraints for requirements” are linked with parameters to validate the system behavior or the requirements. Thus, based on the related parameters, the constraints for system may have n-m relationship with the constraints for requirements.
- **Use Case**
A use case describes the expected service of a system towards stakeholders [22]. Different use cases are defined to describe the services that the solution pattern should provide. These use cases are linked to the stakeholders with an “association” relationship.
- **Sub-system & Interaction**
The architecture of a solution pattern may consists of sub-systems that are finally implemented by physical elements or software components. The sub-systems are modelled as “Block” using SysML, and a “Composite” relationship is used to integrate them to the solution pattern in a *Block Definition Diagram*.
Interactions describe the energy, material and information flows between the sub-systems via the interfaces. The interfaces are defined by the *Interface Block* and are connected to *Proxy Ports* of the corresponding sub-system using SysML.
- **Function & State**
A function is interpreted as an activity in a specific process that a sub-system is able to perform [24]. These functions are assigned to the solution pattern using a “Direct Aggregation” relationship. Based on the activities, specific operations can be called. Activities describe the necessary behavior in an abstract way. Detailed functional simulations can be linked via the operations, which enable verification and cooperation with the domains [25].

States group the functional behavior. Functions are performed in the specific active states of a solution pattern. For this purpose, the activities are assigned to the states.

- **Test Case & Instance**

A Test Case is the definition of the procedure, conditions and expected results of a test to be performed. It is regarded as a flow of *Actions* to check whether the system meets the requirements and is connected to the requirements in the metamodel through verify relationships. It can be described in an *Activity Diagram* through a series of *Actions*. Some of the functions, interactions and constraints of the solution patterns are used in the test cases.

For test execution, the test cases are instantiated in combination with the test object using specific parameters. These test objects with parameters are modeled as “Instances” of solution pattern.

- **Glossary**

A glossary is created to explain terms that are not common or only used in specialist areas. It is required to improve communication efficiency, so that all project participants can clearly understand the relevant terminology. The glossary is integrated directly in the solution pattern.

A solution pattern represents the sub-system in the context of a specific overall system. For the decomposition and alignment process, also the corresponding model of the overall system has to be built based on a metamodel. The model of overall system should support the decomposition of overall system requirements to the specific requirements of sub-system. The overall system metamodel is similar to and derived from the solution pattern metamodel.

3.2 Top-down Process

In the regard of the second aspect of the approach, the methodology should describe the alignment process. It is the alignment between top-down requirements (from the overall system to the sub-system) and the specifications of the sub-system (as-is properties and characteristics) as well as the bottom-up requirements (requirements which the overall system has to fulfill for using the sub-system) that the sub-system additionally contains.

In order to select the right solution pattern for an overall system, the requirements of the overall system must first be top-down analyzed and decomposed in such a way that they can be realized by the sub-system. The requirements meant here not only refer to the stereotype “Requirements” in SysML, but include all the properties that need to be provided by the solution pattern, such as the necessary behavior and related parameters. Furthermore, it is necessary to analyze the functions, context or some specifications of overall system which support the performance of solution pattern.

The result is the requirements for a sub-system as a basis for selection and comparison with the as-is specification of a solution pattern.

The top-down process is a step-by-step procedure. In each step, the result is organized in the package “demand” (see Figure 2) under the structure of the overall system project.

1. Extraction and decomposition of specific requirements

The requirements related to the required sub-system are examined and extracted from the requirements of the overall system and then these requirements are decomposed according to the architecture decisions. The main purpose of the decomposition is to transform the original requirements for the overall system into requirements for the solution pattern. The solution pattern can only implement requirements alone that are specified at the boundary of the sub-system. Requirements that cross several sub-systems cannot be implemented by the solution pattern alone and must therefore be further decomposed. With regard to the “Extraction” operation in this and further steps,

it aims to organize and store the information in a package under the “demand” package for easier and more intuitive analysis of the information in the subsequent assignment process.

Finally, the decomposed requirements are organized and stored in a “requirements” package in “demand” to facilitate subsequent use and access.

2. Extraction and derivation of the corresponding constraints from the relevant requirements

In this step, the constraints are extracted or derived from the requirements associated with the solution pattern to clarify the refinement of requirements. They are then stored in the “constraints for requirement” package under “constraints” in “demand” for further use and access.

3. Extraction of context, functions and states

The solution pattern should work properly in the context of the overall system, so the information involved under the system context should be extracted for subsequent analysis. However, some parameters and work context (e.g., temperature change due to thermal conduction in physical frame) may have changed when they are aligned with the solution pattern due to the structure of the system. These should be considered with special attention.

The analysis of states and functions refers to the analysis of which functions the solution pattern should perform in which states. Certain functions may be reused for different states. This information is extracted into the corresponding package “architecture analysis” under “demand”.

4. Extraction of corresponding constraints from the system architecture

Some constraints can be determined from the system context, functions and states to define important performance parameters. These constraints contain the parameters that need to be provided by the solution pattern and act during the simulation or validation. At the same time, new constraints that directly relate to the solution pattern need to be modified or added according to the requirements of the system. These constraints are extracted to the “constraints for architecture” package under “constraints” in the “demand” package.

5. Extraction of corresponding parameters

Some of the parameters in the system are inherited and used by the solution pattern, e.g. environment variables in context. Moreover, some parameters have to be provided from the solution pattern, e.g., the output values of the solution pattern. These parameters need to be organized in a top-down process and then stored in a new created *Block* called the ‘D-Parameter’ (Demand Parameter) for subsequent comparison and analysis with the solution pattern.

6. Interaction analysis

The interaction between the solution pattern and the other sub-systems in the overall system should be analyzed to guarantee the communication between the sub-system as well as the behavior of the whole system. Thus, in the top-down process, the input and output relationships between them and the ports are identified, the ports are stored in the appropriate packages (“interaction” under “architecture analysis”).

The information required to select the solution pattern is organized. Afterwards the “demand” package is sent to the solution pattern and analyze and compare the information in the “bottom-up process”. The corresponding package in the *Package Diagram* is shown in Figure 2.

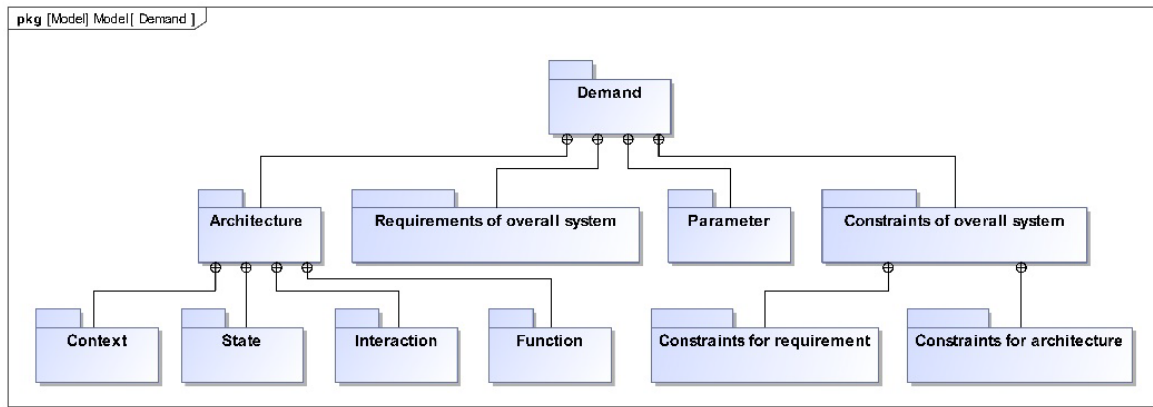


Figure 2: Package Diagram of the demand from top-down process

3.3 Bottom-up Process

Based on the “demand” defined in the top-down process, the information from the solution pattern is compared to the overall system through a bottom-up process for the alignment.

The bottom-up process consists of three tasks:

1. Alignment of the as-is specifications of the solution pattern with the requirements of the overall system;
2. Checking the bottom-up requirements of the solution pattern which the overall system has to fulfill for using the solution pattern as sub-system;
3. Elaboration of the demand for changes or improvements to the overall system based on the bottom-up requirements.

The bottom-up process supports the decision to be made on the suitability of the solution pattern to be selected for the overall system. At the end of the process there is the specification of the sub-system and, if necessary, an adapted specification of the overall system, so that the required and as-is properties fit together sufficiently.

3.3.1 Stereotype definition

For the evaluation of the specification and requirement information correspondence between overall system and solution pattern in the comparison process, relevant comparison stereotypes should be defined. Six stereotypes describing the result of the respective alignment are defined as the derivation of the “Allocate” relation, as well as two stereotypes with suggested modifications, as shown in Figure 3.

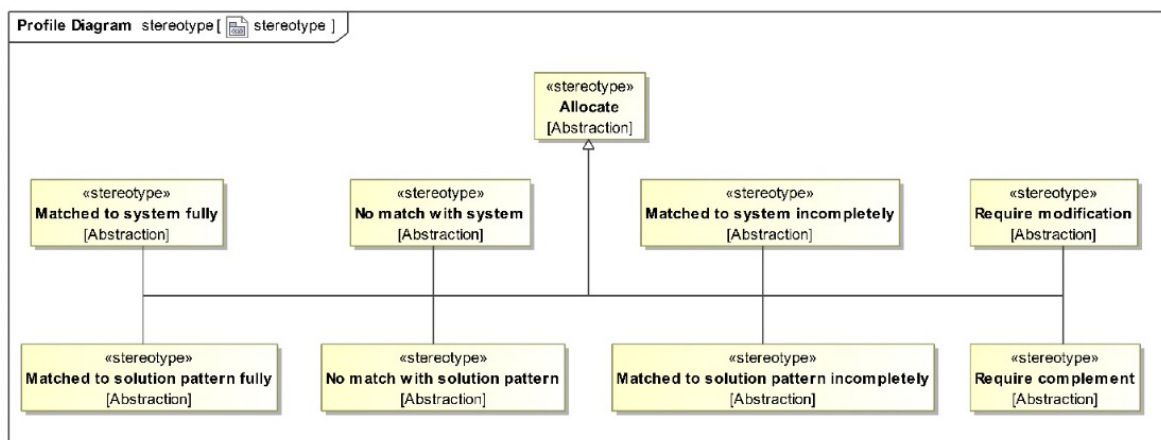


Figure 3: Stereotype definition for the evaluation

The utilization of 8 stereotypes as a criterion for analyzing the requirements and specifications in solution pattern and overall system has several benefits. The different stereotypes of the specific alignment result make it transparent and comprehensible from a holistic perspective. As the assessment is based on the expertise and experience of the engineer, the stereotypes also help to compensate for differences in understanding of the model elements and parameters by different professionals. The limitation is that the application of the stereotypes still depends on the expertise of the engineer and requires a detailed and comprehensive analysis of the content of the solution pattern.

3.3.2 Requirements & Architecture Analysis

As the solution pattern was developed in many cases in a different product and project context [3], the solution pattern was also developed against possibly different requirements. In addition to the requirements, the constraints, states and associated functions in the solution pattern may be defined or cut differently than in the higher-level system. Therefore, for the bottom-up process, the associated elements from the solution pattern are stored in the “solution” package under the structure of the solution pattern project and aligned with the elements of overall system.

The analysis method is performed in the following steps (see also Figure 4 for the packages):

1. Alignment of requirements

The requirements of solution pattern are stored in the “requirements” package under “solution” package. Based on the described solution pattern, the requirements associated with the as-is properties of the solution pattern are compared with the top-down requirements to gain knowledge about whether the pattern can meet the requirements of the overall system.

Some requirements describe the system context of the solution pattern. Only if the overall system meets these context requirements (e.g., sufficient supply of energy), the solution pattern can fulfil the required properties.

Using SysML in Cameo Systems Modeler, the requirements packages from “solution” and “demand” are then imported into the SysML allocation matrix for alignment. The six stereotypes defined above allow the requirements in “demand” package to be aligned with the requirements of the pattern.

Based on the alignment results in the allocation matrix, a preliminary assessment is made as to whether the pattern is suitable for the system. Furthermore, the requirement to be modified or improved is assigned to the corresponding content of the overall system by means of two stereotypes.

2. Alignment of constraints and requirements

In addition to the requirements, the constraints must also be aligned, which are stored in the “constraints for requirement” package under “constraints” and “solution”. The constraints are aligned with the constraints and requirements from “demand” of the overall system in the SysML allocation matrix.

The two stereotypes “constraints to constraints” and “constraints to requirements” are used for the alignment.

The purpose of aligning constraints with requirements is to determine whether constraints from the solution pattern can be used to describe the requirements of the overall system. Since constraints are a further refinement of requirements, analyzing and aligning the constraints from the solution pattern with the requirements of the overall system provides a cross-check for the selection decision and use of the solution pattern.

Constraint to constraint alignment means that the parameters used in the constraints of the solution pattern match with the parameters used in the constraints in “demand” of

the overall system. The constraints may also differ from the solution pattern and overall system. For example, if the result value of the permissible load of a load cell is given in force (in newtons) or in terms of mass (in kilograms), or if the cost limit is given in euros or dollars.

3. Analysis of contexts

In order for the solution pattern to be used in the overall system, the specified context of the solution pattern must also match the overall system. The comparison of the context is relatively special because some elements of the context of the overall system are also elements of the context of the solution pattern. Therefore, the alignment cannot be described exactly by the six stereotypes.

4. Alignment of functions and states

The functions and states of solution pattern are also represented in relevant packages and aligned with the states and function of the overall system in SysML allocation matrices. Also, for the states and functions, the required functions can be described and cut differently by the overall system than the realized functions of the solution pattern (e.g. the measurement function is required as one function in the operation state, but in the solution pattern the function is divided into several functions and possibly sub-states).

5. Alignment of the constraints from architecture

Similar to the constraint alignment in requirements analysis, the constraints in the solution pattern for description of the sub-system are stored in the “constraints for architecture” package under “solution”, and then aligned in the SysML allocation matrix. The alignment here analyzes if the constraints have different description with the constraints of the overall system due to different principles and parameters.

6. Alignment of the parameter

All the parameters (modeled as value properties with SysML) used in the solution pattern are stored in a *Block* called “S-parameter” (solution parameter). In the SysML allocation matrix parameters are aligned with the parameters in the d-parameter (see section 3.2). The elements to be analyzed include the following:

- Whether the parameters from overall system and solution pattern have the same value?
- Whether the parameters have the same name but represent different attributes?
- Whether there are parameters required by the overall system that are not available in the solution pattern?
- Whether there are parameters required by the solution pattern that are not available in the overall system?
- Whether there are parameters in the solution pattern that are not needed or not considered in the overall system?

7. Interaction analysis

The interaction between the solution pattern and the other sub-systems in overall system is analyzed based on the ports and flows used in the model to determine which input and output the solution pattern supports as well as which inputs and outputs are required.

The ports from solution pattern are stored in the “interaction” packages and aligned with the “interaction” from “Demand” in the SysML allocation matrix.

Based on the combination of top-down and bottom-up analysis, insights are gained into the extent to which the overall system or, if applicable, the solution pattern needs to be modified for use in the overall system. With each modification in the overall system or solution pattern, the alignment (top-down and bottom-up) has to be re-evaluated.

The solution package diagram is shown in Figure 4.

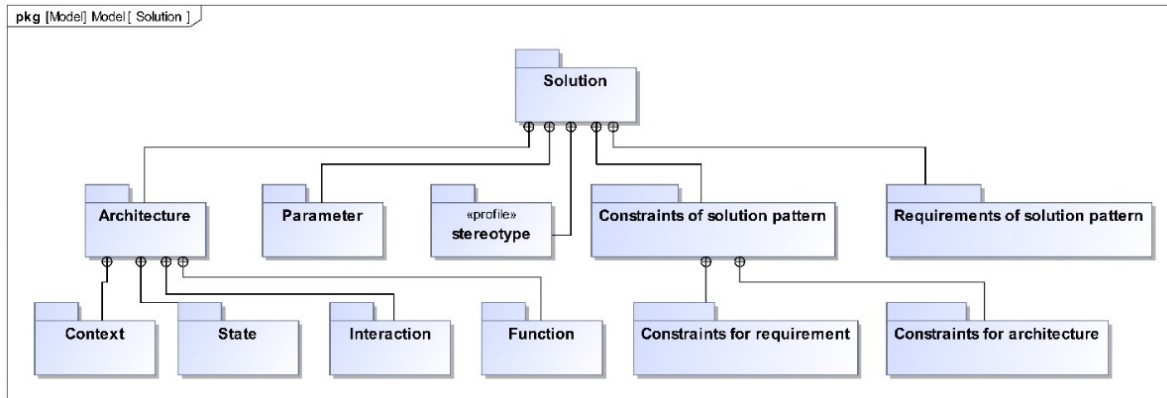


Figure 4: Package Diagram of Solution from Bottom-up Process

4. APPLICATION ON THE EXAMPLE OF LOAD CELL

The concept developed is applied in this paper using a precision engineering application, specifically for a load cell. For this example, the top-down requirements decomposition and the required product properties derived from the requirements of the measurement system to the load cell are explained. The bottom-up requirements for the measurement system from the load cell are also analyzed and described.

4.1 Model of solution pattern and overall system

The application example used in this paper is a measuring system for force measurement. The force measurement is always carried out via a strain measurement with known stiffness. A possible solution for the measurement is a load cell with strain gauges based on the piezoresistive effect. Load cells can be selected as one of the key sub-systems in the development of the measurement system.

In the scope of this paper, the models of the measuring system as well as the load cells are built based on the proposed metamodel. On the one hand the load cells are modeled with reference to some parameters of existing products (but intentionally adapted for the research question in order to investigate relevant aspects) and the OIML standard [26]. On the other hand, the measuring system is modeled based on the following stakeholder requirements.

- The measuring system should provide a static measuring range of 1000 N in the direction of gravity and a dynamic measuring range of +/-10 N;
- The operating temperature of the measuring system should be -10° to 100°;
- The measuring system should be able to resist shear forces.

During the modelling process, some requirements, specifications and structures of overall system and load cell were modified in order to show the alignment process between the measuring system and the load cell, which means the model in the paper doesn't represent the real product. Due to space limitation of the paper, it is not possible to show the detailed model. The system and load cell structure using *Block Definition Diagram* is shown in the Figure 5 & 6.

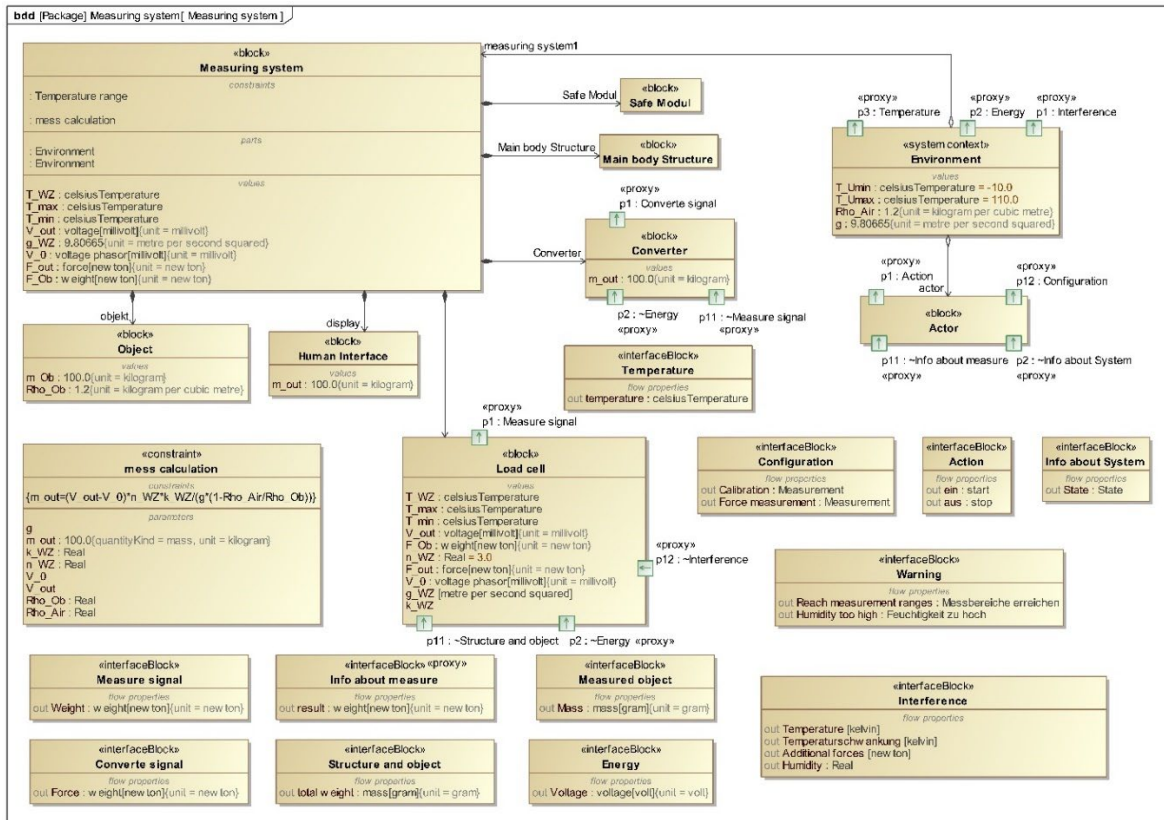


Figure 5: Main Structure of Measuring System

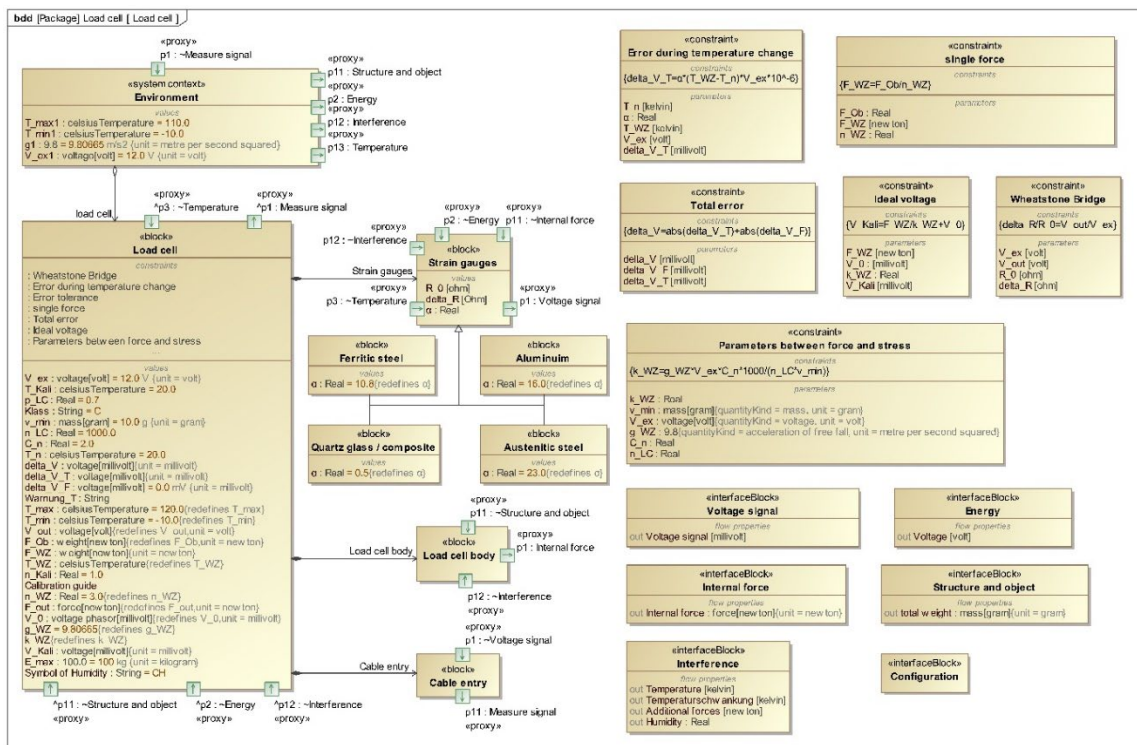


Figure 6: Main Structure of Load Cell

4.2 Top-down Process

Top-down process is performed to select the appropriate solution pattern as sub-system for the measuring system. By analyzing the specification and requirement information that the

measuring system can provide to the load cell and extracting the relevant information to define the top-down requirements, a “demand” package is created under the model to store information about the different elements.

The requirements from stakeholders for the measuring system, as shown in section 4.1, are not a complete reflection of the needs of a measuring system for load cells. In other words, the current requirements contain a large amount of unclear information. Therefore, they should be decomposed into more basic requirements or a more precise description of the requirements through the architecture analysis, as the basis of alignment with the requirements of solution pattern. The result of decomposition for requirement is shown in Figure 7.

| # | Name | Text |
|----|--|--|
| 1 | <input type="checkbox"/> <input checked="" type="checkbox"/> R D1 Force measurement | The load cell should measure the axial force in certain measuring range and show |
| 2 | <input checked="" type="checkbox"/> R D1.1 Static load | In the static cases the load cell should measure the axial force from 0N to 1000N and show |
| 3 | <input checked="" type="checkbox"/> R D1.2 Dynamic load | In the dynamic cases the load cell should measure the axial force with 10N threshold load and show |
| 4 | <input checked="" type="checkbox"/> R D1.3 Force direction | The force to be measured should be in the direction of gravity |
| 5 | <input type="checkbox"/> <input checked="" type="checkbox"/> R D2 Structure | The structure of the load cell should resist the interference |
| 6 | <input type="checkbox"/> <input checked="" type="checkbox"/> R D2.2 Shear Force | The structure of the load cell should resist the disturbance from the shear force |
| 7 | <input type="checkbox"/> <input checked="" type="checkbox"/> R D2.1.1 Internal shear force | The structure of the load cell should resist the internal lateral force |
| 8 | <input checked="" type="checkbox"/> R D2.1.1.1 Thermal strain | The structure of the load cell should resist the shear force by thermal expansion |
| 9 | <input type="checkbox"/> <input checked="" type="checkbox"/> R D2.1.2 External shear force | The structure of the load cell should resist the external lateral force |
| 10 | <input checked="" type="checkbox"/> R D2.1.2.1 Additional forces | The structure of the load cell should withstand the additional forces that are introduced in addition to the axial force |
| 11 | <input type="checkbox"/> <input checked="" type="checkbox"/> R D3 Environment | The load cell should run normally in certain environment |
| 12 | <input type="checkbox"/> <input checked="" type="checkbox"/> R D3.1 Temperature | The load cell should run normally in certain temperature range |
| 13 | <input checked="" type="checkbox"/> R D3.1.1 Static temperature | The load cell should run normally at a certain temperature |
| 14 | <input checked="" type="checkbox"/> R D3.1.2 Dynamic temperature | The load cell should run normally with the temperature variation |

Figure 7: Decomposed requirement table of measuring system

In the parameter analysis, all parameters used in the measuring system model are divided into two categories to distinguish the parameter which the solution pattern should provide with the parameter that required by solution pattern. The parameters that define the internal properties of the measuring system are stored in a new *Block* called “d-parameter-sys” under “parameter” using value properties. For example, the parameters: required amount of load cells, real-time temperature. The other parameters that the measuring system needs from the load cell are extracted and stored in a new *Block* called “d-parameter-dem” under “parameter”. For example, the mass data, maximum and minimum working temperature of the load cell, load limit etc.

Due to space limitations, not all of the process steps in section 3.2 are explained in detail. After the top-down process, a set of information about decomposed requirements and required specification for load cell is defined for the application of the load cell (requirements, constraints, etc.). The bottom-up process from the load cell to the measuring system is carried out based on the as-is characteristics of the load cell.

4.3 Bottom-up Process

In the requirements analysis, the requirements of load cell and measuring system are imported into a SysML allocation matrix under “Requirements” package and aligned.

For example, the static force and dynamic force requirements from measuring system are described in the OIML-load rage, while the force direction requirement corresponds to the OIML-load type, so the “matched to solution pattern fully” is used to allocate the decomposed requirements of the measuring system to the corresponding requirements of the load cell.

The external shear requirement has no correspondence in the load cell, so it has “no match with system”, which means that the structure of the load cell is not able to resist external forces other than axial forces. Thus, the structure of the measuring system needs to be modified to meet this requirement of the measuring system. Therefore, this requirement allocates to itself using “require modification” stereotype.

For load cells, the installation requirements and the safety measurement requirements have no corresponding requirements in the measuring system, so they are linked to their own allocate with “no match with system”. Since the satisfaction of these two requirements is related to the structure of the measuring system, they are linked to the structure requirements of the measuring system by “require complement” and “require modification”, representing if the load cell is selected, the installation and safety measurement requirements of the load cell have to be added to the structural requirements of the measuring system and satisfied by modifying the structure. The completed matrix table is shown in Figure 8.

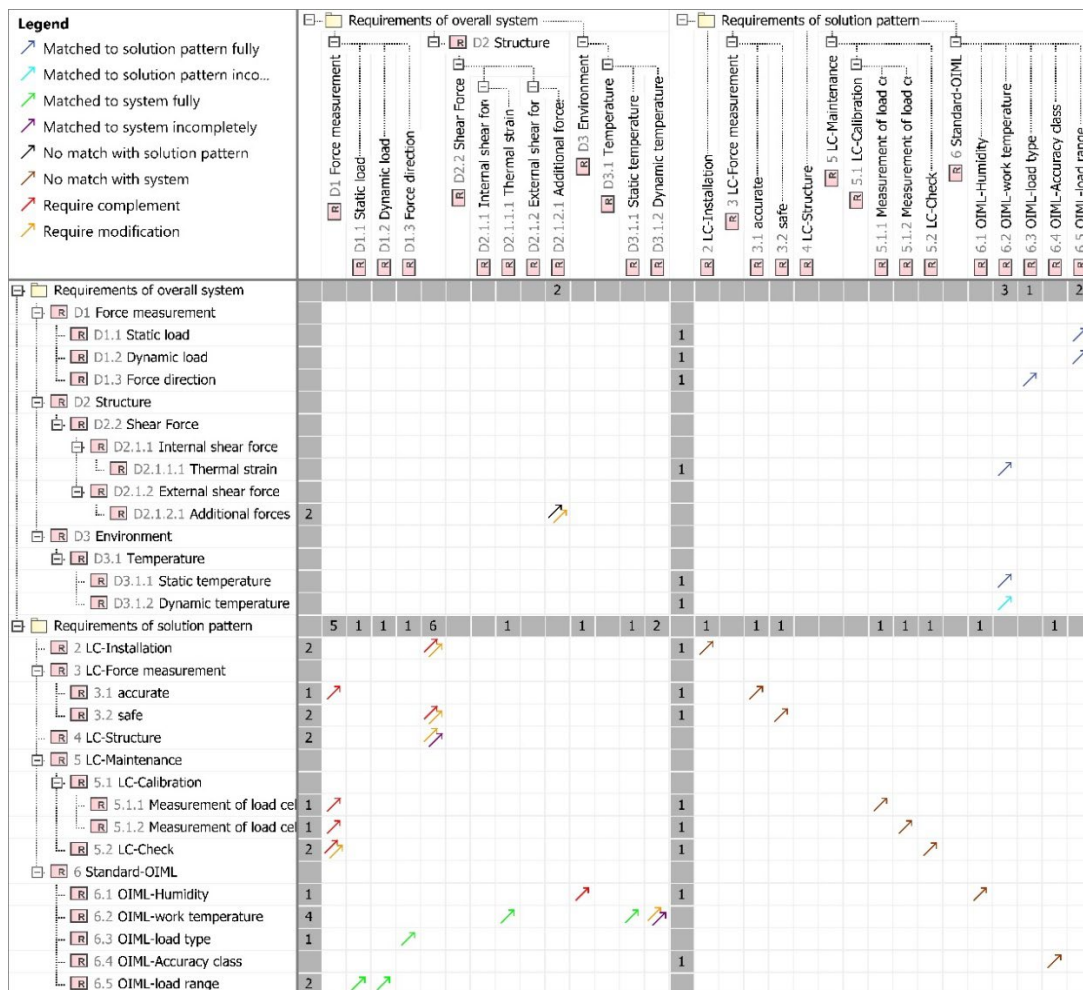


Figure 8: Requirement Allocation Matrix between load cell and measuring system

The above relationships allow a preliminary assessment of whether the load cell currently selected is suitable for the measuring system. The selection logic includes:

- Whether the requirements of the measuring system for the load cell are well satisfied?
- Whether there is a large number of unmatched or incomplete matches between the two requirements?
- Whether the application of this load cell requires a large amount of complementary and modification work on the measuring system?

The matrix shows that the comparison and alignment of information between the top-down process and the bottom-up process is of great importance in the selection of the solution pattern. It also shows that the selection of the solution pattern is not a one-way process, but only when

the overall system and the solution pattern meet each other's demands, the system can operate properly.

Besides the alignment using stereotype of allocation, the analysis process of system context is different. The system context of the load cell is extracted and stored in the "Context" package under "Architecture". The "Context" file under "architecture" is renamed "s-environment" in order to distinguish it from the context of the top-down process. As described in section 3.3.2, since a part of the measuring system exists as a context for the load cell, instead of describing the context aligned by the "Allocation" relation, "D-environment" and "D-Environment" are imported into the *Block Definition Diagram* called "Context".

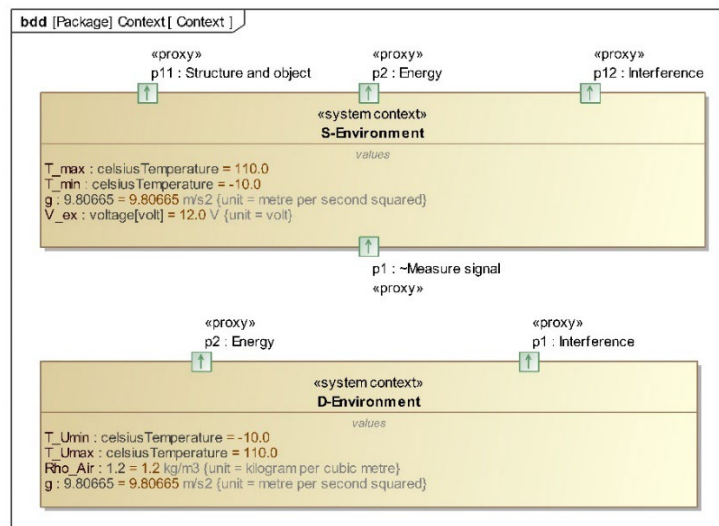


Figure 9: Context of the load cell and the measuring system




As shown in Figure 9, the two contexts have a lot in common, such as the maximum temperature of the environment, the minimum temperature, and the value of gravitational acceleration. It can also be found here that for the environment maximum and minimum temperatures, the two contexts do not use the same symbols, but they indicate the same meaning. In addition, the two proxy ports "Energy" and "Interference" stay in both contexts, indicating that both contexts provide the flow of energy and interference to the system. Further analysis is required to identify the difference between contexts and related suggestions of modifications.

The rest of the alignment process following the method described in the previous section has not been presented due to the limitation of the length of the paper.






4.4 Complement & Modification

If the selected load cell basically satisfies all the requirements of the measuring system and provide a good match in terms of function, interaction interface, and parameters, the solution pattern can be determined as suitable sub-system for utilization by the measuring system.

Nevertheless, following the *Allocation Matrices*, the requirements and specifications marked with "Require complement" and "Require modification" in the aligning result should be taken into consideration for utilization of load cell. The corresponding suggested information can be stored in a generic table in Cameo Systems Modeler, as shown in the Figure 10, where a few simple examples of suggested modifications are presented. The load cell can only be used in the measuring system after all the related information or structure has been modified and complemented in the model according to the corresponding suggestions.

| # | Name | Documentation |
|---|--|---|
| 1 |  Additional shear force | The structure of load cell can't withstand additional forces, should realize it by measuring system |
| 2 |  Dynamic load | The unit of dynamic load is Newton, not match with the load cell, which use kilogram to define the load range |
| 3 |  Static load | The unit of static load is Newton, not match with the load cell, which use kilogram to define the load range |

(a) Complement suggestions

| # | △ Name | Documentation |
|---|---|--|
| 1 |  Accuracy | The load cell has its accuracy data |
| 2 |  Calibrate load cell | The load cell should work in a calibration state, which should be added to measuring system and let measuring system provide the corresponding function and state. |
| 3 |  LC-Calibration | The load cell has a requirement about calibration that should be satisfied |
| 4 |  LC-Check | The load cell has a requirement, that it should be checked in specific time |
| 5 |  reference voltage | The load cell has the reference voltage |

(b) Modification suggestions

Figure 10: Suggestions in Generic Table

Furthermore, an instance based on the measuring system with the load cell can be generated, through which the measuring system can run a simulation of the state machine in which the achievement of the constraints is indicated. Based on the result of simulation, it is possible to intuitively verify whether the requirements of the overall system have been fulfilled.

5. CONCLUSION

Mechatronic systems consist of an increasing number of sub-systems in order to solve increasingly complex tasks. In terms of a comprehensive and efficient description of the available knowledge about the sub-systems, the application of model-based solution patterns is an effective approach.

This paper proposes a methodology to enable the selection and application of solution patterns. The methodology consists of two aspects. On the one hand, a metamodel about the description of the solution pattern and related overall system is presented. With the metamodel, the development of a solution pattern for load cells is possible, which also support the selection and alignment process for the reuse in different measuring system.

On the other hand, the methodology explains the selection and alignment process, including the top-down requirements decomposition and the specifications analysis of the solution pattern, as well as the generation of bottom-up solution and suggestions. Meanwhile, the methodology describes an evaluation approach to analyze the suitability of the solution pattern for the overall system. The bottom-up requirements from the solution pattern to the overall system are also taken into account.

Although the methodology has been evaluated on a concrete application scenario with load cells, the work still reveals open questions that are considered to be the subject of further research. On the one hand the metamodel of solution patterns should be further optimized. A more detailed description of metamodel can effectively reduce the difficulty of applying the metamodel for the design of complex solution patterns in specific domains. However, the exhaustive description will increase difficulty about applicability of the metamodel. Therefore, how to balance the generalization and complexity of the metamodel is worth researching.

On the other hand, the systematic evaluation approach between the solution pattern as a sub-system and the overall system should be further discussed. The current evaluation only deals with the alignment of information and the need for modification or refinement using the

engineers' experience, but how to standardize the suitability of the solution pattern for selection by alignment is not discussed in the current methodology. A possible way is to use the corresponding scores to measure the impact of the information mismatch on the overall system and the impact of specific requirements not being met on the overall system, such as increased costs, longer development cycles and complexity of the development process, etc. In addition, the application of the methodology to more fields than measurement systems are much worth exploring.

REFERENCES

- [1] D. D. Walden, G. J. Roedler, K. Forsberg, R. D. Hamelin, and T. M. Shortell, *Systems engineering handbook: A guide for system life cycle processes and activities*, 4th ed. Hoboken, NJ: Wiley, 2015.
- [2] A. Albers *et al.*, "The Reference System in the Model of PGE: Proposing a Generalized Description of Reference Products and their Interrelations," in *22nd International Conference on Engineering Design - ICED19*, Delft, The Netherlands, 2019, pp. 1693–1702, doi: 10.1017/dsi.2019.175.
- [3] A. Mahboob and S. Husung, "A Modelling Method for Describing and Facilitating the Reuse of SysML Models during Design Process," in *INTERNATIONAL DESIGN CONFERENCE – DESIGN 2022*, 2022, doi: 10.1017/pds.2022.195.
- [4] C. Weber and S. Husung, "Solution patterns - their role in innovation, practice and education," in *14th International Design Conference (DESIGN 2016)*, Cavtat, Dubrovnik, Croatia, vol. Design Theory and Research Methods, 2016, pp. 99–108.
- [5] H. Anacker, R. Dumitrescu, A. Kharatyan, and A. Lipsmeier, "Pattern Based Systems Engineering – Application of Solution Patterns in the Design of Intelligent Technical Systems," in *16th International Design Conference (DESIGN 2020)*, 2020, pp. 1195–1204, doi: 10.1017/dsd.2020.107.
- [6] H. Stachowiak, *Allgemeine Modelltheorie*. Wien, New York: Springer, 1973.
- [7] J. P. van Gigch, *System Design Modeling and Metamodeling*. Boston, MA: Springer US, 1991.
- [8] M. M. Schmidt and R. Stark, "Model-Based Systems Engineering (MBSE) as computer-supported approach for cooperative systems development," in *European Conference on Computer-Supported Cooperative Work*, 2020.
- [9] W. Schindel and T. Peterson, "Introduction to Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques," in *23. INCOSE International Symposium*, vol. 23, 2013, p. 1639.
- [10] S. Husung, C. Weber, and A. Mahboob, "Model-Based Systems Engineering: A New Way for Function-Driven Product Development," in *Design Methodology for Future Products*, D. Krause and E. Heyden, Eds., Cham: Springer International Publishing, 2022, pp. 221–241.
- [11] Q. Wu, D. Gouyon, E. Levrat, and S. Boudau, "Use of Patterns for Know-How Reuse in a Model-Based Systems Engineering Framework," *IEEE Systems Journal*, vol. 14, no. 4, pp. 4765–4776, 2020, doi: 10.1109/JSYST.2020.2975116.
- [12] S. Husung, C. Weber, A. Mahboob, and S. Kleiner, "Using Model-Based Systems Engineering for need-based and consistent support of the design process," in *23rd International Conference on Engineering Design (ICED21)*, 2021, doi: 10.1017/pds.2021.598.
- [13] J. A. Estefan, "Survey of model-based systems engineering (mbse) methodologies," Rep. INCOSE-TD-2007-003-02, 2008.
- [14] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: The systems modeling language*, 3rd ed. The MK/OMG Press. Burlington, 2015.

- [15] S. Husung, C. Weber, and A. Mahboob, “Integrating Model-Based Design of Mechatronic Systems with Domain-Specific Design Approaches,” *Proc. Des. Soc.*, vol. 2, pp. 1895–1904, 2022, doi: 10.1017/pds.2022.192.
- [16] Q. Wu, D. Gouyon, S. Boudau, and E. Levrat, “Capitalization and reuse with patterns in a Model-Based Systems Engineering (MBSE) framework,” in *IEEE ISSE 2019: 5th IEEE International Symposium on Systems Engineering : 2019 symposium proceedings : Edinburgh, Scotland, UK, October 1-3, 2019*, Edinburgh, United Kingdom, 2019, pp. 1–8, doi: 10.1109/ISSE46696.2019.8984571.
- [17] S. Gao, Y. Cao, Z. Fang, and S. Xie, “Reusable MBSE Modeling and Simulation for Satellite Network,” in *2021 16th International Conference of System of Systems Engineering (SoSE)*, Västerås, Sweden, 2021, pp. 168–173, doi: 10.1109/SOSE52739.2021.9497494.
- [18] F. Pfister, V. Chapurlat, M. Huchard, and C. Nebut, “A Design Pattern meta model for Systems Engineering,” *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11967–11972, 2011, doi: 10.3182/20110828-6-IT-1002.03005.
- [19] Bill Schindel and Troy Peterson, Eds., *Pattern Based Systems Engineering – Leveraging Model Based Systems Engineering for Cyber-Physical Systems*. 2014 NDIA Ground Vehicle Systems Engineering and Technology Symposium, Systems Engineering Technical Session, 2014.
- [20] D. Ernadote, “An ontology mindset for system engineering,” in *2015 IEEE International Symposium on Systems Engineering (ISSE)*, Rome, Italy, 2015, pp. 454–460, doi: 10.1109/SysEng.2015.7302797.
- [21] D. Ernadote, “Ontology-Based Pattern for System Engineering,” in *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Austin, TX, 2017, pp. 248–258, doi: 10.1109/MODELS.2017.4.
- [22] T. Weilkiens, *Systems engineering with SysML/UML: Modeling, analysis, design* (OMG Press series). Amsterdam, Boston, Heidelberg, London: Elsevier Morgan Kaufmann OMG Press, 2007.
- [23] A. Aleksandraviciene and A. Morkevicius, Eds. *MagicGrid® Book of Knowledge -A practical guide to Systems Modeling using MagicGrid*. LT-44146 Kaunas, Lithuania: Vitae Litera, UAB, 2018.
- [24] R. Barker and C. Longman, *Case Method: Function and Process Modelling*. Addison-Wesley, 1992. [Online]. Available: <https://books.google.de/books?id=ipHTngEACAAJ>
- [25] T. Zerwas *et al.*, “Model Signatures for the Integration of Simulation Models into System Models,” *Systems*, vol. 10, no. 6, 2022, doi: 10.3390/systems10060199.
- [26] *International Recommendation OIML R60 for load cells*, Paris, France, Edition 2000.

CONTACTS

M.Sc. Zirui Li

email: zirui.li@tu-ilmenau.de

ORCID: <https://orcid.org/0009-0007-7983-7901>

M.Sc Faizan Faheem

email: faizan.faheem@tu-ilmenau.de

ORCID: <https://orcid.org/0009-0009-1014-5389>

Univ.-Prof. Dr.-Ing. Stephan Husung

email: stephan.husung@tu-ilmenau.de

ORCID: <https://orcid.org/0000-0003-0131-5664>