INTERNATIONAL DESIGN CONFERENCE – DESIGN 2022
https://doi.org/10.1017/pds.2022.195

A Modelling Method for Describing and Facilitating the Reuse of SysML Models during Design Process

A. Mahboob and S. Husung

Technische Universität Ilmenau, Germany

atif.mahboob@tu-ilmenau.de

Abstract

MBSE and SysML are increasingly finding their applications in industry as well as in academia. The reuse of the information described in SysML models depends, among others, on the modelling methods, management of dependencies between elements and on the needed remodelling effort. In this paper, a modelling method is presented that address the reuse of SysML models and descriptions as well as reuse of model variants in SysML. A case example is presented to explain the modelling methods and the gained experience is summarised in the form of general recommendations for further use.

Keywords: model-based systems engineering (MBSE), modelling, variant management, resuse of SysML models, information management

1. Introduction

The development of complex mechatronics systems (e.g. because of cross-domain functions) require close collaboration between multidisciplinary teams that are involved in the product development process. Starting from the stakeholder requirements, the desired functions from a system are described in a top-down fashion (system level). The realisation of a specific system function may require further sub-functions. These sub-functions might be realised by sub-systems. Sub-systems sometimes already exist (e.g. through previous generations) and may well be used multiple times in multiple different systems (e.g. a sensor is used in different measuring devices). An As-is description could already exist for the sub-systems, which can be integrated bottom-up into the current development. The reuse of sub-systems descriptions containing information about the implementation of sub-functions is important because this reuse can contribute to increasing quality as well as saving time and resources in the development. The reuse of any given sub-system inside any system requires the integration of top-down (system) with the bottom-up (sub-system) descriptions. A seamless integration of both (top-down and bottom-up) is only ensured by considering the reuse right from the start of descriptions of the sub-systems.

Each sub-system may have multiple different variants (for example, with different performance parameters, measuring ranges or specific implementations of functions) that have to be analysed against the requirements to find the most suitable solution. These variants can be described in the form of models containing overlapping and variant specific information from which concrete sub-system instances can be derived. During development, a decision must be made on the basis of the requirements as to which variant of the sub-system adequately fulfils them. Therefore, it is important to consider the variants of sub-system and their integration on the system level.

For concretising of the reuse of a sub-system, figure 1 shows two cases of the reuse of a sub-system. The first case refers to the situation in which a sub-system is initially described as a part of a system description (System 1 in figure 1) and this sub-system description has to be reused in another system (System 2 in figure 1). The second case refers to the situation in which the sub-system is described independent of the system it is going to be used in and is than used in multiple systems. Furthermore, the sub-system can itself have multiple implementation (System N and M in figure 1) or in other words instances as shown in figure 1. In order to make such a reuse of a sub-system in multiple systems and the integration of variants in the current system development possible, the sub-system descriptions:

1. on one side must match the specification of the required sub-system (from top-down) and
2. on the other side, the sub-system itself as well as its variants must be described in a generic fashion so that it can be later integrated/used in other system descriptions as well.

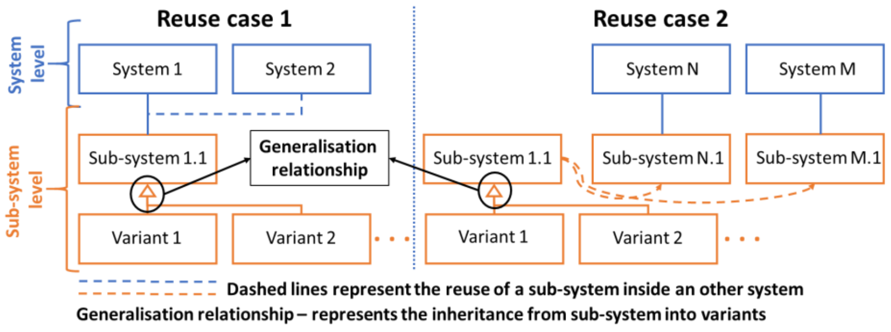
Reuse case 1
Reuse case 2
System level
Sub-system level
System 1
System 2
System N
System M
Sub-system 1.1
Generalisation relationship
Sub-system 1.1
Sub-system N.1
Sub-system M.1
Variant 1
Variant 2
Variant 1
Variant 2
Dashed lines represent the reuse of a sub-system inside an other system
Generalisation relationship – represents the inheritance from sub-system into variants

Figure 1. Two cases of development and reuse of a sub-system description

The description and integration/ reuse of the sub-systems (bottom-up) inside multiple systems (top-down) is addressed inside this paper while using Model Based System Engineering (MBSE). In MBSE, the information about the system (requirements, solution element) and development related information (e.g. decisions) are captured in model descriptions with static and dynamic parts. A main characteristic of these models is that these contain links between modelling elements that are realising different aspects (e.g. requirements view, structure view and solution view) of the model (i.e. representing the system/ a sub-system). For instance, starting from capturing the stakeholder requirements to definition of system architecture and behaviour, these aspects can be linked to achieve traceability between different model elements (Aleksandraviciene and Morkevicius, 2018). Today, MBSE is being applied in the development of complex mechatronics systems in the industry and is also an active topic of research in academia (Mandel et al., 2020). Among the languages used to develop the MBSE models, Systems Modelling Language (SysML) is a standardized graphical language (OMG, 2019) that finds widespread application in academia as well as in the industry. SysML facilitates the capturing of the information about the systems, i.e. requirements, structure, behaviour, use cases etc. in the form of graphical diagrams. It is an abstract language and therefore, can be used to model systems from different industrial sectors. However, this abstraction leads to certain challenges as well, for instance, there are multiple possible ways of describing a system aspect or different modelling architecture can be used to reach similar modelling goals (Mahboob, A. 2021). Due to this reason, a number of researches are focusing on defining the scope of different modelling architecture and modelling elements for describing specific aspects related to the product/system under development (Husung et al., 2021).

As most product developments are further developments of existing products/systems, the information (especially the findings) from previous development processes is often reused (Albers et al., 2015). Therefore, the reuse of already modelled SysML descriptions is an active topic of research (Mendieta et al., 2017). Furthermore, the reuse of model descriptions is vital in reducing the needed modelling effort for each development cycle. Depending upon the model architecture, modelling method and elements used to describe a system with SysML, the reuse requires a certain level of remodelling effort (Trujillo and Madni, 2020). Therefore, in this paper, the authors also discuss the reuse of already existing SysML models while reducing the need of remodelling. Also, a sub-system description may well be reused in multiple systems in parallel (a more detailed graphical representation of the reuse situations from figure 1 is shown in figure 2). Figure 2 shows a reuse situation involving the use of a sub-system in multiple systems, along with the graphical depiction of top-down and bottom-up development. Based on the needs of stakeholders, the required system specifications are developed (top-down). On the logical decomposition level, the specifications are detailed and further divided into required sub-system specifications that are needed to realise the overall requirements. The required sub-system may have multiple variants (multiple As-is-descriptions) that can be used inside a system. Therefore, it is important to compare a given sub-system's descriptions (is-descriptions) with the required specification (from top-down) to find the best suitable sub-system descriptions (bottom-up integration). Furthermore, one is-descriptions of a sub-system may well be used in the realisation of multiple systems (as shown in figure 1 on the right and in figure 2 where *sub-system1v2* can be used in *System 1, 2 and N*).

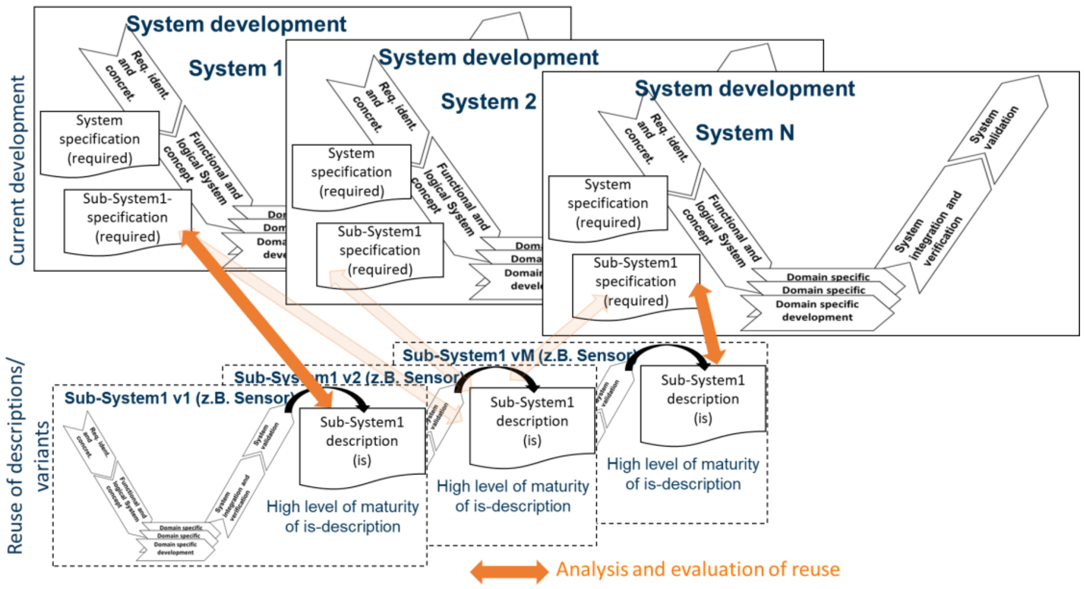
Current development
System development
System 1
System 2
System N
Req. ident. and concret.
System specification (required)
Functional and logical System concept
Sub-System1-specification (required)
Domain specific development
System integration and verification
System validation
Reuse of descriptions/ variants
Sub-System1 v1 (z.B. Sensor)
Sub-System1 v2 (z.B. Sensor)
Sub-System1 vM (z.B. Sensor)
Sub-System1 description (is)
High level of maturity of is-description
Analysis and evaluation of reuse

Figure 2. Reuse of a sub-system in multiple systems

The reuse of a sub-system inside a system (figure 1 left) requires that the specification of the sub-system from top-down (*sub-system 1 specification (required)* from figure 2) must be considered in the development of bottom-up description of the sub-system (*sub-system1v1 description (is)* from figure 2). Only then, it can be ensured that the is-description can be used against the required-specification. The reuse of independently modelled sub-system in multiple systems (figure 1 right) requires the management of the dependencies between the required-specification and As-is-descriptions in such a way, the is-description can be used in other systems as well. For instance, the multiple As-is-descriptions of *sub-systems1* in figure 2 (i.e. *sub-system1v1, sub-system1v2…sub-system1vM*) are the different variants of *sub-systemv1* and they must be described in a way that they can be used in multiple systems. This is vital in achieving, for instance, an overall system model, trade-off studies to find the best suitable sub-system variants. The development of a sub-system description, that is compliant to the required specification as well as generic enough for reuse in multiple systems, is a very challenges task.
In the light of the above discussion, the authors will thrive to address the following research questions over the course of this paper:

1. How the bottom-up descriptions of a sub-system can be modelled in MBSE that match the required (top-down) specification?
2. How can the system descriptions or part of these descriptions from previous developments can be reused with minimum remodelling effort in the current development?
3. How different model variants can be built in such a way that the variant descriptions are consistent with required specification and are reusable at the same time?

The above research questions will be addressed systematically employing a case example of a mechatronic system, a load cell. The concept of generalization and inheritance in SysML are discussed with additional focus on the aspects relating to the modelling of different system variants as well as their reuse and integration beyond the scope of the system, they were developed for.

https://doi.org/10.1017/pds.2022.195 Published online by Cambridge University Press

2. State of the art

An important aspect associated with the use of models is enabling the reuse of information that is already captured in models into later/future development or reuse of different model variants. The reuse of the models can be planned or unplanned as well (Wu et al., 2020). Keeping the later reuse in mind, there are a number of contributions that deal with the reuse (Lange et al., 2018; Shani and Broodney, 2015; Wu et al., 2019) of models. Furthermore, there are several contributions that deal with the modelling of variants in SysML (Grönniger et al., 2008; Weilkiens, 2016). Although, the already developed SysML models can be reused, it is not always the most efficient solution if the reuse requires extensive remodelling effort. Therefore, it is at times necessary to estimate the remodelling effort needed to achieve the reuse (Trujillo and Madni, 2020).The linking (direct dependencies) between model elements corresponding to requirements, structure, behaviour etc. and between similar elements of two interacting models although contributes to the traceability, but it can at times render the reuse of such models difficult by increasing the amount of needed remodelling. Therefore, it is necessary to address the later reuse of MBSE model right from the start of the model development and modelling process.

Often, the focus is put on the development of a High-Level Solution Architecture (HLSA) that defines the overall model structure and enables the modelling of different variants that can be reused as well (Aleksandraviciene and Morkevicius, 2018; Mahboob et al., 2019). Such approaches rely on the use of the concept of *Generalization & Inheritance* (Trujillo et al., 2020) in SysML to build specialized variants of a generic model description, i.e. HLSA. The generalization makes it possible that structural properties (e.g. value or flow properties) and all the interfaces of a system under development can be kept consistent throughout all the variants by inheriting them from the parent, however, the reuse of any variant into any other model will require remodelling or relinking of the inherited properties and interfaces. This limitation can be visualised by means of figure 3 (based on reuse case 1 from figure 1).

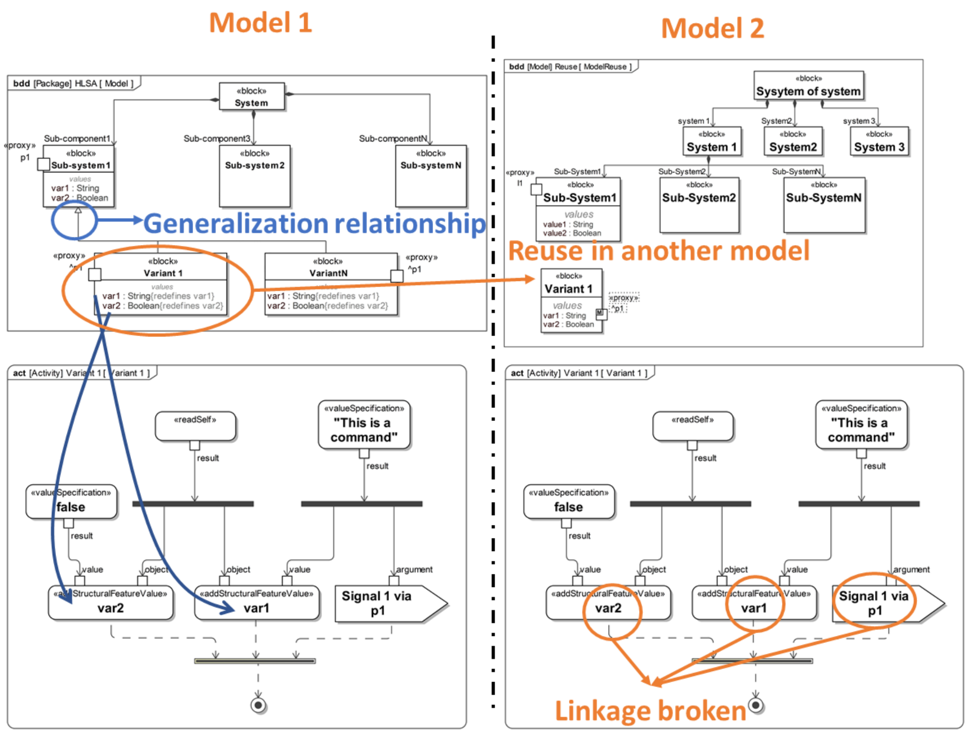

Figure 3. Reusing a variant in another model

The left side of figure 3 shows the already developed *"Model 1"* containing *"Variant 1"* as a variant of *"Sub-system1"* that inherits the ports and the properties. The inherited port and the properties are used inside the activity (bottom-left in the Figure 3) describing the example behaviour of the *Variant 1*. The reuse of the *Variant 1* inside another model (*Model 2*) or in another system (also see figure 1 right) is shown on the right side of the figure 3. Because in the modelling of *Variant 1*, the use of inheritance and redefinition of properties from the parent (*Sub-system1* in *Model 1*) builds direct dependencies with the elements present in the HLSA of the *Model 1*, the reuse of *Variant 1* in another model (*Model 2*) breaks all these dependencies (see the activity bottom-right in the figure 3). Therefore, manual rework will be needed to identify and connect the interfaces and properties to that of the *Model 2*. This might seem to be an easy task in the example explained here, however, it is extremely difficult and prone to error while dealing with a variant containing a detailed structural and behavioural description. On the other hand, the generalization cannot be fully left out if the descriptions contain variants of the system, sub-systems or sub-systems components. Therefore, the use of generalization has to be addressed while keeping the interdependencies between model elements in such a way that the reuse can be facilitated.

3. Method

The focus of the presented method and this paper with the product development activities can be understood by looking at the V-model. Figure 4 shows a simplified V-model representing the current product development process. The product development process starts with the identification of the needs and requirements of different stakeholders. Based on the identified needs, a system concept or multiple system concepts may be developed that contain the required specifications of system, sub-systems, components etc. These required specifications can be modelled in the form of an HLSA. These required specifications are used further in the development of specialized descriptions (e.g. sub-system and its variants) that eventually realise the desired functionality. There may be multiple possible solutions for the realisation of a function or sub-functions and therefore, a trade-off to find out the best possible solution from a number of available possibilities for a given sub-system may be required. These multiple possibilities can be implemented as variants and be evaluated by building instances in SysML.

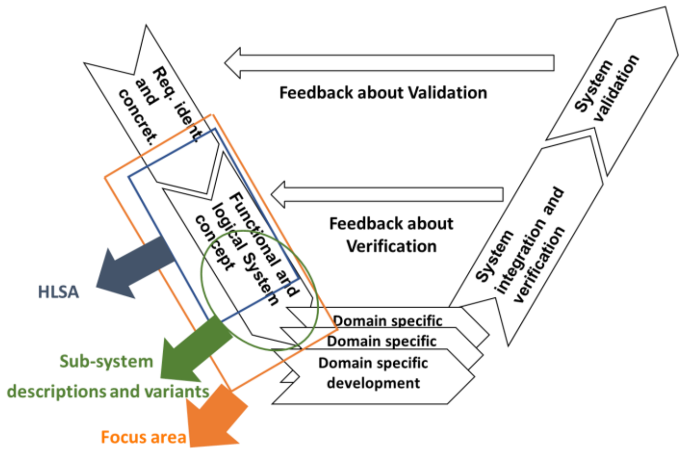
Req. ident. and concret.
Feedback about Validation
System validation
Feedback about Verification
Functional and logical System concept
System integration and verification
HLSA
Domain specific
Domain specific
Domain specific development
Sub-system descriptions and variants
Focus area

Figure 4. V-model and focus area of the presented method[1]

The same development process can now be seen from the system description standpoint using MBSE/SysML. The *"Functional and logical system concept"* can be realised by developing an abstract HLSA containing the requirements, specifications of the system and its sub-systems identified so far along with system environment and interfaces etc. This HLSA is then used as input to develop individual sub-system descriptions. This means that the HLSA is used as the reference model for the development of sub-system descriptions that intern refers to the reuse of elements defined inside HLSA into the sub-system descriptions. There may be multiple systems concepts and multiple sub-system variants under development and therefore, an HLSA is vital for achieving consistency in all variant descriptions. However, each system's logical concept or parallel developing system (see *System 1* to *System N* in figure 2) may have its own HLSA that is reusing a certain description of a sub-system or its variant (see *Sub-system1v1* to *Sub-system1vM* in figure 2). This raises the challenge of defining the sub-systems in such a way that their descriptions on one side, conform to the HLSA specifications (structure, interfaces etc.) and on the other hand, possess minimum direct dependencies with the elements of one particular HLSA so that they can be reused in another HLSA as well.

[1] Req. ident. and concret.: Requirements identification and concretisation

https://doi.org/10.1017/pds.2022.195 Published online by Cambridge University Press

3.1. Modelling

In order to address the above discussed challenges, the authors propose that the scope of linking between the elements of HLSA and that of the sub-system/ a variant has to be managed. In other words, it has to be checked which elements of an HLSA can be inherited and reused in describing a sub-system or a variant in such a way that they do not limited the reuse capability of a variant description. Furthermore, the inherited elements of a HLSA that may require remodelling for a reuse scenario must not be used directly in the variant specification. This concept can be visualised as shown in the figure 5.

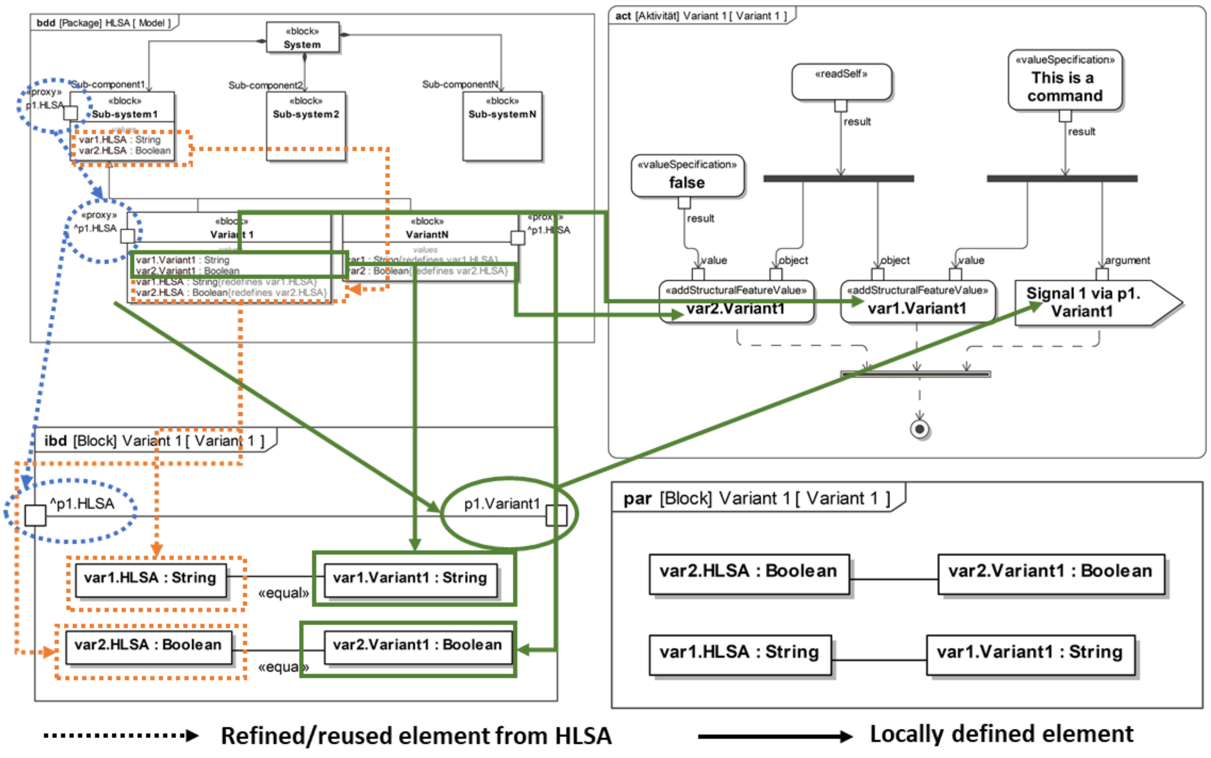

Figure 5. Modelling approach for facilitating the reuse

The upper-left corner of figure 5 shows an example HLSA model and the rest of the three diagrams (ibd (bottom-left), par (bottom-right) and act(up-right)) are the part of *Variant 1*. In order to keep the structural specification of *Variant 1* compliant to that of HLSA, a generalization relationship is established and the port as well as the properties are refined from *Sub-System1* to its variant, i.e. *Variant 1*. Keeping the later reuse of *Variant 1* in mind, the refined elements must not be used directly in the detailing, instead the same elements can be locally defined and linked with the refined elements using an ibd or a parameter diagram in SysML. Therefore, a similar port and two similar properties are defined locally (i.e. *var1.Variant1*, *var2.Variant1* and *p1.Variant1*) and are bound (over either ibd or par diagram) to the inherited elements of HLSA (i.e. *var1.HLSA*, *var2.HLSA* and *p1.HLSA*). From here on, the locally defined elements can be used in detailing the behaviour of the *Variant 1* as shown in the upper-right side of figure 5 in the form of an activity (act) diagram. Comparing the activity diagram from figure 4 to the activity diagram in *Model 2* in figure 2, it can be identified that there are no direct dependencies between the HLSA elements and the modelled elements of *Variant 1*. In this way, the linkage between the elements of the activity diagram of *Variant 1* will not break even if it is used outside the scope of current HLSA. The only remodelling needed to enable this reuse will be in the par and ibd diagrams, where the port and properties of the *Variant 1* have to be linked with that of the new HLSA. This remodelling will be greatly reduced in terms of the needed effort as compared to linking all the elements from the complete *Variant 1* specification (structure as well as behaviour), as it was the case in the figure 2. Following these lines, a sub-system can be developed that is compliant to the HLSA and can be reused with less effort in multiple (parallel developments) systems.

3.2. Modelling elements and analysis

In the modelling concept presented in the last sub-section, only one inherited port and two properties were shown for the sake of simplicity and easy understanding. The modelling concept was applied and analysed inside a detailed example of a load cell that is explained later in section 4. The gained experience in doing so is summarized in the table below as general recommendations from the authors on the usage of other SysML elements in HLSA and in the variant descriptions. These recommendations along with their scope of definition can be seen in table 1.

Table 1. Recommendation for element usage in HLSA and in the variants

SysML element	HLSA scope (System level)	Variant scope (Sub-system level)
Generalization relation	✓	Reused from HLSA
Ports	✓	Refined ports from HLSA bound to local ports in the variant. Local ports used in variant descriptions
Value properties or flow properties etc.	✓	Refined properties bound to local properties in the variant. Local properties used in variant descriptions
Element linking	×	Element linking done in variant descriptions
Signals	✓ (only for reference)	Signal reception, only possible with a direct dependency to source signal (explained towards the end of section 4)
Model sharing	✓(HLSA packages saved as a separate project and shared with variants)	Variants use the HLSA packages (read only) as reference and are saved also as individual projects
Interface specification	✓	Refined and linked to local interface specification
Actions, states, activities, state machines and other behaviour elements	✓(Limited definition. Only for the higher-level abstract model detail)	Defined locally for the detailed specifications in the variant descriptions
Instances	✓	×

4. Example and discussion

In order to demonstrate the modelling method presented in this paper, a load cell example was used. A load cell is a mechatronic system and it is chosen as the case example because it also depends on the context (e.g. environmental conditions) for its working. It measures the amount of applied force by measuring the strain caused by the applied force onto a strain gauge. There are multiple different constructions available for different kind of strain gauges that are available in the market, however, their discussion lies beyond the scope of this paper. Figure 6 (left) shows an example (simplified) strain gauge for understanding. The applied force (F) causes a deformation in the body of the load cell that induces strain in the strain gauge attached to the body of the load cell. This strain leads to changes in the resistance that can be directly measured (*U measuring*) for a given supply voltage (*U0*). Furthermore, the environmental conditions e.g. ambient temperature, humidity etc. can also have an influence on the output of a strain gauge load cell.

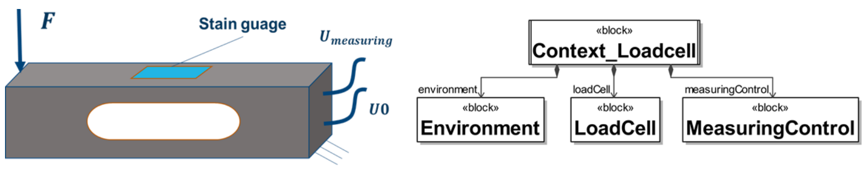

Figure 6. Strain gauge load cell (left) and HLSA of load cell sub-system (right)

Figure 6 (right) shows the HLSA of the load cell as a sub-system of a system containing load cell and environment model, along with the *MeasuringControl* representing the use of load cell output. This HLSA contains the specification of the load cell, the environmental aspects and the detail about the usage of the load cell output. The development of the load cell can be seen as a sub-system that can have multiple different versions/variants as well, therefore, three variants of the load cell model were developed that are shown in the figure 7.

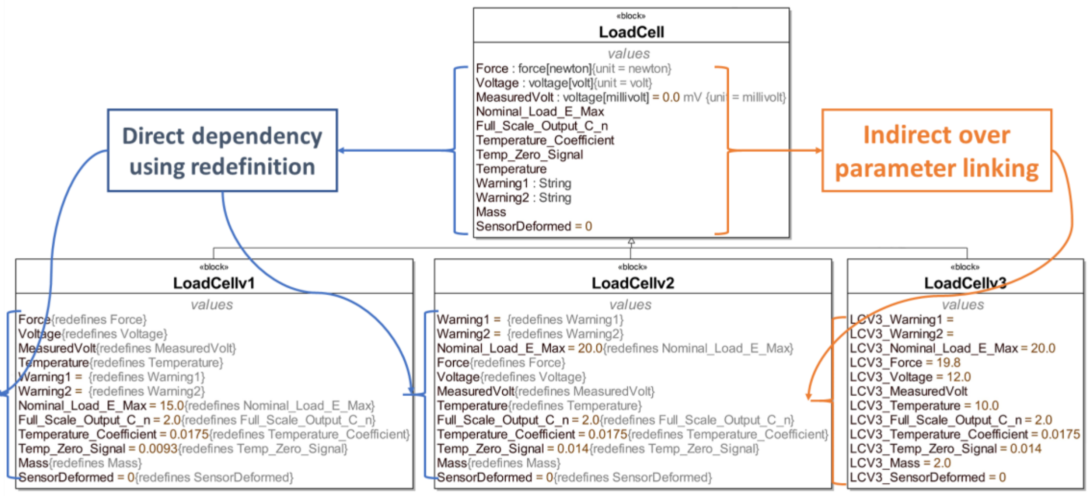
«block»
LoadCell
values
Force : force[newton]{unit = newton}
Voltage : voltage[volt]{unit = volt}
MeasuredVolt : voltage[millivolt] = 0.0 mV {unit = millivolt}
Nominal_Load_E_Max
Full_Scale_Output_C_n
Temperature_Coefficient
Temp_Zero_Signal
Temperature
Warning1 : String
Warning2 : String
Mass
SensorDeformed = 0
Direct dependency using redefinition
Indirect over parameter linking
«block»
LoadCellv1
values
Force{redefines Force}
Voltage{redefines Voltage}
MeasuredVolt{redefines MeasuredVolt}
Temperature{redefines Temperature}
Warning1 = {redefines Warning1}
Warning2 = {redefines Warning2}
Nominal_Load_E_Max = 15.0{redefines Nominal_Load_E_Max}
Full_Scale_Output_C_n = 2.0{redefines Full_Scale_Output_C_n}
Temperature_Coefficient = 0.0175{redefines Temperature_Coefficient}
Temp_Zero_Signal = 0.0093{redefines Temp_Zero_Signal}
Mass{redefines Mass}
SensorDeformed = 0{redefines SensorDeformed}
«block»
LoadCellv2
values
Warning1 = {redefines Warning1}
Warning2 = {redefines Warning2}
Nominal_Load_E_Max = 20.0{redefines Nominal_Load_E_Max}
Force{redefines Force}
Voltage{redefines Voltage}
MeasuredVolt{redefines MeasuredVolt}
Temperature{redefines Temperature}
Full_Scale_Output_C_n = 2.0{redefines Full_Scale_Output_C_n}
Temperature_Coefficient = 0.0175{redefines Temperature_Coefficient}
Temp_Zero_Signal = 0.014{redefines Temp_Zero_Signal}
Mass{redefines Mass}
SensorDeformed = 0{redefines SensorDeformed}
«block»
LoadCellv3
values
LCV3_Warning1 =
LCV3_Warning2 =
LCV3_Nominal_Load_E_Max = 20.0
LCV3_Force = 19.8
LCV3_Voltage = 12.0
LCV3_MeasuredVolt
LCV3_Temperature = 10.0
LCV3_Full_Scale_Output_C_n = 2.0
LCV3_Temperature_Coefficient = 0.0175
LCV3_Temp_Zero_Signal = 0.014
LCV3_Mass = 2.0
LCV3_SensorDeformed = 0

Figure 7. Variants of the load cell

Figure 7 shows the load cell block from the HLSA (figure 6) and its three variant (i.e. *LoadCellv1*, *LoadCellv2* and *LoadCellv3*). For easy understanding, only the values of each Block are shown in the figure. The first two variants are developed using generalization and redefinition of the elements from HLSA. Their redefined elements are then used in the behaviour description of both of the load cells. This builds direct linking /dependencies of the behaviour descriptions of the variant with the elements of HLSA. Therefore, the reuse of a variant defined in such a way leads to the problem of broken linkages (see figure 3) and remodelling of the behavioural descriptions of the variant. These direct dependencies are avoided in the modelling of *LoadCellv3* and the modelling method mentioned in section 3 is followed. HLSA is used in the modelling of third variant with generalization as well to inherit the specifications of the HLSA in the specific variant to achieve consistency. However, all the elements of interest that will be needed in detailing the variant or in modelling of its behaviour are defined locally first. These locally defined elements are then used in describing the behaviour of the variant. In this way, the direct dependencies between HLSA elements and the elements of *LoadCellv3* can be avoided. The locally defined elements of *LoadCellv3* have to be linked with their counterparts from HLSA to have consistency and to keep the variant description complaint with that of HLSA. Also, such a linkage is important considering instance building, trad-off studies or requirements verification etc. This linkage can be done using a parameter diagram defined within *LoadCellv3* description and is shown in figure 8.

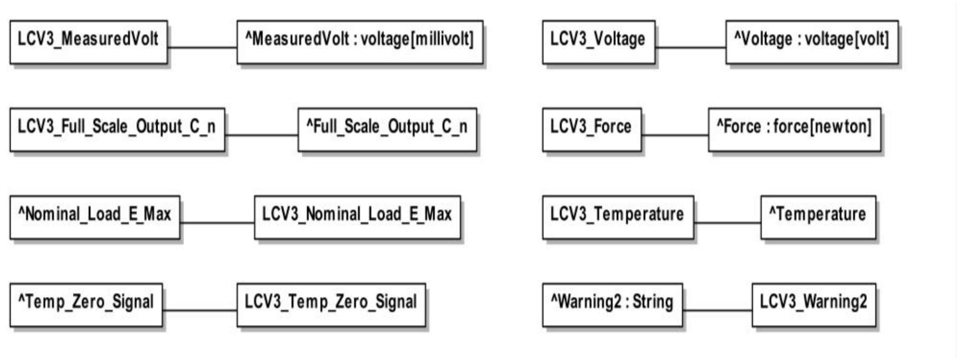

Figure 8. Linking HSLA parameters to that of variants ("^" represents redefinition from HLSA)

Figure 9 shows the binding of locally created elements of the variant with that of the HLSA. Now, if we consider the reuse to a variant defined in a similar way as the *LoadCellv3*, there will be no broken linkages (see figure 3) requiring the remodelling. Instead, the parameter diagram containing the element binding has to be remodelled only that is fairly easy task compared to figuring out all the broken linkages. The same approach can be used for the inherited ports and the inherited ports can be connected to the locally created ports as shown in figure 9.

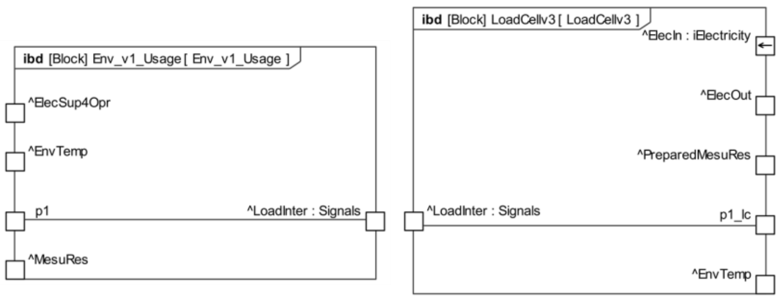

Figure 9. Linking inherited interfaces (environment model variant left, load cell variant right)

The local definition of ports is important in the modelling of local properties and signals while keeping the specifications of them compliant to that of HLSA. In the modelling example presented here, it is possible to let a signal flow from a load cell variant into a variant of environment model. The signal that originates from the load cell model flows over the port *p1_lc* from the load cell, through the *LoadInter : Signals* to the *p1* port in the environment model. In the behaviour description of the variant of the environment model, this signal can be received over the port *p1*. However, the original signal that is defined in the load cell model has to be directly linked to the signal receiving element (Accept Event Action in SysML) inside the description in the environment model. This although enables the reception of the signal, but it builds a direct dependency that will require the remodelling/manual linking of such signals in the individual descriptions. At its current maturity level, the presented method cannot cover the scope of signal modelling without direct dependencies, and thus it is a limitation at this point. As a workaround, the authors recommend either avoiding the use of signals across interacting variants or the signal receptions points in the variant descriptions can be clearly documented (for instance, using comments in the model or as a part of additional documentation) to make them easily identifiable for the reuse scenarios.

The goal of developing the method is to facilitate the reuse of sub-system description or variant specifications in the HLSA to perform trade-off or to verify the requirements. Therefore, an instance containing the *LoadCellv3* along with one variant each of environment and measurement control unit was created finally. Executing this instance, executes each variant in parallel and the requirements that were defined in the HLSA, their fulfilment can also be verified. In this way, the presented modelling method facilitates the systematic modelling of sub-system descriptions as well as the variants and there by answers the research question 1. The use of HLSA and avoidance of direct dependencies cover the last two research questions by showing the reuse of a sub-system beyond its original scope of development and their integration in another HLSA. Also, the discussion about individual modelling elements and their scope of definition helps in avoiding the remodelling effort needed to reuse an already defined descriptions or the description of a variant.

5. Conclusion and further steps

In this paper, a modelling method is developed to address the reuse of SysML models (including structural and behavioural descriptions) of sub-systems and their variants in multiple system descriptions. This paper starts by identifying the need, advantages and challenges associated to the reuse of system descriptions during product development. After highlighting the significance of direct dependencies between the modelling elements in system and variant descriptions, their influence on remodelling effort needed to enable the reuse is discussed. An enhanced modelling method is presented that address the reuse perspective of description in future development right from the early modelling stage and the knowledge gained in the development of this method is presented by clarifying the scope of individual SysML model element as well as by including general recommendations for facilitating a later model reuse. Furthermore, the integration of sub-system descriptions (bottom-up) and of its

variants into the current system (top-down) development is discussed as well. The method is explained with the help of a case example of a mechatronic system that also takes the context of the system in account. As a result, the descriptions developed using the presented method facilitates the integration and reuse of sub-system descriptions, descriptions of their variants and already developed descriptions from previous developments in the current development.
Although the presented method facilitates the use of signals across different model descriptions that are interacting with each other, it still requires a direct linking with the original signal in HLSA that builds direct dependencies which eventually leads to remodelling. In the future, the method will be developed further to address this issue. Also, further SysML elements will be included to discuss their scope and effect of the reuse of SysML descriptions.

References

Albers, A., Bursac, N. and Wintergerst, E. (2015) 'Product Generation Development –importance and challenges from a design research perspective.', in *International Conference on Theoretical Mechanics and Applied Mechanics (TMAM 2015),* pp. 16–21.

Aleksandraviciene, A. and Morkevicius, A., eds. (2018) *MagicGrid® Book of Knowledge -A practical guide to Systems Modeling using MagicGrid*, LT-44146 Kaunas, Lithuania, Vitae Litera, UAB.

Grönniger, H., Krahn, H. and Pinkernell, Claas and Rump, Bernhard (2008) *Modeling Variants of Automotive Systems using Views* [Online]. Available at http://se-rwth.de/~rumpe/publications20042008/kr_mbeff_08.pdf#page=81.

Husung, S., Weber, C., Mahboob, A. and Kleiner, S. (2021) 'Using Model-Based Systems Engineering for need-based and consistent support of the design process', In: *23rd International Conference on Engineering Design (ICED21),* pp. 3369–3378. DOI: https://doi.org/10.1017/pds.2021.598.

Lange, C., Grundmann, J. T., Kretzenbacher, M. and Fischer, P. M. (2018) 'Systematic reuse and platforming: Application examples for enhancing reuse with model-based systems engineering methods in space systems development', *Concurrent Engineering*, vol. 26, no. 1, pp. 77–92.

Mahboob, A. (2021) Modelling and use of SysML behaviour models for achieving dynamic use cases of technical products in different VR-systems (Dissertation), Ilmenau.

Mahboob, A., Husung, S., Weber, C., Liebal, A. and Krömker, H. (2019) 'The Reuse of SysML Behaviour Models for Creating Product Use Cases in Virtual Reality', In: *22nd International Conference on Engineering Design - ICE19*, Volume: 1. Delft, The Netherlands, pp. 2021–2030. DOI: https://doi.org/10.1017/dsi.2019.208.

Mandel, C., Stürmlinger, T., Yue, C., Behrendt, M. and Albers, A. (2020) 'Model-Based Systems Engineering Approaches for the integrated development of product and production systems in the context of Industry 4.0', In: 2020 *IEEE International Systems Conference (SysCon),* pp. 1–7. DOI: https://doi.org/10.1109/SysCon47679.2020.9275902.

Mendieta, R., La Vara, J. L. de, Llorens, J. and Álvarez-Rodríguez, J. M. (2017) 'Towards Effective SysML Model Reuse', In: *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development.* Porto, Portugal, 19.02.2017 - 21.02.2017, SCITEPRESS - Science and Technology Publications, pp. 536–541. DOI: https://doi.org/10.5220/0006267605360541.

OMG (2019) The OMG Systems Modeling Language™ Version 1.6.

Shani, U. and Broodney, H. (2015) 'Reuse in model-based systems engineering', In: 2015 *Annual IEEE Systems Conference (SysCon) Proceedings.* Vancouver, BC, Canada, 13.04.2015 - 16.04.2015, IEEE, pp. 77–83. DOI: https://doi.org/10.1109/SYSCON.2015.7116732.

Trujillo, A., Weck, O. L. de and Madni, A. M. (2020) 'An MBSE Approach Supporting Technical Inheritance and Design Reuse Decisions', In: *ASCEND 2020.* Virtual Event. Reston, Virginia, American Institute of Aeronautics and Astronautics. DOI: https://doi.org/10.2514/6.2020-4167.

Trujillo, A. E. and Madni, A. M. (2020) 'Assessing Required Rework in a Design Reuse Scenario', In: *IEEE International Conference on Systems, Man, and Cybernetics (SMC).* Toronto, ON, 11.10.2020 - 14.10.2020, IEEE, pp. 2946–2951. DOI: https://doi.org/10.1109/SMC42975.2020.9283430.

Weilkiens, T. (2016) *Variant modeling with SysML*, Fredesdorf, MBSE4U. 9783981787573.

Wu, Q., Gouyon, D., Boudau, S. and Levrat, E. (2019) '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, 10/1/2019 - 10/3/2019. Piscataway, NJ, IEEE, pp. 1–8. DOI: https://doi.org/10.1109/ISSE46696.2019.8984571.

Wu, Q., Gouyon, D., Levrat, E. and Boudau, S. (2020) 'Use of Patterns for Know-How Reuse in a Model-Based Systems Engineering Framework', *IEEE Systems Journal*, vol. 14, no. 4, pp. 4765–4776.