Chaining Model Transformations for System Model Verification : Application to Verify Capella Model with Simulink

Christophe Duhil, Jean-Philippe Babau, Eric Lépicier, Jean-Luc Voirin, Juan Navas

Chaining Model Transformations for System Model Verification : Application to Verify Capella Model with Simulink

Christophe Duhil[1], Jean-Philippe Babau[2], Eric Lepicier[1], Jean-Luc Voirin[1], Juan Navas[3]
[1]*Thales Defense Mission System, Brest, France*
[2]*Lab-STICC, CNRS, UMR6285, Université de Bretagne Occidentale, Brest, France*
[3]*Thales Corporate Engineering, Velizy-Villacoublay, France*
{christophe.duhil, eric.lepicier, jean-luc.voirin}@fr.thalesgroup.com, jean-philippe.babau@univ-brest.fr, juan.navas@thalesgroup.com

Keywords: Cyber Physical System, Model Transformation, Model Simulation, Model Verification

Abstract: In the context of Model-Based System Engineering (MBSE), Thales has developed a method called Arcadia, and its dedicated workbench Capella. This approach provides engineer generic practices and tools to design system models in a coherent way. While models grew in complexity, the need emerged for model Simulation and verification. In this paper, a model based approach is proposed to provide an interpretation of the Capella dynamic behavior description of modeled systems. The approach allows targeting different semantics and facilitating reuse of legacy semantics. The idea is to enforce separation of concerns of semantics definition by defining a chain of five transformations. The approach ensures traceability between Capella source models and target models, facilitating interpretation of the verification results. We apply our approach to analyze dataflow diagrams of a Capella "clock radio" model. For this purpose we transform the Capella dataflow model to a Simulink model. The experimentation on the use case demonstrates the ability of the tool to catch model inconsistency problems.

1 INTRODUCTION

In the context of Model-Based System Engineering (MBSE), system engineers build a model of the system architecture to capture customer's requirements and needs. Designing architecture for complex cyber physical systems like airplanes, satellites or trains, implies to build wide and heterogeneous models shared by a large community of stakeholders. In this context, Thales has deployed a MBSE method called Arcadia (Jean-Luc. Voirin, 2017) and a modeling tool called Capella (Pascal Roques, 2017). Arcadia and Capella workbench provide guidelines and tools to describe architecture of complex cyber physical systems.

In an industrial context, a modeling error could dramatically impact the progress of a project whenever it's detected at the late stage of integration. So, there is a strong need of model verification and validation tools at the early stage of the design.

As discussed in (Kai Chen et al., 2005), the semantic of a Domain Specific Modeling Language may be either structural (how to describe concepts and relationships between concepts) or behavioral (how to interpret the execution of the concepts). Even if Capella allows the description of the behavior of the modeled systems, it does not propose a behavioral semantic. So, the verification of behavior part of Capella models requires the definition of an executable semantic for Capella. In the literature (Andrea Sindico et al., 2001) (Bassim Chabibi et al., 2018) (Daniel Chaves Café et al., 2013) (Slim Medimegh et al., 2018), the common solution is to transform the cyber physical system model (usually expressed with SysML (Sanford Friedenthal et al., 2012)) to a formal model providing an executable semantic. The first limitation of these approaches is that they consider only one target semantic when different heterogeneous semantics should be targeted, for different application domains (a hardware component does not have the same behavior as a software component) and different analysis domains (simulation environment or formal description). Another point is that such model transformation integrates different aspects of semantic definition such as the mapping of source and target concepts, the refactoring of source concepts to facilitate alignment of concepts, the definition of the analyzable subset of source model, or the necessary

model adaptations and enrichments for analysis. Implementing a unique model transformation is not suitable for such transformation. The application of the separation of concerns facilitates the reuse of the different parts of the semantic definition. For instance, subset definition and adaptations may be reused to target both simulation and formal languages. Another important point is that each aspect of the transformation plays a specific role in the semantic definition. It must be explicitly specified, independently of other aspects. And last, for each step, the traceability is a key point to help the engineer to interpret analysis results by making a link between the target concepts and the source Capella concepts.

In this paper we propose a model-based method to transform Capella models, to models conforming a simulation meta-model. In the first section of this paper we introduce the Arcadia method and the Capella tool. In the second section we discuss the related works. Then, we propose our methodological approach based on five transformations. The fourth section is dedicated to the experiments results. The conclusion proposes some future works.

2 ARCADIA AND CAPELLA

Arcadia is a model-based engineering method developed by Thales to meet its engineering needs for hardware and software systems design.

The Capella workbench implements the Arcadia method. The objective of Capella is to propose a common description of system architecture, considering different purposes and levels of abstraction. The language of Capella makes it possible to describe the structure and the behavior of the systems but it does not provide an operational semantic. Verifying some behavior-related properties based on simulation (among others) may require defining an operational semantic. This is the purpose of this article.

We illustrate our approach by translating a data flow of a Capella model to a Simulink model in which it's possible to perform simulations and model verification.

Data flow is defined in the Arcadia method as the description of the relationships between different elements of the model in terms of interactions or exchanges (mainly functions or operational activities). In our case study, *Data flow* is used to describe the dependency relations between *Functions*. *Functions* are linked together through *Functional Exchanges*. A *Functional Exchange* defines a functional dependency between a source *Function* and a target *Function*.

Now we present representative related works addressing the formalization and verification of system modeling.

3 RELATED WORKS

In this paper, we propose to add a behavioral semantic to Capella. For such purpose, (Benoit Combemale et al., 2009) proposes a taxonomy based on three approaches: by defining an axiomatic semantic dedicated to Capella, by extending the Capella meta-model with an operational semantic or by transforming a Capella model to a model conforming to a meta-model containing a behavioral (axiomatic or operational) semantic.

For the first approach, (José E. Riviera and Antonio Vallecillo, 2007) proposes to add a formal semantic to DSL by describing the meta-model using Maude. So, even if it follows the first approach by providing an axiomatic semantic to the DSL, the idea is to apply the third approach at meta-level: the Ecore model is transformed into a Maude model, providing sufficient features to define an axiomatic semantic.

The authors of (Benoit Combemale et al., 2016) propose to follow the second approach by extending the Capella meta-model to add an operational semantic to data flow and state-machines. The model is then executed in a simulation environment built from GEMOC methods and tools(Erwan Bousse et al., 2016). The approach demonstrates the necessity of adding elements on Capella models to define a precise semantic. The limitation of this approach is that only one behavioral semantic is targeted. And, as illustrated by (Cécile Hardebolle and Frédéric Boulanger, 2009), a system designer may have to specify different semantics for the same part of a model, here the semantic of adaptation between heterogeneous models.

Due to the size of Capella meta-model and the consideration of heterogeneous targeted domains, we consider the third approach as the more adapted (see motivation section after). But, from our knowledge, (Benoit Combemale et al., 2016)is the only work, proposing an operational semantic to Capella. So for the third approach, we consider works based on SysML. SysML is a general-purpose graphical modeling language defined by OMG for complex and heterogeneous system modeling. From modeling objectives, SysML is close enough to Arcadia language to be considered in this section in place of our language (Polarsys, 2019).

(Andrea Sindico et al., 2001) (Bassim Chabibi et al., 2018) (Daniel Chaves Café et al., 2013) (Slim

Medimegh et al., 2018) propose a direct transformation from SysML to a target model via a model to text transformation. For such transformation, the authors propose first a mapping of concepts between SysML and the target domain (Simulink for (Andrea Sindico et al., 2001) and (Bassim Chabibi et al., 2018), SystemC for (Daniel Chaves Café et al., 2013) and hybrid model for (Slim Medimegh et al., 2018)). To prepare the mapping, in (Bassim Chabibi et al., 2018) and (Slim Medimegh et al., 2018), the authors use SysML stereotypes applied to a subset of SysML elements (structure of blocks, blocks, ports and flows). The stereotypes contain additional information for the target domain. In (Andrea Sindico et al., 2001) and (Daniel Chaves Café et al., 2013), the same SysML structural elements (blocks, ports and flows) are mapped to the target domain and, after mapping, the result is enriched with necessary additional information for execution. We can notice that (Benoit Combemale et al., 2016) also enriches the initial model to perform simulation. In (Daniel Chaves Café et al., 2013), adaptations are performed in order to generate a correct code. Model-based code generator (Acceleo, 2019) are used in (Andrea Sindico et al., 2001) (Bassim Chabibi et al., 2018) (Daniel Chaves Café et al., 2013) (Slim Medimegh et al., 2018) to implement the final transformation. All these works propose at least five operations to transform the source model in an executable one. First is selection of required and analyzed concepts. All approaches consider a limited subset of SysML, even if the subset definition is implicit. Secondly an operation of mapping of concepts is performed to define the semantic of the source elements. This operation defines how the source concepts become target concepts. The approaches consider implicitly that the target concepts exist (with different names) in the SysML meta-model. But in general case, a preparation (refactoring operation) is necessary to prepare the alignment of concepts. The third operation is an enrichment operation. The missing information is added to the model to conform to the simulation environment. And finally, an adaptation operation may be performed before the last code generation operation. When no adaptation is proposed, the limitation of the approach is the implicit usage of a SysML pattern defined by the transformation. During the transformation process, (Benoit Combemale et al., 2009) shows the necessity of typing or equivalence relationship between models. If the model equivalence is not established, the result of the validation in the target domain cannot be interpreted in the source model.

All these approaches are adapted but limited to target one specific semantic. The different concerns of the semantic definition are composed in a same transformation and the traceability of links between source and target concepts is not enough detailed. In the next section, we present our approach to provide an operational semantic to a model by performing a chain of five operations: selection, refactoring, mapping, adaptation and enrichment.

4 APPROACH

4.1 Motivation

As discussed before, Capella doesn't provide behavioral semantic. A solution to define one may be to extend the Capella meta-model by providing a specific axiomatic or operational semantic. This approach is not suitable for our case because Capella is independent of a specific usage. The operational semantic has to be defined outside the tool and should fit the specifics designer needs (principle of separation of concerns).Furthermore, adding full semantics likely to allow behavior simulation would significantly increase the complexity of the tool, increase model maintenance and evolution costs.

If the behavioral semantic is defined in a separate model, it is important to not define yet another semantic: the approach has to improve reuse of legacy semantics.

In this paper, we propose to add an operational semantic by operating a chain of transformation on Capella models. Each transformation concerns a specific aspect of the operation of adding an operational semantic to a model. Applying our approach allows to transform the Capella data flow model to a target data flow model from which it is possible to perform different analysis according to the engineer needs.

As viewed in the related works, such transformation requires making the operations of selection, mapping and enrichment. In addition, it appears necessary to modify the model (refactoring) to prepare the mapping, and adapt the model to allow its execution. In our approach, the transformation is defined by these five chained transformation steps (selection, refactoring, mapping, adaptation, enrichment). Each transformation step follows a pattern defined by its intention, its principles and implementation guidelines. The intention gives the objective of the transformation. The principles define a set of constraints on transformation implementation. The implementation proposes guidelines on how to implement the transformation by reusing existing tools.

We present now the five transformation steps of the approach.

4.2 The Transformations

4.2.1 First Step: Selection

Intention The goal of the first step is to select model elements whose concepts are relevant for the analysis.
Principles We select a subset of Capella classes, attributes and references involved in the sub-domain to analyze. The resulting CapellaSelection meta-model is a type of the Capella meta-model in the sense of (Jim Steel and Jean-Marc Jézéquel, 2007) (Wuliang Sun et al., 2013). Because of the typing relationship, the operations defined on CapellaSelection are relevant for Capella.
Implementation We first select the concrete classes and the features representing the objects which are relevant for the analysis. To ensure meta-model correctness, all the kept references are linked to existing classes. We also select abstract classes containing attributes and references useful for the analysis. From these abstract classes, we select all the inheritance paths, including intermediate classes, ensuring an inheritance relationship between the abstract classes and the concrete classes. Finally, we select a class playing a role of root class for the selected classes. From this root class, we select all the classes and references insuring a containment path between the root class and all concrete classes.
Case Study The Capella meta-model contains 428 meta-classes, distributed in 20 meta-models and 25 packages. To analyze the Capella physical architecture data-flow, we select 12 concrete classes and 17 abstract classes.
Comments This step is usually implicit in most of the approaches proposed in the literature. From our point of view, it appears fundamental to explicit the subset of the Capella meta-model involved in the data flow.

4.2.2 Second Step: Refactoring

Intention The goal of refactoring is to organize the Capella Selection meta-model in order to prepare mapping.
Principles The resulting meta-model is only a compound of classes that can be directly mapped to the target meta-model. All the operations involved in this transformation are refactoring operation. We do not allow creation of information in this step. All the new information is derived features.
Implementation We use a library of refactoring operators (flatten, hide, move ...) provided by co-evolution approaches such as ModifRoundrip (Paola Vallejo et al., 2016) or Epsilon Flock (Louis M. Rose et al., 2010). For instance, one can flatten all the features of abstract classes before deleting them (combination of flatten and hide operators defined by ModifRoundtrip co-evolution tool).
Case Study In the new meta-model (see figure 1) all the attributes and references are moved to the concrete classes. Abstract classes are deleted. The classes involved in the structure description (Physical Architecture, Packages, Actor, and Component) are hidden. The classes and the objects are removed but the implicit references between Physical Functions and System Engineering remain. We obtain a new meta-model without any generalization. Objects involved in system hierarchy have been hidden. The resulting object graph is simpler than the original one. Only information needed for the verification objective is present. The figure 1 shows the impact of the refactoring stage on the object graph. The classes needed by the Capella structure are hidden.

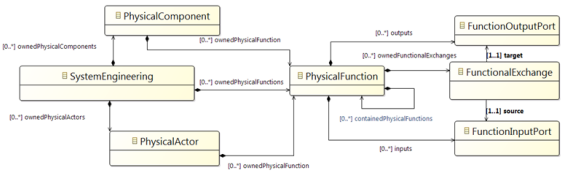

Figure 1: Meta-model after refactoring step.

Comments Selection and refactoring prepare the mapping; no extra-information is added to the model before mapping. If the mapping does not produce complete result, we may add information after, in the context of the target meta-model. By facilitating alignment of concepts between Capella and target meta-models, refactoring enforces reuse of legacy semantics. From our point of view, there exist enough formal models to analyze the different Capella aspects.

4.2.3 Third Step: Mapping

Intention The goal of the mapping step is to translate the refactored Capella model to a model conform to the concepts defined by the target meta-model. The target meta-model is a legacy meta-model providing a behavioral semantic, equipped with simulation and analysis tools.
Principles We map each Capella concept (class, attribute and reference) to one concept of the target meta-model.
Implementation The mapping is implemented as a simple and direct model to model transformation. A wide range of tooling can implement such transformation (QVT (Ivan Kurtev, 2008), ATL (Frédéric Jouault et al., 2008), ETL (Dimitrios S. Kolovos et al., 2008), XTend (Xtend, 2019)). Using co-evolution tools like ModifRoundtrip, the only operator to use is the re-

name operator. To ensure the traceability from the source objects, the target objects keep the information of the source objects id.
Case Study The mapping of concepts to the target data flow concepts is given by the table 1.

Capella Concept	Target Concept
SystemEngineering	DataFlow
PhysicalComponent	Component
PhysicalActor	Actor
PhysicalFunction	FunctionBlock
FunctionInputPort	InputPort
FunctionOuputPort	OutputPort
FunctionalExchange	Link

Table 1: Concept mapping.

The resulting object graph is equivalent to the refactored one. From initial Capella model, only information required by the verification purpose is kept. In the approach, the modification of the object graph is only performed by the refactoring step.
Comments The mapping is a fundamental transformation to explicit the semantic: the source concepts are interpreted in the sense of the target domain. In the literature, it is usually the more commented aspect. If it is the core of the transformation, our approach proposes data selection and preparation to be complete.

4.2.4 Fourth step: adaptation

Intention In this step, the generated model is adapted to be executable in the chosen simulation environment. The goal of the adaptation step is to adapt the resulting model to respect the analysis tools constraints.
Principles The transformation is endogenous. The resulting model conforms to the target meta-model. Necessary information is added but no information is lost during this transformation. From the result, it is always possible to get the source model by using refactoring operators.
Implementation We add new objects and corresponding features to transform the model structure.
Case Study To simulate the model under Simulink we adapt the Capella data-flow structure to integrate the Simulink constraints. We find two critical cases: First of all, using Simulink (as opposed to Capella), it is not possible to link two different block output ports to the same block input port. To solve this problem, we add a Merge block receiving the two signals in two different inputs ports, the output of the Merge block is then linked to the block input port. Secondly, Simulink considers a cycle as an algebraic loop and requires additional information to execute it. So, when a cycle is detected, we add an Initial Condition block to the cycle.
Comments This step is fundamental for the approach. If this transformation is not included in the approach, the previous cases lead to non-executable models. Facing this problem, the designer would adapt its Capella model to respect the structural constraints given by the analysis tool. So the designer implicitly embeds, during system modeling, the tool verification concerns. In our approach, we hide this complexity during system modeling by adding a model correction in the adaptation transformation. This transformation is a part of the semantic given to the Capella model (here for multiples inputs and for cycles).

4.2.5 Fifth Step: Enrichment

Intention The goal of the final step is to add information required by the simulation environment.
Principles This transformation is endogenous. The resulting models have to contain all the required information for analysis. The parts of the model obtained by the previous transformations are not modified. We just add extra information. At the end the model is complete and correct for analysis.
Implementation Considering the target meta-model, we add the missing objects and set the non-generated features. Each added objects and features are set by a default object constructor. For specific simulation or evaluation, new features and new objects may be edited.
Case Study For Simulink simulation, we choose to verify the completeness of the data flow by propagating a signal through the data flow. In this case, typing is required for Simulink signals but it is not a part of Capella. So we type the signal as double. This choice makes easier the integration of loop in the data flow.
Comments By adding new information for analysis, this last step is a part of Capella semantic. Because, the operational semantic is dependent of verification objectives, new information may be edited to fit different verification objectives. The obtained model is then a refactored Capella model extended by adaptation and enrichment operations.

4.2.6 Discussion

The proposed approach is based on a chain of five consecutive transformations: selection, refactoring, mapping, adaptation and enrichment. The approach proposes to separate each concern of the transformation process dedicated to the semantic definition in different transformations:

- the first transformation impacts the semantic definition by limiting its definition area;
- the second transformation impacts semantic definition by refactoring source concepts to identify target concepts;
- the third transformation adds semantic to the model by mapping source and target domains : a source concept is interpreted as a target concept;
- the fourth transformation adds semantic by adding information to adapt to target tool specific patterns;
- the fifth transformation adds semantic by adding extra-information for analysis.

By applying the five transformation steps, we obtain an extended Capella, providing an operational semantic. The extension is due to the adaptation and enrichment transformations. Then by changing one step, we can target other specific semantics.

In the next section we present the details of the implementation of the transformations.

5 EXPERIMENTS

5.1 Implementation

A Capella model transformations have been implemented as an eclipse plugin. The plugin is integrated in the Capella workbench, taking as input a *SystemEngineering* object. It produces as output a MATLAB script (.m file). Because, the Simulink metamodel is not public, we generate a script, executed by MATLAB to build the Simulink model. The transformations are implemented using Xtend (Xtend, 2019). Xtend provides a language to easily implement both M2M and M2T transformations.

5.2 Clock Radio Case Study

We experiment our approach on a model describing the operations of a clock radio. The behavior of this system is described in a Capella physical architecture, by a data flow and a state machine. Three modes are available for the user. In the mode off, only the time display is available. The radio mode, broadcasts the radio and displays the time. The Alarm mode triggers the alarm and broadcasts the radio at a preset alarm time. The Capella state machine in figure 2, describes this three modes. Therefore the data flow describes the dependency relations between functions. For the radio clock case study, the data flow describes how

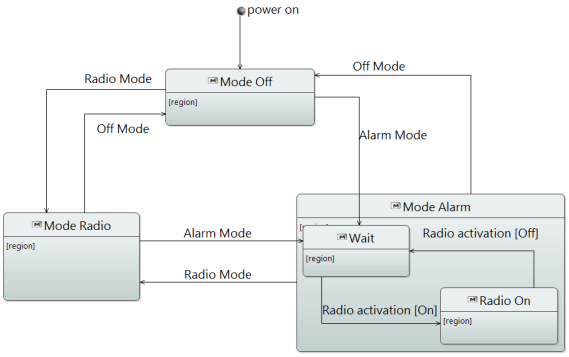

Figure 2: Capella Clock Radio State Machine.

is managed the human-machine interface (HMI), the time and the alarm.

The Data flow is linked to the state machine through functional exchanges and functions. Each state of the state machine defines a set of enabled functions. Considering the enabled functions, the data flow defines a set of available functional exchanges. Then an available functional exchange may trigger a transition of the state machine as shown in figure 3.

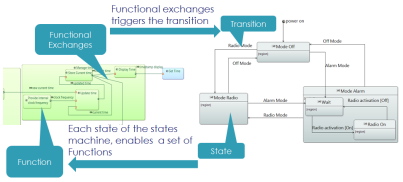

Figure 3: Interactions between data flow and state machine.

5.3 Model Transformation

We apply the transformation process to the clock radio data flow to obtain an image of the Capella data flow in Simulink.

During the adaptation step, the transformation tool produces a warning if a cycle is detected in the data flow and if two functional exchanges are plugged to a same function input port.

5.3.1 Warning: 'Cycle Detected'

In the Clock radio data flow, a cycle exists (see figure 4) between the *Update_time* function and the *Store_Current_time* function. By default the tool atomically adds an initial condition *ic_update_time* to validate at least one transition in the cycle. The goal of this warning is to warn the system engineer on the presence of a cycle in the Capella model.

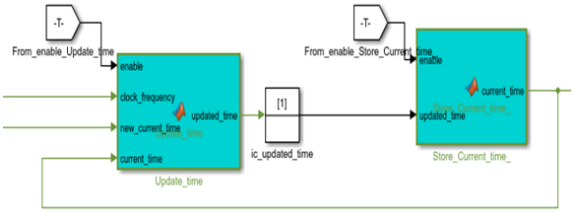

Figure 4: Simulink Initial Condition added to a cycle.

5.3.2 Warning: 'Multiple Connections on a Single Port Found'

The transformation tool finds two cases of multiple functional exchanges connected to a single function input port. In the first warning the functional exchanges *Radio_activation* (see figure 5), arriving on the same input port, is homogeneous and compatible (same type). The *Broadcast_Radio* function can receive both functional exchanges without any problem. This is not a modeling error.

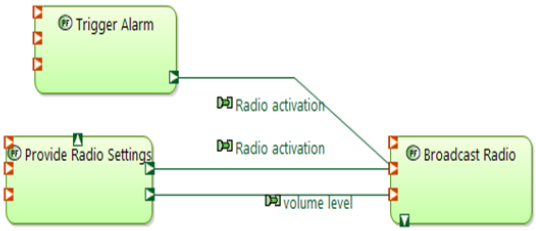

Figure 5: Capella model of multiple exchanges on a same input port.

For the second warning , functional exchanges arriving on the input port of the function *Switch power supply* are different and could not be compatible. It could be considered as a modeling error. The design may be re-factored by adding distinct input ports (see figure 6).

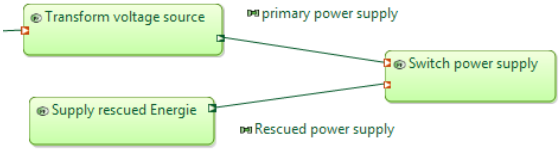

Figure 6: Capella model of the re-factored exchanges.

5.4 Simulation with Simulink

In the Simulink environment, we manually implement the clock radio state machine and build a test environment. This test environment is made to interact with the image of the Clock Radio data flow automatically created by the transformation tool. The goal of the simulation is to verify the completeness of the data flow in each state of the state machine. the data flow is complete when all the enable functions have their needed data available in input.

5.4.1 Error Found

In the first definition of the *alarm_mode state*, the functions *Trigger_Alarm and Broadcast_Radio* are set to enable. But the function *Decode_Radio_Waves* is set to disabled. This last function is not able to produce data, and then the *radio_signals* port receives a null data making *Broadcast_radio function invalid*. The data-flow is not complete. The figure 7 shows an excerpt of the Simulink diagram. During the simulation, enable-valid functions are colored in blue. None enable functions are colored in white. And invalid functions are colored in red. To correct the error the system engineer should include the *Decode_Radio_Waves* in the list of *enabled* function of the state *Alarm_mode*.

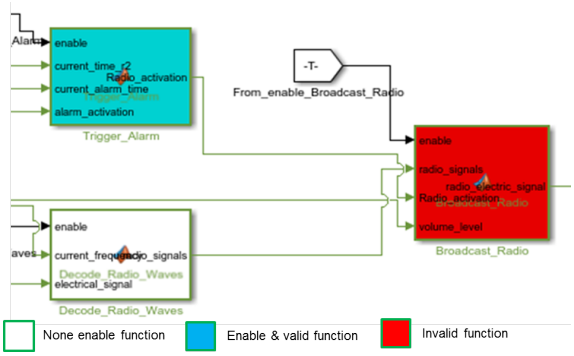

Figure 7: Simulation result of an incomplete data flow.

6 Conclusion

In this paper, we present a transformation method to add operational semantic to a descriptive Capella model. The transformation is based on five steps: selection, refactoring, mapping, adaptation and enrichment. The transformation chain ensures the consistency of the data through the transformation process. Each step plays a specific role in the semantic definition. We apply the approach to a Capella data flow model. From this data flow, the five transformation steps produce an executable Simulink model. Then we are able to detect modeling error and potential inconsistencies in the model. The simulation detects an

error due the incompleteness of the data flow and incompatible signals arriving on a single port. The application of the approach to the Capella data flow is a first step to verify the consistency of more complex heterogeneous Capella models. For future works, we will apply the approach to transform Capella models of state machines and scenarios with data flow.

REFERENCES

Acceleo (2019). Acceleo web-page. Accessed September 01, 2019. www.eclipse.org/acceleo/.

Andrea Sindico, Marco Di Natale, and Giampiero Panci (2001). Integrating SysML with Simulink using Open-source Model Transformations. In *1st Int. Conf. on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH'11).*, pages 45–56, Noordwijekr, The Netherlands.

Bassim Chabibi, Adil Anwar, and Mahmoud Nassar (2018). Towards a Model Integration from SysML to MATLAB/Simulink. *Journal of Software*, pages 630–645.

Benoit Combemale, Cédric Brun, Joèl Champeau, Xavier Crégut, Julien Deantoni, and Jérome LE Noir (2016). A Tool-Supported Approach of Concurrent Execution Of Heterogeneous Models. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, Toulouse.

Benoit Combemale, Xavier Crégut, Pierre-Loïc Garoche, and Xavier Thirioux (2009). Essay on Semantics Definition in MDE An Instrumented Approach for Model Verification. *journal of Software*, pages 943–958.

Cécile Hardebolle and Frédéric Boulanger (2009). Multi-Formalism Modelling and Model Execution. *International Journal of Computers and Applications*, 31,(3):193–203.

Daniel Chaves Café, Frédéric Boulanger, Filipe Vinci dos Santos, Christophe Jacquet, and Cécile Hardebolle (2013). Multi-Paradigm Semantics for Simulating SysML Models using SystemC-AMS. In *Forum on specification and Design Languages (FDL)*, Paris, France. IEEE.

Dimitrios S. Kolovos, Richard F. Paige, and Fiona A.C. Polack (2008). The Epsilon Transformation Language. In *International Conference on Theory and Practice of Model Transformations (ICMT)*, volume 5063 of *Lecture Notes in Computer Science*, pages 46–60, Zurich, Switzerland. Springer, Berlin, Heidelberg.

Erwan Bousse, Thomas Degueule, Didier Vojtisek, Tanja Mayerhofer, Julien Deantoni, and Benoit Combemale (2016). Execution Framework of GEMOC Studio (Tool Demo). In *ACM SIGPLAN International Conference on Software Language Engineering*, pages 84–89, Amsterdam, Netherlands. ACM.

Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev (2008). ATL: A model transformation tool. *Science of computer programming*, 72(1-2):31–39.

Ivan Kurtev (2008). State of the Art of QVT: A Model Transformation Language Standard. In Andy Schürr, Manfred Nagl, and Albert Zündorf, editors, *Applications of Graph Transformations with Industrial Relevance*, pages 377–393, Kassel, Germany. Springer Berlin Heidelberg.

Jean-Luc. Voirin (2017). *Model-based System and Architecture Engineering with the Arcadia Method*. ISTE Press. ISTE Press - Elsevier.

Jim Steel and Jean-Marc Jézéquel (2007). On model typing. *Software & Systems Modeling*, 6(4):401–413.

José E. Riviera and Antonio Vallecillo (2007). Adding Behavioral Semantics to Models. In *Enterprise Distributed Object Computing Conference*, pages 169–180, Annapolis, MD, USA. IEEE.

Kai Chen, Janos Sztipanovits, Sherif Abdelwalhed, and Ethan Jackson (2005). Semantic Anchoring with Model Transformations. In Alan Hartman and David Kreische, editors, *Model Driven Architecture - Foundations and Applications*, volume 3748 of *Lecture Notes in Computer Science*, pages 115–129, Nuremberg, Germany. Springer Berlin Heidelberg.

Louis M. Rose, Dimitrios S. Kolovos, and Richard F. Paige (2010). Model Migration with Epsilon Flock. In L. Tratt and M. Gogolla, editors, *Theroy and Practice of Mdel Transformation*, volume 6142 of *Lecture Notes in Computer Science*, pages 184–198, Malaga, Spain. Springer, Berlin, Heidelberg.

Paola Vallejo, Jean-Philippe Babau, and Mickaël Kerboeuf (2016). ModifRoundtrip: A Model-Based tool to reuse legacy transformations. In *MoDELS 2016 Demo and Poster Sessions co-located with ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2016).*, Saint Malo, France. IEEE Xplore.

Pascal Roques (2017). *System Architecture Modeling with the Arcadia Method : A Practical Guide to Capella*. Iste press - elsevier edition.

Polarsys (2019). Equivalences and Differences Between SysML and Arcadia/Capella, web-page. Accessed September 01, 2019. www.polarsys.org/capella/arcadia_capella_sysml_tool.html.

Sanford Friedenthal, Alan Moore, and Rick Steiner (2012). *A practical Guide to SysML, The Systems Modeling Language*. Object Management Group. Morgan Kaufmann, elsevier edition.

Slim Medimegh, Jean-Yves Pierron, and Frédéric Boulanger (2018). Qualitative Simulation of Hybrid Systems with an Application to SysML Models. In *6 th International Conference on Model-Driven Engineering and Software Development*, Funchal Portugal. SCITEPRESS - Science and Technology Publications.

Wuliang Sun, Benoit Combemale, Steven Derrien, and Robert B. France (2013). Using Model Types to Support Contract-Aware Model Substitutability. In Pieter Van Gorp, Tom Ritter, and Louis M. Rose, editors, *Modelling Foundations and Applications*, volume 7949 of *Lecture Notes in Computer Science*, pages 118–133, Montpellier, France. Springer Berlin Heidelberg.

Xtend (2019). Xtend web-page. Accessed September 01, 2019. www.eclipse.org/xtend/.