Capella to SysML Bridge: A Tooled-up Methodology for MBSE Interoperability

Nesrine BADACHE, ARTAL Technologies, nesrine.badache@artal.fr

Pascal ROQUES, PRFC, pascal.roques@prfc.fr

Keywords:

Modeling, Model, MBSE, System Engineering, DSML, Modeling Tool, Bridge, Interoperability, SysML, Arcadia, Capella, Model to Model mapping, Papyrus

Abstract:

Model to model transformation is a critical task in Model Based System Engineering (MBSE). Indeed, there is a growing need to migrate existing models, for instance in UML or SysML, to fit new modeling approaches such as Arcadia/Capella. The reverse is also desirable, as organizations have since long invested time and money in models and tools on top of these standards. In many situations a smooth integration of new tools requires compliancy with the in-place standards.

Capella [1] is a model-based engineering solution that has been successfully deployed in a wide variety of industrial contexts. Based on a graphical modelling workbench, it provides system architects with rich methodological guidance relying on Arcadia [9], a comprehensive model-based engineering method.

SysML [2] supports complex modeling for system engineering applications at different steps of the system life cycle. SysML provides architects and system engineers an easy way to collaborate using a unique common language. It enables the management of systems with growing complexity across different development teams with many modeling capabilities: requirement, behavioral and structural definitions. Refer to OMG SysML definition [2] for further details on SysML.

To take advantage of the power of the Capella tool, as well as the conformance to the standardized SysML language, this paper depicts a first attempt of Capella to SysML mapping and a prototype of transformation tool as proof-of-concept. This work is part of the Clarity project [3].

Motivation

Model to model transformation is a critical task in Model Based System Engineering (MBSE). Indeed, there is a growing need to migrate existing models, for instance in UML or SysML, to fit new modeling approaches such as Arcadia/Capella. The reverse is also desirable, as organizations have since long invested time and money in models and tools on top of these standards. In many situations a smooth integration of new tools requires compliancy with the in-place standards. Another typical use case is the exchange of models in the context of an extended enterprise, even if participants do not use the same modeling tools or even modeling languages.

To be able to interoperate efficiently between the new Capella tool [10] and existing SysML tools, a mapping of concepts should be performed first and then transformation tools should be provided to modelers. As part of the Clarity project [3], we have tried to initiate this mapping of concepts, considering only a subset of Arcadia/Capella concepts that focus on the most relevant features, and to build a

prototype transformation tool as proof-of-concept. The reverse mapping (SysML to Capella) is out of the scope of this paper.

We present hereafter this transformation specification and the associated strategies, as well as a mapping tool with an example based on a simple use case of an alarm clock available in the SysML literature. The paper focuses on Capella Components, Functions and the related Exchange modeling concepts in the Logical Architecture level. Similar mapping rules could be generalized over the other Capella levels (Operational, System and Physical). The transformation prototype tool is briefly presented as a proof of concept but is not detailed in this paper. It must be noted that the resulting model will require full review to ensure correctness as the tool cannot itself guarantee any. The resulting SysML model use case is explored in Papyrus/SysML [4]. Papyrus was chosen to be the SysML target, as it is technically based on the same Eclipse components as Capella, and is part of the same Polarsys project.

SysML with a tool vs Arcadia / Capella: elements of comparison

Arcadia method and Capella Tool

Arcadia [9] - Architecture Analysis and Design Integrated Approach (Cf. figure 1) is a Model-Based Engineering method for systems, hardware and software architectural design. It has been developed by Thales between 2005 and 2010 through an iterative process involving operational architects from all the Thales business domains (transportation, avionics, space, radar, etc.).

The Arcadia modeling language is inspired by UML/SysML and NAF standards, and shares many concepts with these languages. It enforces an approach structured on successive engineering phases which establishes clear separation between needs (Operational need analysis and System need analysis) and solutions (Logical and Physical architectures), in accordance with the IEEE 1220 standard. It should be noted that Arcadia provides a great variety of modeling concepts and diagrams and should be seen as a modeling framework, instead of a straightforward modeling methodology. If a project wants to use mainly artifacts also provided by SysML (such as sequence or state diagrams), Arcadia can be customized by methodological guidelines.

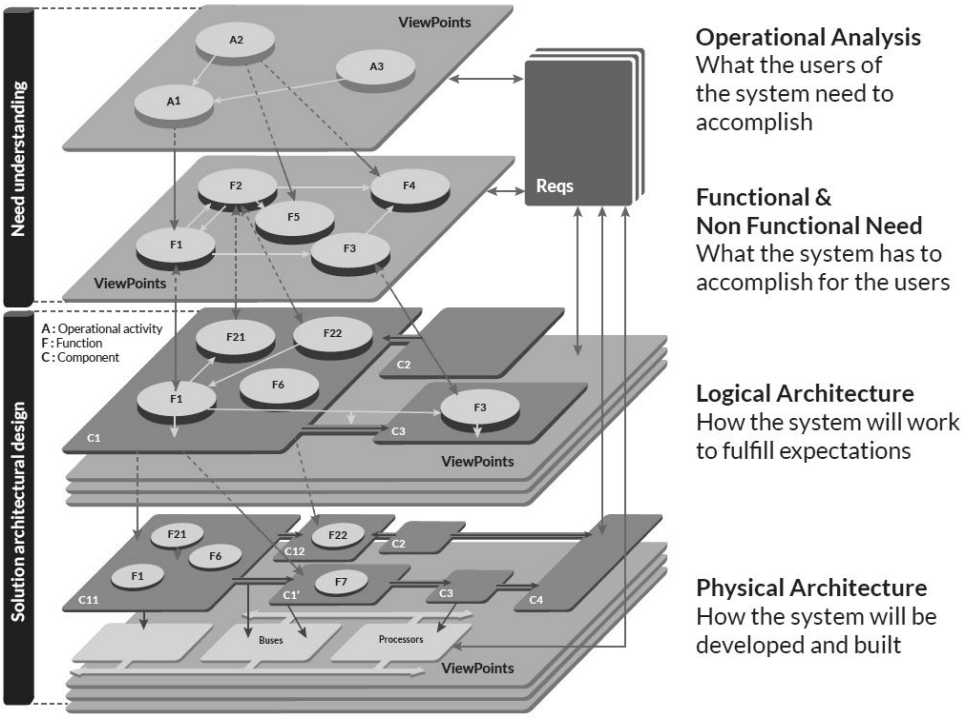

Figure 1 - Summary of ARCADIA levels

The Capella workbench is an Eclipse application implementing the Arcadia method providing both a Domain Specific Modeling Language (DSML) and an associated toolset.

SysML (System Modeling Language)

SysML™ is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities. It is a specialized UML profile targeted to system engineering. It defines nine different types of diagrams among four different pillars: Behavior, Structure, Requirements and Parametrics (Cf. figure 2). We focus hereafter on three important types of diagrams: block definition diagram (bdd), internal block diagram (ibd) and activity diagram (act).

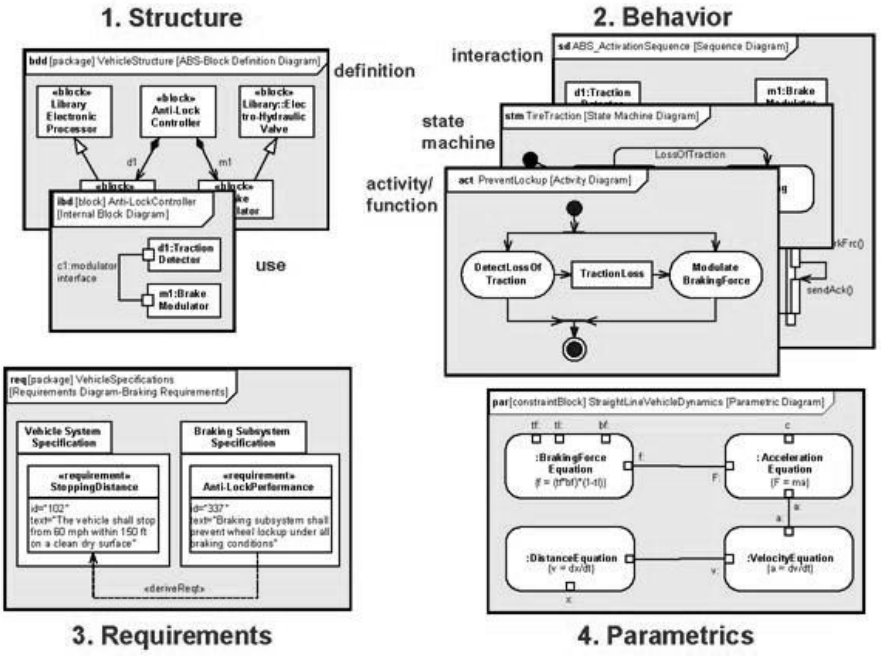

Figure 2 - The "Four Pillars" of SysML (OMG website)

Elements of comparison

As we explained earlier, the Arcadia DSML is inspired by UML/SysML and NAF standards, and shares many concepts with these languages. But a Domain-Specific Modeling Language was preferred in order to ease appropriation by all stakeholders, usually not familiar with general-purpose, generic languages such as UML or SysML. Previous experiments inside Thales proved that System Engineers not coming from software were not at ease with the object-oriented concepts proposed by UML (and subsequently by SysML). Therefore, Arcadia is mostly based on Functional Analysis, and then allocation of Functions to Components. The vocabulary of the DSML has proven to be easily understood by System Engineers.

Arcadia was defined first in Thales, from the engineering problems encountered in real projects. Then came the need for a software tool enabling to create and manage Arcadia models. The first experiments were done using existing UML tools such as Rational Software Modeler, Objecteering and Rhapsody, and defining UML profiles on top of them. At the time of these first tries, the commercial tools were not easy at all to customize, and it was difficult to remove unused commands or menus. That is why Thales people decided to create their own tool, dedicated to Arcadia, encouraged by the emergence of enabling technologies based on the Eclipse platform, such as EMF, etc. Arcadia definition can thus really be seen as the specification of the Capella modeling tool.

If we try to compare with other possible solutions, namely use a standard modeling language, such as SysML, and an existing commercial tool, such as Rhapsody, we can spot several important differences. SysML and Rhapsody (as the other commercial SysML tools) are based on UML, which is a

disadvantage for System Engineers who have not been exposed to object-oriented concepts: notions of operation, generalization / specialization, type / instance in block definition diagrams, and even of “Object” flow and “Object” node in the activity diagram. These object-oriented origins are clearly an obstacle to adoption by System Engineers who are not familiar with software development [5].

Another big issue is that SysML is only a language, and each company needs to elaborate an adapted modeling strategy. But then, how to teach the method to the modeling tool? Each commercial tool claims to offer an API to build specific add-ons, but this represents clearly a big investment, with software skills needed. A prototype was for instance provided by IBM with the “Harmony for SE” toolkit, but experiments in Thales proved that this toolkit was no more than a proof of concept when the decision to create Capella was taken. For instance, the automated transitions between modeling phases were not iterative and incremental, as is the case with Capella, but merely one-shot.

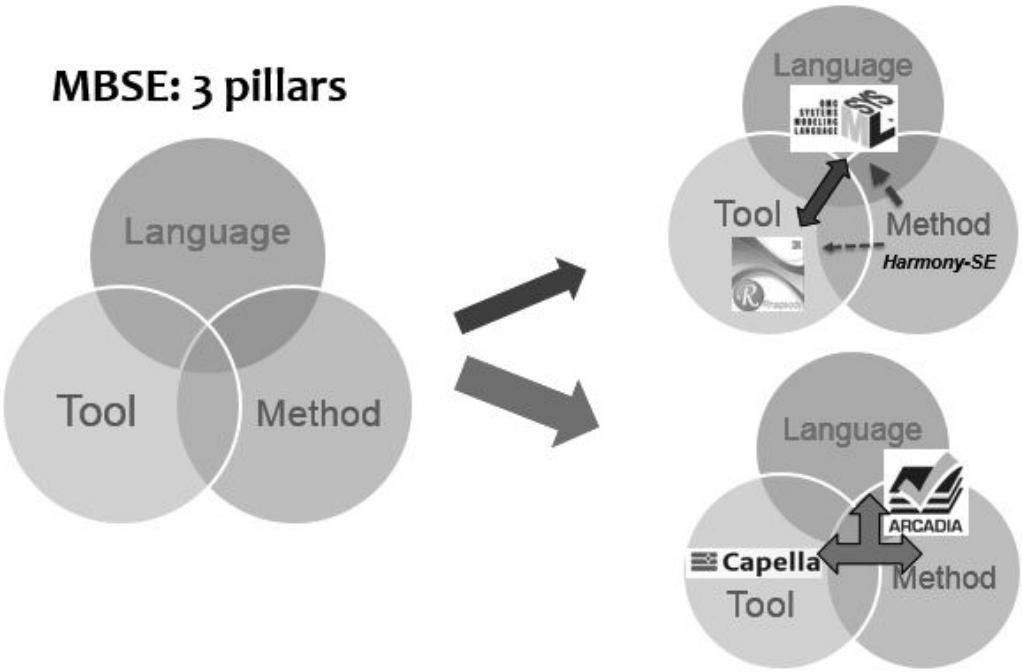

Figure 3 - MBSE 3 pillars implementation: a comparison

Capella to SysML transformation specification and strategies

The transformation strategy focuses on the transformation of Capella elements to SysML 1.4 elements with no Capella stereotype. This choice was motivated by the fact that a specific Capella profile would constrain the existing SysML modeling tools, and hinder interoperability.

Capella and SysML share many similar modeling concepts as Arcadia is highly inspired by SysML. One main difference is that Capella explicitly separates between different modeling abstraction levels: Operational Analysis - OA, System Analysis - SA, Logical Architecture - LA, Physical Architecture - PA, EPBS. These levels refine one another and express the same System on different levels of interest for different specialties, but keep the same structural and behavioral diagram types.

Logical Architecture to SysML 1.4 Transformation

SysML does not define modeling levels/layers. It is the engineer’s task to add a semantic if needed (with stereotypes). For instance, SysML packages are effective modeling elements to separate modeling concerns as the Capella layers propose natively. SysML provides Connectors and Block Ports, for Components Exchanges mapping. Ports can be typed using Blocks, Value Types, and Flow Parameters to translate Capella Exchange Items.

Components and their exchanges transformation

Capella Components and Actors are similar to SysML Blocks (Cf. Figure 4). Components imbrication is materialized in SysML using composition relationships. Communication and exchanges between Capella components and functions are fundamental to components cooperation for system execution. Exchange items, which flow through exchanges (as well as through components or ports), describe the system data type and structure.

Capella element/concept	Capella metamodel	SysML metamodel mapping	Note
- Logical Architecture -Logical Actors Package	- LogicalArchitecture - LogicalArchitecturePkg	UML Package	SysML does not model abstraction layers. Packages are used to group modeled elements per level. A stereotype can be used.
Each Capella level is transformed to a separated package. *A stereotype could be used for each different level.*			
- Logical Context - Logical Actor - Logical Component	- LogicalContext - LogicalActor - LogicalComponent	UML Class + SysML Block Stereotype	A stereotype can be used. Actor blocks are contained in the package Logical Actors, while component blocks are contained in the Logical Architecture. Contained Class/elements in a Packag are UML packaged elements.
Containment of Component	ownedLogicalElement	UML Association with aggregation = Composite, and member End = Owner class, owned class + UML property at the owned Class	The owner Class has a property typed with the contained Class. The item association refers to the association relationship between the owned and the owner classes.
Component Port	ComponentPort	UML Port + Block as port Type	Port is an owned attribute of the container Block. Each Port is typed with a generic Block. The exchanged data are Flow properties in the generic Block type.
Component Exchange	ComponentExchange, source = source port, target = targetport	UML Connector, end ports	Connector ends are the interconnected source and target ports, and the associated block parts.

Table 1 – Logical Architecture component and exchanges transformation to SysML 1.4

Figure 4 shows the transformation result of Capella Logical components and component exchanges (left Clock Radio (LA)) to SysML Blocks and Connectors (Right <<Block>> Clock Radio (LA)).

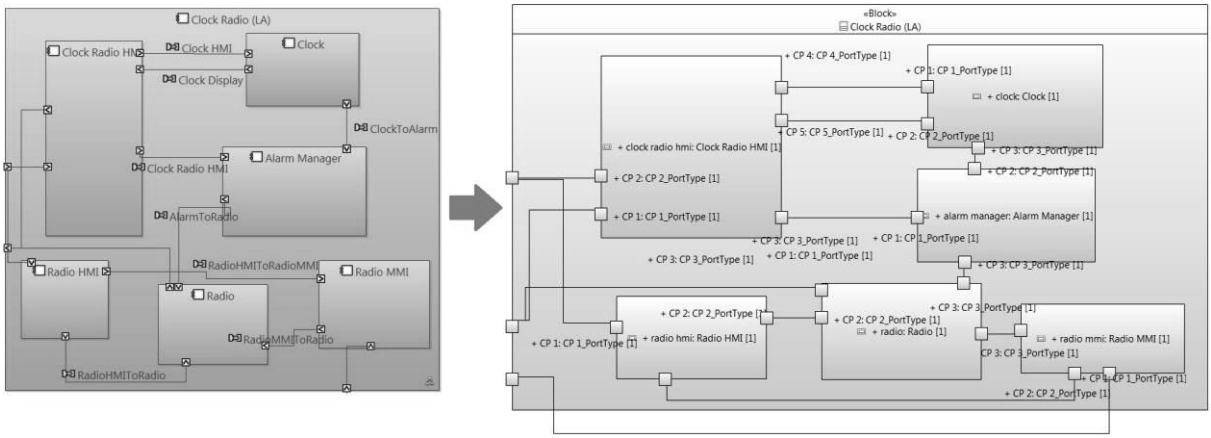

Figure 4 - Capella logical components and component exchanges to SysML blocks and connectors

Functions transformation

The concept of “Function” is not directly supported in SysML. The closest concept is the mapping to Activities and Actions in the Activity diagram. In this case, Pins, Parameter Nodes and Object flows are used for communication (Cf. Figure 5). Another option would be to use Blocks with a “function” stereotype [11]. We decided for this experiment to use only UML/SysML existing elements with no stereotypes (so not to create a UML/SysML profile for Arcadia/Capella).

Capella element/concept	Capella metamodel	SysML metamodel mapping	Note
Logical function	LogicalFunction	Activity + Call Behavior Action that refers the activity	One Capella function is transformed to an activity and a call behavior action that is contained in it.
Actions are atomic elements. It is the reason why a function is transformed to both Activity, which can express the imbrication and uses Activity parameter node for flow.			
Function Input / output ports	FunctionInputPort / FunctionOutputPort	Activity parameter node + Input/Output pin (argument/result)	An Activity parameter node is contained in activity. A pin is contained in action. They are connected to each other using an ObjectFlow
Functional exchange	functionalExchange target = FunctionInputPort ,source = FunctionOutputPort	Object Flow , source = Activity parameter node source, target = Activity parameter node	The object flow interconnects activities. Objects flows are used also to interconnect activity parameter node and pin.

Table 2 – Logical function and exchanges transformation to SysML 1.4

Figure 5 shows the Capella Logical functions (top green blocks) and the functional exchanges transformation to SysML activities and actions (bottom blue blocks), applying Table 2.

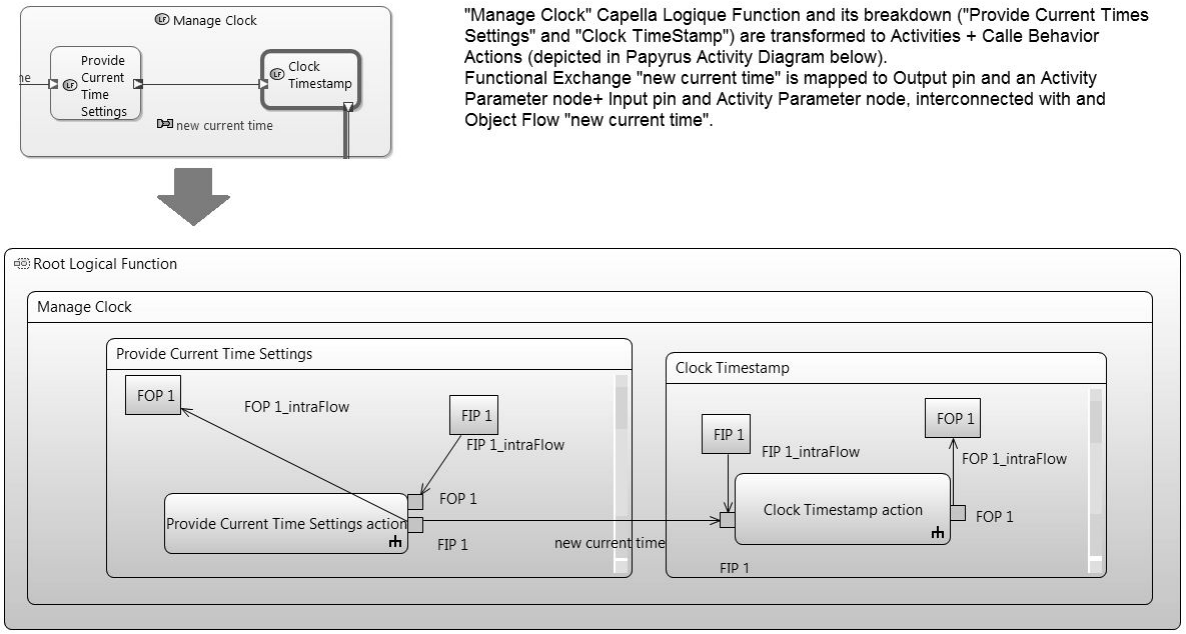

Figure 5 - Capella Logical Functions to SysML Activities and Call Behavior actions

Exchanges and Exchange Items transformation and allocation

Capella element/concept	Capella metamodel	SysML metamodel mapping	Note
Data Package	Datapkg	UML Package	Data package is contained in logical architecture package. Note that each Capella level has its own Data package.
Exchange Item	exchangeItem	UML Block or Value Type	Block/Value type represents the exchanged data. The block/value type types flow property, contained in a port or pin block type, in the case of exchange item allocation (cf. figure 5)
We only transform, at first, exchange items of kind = unset or kind = flow			
Class	class	UML Block	The resulted block will type the property (the result of exchange item element transformation).
Exchange Item Element	exchangeItemElement	UML Property	The created property is added to the “exchange item” Block/Value Type (see exchange item row)
Data Type (Numeric)	Data type	Block Or DataType	See transformation result figure 6.
Property / Literal Numeric Value	property	Property + type	The created property is added to the Block resulted from the transformation of “Class” or of” Data Type”

Predefined Types package	Package	Package	This Package is present in the Data package of SA layer
Predefined Type	DataType	Block Or DataType	See transformation result figure 6.
Exchange Item Allocation In Functional Exchange	Functional Exchange -> exchangeitems = …,…	InPin + outpin type = the allocated exchange item block/value type	Pins are typed by the related exchange items block/ value type from the transformation.
Exchange item Allocation on Component exchange	Component exchange -> exchangeitems = …, …	Port type = Block -> flow property = the allocated exchange item block/value type	Ports are typed by a Block. The flow property contained in the block is to be typed by the corresponding allocated exchange item block/ value type (resulted from the transformation)

Table 3 - Exchange items, Type and exchange item allocation transformation to SysML 1.4

Figure 6 shows SysML pin FIP1 and its type “alarm time”. Alarm time is described as a block with an attribute timestamp, which consists of attributes “Hour” and “Minute”.

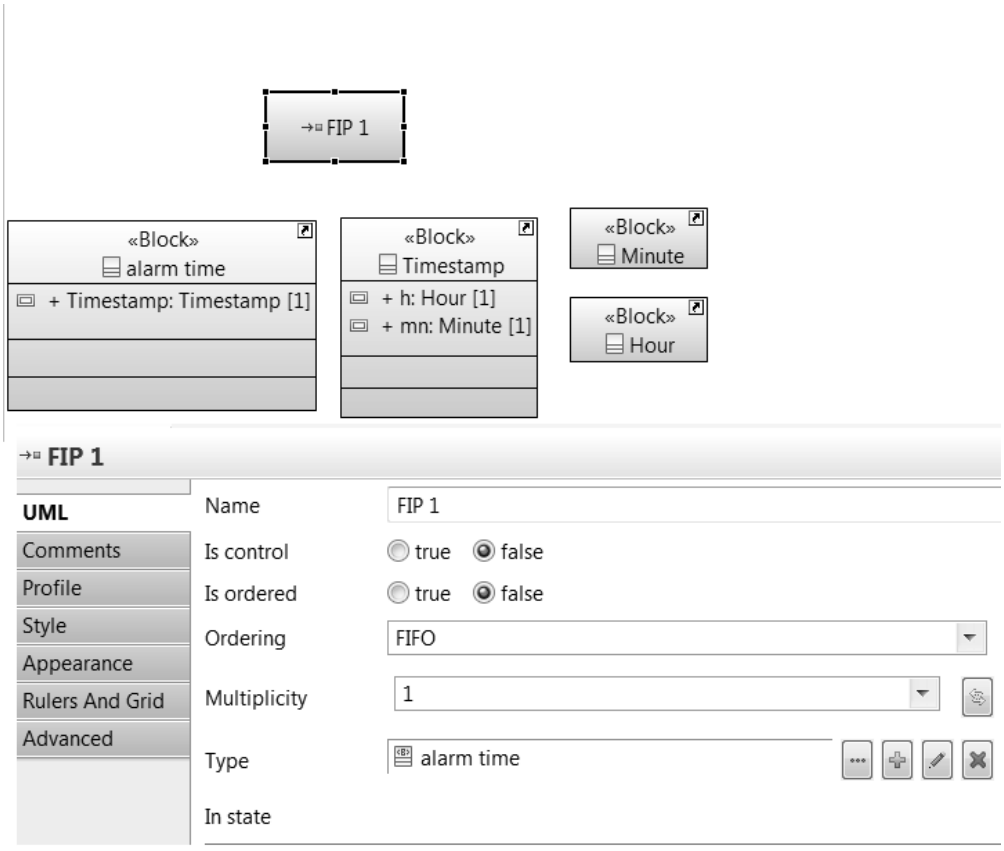

Figure 6 - Pin FIP1 typed with alarm clock

Figure 7 shows the application of the mapping to SysML 1.4 of table 3.

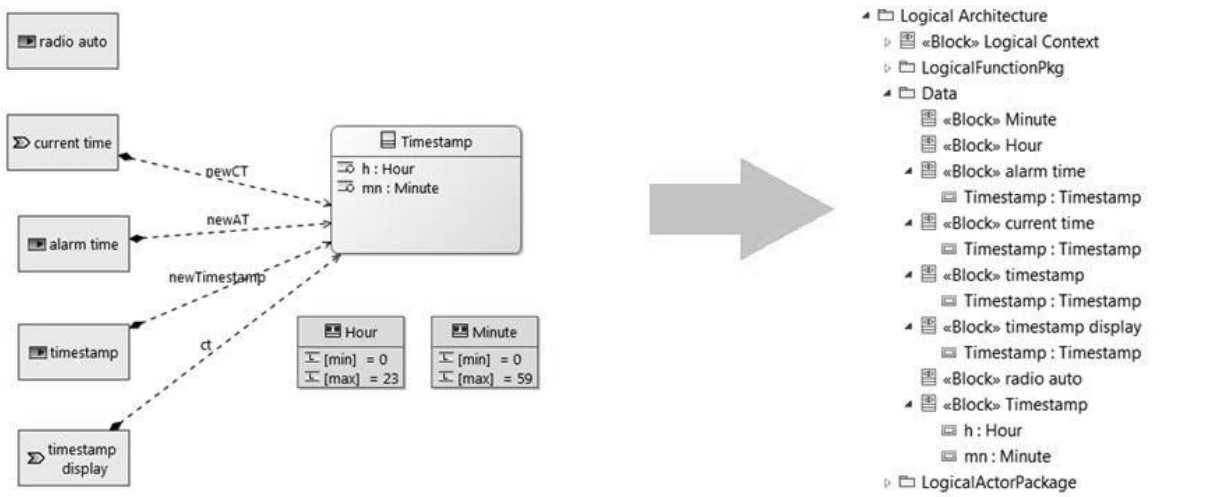

Figure 7 - Predefined Types transformation in SysML 1.4

Allocation and realization transformation

In Arcadia, the concept of "allocation" is very important as Functions should be allocated to Components (System/ Logical/ Physical) at each engineering level. Figure 8 depicts the allocation table in Papyrus, resulted from the transformation of Capella allocations to SysML allocation relationships.

		B	C
		client : NamedEleme...	/owner : Element [0....
1	<Abstraction> Radio	Broadcast Radio action	LogicalFunctionPkg
2	<Abstraction> User	Set Alarm Time action	LogicalFunctionPkg
3	<Abstraction> User	Set Current Time action	LogicalFunctionPkg
4	<Abstraction> User	Set Radio action	LogicalFunctionPkg
5	<Abstraction> User	Listen to Radio action	LogicalFunctionPkg
6	<Abstraction> User	Get Time action	LogicalFunctionPkg
7	<Abstraction> Radio Transmitter	Emit Radio Waves action	LogicalFunctionPkg
8	<Abstraction> CP 3	FOP 1	LogicalFunctionPkg
9	<Abstraction> CP 3	FOP 1	LogicalFunctionPkg
10	<Abstraction> FIP 1	FIP 1	LogicalFunctionPkg
11	<Abstraction> Clock Radio HMI	Provide Current Time Settings a...	LogicalFunctionPkg
12	<Abstraction> Clock Radio HMI	Provide Alarm Settings action	LogicalFunctionPkg
13	<Abstraction> CP 1	FOP 1	LogicalFunctionPkg
14	<Abstraction> CP 5	FIP 1	LogicalFunctionPkg
15	<Abstraction> Radio MMI	Decode Radio Waves action	LogicalFunctionPkg
16	<Abstraction> CP 1	FOP 1	LogicalFunctionPkg
17	<Abstraction> CP 2	FIP 3	LogicalFunctionPkg
18	<Abstraction> CP 2	FIP 4	LogicalFunctionPkg
19	<Abstraction> CP 3	FIP 2	LogicalFunctionPkg
20	<Abstraction> CP 4	FIP 1	LogicalFunctionPkg
21	<Abstraction> Alarm Manager	Trigger Alarm action	LogicalFunctionPkg

Figure 8 – Transformation of Capella Allocation between elements to the SysML 1.4 allocation relationship

Table 4 describes allocations in Papyrus for SysML, allocation tables are used.

Capella element/concept	Capella metamodel	SysML metamodel mapping	Note
Allocation	Allocation, targetElement = <allocated_element>, sourceElement = <element_to_allocate_on >	uml:Abstraction, client = < allocated_element> , supplier = < element_to_allocate_on > + allocation stereotype	Allocation relationship is the transformation for function to component allocation, function port to component port etc, …

Finally, Realization relationship in SysML are used as a mapping for "Realize" Capella relationship, and express the refinement among modeling level, e.g. Logical Architecture <Realize> System Analysis.

Capella element/concept	Capella metamodel	SysML mapping	MM SysML
Realization	<….Realization>, targetElement = <realized_element>, sourceElement = <realizer_element>	Realization relationship	-uml:Realization, client = <realizer_element(LA)>, supplier= <realized_Element(SA)>

CapellaTo SysML Bridge: the transformation tool

Capella2SysML is an Eclipse based tool developed to support Capella to SysML mapping. Capella2SysML is based on the Transposer [6] technology and developed in Capella Studio 1.1.x [7] It also embeds the CoEvolution technology [8] that allows incremental transformation. The source model is Capella 1.1 and the target model is SysML 1.4 for Papyrus 2.0. Capella2SysML Bridge is provided as an update site in Capella. It is active on a Capella Project. Once Capella2SysML is executed, a SysML Papyrus compliance model is created, in addition to a trace file for incremental transformation. As with most XMI bridges between SysML tools, the diagrams themselves are not kept by the transformation.

Conclusion

To take advantage of the power of the Capella tool, as well as the conformance to the standardized SysML language, this paper depicts a first attempt of Capella to SysML mapping and a prototype of transformation tool. As part of the Clarity project [3], we have tried to initiate this mapping of concepts, considering only a subset of important Arcadia/Capella concepts, and to build a prototype transformation tool towards Papyrus SysML as proof-of-concept.

The paper focuses only on Capella Components, Functions and the related Exchange modeling concepts in the Logical architecture level. We developed a tool that supports this transformation as a first bridge between Capella and SysML. It enables the transformation of a given Capella source model to a target SysML model for model interoperability. The mapping strategies will be enhanced with additional transformations, such as Sequence diagram elements and the Physical architecture level. Stereotypes could also be defined to create a real Capella “Profile”, as an alternative solution.

References

[1] https://polarsys.org/capella/

[2] http://www.omg.org/spec/SysML/1.4/

[3] http://www.clarity-se.org/project/

[4] https://projects.eclipse.org/projects/modeling.mdt.papyrus/releases/2.0.0

[5] VOIRIN J.L.., BONNET S., EXERTIER D., NORMAND V. Simplifying (and enriching) SysML to perform functional analysis and model instances », INCOSE IS, Edinburgh, Ecosse, July 2016.

[6] https://wiki.polarsys.org/Kitalpha/CTK/Transposer

[7] https://wiki.polarsys.org/Capella/Studio

[8] https://wiki.eclipse.org/EMF_DiffMerge/Co-Evolution

[9] VOIRIN J.L., *Model-based System and Architecture Engineering with the Arcadia Method*, ISTE Editions, London, 2017.

[10] ROQUES P., *Systems Architecture Modeling with the Arcadia Method*, *Practical Guide to Capella*, ISTE Editions, London, 2017.

[11] LE BASTARD J., *SysML and System Engineering functional analysis*, Alstom, 2012.