Project: COMPASS

Grant Agreement: 287829

Comprehensive Modelling for Advanced Systems of Systems

SysML Blocks in CML

COMPASS White Paper WP02

April 2013

Public Document

http://www.compass-research.eu

Authors:
Alvaro Miyazawa, Lucas Lima, Ana Cavalcanti, University of York, UK

Abstract:

SysML is an extension of UML 2.0 to support modelling for systems engineering; it inherits some of UML's diagrams, modifies others and adds two new diagrams. SysML supports modelling various aspects of a system, and in recent years, SysML has increasingly been supported by a number of tool vendors such as IBM, Atego and Sparx Systems.

Our aim is to support the application of formal analysis tools and techniques at the level of the graphical notations used in current industrial practice. In particular, in this white paper, we present our results on formalising the notion of SysML blocks including their related elements such as associations, compositions, generalisations, ports, interfaces and connectors. This is achieved by a denotational semantics of SysML blocks in the COMPASS modelling language (CML), a formal specification language that supports a variety of analysis techniques.

The main characteristics of the approach proposed are the compositionality of the generated models, the use of parallelism to compose different aspects of the system and the support for refinement. These aspects make it possible the application of compositional analysis techniques, and refinement strategies to obtain equivalent models better suited to alternative analysis techniques (e.g., model checking). The most interesting aspects of our models are the treatment of operation calls, the use of interleaving to model inheritance, and the use of parallelism to model block composition. Finally, whilst the models of ports are simple, the use of interface classes in the specification of a port's communication protocol proved important to preserve the compositionality of our models.

SysML blocks in CML

Alvaro Miyazawa Lucas Lima Ana Cavalcanti

April 2013

1 Introduction

SysML is an extension of UML 2.0 to support modelling for systems engineering; it inherits some of UML's diagrams, modifies others and adds two new diagrams. SysML supports modelling various aspects of a system, and in recent years, SysML has increasingly been supported by a number of tool vendors such as IBM[1], Atego[2] and Sparx Systems[3].

Our aim is to support the application of formal analysis tools and techniques at the level of the graphical notations used in current industrial practice. In particular, in this paper, we present our results on formalising the notion of SysML blocks including their related elements such as associations, compositions, generalisations, ports, interfaces and connectors. This is achieved by a denotational semantics of SysML blocks in the COMPASS modelling language (CML) [13], a formal specification language that supports a variety of analysis techniques [4].

Whilst SysML is an informal graphical notation, CML builds on well known and widely used formal specification languages: VDM [5] and CSP [6]. Its approach to modelling reactive behaviour and its semantic model are those adopted in the *Circus* [3] family of refinement languages. The semantics of both CML and *Circus* use the Unifying Theories of Programming to cater for object-orientation [11], time [12], and synchronicity [1], for instance. *Circus* has been successfully used in practical applications [2, 8].

We present a denotational semantics for blocks in SysML models using CML, and discuss how this formalisation can be used as an integration context for formal models of other SysML elements such as state machine, activity and sequence diagrams. The semantic function is formalised via translation rules, which can be used to generate CML models of blocks automatically. Currently, our translation rules are being used as a basis for an implementation of a CML semantics of SysML based on the Atego's Artisan Studio. As far as we know, there are no formal accounts in the literature of the behavioural semantics of SysML blocks that support integration with other diagrams.

The CML semantics enables a variety of refinement-based analysis of SysML models. CML tools [4] include an Eclipse-based development environment (parser and type-checker) with links to Artisan Studio to support design using SysML and RT-Tester[4] for test automation, and plug-ins that support the generation of proof obligations, simulation, theorem proving based on Isabelle/HOL [10], model checking, and the application of a refinement calculus. The use of CML to reason about systems of systems is discussed in [13], and compositional refinement-based reasoning techniques are described and formalised in [9]. The CML semantics enables a variety of refinement-based analysis of SysML models. CML tools [4] include an Eclipse-based development environment (parser and type-checker) with links to Artisan Studio to support design using SysML and RT-Tester[5] for test automation, and plug-ins that support the generation of proof obligations, simulation, theorem proving based on Isabelle/HOL [10], model checking, and the application of a refinement calculus. The use of CML to reason about systems of systems is discussed in [13], and compositional refinement-based reasoning techniques are described and formalised in [9].

In SysML, the behaviour of blocks may be specified by state machine diagrams, and operations may be specified by activity diagrams, which describe a form of flowchart. Sequence diagrams may be used to model

[1] www-142.ibm.com/software/products/us/en/ratirhaparchforsystengi
[2] www.atego.com/products/artisan-studio
[3] www.sparxsystems.com/products/mdg/tech/sysml
[4] www.verified.de/en/products/rt-tester
[5] www.verified.de/en/products/rt-tester

particular scenarios of interaction between elements of the model. Our approach considers process models for state machine, activity and sequence diagrams. An approach to the construction of theses models is described in [8].

2 Formalising SysML blocks

We assume that the SysML model is sufficiently complete to allow the derivation of a well formed CML model. This assumption is decomposed in a number of guidelines that can be divided into three groups:

Entity definition. These guidelines require that elements such as operations, blocks and associations are defined somewhere in the model. For instance, it requires that operations are defined either via the action language, a state machine or an activity diagram.

Instance definition. These guidelines require that enough information about the instances of composite blocks is available. For instance, it requires that the parts of a composite block and their interconnections are specified.

Simplification assumptions. These guidelines provide alternatives to the use of certain elements, where they have an equivalent counterpart, or define how they can be used. An example of such guidelines is the requirement that asynchronous operations are modelled as signals. This requirement stems from the fact that the meaning in SysML of an asynchronous operation with return value is unclear and that asynchronous operations without return values and signals can be considered equivalent. In this case, the guidelines propose the use of signals as an alternative to asynchronous operations.

The full list of guidelines can be found in [7]. A SysML model that respects our guidelines has the following structure. It is formed by blocks that may have properties and offer services defined by operations and signals. A simple block contains properties, operations, signals and ports, whilst a composite block contains parts that are typed by blocks and ports. A block is defined in the SysML metamodel and its content is obtained from the diagrams that refer to it.

Our CML definition of a SysML block is specified by a semantic function that calculates the model of a block in terms of the models of its parts. This semantic function is formalised by translation rules, whose compositional nature makes traceability viable as they allow us to identify which parts of the CML model that they define correspond to particular elements of a given SysML model. Moreover, as opposed to *ad hoc* rules for model transformation, the semantic function formalised by our translation rules can be encoded in a theorem prover to support both the validation of the semantics and the analysis of the model using techniques based on theorem proving.

2.1 Structure of models

We model a block as a CML process, where the state characterised by its properties is encapsulated (not accessible externally). For this reason, access to properties as well as interaction via operations and signals are modelled as communications through channels. The CML process that models a block can receive requests to read and write to the block's properties, as well as signals and operation calls. Each of these requests are received through CML channels whose names are the names of the blocks properties prefixed by `set_` and `get_`, or the name of the block appended with `_op` and `_sig`. Finally, a channel `_addevent` is used to delegate the treatment of event to the environment, which, for instance, can be a process that models an activity or a state machine diagram.

In general, a SysML model may contain a number of blocks that are not related to each other. In this case, our CML model consists of a number processes, one for each block, that are also not related to each other. An analysis needs therefore to focus on a particular process.

The model of a simple block is formed by the parallel composition of two or more basic processes: one specifying the behaviour of the block's parent, another called `simple_` process, modelling the behaviour of the block itself, and the remaining modelling the block's ports. This allows the reuse of the model of the parent block to reflect the structure of the SysML model in CML. A CML process that models a composite block is defined in terms of the processes that model its parts and ports.

SysML element	CML element
Simple block	Process
Composite block	Process
Port	Process
Connector	Channel
Interface	Class
Operation call	Record type
Signal	Record Type
Event	Communication

Table 1: SysML-CML correspondence

A port is modelled by a process that uses four channels: `ext_op`, `ext_sig`, `int_op` and `int_sig`. The first two allow the port to interact with a component external to the block by sending and receiving signals and operation calls as well as responses to operation calls. The last two are used to communicate with the model of the block. The behaviour of a port is to restrict which values can be received at each channel, and relay the accepted values to the equivalent channel on the other side of the port.

Table 1 shows the correspondence between elements of a SysML model and elements of CML. In general, a SysML element that exhibits some form of behaviour, namely, blocks and ports, are modelled by CML processes; connectors, which specify communication links, are modelled by channels; static elements (i.e, without intrinsic behaviour), namely operational calls and signals, are modelled by record types, and interfaces, which are collections of static elements are modelled by classes. Operation calls are considered static because they specify the message that is sent to blocks, not the behaviour of the operation itself, which is usually specified by a state machine or activity diagram. Events, like the communication of signals, are modelled by CML communications.

2.2 Integration with other SysML model elements

In our approach, the processes that model state machine and activity diagrams accept events through a channel `addevent`. These events are added to an event pool and are processed according to the semantics of the element (state machine or activity diagram). The processing of events may lead to the generation of new events as well as to changes to the state of the block that is associated with state machine or activity diagram. The models of sequence diagrams use the channels of the blocks included in the diagram.

To obtain the integrated model of a block whose behaviour is specified by a state machine diagram or whose operations are specified by activity diagrams, the processes that model the state machine and activity diagrams are composed in parallel with the process that models the block. Each of these processes synchronise on the events associated with a channel `_addevent` whose parameters include the operations and signals treated by the activity or state machine diagrams. For instance, if a state machine diagram treats the operations check, and an activity diagram responds to a signal off, the synchronisation sets should be as follows. The first synchronisation set includes all events of the channel `addevent` where the first three parameters (representing the instance, source and target of a signal or operation call) are unrestricted and the fourth is limited to values whose type is the input record type of the operation check. Similarly, the second synchronisation set includes all events of the channel `addevent` where the first three parameters are unrestricted and the fours is limited to values of the record type of the signal off.

Figure 1 illustrates an approach to the analysis of SysML models by establishing consistency between a sequence diagram, which describes possible valid traces, and the SysML model described by the blocks, and activity and state machine diagrams. The integrated CML model can be validated by reasoning tools, like a model checker, in order to check whether the system model is compatible with the flows of execution specified by the sequence diagram.

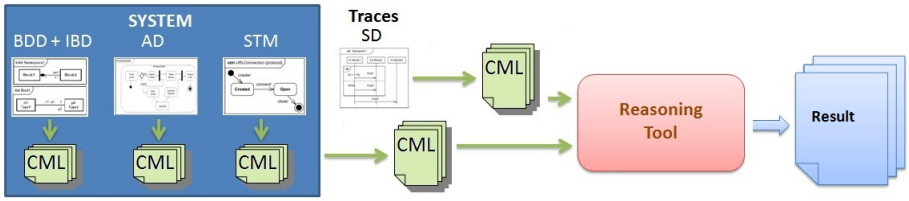

Figure 1: Integrated Model Analysis Approach.

3 Conclusions

We have presented a behavioural model of SysML blocks that includes simple and composite blocks, generalisation, association and composition relations, standard ports and connectors, interfaces, operations, properties and signals. To the best of our knowledge, this is the first formalisation of the behavioural semantics of a comprehensive subset of the block notation.

The main characteristics of the approach proposed in this paper are the compositionality of the generated models, the use of parallelism to compose different aspects of the system and the support for refinement. These aspects make it possible the application of compositional analysis techniques [9], and refinement strategies to obtain equivalent models better suited to alternative analysis techniques (e.g., model checking).

The most interesting aspects of our models are the treatment of operation calls, the use of interleaving to model inheritance, and the use of parallelism to model block composition. Finally, whilst the models of ports are simple, the use of interface classes in the specification of a port's communication protocol proved important to preserve the compositionality of our models.

The functions that characterise our semantics are specified by translation rules, which take elements of the SysML abstract syntax and produce the corresponding CML elements. The complete set of translation rules for SysML blocks can be found in [8]. These rules are currently being implemented in Artisan Studio to support the automatic generation of CML from SysML models. This work is being carried out by our industrial partners at Atego, and the revision of the rules by a SysML expert and the process of mechanising the rules have helped us partially validate our semantics.

References

[1] A. Butterfield and P. Gancarski. The denotational semantics of slotted-circus. In Ana Cavalcanti and Mads Damm, editors, *FM 2009*, LNCS. Springer, 2009.

[2] A. Cavalcanti, P. Clayton, and C. O'Halloran. From control law diagrams to Ada via *Circus*. *Form. Asp. Comp.*, pages 1–48, 2011. 10.1007/s00165-010-0170-3.

[3] A. L. C. Cavalcanti, A. C. A. Sampaio, and J. C. P. Woodcock. A Refinement Strategy for *Circus*. *Form. Asp. Comp.*, 15(2 - 3):146 — 181, 2003.

[4] J. W. Coleman, A. K. Malmos, P. G. Larsen, J. Peleska, R. Hains, Z. Andrews, R. Payne, S. Foster, A. Miyazawa, C. Bertolini, and A. Didier. COMPASS Tool Vision for a System of Systems Collaborative Development Environment. In *Proceedings of the 7th SoSE*, volume 6 of *IEEE Systems Journal*, 2012.

[5] J. Fitzgerald and P. G. Larsen. *Modelling Systems – Practical Tools and Techniques in Software Development*. Cambridge University Press, Second edition, 2009.

[6] C. A. R. Hoare. *Communicating sequential processes*. Prentice-Hall, Inc., 1985.

[7] A. Miyazawa, L. Albertins, J. Iyoda, M. Cornélio, R. Payne, and A. Cavalcanti. Final report on combining SysML and CML. Technical report, 2013.

[8] A. Miyazawa and A. Cavalcanti. Refinement-oriented models of stateflow charts. *Science of Computer Programming*, 77(10):1151–1177, 2012.

[9] M. Oliveira, A. Sampaio, P. Antonino, R. Ramos, A. Cavalcanti, and J. Woodcock. Compositional analysis and design of CML models. Technical report, 2013.

[10] Lawrence C. Paulson. *Isabelle: A Generic Theorem Prover*. LNCS. Springer, 1994.

[11] T. Santos, A. L. C. Cavalcanti, and A. C. A. Sampaio. Object-Orientation in the UTP. In S. Dunne and B. Stoddart, editors, *UTP 2006*, volume **4010** of *LNCS*, pages 20–38. Springer-Verlag, 2006.

[12] A. Sherif, A. Cavalcanti, H. Jifeng, and A. Sampaio. A process algebraic framework for specification and validation of real-time systems. *Form. Asp. Comp.*, 22:153–191, 2010.

[13] J. Woodcock, A. Cavalcanti, J. Fitzgerald, P. Larsen, A. Miyazawa, and S. Perry. Features of CML: a Formal Modelling Language for Systems of Systems. In *Proceedings of the 7th SoSE*, volume 6 of *IEEE Systems Journal*. IEEE, July 2012.