A Model-Driven Based Environment for Automatic Model Coordination

Matias Ezequiel Vara Larsen, Julien Deantoni, Benoit Combemale, Frédéric Mallet

A Model-Driven Based Environment for Automatic Model Coordination

Matias Ezequiel Vara Larsen*, Julien DeAntoni*, Benoit Combemale†, and Frédéric Mallet*
*Université Nice-Sophia Antipolis, I3S, INRIA, France
†INRIA and University of Rennes 1, France

***Abstract*—We present the integration of the Behavioral Coordination Operator Language (B-COoL) into *the GEMOC Studio*. B-COoL enables the system designer to automate the coordination of models by specifying Operators between Domain-Specific Modeling Languages. In this demonstration, we present how B-COoL is used to coordinate the heterogeneous model of a video surveillance system. To this propose, we define operators between timed finite state machines and activity diagrams. These operators are used to generate an explicit model of coordination that can be executed and verified [1]. This demonstration comes as a support for the paper accepted into the main conference.**

***Index Terms*—Heterogeneous Modeling, Coordination Languages, DSMLs**

I. Introduction

The development of complex software intensive systems involves interactions between different subsystems. For instance, embedded and cyber-physical systems require the interaction of multiple computing resources (general-purpose processors, DSP, GPU), and various digital or analog devices (sensors, actuators) connected through a wide range of heterogeneous communication resources (buses, networks, meshes). The design of complex systems often relies on several Domain Specific Modeling Languages (DSMLs) that may pertain to different theoretical domains with different expected expressiveness and properties. As a result, several models conforming to different DSMLs are developed and the specification of the overall system becomes heterogeneous.

To ease the development of such complex systems, tools must allow system designers to edit, execute, simulate, and animate their models. Furthermore, to verify and validate the system as a whole, they should be able to specify how models interact/coordinate with each other. Current Coordination frameworks [1], [2] propose an environment to develop and coordinate heterogeneous models. However, they are bound to a fixed set of coordination patterns (*e.g.*, hierarchical coordination of models in Ptolemy).

In this demonstration, we present the integration of B-COoL [3] into the GEMOC studio[2]. B-COoL is a dedicated language that allows for capturing coordination patterns for a given set of DSMLs. These patterns are captured at the language level, and then used to derive a coordination specification automatically for models conforming to the targeted DSMLs. While the GEMOC studio supports several facilities *e.g.*, editing, graphical representation, animation and execution of domain-specific tools, B-COoL adds coordination facilities. While the main paper [3] introduces the main research challenges, this demonstration focuses on the technical choices and implementation difficulties.

This paper is organized as follows. Section II motivates our work by comparing with state of art approaches. This section also presents the background needed to understand the approach. Section III presents B-COoL and how it is integrated into the GEMOC studio. In Section IV, we describe the demonstration, which relies on B-COoL to capture three coordination patterns between two languages: TFSM and fUML Activities. The patterns are captured as four operators later used to coordinate the model of a surveillance video system. Section V gives a comparison with related work.

II. Motivation and Background

The coordination between models can be explicitly modeled by using a *coordination language* (*e.g.*, Linda [4], Esper [5]). A system designer can define one or more coordination specifications to specify how models interact. This results in a global behavior that is explicit and amenable for reasoning (for instance for Verification and Validation activities). The major drawback of these approaches is that the coordination is done manually by the system designer. With the increasing number and heterogeneity of the model behavior, this task can quickly become difficult and error prone.

More recent approaches [1], [2], [6], [7] identified that the coordination of models can be a systematic activity the system designer repeats many times and can consequently be defined as a pattern. Such a pattern is based on the know-how of the system designer and sometimes on naming or organizational conventions adopted by the models. Thus, they have captured the specification of such a behavioral coordination pattern into a tool. They specify the coordination between heterogeneous languages instead of specifying it between particular models. Such specification is then applied on a set of models to automatically instantiate a model of coordination. This is the case of Ptolemy [1], a framework that enables the system designer to hierarchically coordinate heterogeneous models. However, these approaches embed the coordination pattern inside a tool by using a general-purpose language (GPL). This has mainly two drawbacks:

- The semantics of the coordination is hidden into a tool and is not made explicit thus potentially leading misun-

[1]https://youtu.be/skXrpKlv6C4
[2]http://www.gemoc.org

derstanding and errors. In addition, since the coordination relies on GPL, the validation and verification of the coordinated system remains limited.
- The system designer cannot change the coordination specification without altering the core of the tool. This work needs a good knowledge of the language itself (*e.g.*, Java in Ptolemy), which is beyond the expected skills of a system designer.

In our approach, we propose B-COoL: a (meta)language to capture behavioral coordination patterns between DSMLs. B-COoL is a dedicated language to system designer that eases the understanding and adaptation of coordination patterns. By instantiating a coordination pattern, models are automatically coordinated. The generated coordination specification conforms to a formal language, thus enabling verification and validation activities. To be able to specify the coordination between languages, a partial representation of the language behavioral semantics is mandatory. In our approach, the semantics of languages is abstracted by using a *language behavioral interface* made of *Domain Specific Events* (DSE) [8]. DSE act as *coordination points* on the behavioral semantics of languages. Coordination patterns are thus captured as constraints at the language level on the DSE. When models conformed to the coordinated languages are known, the coordination between them can be automatically generated.

In the following, we first present the current integration of B-COoL into the GEMOC studio[3], and then we demonstrate how the approach is used to:

- capture explicitly the specification of coordination patterns between the TFSM and fUML Activity languages.
- generate the coordination specification for a video surveillance system.
- execute the coordinated system.

III. Description of the Approach

B-COoL is a dedicated (meta)language that enables system designers to capture coordination patterns between DSMLs. In B-COoL, the coordination is specified between languages by relying on a language behavioral interface made of DSE. Coordination patterns are captured by using *Operators* that specify how the DSE of different language behavioral interfaces are related. Operators rely on a *Correspondence matching* and a *Coordination rule*. The correspondence matching identifies what elements from the behavioral interfaces (*i.e.,* what instance of DSE) must be selected. The coordination rule operates on elements of the semantics (*i.e.,* instances of DSE), and it specifies the, possibly timed, synchronizations and causality relations between the selected instances of DSE. From a B-COoL specification, the coordination specification is automatically generated as constraints between instances of DSE of specific models. Therefore, the generated coordination specification is an instance of a given coordination pattern.

[3]http://www.gemoc.org

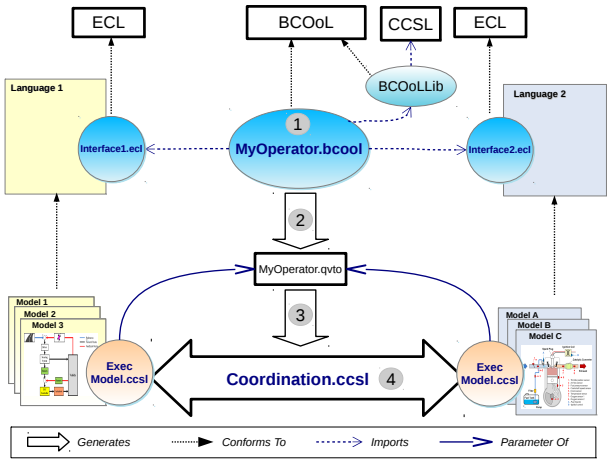

Fig. 1. B-COoL Language Workbench

B-COoL is developed as a set of plugins for Eclipse[4] as part of the GEMOC studio[5]. B-COoL is itself based on EMF and its abstract syntax has been developed using Ecore (*i.e.,* the meta-language associated with EMF). The textual concrete syntax has been developed by using Xtext[6] thus providing advanced editing facilities.

The B-COoL specification is defined between languages by relying on its language behavioral interface made of DSE. B-COoL takes advantage of the ECL (standing as Event Constraint Language [9]) specification of a DSML to extract the language behavioral interface. ECL is an extension of OCL [10] with events that allows augmenting AS metaclass and add DSE. To get the DSE, the ECL specification of each language must be imported (step 1 in Figure 1). Once defined, the B-COoL specification is used to generate an executable coordination model used to simulate the coordinated execution of some input models. From a B-COoL specification, the automatic generation of the coordination is made in two steps (step 2 and step3 in Figure 1). To ease this task, the studio provides two plug-ins.

The first step consists in the automatic generation of a transformation by using a higher order transformation written in acceleo[7] (step 2 in Figure 1). The acceleo transformation translates the B-COoL specification into a QVTo transformation. Then, the second step (step 3 in Figure 1) consists in applying the QVTo transformation between the models.

The QVTo transformation takes as an input any models conforming to the languages used in the operator and generates as an output the coordination model for these models. The resulting coordination model is a CCSL [11] specification, which is a formal language dedicated to define the possibly timed synchronizations and causality relationships between some events. Note that CCSL is also used to specify the

[4]http://www.eclipse.org
[5]http://gemoc.org/studio/
[6]http://www.eclipse.org/Xtext/
[7]http://www.eclipse.org/acceleo/

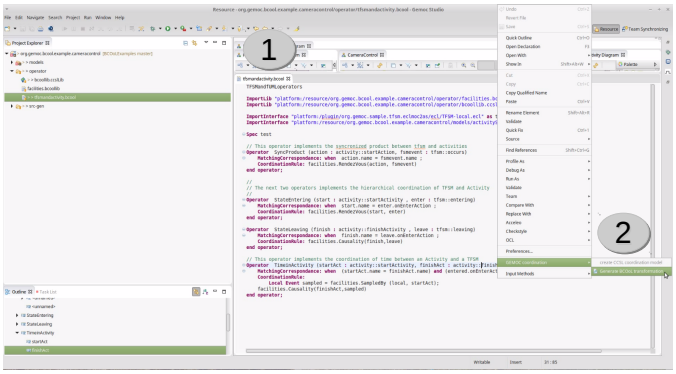

Fig. 2. Definition of Coordination Operators between the TFSM and fUML languages

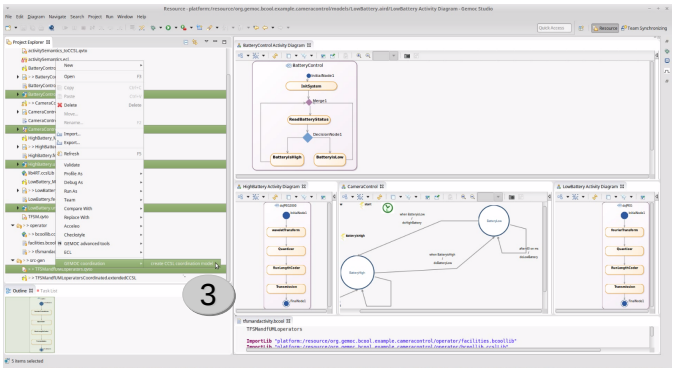

Fig. 3. Generation of the coordination specification for a surveillance camera system

execution model of a model. This is convenient in our case since eases the execution of the coordinated model.

The GEMOC studio can then be used to execute this coordination specification (step 4 in Figure 1). For instance, it is possible to obtain a timing output of the execution of the coordinated system by using TimeSquare [8]. The workbench also offers the possibility to obtain by exploration quantitative results on the scheduling state-space.

IV. Description of the Demonstration

In the demonstration, we use the integrated workbench to automatically coordinate the model of a video surveillance system. To do so, we use B-COoL to define four operators. Each operator captures a given coordination pattern between two different languages: TFSM and fUML activity language [12]. Then, we use these operators to generate the coordination for the video surveillance system. A more precise description of this example is presented in [3].

In a B-COoL specification named *TFSMandfUMLoperators*, we define four coordination operators between the TFSM and fUML language. The first operator, named *SyncProduct*, coordinates the occurrences of TFSM events with the start of fUML *Actions*. In the second and third operators, named *StateEntering* and *StateLeaving*, we specify a hierarchical coordination between the TFSM and fUML language. In our case, we chose the semantics in which entering a specific state of a TFSM model triggers the execution of a given fUML activity. The fourth operator deals with the temporal aspects of the coordination. It specifies how the time in the TFSM elapses during the execution of the activities. This coordination is also hierarchical but only considers the timing aspects. The operator enforces the execution of the "internal" activity to be atomic with respect to the time in the TFSM model.

We use the operators previously defined to coordinate the heterogeneous model of a surveillance camera system. The model of the system has been developed by using the graphical tooling proposed in the GEMOC studio. Figure 3 illustrates the whole model of the video surveillance system by using TFSMs and Activities. Roughly speaking, the video surveillance system is composed of a camera (*CameraControl* in Figure 3) and a battery control (*BatteryControl* in Figure 3). The camera takes pictures by using either the *JPEG2000* (*HighBattery* in Figure 3) or *JPG* (*LowBattery* in Figure 3) algorithm and is powered by a battery. When the battery is low, the battery control makes the camera use the *JPG* algorithm, thus reducing the quality of the picture but also the energy consumption [13]. When the battery is high, the JPEG2000 algorithm is used instead.

To coordinate the models, we have to specify a timing and hierarchical coordination between the states of the TFSM CameraControl and the activities doJPEG and doJPEG2000. In addition, we have to synchronize the activity BatteryControl and the TFSM CameraControl by coordinating the corresponding Actions and TFSM events.

To generate the coordination for these models, we first generate the corresponding QVTo transformation by invoking the plug-in provided by the studio (step 2 in Figure 2)[9]. Then, we apply the qvto transformation on these models. We select the models, and then, we invoke the plug-in (step 3 in Figure 3). The generated coordination specification corresponds to eight CCSL relations that can be executed and analyzed by using TimeSquare. Figure 4 shows the resulting timing execution of the coordinated system.

A video presenting the whole demonstration (definition, compilation, execution) can be found on the companion webpage[10]. In addition, the webpage contains other examples. All examples are hosted in Github[11] at BCOoLExamples[12]

A. Take-Away Lessons

The coordination of the surveillance camera system requires the specification of eight CCSL relations. By manually coordinating the models, this would require specifying each relation manually. The reader can notice that the number of relations increases with the number of model elements involved in the coordination. For instance, for a system with N cameras, the

[8]http://timesquare.inria.fr

[9]In this example, the generated QVTo contains 846 lines

[10]http://timesquare.inria.fr/BCOoL

[11]http://www.github.com

[12]http://matiasvara.github.io/BCOoLExamples/

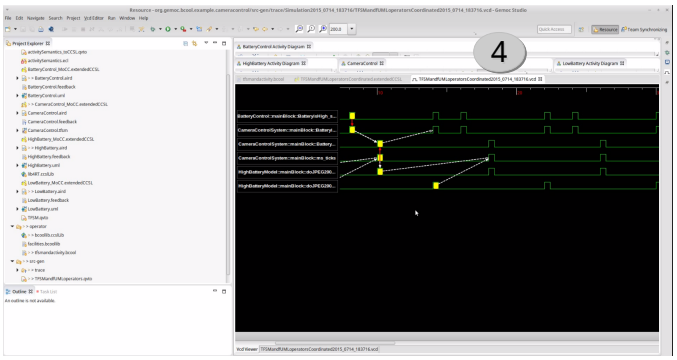

Fig. 4. Execution of the coordinated system by using TimeSquare

system designer would need to specify 8*N relations. Our proposition is to leverage this task for the system designer at the language level and then to generate all the required relations accordingly.

We want to highlight that variations of the semantics of the resulting coordination can be done by only modifying the coordination rules of the operators. In frameworks like Ptolemy, such a variation is only supported by changing the current implementation of a *director* written in Java. The same problem appears in ad-hoc translational approaches [6], where the transformation needs to be changed. Since this state of the art approach is using general-purpose transformation frameworks, this work needs a good knowledge of coordinated languages as well as a good knowledge of the transformation language itself. This is beyond the expected skills of a system designer. In our approach, we are using a language dedicated to system designer thus easing the understanding and adaptation of the B-COoL specification.

V. Related Work

We consider as related work the coordination frameworks Ptolemy [1] and ModHel'X [2] that provide a dedicated environment to develop and coordinate heterogeneous models. These frameworks rely on a common syntax based on actors and a semantics given by Models of Computation (MoC). Models are represented as actors that can be atomic or composite, *i.e.,* made of internal actors. Each composite actor follows an explicit model of computation implemented as a *Domain*. A domain defines the communication semantics and the execution order among internal actors. Based on a fixed syntax, these approaches provide a dedicated environment to develop heterogeneous models. In addition, they enable the system designer to hierarchical coordinate models. The environments include a graphical editor, an execution engine, plotters and so on. These environments, however, are ad-hoc solutions to manage both the development and the coordination of heterogeneous models. Differently, in our approach, the workbench is the integration of several plug-ins that deal with different aspects of the heterogeneous development of models, *e.g.*, the GEMOC studio for the design and implementation of DSMLs, the Sirius animator for graphical representation, TimeSquare for the analysis of model execution. Our approach takes advantages of this collaborative environment, and it provides the means for model coordination.

Acknowledgment

This work is partially supported by the ANR INS Project GEMOC (ANR-12-INSE-0011).

References

[1] B. Evans, A. Kamas, and E. A. Lee, "Design and simulation of heterogeneous systems using ptolemy," in *Proceedings of ARPA RASSP Conference*, 1994.
[2] F. Boulanger and C. Hardebolle, "Simulation of Multi-Formalism Models with ModHel'X," in *ICST*, 2008.
[3] M. E. Vara Larsen, J. Deantoni, B. Combemale, and F. Mallet, "A Behavioral Coordination Operator Language (BCOoL)," Aug. 2015. [Online]. Available: https://hal.inria.fr/hal-01182773
[4] D. Gelernter and N. Carriero, "Coordination languages and their significance," *Commun. ACM*, 1992.
[5] Esper, "Espertech," 2009.
[6] M. Di Natale, F. Chirico, A. Sindico, and A. Sangiovanni-Vincentelli, "An MDA approach for the generation of communication adapters integrating SW and FW components from Simulink," in *ACM/IEEE Models*, 2014.
[7] P. Bjureus and A. Jantsch, "Modeling of mixed control and dataflow systems in MASCOT," *VLSI Systems, IEEE Transactions on*, 2001.
[8] B. Combemale, J. Deantoni, M. Vara Larsen, F. Mallet, O. Barais, B. Baudry, and R. France, "Reifying Concurrency for Executable Metamodeling," in *SLE*, 2013.
[9] J. Deantoni and F. Mallet, "ECL: the Event Constraint Language, an Extension of OCL with Events," INRIA, Research report, 2012.
[10] *UML Object Constraint Language (OCL) 2.0*, OMG, 2003.
[11] C. André, "Syntax and Semantics of the Clock Constraint Specification Language (CCSL)," Tech. Rep., 2009.
[12] *Semantics of a Foundational Subset for Executable UML Models (fUML), Version 1.0*, OMG, 2011.
[13] M. Rhepp, H. Stgner, and A. Uhl, "Comparison of jpeg and jpeg 2000 in low-power confidential image transmission," in *SPC*, 2004.