Domain Specific Modeling Language for Cyber Physical Systems

Muhammad Waqar Aziz
Science and Technology Unit,
Umm Al-Qura University,
Makkah, 21955, Saudi Arabia
e-mail: mwabdulaziz@uqu.edu.sa

Muhammad Rashid
Department of Computer Engineering,
College of Computer and Information Systems,
Makkah, 21955, Saudi Arabia
e-mail: mfelahi@uqu.edu.sa

***Abstract*—The benefits of Domain Specific Modeling Languages (DSML), for modeling and design of cyber physical systems, have been acknowledged in previous years. In contrast to general purpose modeling languages, such as Unified Modeling Language, DSML facilitates the modeling of domain specific concepts. The objective of this work is to develop a simple graphical DSML for cyber physical systems, which allow the unified modeling of the structural and behavioral aspects of a system in a single model, and provide model transformation and design verification support in future. The proposed DSML was defined in terms of its abstract and concrete syntax. The applicability of the proposed DSML was demonstrated by its application in two case studies: Traffic Signal and Arbiter case studies. The results showed that the proposed DSML produce simple and unified models with possible model transformation and verification support.**

Keywords-Domain Specific Modeling Language; meta-modeling; cyber physical systems; Unified Modeling Language; abstract syntax; concrete syntax.

I. INTRODUCTION

In cyber physical systems, where system failure can result in severe loss, system modeling is very challenging due to the complexity and diversity involved [1, 2]. To reduce the design complexity, Model-Driven Development (MDD) is used that allows the development of a system using models [3]. Models in MDD can be formulated either using general purpose modeling languages, such as Unified Modeling Language (UML) [4] or through a domain-specific modeling language (DSML). Despite having rich tool support, general-purpose modeling languages (e.g., UML) are usually very large, where it is clumsy to add domain-specific restrictions. In addition, these languages lack detailed formal semantics which are needed for formal analysis [5]. In contrast, DSML is a specialized modeling language, which is tailored for the needs of a specific application domain.

To combine the advantages, a DSML is often defined in terms of UML. Among the UML profile based solutions, some are limited to allow modeling of only one aspect, i.e., structure or behavior of the system. While others, if covers both of these aspects, result usually in sets of discrete models [6], which are difficult to handle and comprehend [7]. While the standard UML profiles such as MARTE [8] and SysML [9] (or their combination) can be used to fulfill the modeling requirements of cyber physical systems, they do not provide any verification facility at the high-level design [10]. Moreover, defining a DSML in this way by using a UML profile is not enough for model transformation perspective in MDD. Furthermore, it is difficult to align the systems modeled by using the existing profiles with the widely used concepts of web services and cloud computing (resource-as-a-service). To handle these problems, a new simplified and systematic DSML is proposed in this work.

The rationale of the proposed DSML is to simplify the design models of cyber physical systems, while allowing the representation of their structure and behavior in a unified way. The proposed DSML is defined systematically in terms of a meta-model (to interpret any particular model instance formally) and implemented as a UML profile. In this way, the drawback of relying on UML formalism is eliminated and the benefits of the using the existing UML tool are obtained. The proposed DSML uses the Service-Oriented Computing (SOC) [11] concepts to align with web services and cloud computing domains. As a part of the MODEVES[1] project, this work further aims at producing design models, which are explicit enough to be used later in the transformation and verification phases of the project (which are not dealt with in this paper).

To illustrate the application of the proposed DSML, it has been applied to model two case studies: *Traffic Light* and *Arbiter*. Although the proposed DSML was applied in these case studies to check the soundness of the presented concepts, it is general enough to be applied to any cyber physical system. The rest of the paper is organized as follows: The next section provides the background of modeling mechanisms that can be used. The methodology of developing the DSML is explained in Section 3. The proposed DSML is presented in Section 4, which is then applied to the two case studies in Section 5. The discussion and related work are provided in Section 6, whereas the paper is concluded in the last section.

II. BACKGROUND

Classically, a DSML can be defined in the following ways [12, 13]: (1) by lightweight extension of UML via stereotypes and tagged values (known as UML *profile*), (2)

by either extending the UML meta-model or by defining a fully dedicated meta-model independent of UML using the Meta Object Facility (MOF) [14]. The latter option of defining a DSML is one of the most important aspects of MDD [12]. This is because a meta-model formally specifies a DSML by describing its *abstract syntax* and *static semantics*. The abstract syntax simply specifies the basic constructs of the language and their relationship, which then can be realized through the notations provided by the *concrete syntax*. The static semantics are the constraints on the abstract syntax that tells the well-formedness of the models. Developing a meta-model is useful not only to validate the developed models based on the defined constraints, but also in model-to-model transformations, where the transformations are defined as mapping rules between two meta-models. Moreover, a meta-model can act as generation templates for code generation, and can be used as a basis for tool integration [12].

On the other side, a UML profile can be defined by introducing new stereotypes of the important domain concepts and tag values to provide new domain specific semantics. Most UML tools are readily available to define a UML profile, due to their support for defining custom stereotypes and tagged values. However, some aspects of the abstract syntax cannot be conveniently defined through profiling. In addition, defining a UML profile or extending an existing one would inherit UML complexity and ambiguity. In contrast, extending the UML meta-model has the disadvantages of losing tool support, familiarity and standard conformity. Therefore, this alternative is not much used in academia and industry [15]. In existing work, DSMLs for safety-critical systems are defined in terms of UML profiles [7, 16, 17, 18], resulting in merely increasing the set of concrete syntax. In this sense, these solutions cannot be considered as holistic, as model transformation issues are not dealt. To be useful for model transformation and code generation these profiles still depend on the UML abstract syntax.

In the proposed modeling mechanism, a new DSML is defined systematically, for cyber physical systems, by specifying its abstract and concrete syntax. The abstract syntax is defined by developing a MOF based meta-model that is not tied to the UML meta-model, whereas the concrete syntax is defined by developing a profile. In this way, the proposed modeling mechanism can be easily used by implementing it in a UML tool, such as Papyrus [19].

III. METHODOLOGY FOR DEFINING THE DSML

To make an explicit interpretation and a common understanding, the proposed DSML was defined by developing a meta-model. To achieve this, the first step was the collection of domain concepts to develop a domain model. Although most of the concepts of cyber physical systems are domain specific, special care was given to represent the basic concepts in a generic way. The important concepts and their definitions are provided:

Device: Device is a physical entity, which takes some input, process it and performs some actions. It can range from small IC to large standalone devices such as a printer.

Service: Service is a logical entity that represents the functionality of a device. It can be considered similar to the concept of service as used in SOC. Instead of representing a software component, this concept has been adopted and given a broader meaning here to represent any facility offered by a device. To keep the models simple, other concepts of SOC, such as service composition, orchestration, service repository etc. are not included.

Event: An event is any action that takes place in the system, on the satisfaction of a particular condition. It can be caused both by system internal or external entities. When an event occurs, the states of one or more devices change. In other words, an event triggers a state change.

State: A state is a particular value of a device property at a certain time. A device can have one or more states, which change based on the events occurring in the system.

Once the domain concepts were collected to develop a domain model, the following list of requirements was defined which the meta-model should fulfill:

- Expressing the system structure in terms of devices and the communication among them. This includes the concepts related to a device, such as its properties, inputs/outputs etc.
- Representing the basic functionality of a device as a service and finding a way of representing the services of a device. In addition to service specific properties, the way different services communicate with each other need also to be represented.
- Finding a way to express the system behavior in terms of different events occurring in the system and subsequent changes in the states of the devices in response.
- Representing different states of a device and finding a suitable way to represent the state changes based on the occurred events.

To make the proposed DSML usable, a profile was then created based on the proposed meta-model, through UML meta-classes using façade meta-modeling approach [12]. To define the concrete syntax, different stereotype notations were defined. The basic principle was the mapping of concrete elements to abstract elements.

IV. THE PROPOSED DSML

Based on the methodology described in the previous section, a DSML is proposed in this work for cyber physical systems. The proposed DSML is specified in terms of its abstract and concrete syntax:

A. Abstract syntax

The abstract syntax is defined in terms of MOF based meta-model that is independent of UML. The meta-model consists of distinct entities: the device, the service and its components, and the relationships between them, as shown in Figure 1. The details of the elements of the proposed meta-model are provided below: A device is a physical entity that provides and manages different functionalities (services). Therefore, a device acts like a container of services. The NFP of a device can be included as *property*, which holds for all the contained services. The communication among devices takes place via different input/output signals. A

service represents a basic functionality offered by a device. A device can provide one or more services, which can be accessed simultaneously. In this way, services are considered as concurrent computing units, where each service has a set of associated properties. The service specific characteristics are included as *service property*. A service can interact with other services through service interface, which can either be *provided* or *required*.

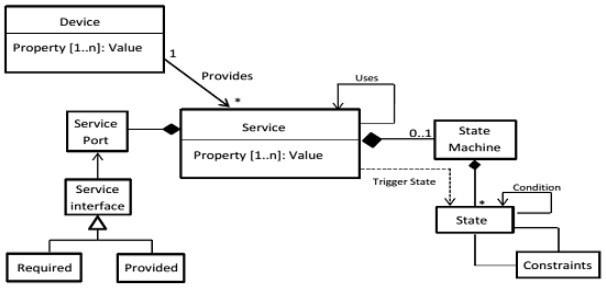

Figure 1. The proposed meta-model.

To define its behavior, a service may consist of a state machine that represents different states of the device and the way transitions among them happen. At the system initialization or reset, every state machine is at its start state. When an event occurs, a state generates one or more outputs, which become the inputs to states of other devices. These inputs cause the transition from one state to another, either in the same or the next clock cycle. The events are represented as a dashed line, whereas the state transition as a solid line. The transition specific requirements (named as a *condition*), such as durations and delays are represented along the state transition lines.

B. Concrete syntax

To make the proposed modeling mechanism useable and to allow its implementation in any UML tool, the DSML is also defined in terms of its concrete syntax (profile). This definition is based on the abstract syntax (proposed meta-model). Initially, the profile is developed by defining the stereotypes of different elements of the meta-model. In the future, the tagged values would be added to represent the attributes of the elements with their types and initial values. For example, the devices are represented by using «Device» stereotype, whereas the services they provide by the «Service» notation. The notations used to represent these concepts are presented in Table 1. The concrete syntax was implemented by creating the defined stereotypes in the Eclipse tool Papyrus, as shown in the next section.

V. APPLICATION OF THE PROPOSED DSML

This section presents the validation of the proposed modeling mechanism via its application in the *Traffic Light* case study. To ensure the completeness and consistency of the proposed DSML, it is also applied to the *Arbiter* case study. The models produced in the case studies are high-level (platform independent); therefore they do not provide the platform specific details.

TABLE I. CONCRETE SYNTAX OF THE PROPOSED DSML

Stereotype notations	***Concept***	***Symbol***	***Description***
<<Device>>	Service Provider	Basic block	The some service entity providing
<<Service>>	Basic Functionality	Basic block with shaded header	Functionality provided by a device
<<Events>>	Relationship	Dash line	Represents the events occurring in the system, where the direction of arrow points from source to destination
<<Condition >>	Requirement	Condition written on the Solid line	Represents the condition(s) that must be fulfilled to allow the state transition
<<port>>	Transition	Solid line	Represents the communication between devices

A. Traffic Light Case Study

The case study design model represents a simple traffic light controller for a North-South and East-West intersection. The North-South is the main road, and is given the GREEN light unless a sensor on the rarely used East-West farm road is activated. When that occurs, and the North-South light was GREEN for enough time, then the light will change to give way to the East-West traffic. The design also takes into account emergency vehicles that can activate an emergency sensor. When the emergency sensor is activated, then the North-South and East-West lights will turn RED, and will stay RED for a minimum period of 3 cycles.

In terms of the proposed modeling mechanism, the case study comprises of four devices, which are the two traffic signals (NS and EW), and two sensors (EW and emergency). Each of the devices is providing exactly one service, as shown in Figure 2. Each service consists of a state machine representing different states of the device.

The *start* state is represented by a solid circle in each state machine. A transition occurs on receiving an input signal. For instance, both *NSTrafficSignal* and *EWTrafficSignal* devices change to *Yellow* state, on receiving *emg-sensing* signal from the *EmergencySensor* device. Similarly, these devices change to *Red* state when the timer sends a signal that *EWTrafficSignal* is in *Green* state for 3 clock cycles (it is assumed that every device has a built-in timer). There are some states which generate outputs, such as sensing states of *EmergencySensor* and *EWSensor* devices. For simplicity, the service properties and service interfaces are not displayed at this level.

B. Arbiter Case Study

The arbiter system provides a link between 3 master devices and 2 target devices. A link is established between the master and the target devices by the mediator. At a given time, only one master can conduct a read or a write transaction and with only one target device. Any master device can conduct a transaction with any target device. The mediator contains arbiter logic that decides which master will be allowed to conduct a transaction. The arbiter uses a simple round robin technique. The mediator also contains

glue logic that actually decodes the master information for the target device and vice versa. The glue logic helps establish the link between a specific master device and the target device to conduct the transaction successfully.

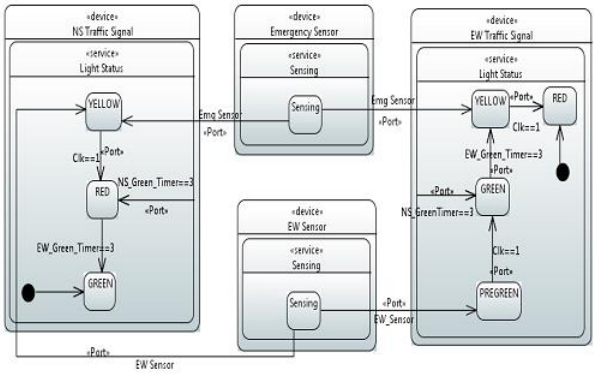

Figure 2. Traffic Light case study as implemented in Papyrus tool.

The master device can perform a read and a write transaction. It can support 2 target devices in a single system. When the master device gets the instruction "ask_for_it," it is ready to perform a transaction. It sends an active low pulse on the "req" signal and waits for a "gnt." The "gnt" signal is an active low signal. If the "gnt" signal does not come within 2 to 5 clock cycles, then the master will retry to get access at a later time. If the "gnt" is acquired, then the master will immediately assert the "frame" and "irdy" signals acknowledging the arrival of the "gnt" signal ("frame" and "irdy" are active low signals). In the same clock cycle, it also selects the target device it will have the transaction with. The master uses the output signal "rsel" to indicate this. If signal "rsel" is set to 1, then the master will have a transaction with target device 1. If the signal "rsel" is set to 0, then the master will have a transaction with the target device 0.

Once the signal "rsel" is updated, the target device is expected to identify itself to the master. The target device uses the signal "trdy" to acknowledge its readiness. If the target does not acknowledge itself within 3 clock cycles from the point when "rsel" is assigned, it is an error condition. If the target does acknowledge itself, then the master decides whether to read or write. The master sends the data and the instructions whether read or write through the "datac" bus.

Applying the concepts of the proposed modeling mechanism, the case study comprises of six devices, which are the three *Master* devices (Master 1, Master 2, and Master 3), one *Mediator* and two *Target* devices (Target 0 and Target 1). As the structure of all *Master* devices is same, the detail of only one *Master* and one *Target* device is shown in Figure 3 for simplicity. The *Master* and *Target* devices are providing one service each, but the *mediator* is providing two services, which are *Arbiter* and *Linker* services. Each service includes a state machine to represent different states. Some state waits for an inputs (e.g., *Ready* state in *ConductTrans* service), while other states are triggered by the arrival of the particular signal (e.g., *Acknowledge* state in *ConductTrans* service). Normally the states are taking one input and producing one output, but there are some states which are taking more than one input (e.g., *Decision* state in *Linker* service) and producing more than one output (e.g., *Acknowledge* state in *ConductTrans* service). The requirements that must be satisfied to perform a state transition (e.g., duration = 3) are written along with the transition arrow.

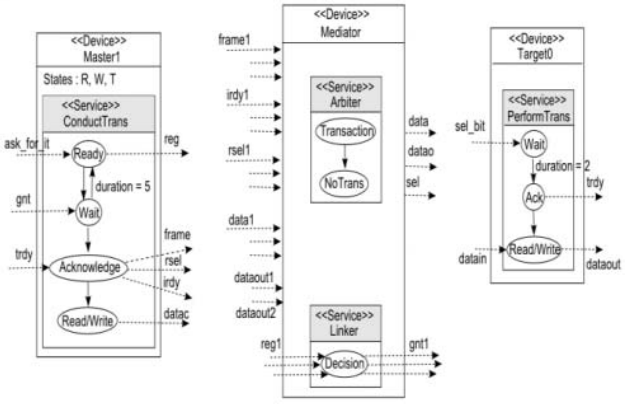

Figure 3. The unified model of Arbiter case study based on the proposed modeling mechanism.

VI. Discussion and Related Work

As stated earlier, the existing work is mostly related with the modeling of cyber physical systems using UML profiles. For example, *SafeUML* profile [20] is defined to model safety related concepts of aerospace systems. However, this profile is related to the generation of safety-related certification information from UML models. Similarly, another UML profile [21] includes safety-requirements in a UML model. But this profile defines the stereotypes that can only be used for safety analysis. RCSD profile [22] is another UML based profile proposed specifically for railway and tramway control systems. The stereotypes defined, however, are specific enough to the domain and cannot be used in other cyber physical systems.

Although requirements for domain-specific languages in MDD of safety-critical systems are defined [21, 23], there is no work available regarding the development of a MOF based meta-model for these systems. The facility to describe the system behavior as provided in the proposed mechanism reduces the potential safety problems that originate due to incomplete and ambiguous specifications.

In MODEVES project, a Systematic Literature Review [1, 2] has been performed to investigate latest MDD trends, approaches and tools to perform modeling, model transformation, model verification and simulation activities for the development of cyber physical systems using MDD. This includes Object Constraint language (OCL), MARTE Clock Constraint Specification Language (CCSL), Property Specification Language (PSL) and other property specification techniques. Consequently, OCL has been selected and its major constructs have been investigated in detail.

It is intended to generate System Verilog RTL and assertion code, in MODEVES project, from the developed

models. Therefore, a methodology, which will be logically equivalent to System Verilog semantics, needs to be developed to specify the constraints/properties in these models so that the logical mapping of concepts can be performed [24]. It has been analyzed that all OCL constructs cannot be mapped directly to System Verilog constructs. Hence, it is required to extend OCL in order to support basic constructs of System Verilog. The proposed methodology will be based on the OCL extension that supports System Verilog constructs. It is believed that this will significantly reduce the transformation efforts.

VII. CONCLUSION

As a first step towards developing a holistic design approach, a modeling mechanism for cyber physical systems is proposed in this work. A simple DSML is proposed in terms of MOF based meta-model and the concrete syntax is defined, based on which the models are implemented using the UML Papyrus tool. The proposed mechanism allows the unified modeling of the structural and behavioral aspects of a system in a single model. The applicability of the proposed mechanism is demonstrated through two case studies. In the future, it is planned to define the static semantics of the proposed DSML in terms of the meta-model constraints using the Object Constraints Language (OCL). To this end, the detailed analysis of the major constructs of the OCL has been performed, to investigate its use for formal specification of the constraints of the proposed meta-model. Based on this, the logical mapping of concepts would be defined for transformation into System Verilog assertions. It is also planned to enrich the proposed modeling mechanism by adding more concepts, safety requirements and temporal properties of cyber physical systems in our meta-model.

ACKNOWLEDGMENT

This project is funded by NSTIP (National Science, Technology, Innovative Plan), Saudi Arabia (grant no. 13-INF761-10). The authors acknowledge the support of STU (Science and Technology Unit), Umm Al-Qura University, Saudi Arabia.

REFERENCES

[1] M. Rashid, M. W. Anwar, A. M. Khan, "Identification of Trends for Model Based Development of Embedded Systems", In 12th IEEE International Symposium on Programming and Systems (ISPS), Algiers, April 2015, pp. 1 – 8.

[2] M. Rashid, M. W. Anwar, and A. M. Khan, "Towards the Tools Selection in Model Based System Engineering for Embedded Systems-A Systematic Literature Review," Journal of Systems and Software, vol. 106, 2015, pp. 150-163.

[3] F. Herrera, H. Posadas, P. Peñil, E. Villar, F. Ferrero, R. Valencia, and G. Palermo, "The COMPLEX methodology for UML/MARTE Modeling and design space exploration of embedded systems," Journal of Systems Architecture, vol. 60, Issue 1, January 2014, pp. 55-78.

[4] OMG Unified Modeling Language (OMG UML). Superstructure (vol 2.3). Object Management Group, Inc., 2009.

[5] A. D. Brucker, and J. Doser, "Metamodel-based UML notations for domain-specific languages," 4th International Workshop on Software Language Engineering (ATEM 2007). October, 2007.

[6] K. Berkenkötter, and U. Hannemann, "Modeling the railway control domain rigorously with a UML 2.0 profile," Computer Safety, Reliability, and Security. Springer: Berlin, Heidelberg, 2006, pp. 398-411.

[7] K. Mewes, Domain-specific modelling of railway control systems with integrated verification and validation. Verlag Dr. Hut. 2010.

[8] OMG UML Profile for MARTE: modeling and analysis of real-time embedded systems (vol 1.0, pp 738). Object Management Group, Inc., 2009.

[9] OMG Systems Modeling Language. http://www.omgsysml.org/.

[10] M. Rashid, M. W. Anwar and F. Azam, "Expressing Embedded Systems Verification Aspects at Higher Abstraction Level - SystemVerilog in Object Constraint Language (SVOCL), IEEE International Systems Conference (SysCon), Florida, USA, April 2016.

[11] M. N. Huhns, M. P. Singh, "Service-oriented computing: key concepts and principles," IEEE Internet Computing, vol. 9, issue 1, 2005, pp.75-81.

[12] T. Stahl et al., Model-driven software development: technology, engineering, management, Wiley, New York, 2006.

[13] F. Noyrit, S. Gérard, and B. Selic, Façade Metamodel: masking UML. Springer: Berlin, Heidelberg, 2012, pp. 20-35.

[14] Meta object facility (MOF) specification, OMG document formal/05-05-05. Also available as iso/iec 19502. 2005.

[15] M. W. Aziz, R. Mohamad, D. N. A. Jawawi, and R. Mamat, "Service based meta-model for the development of distributed embedded real-time systems," Real-Time Systems, vol. 49, issue 5, 2013, pp. 563-579.

[16] D. Kuschnerus, F. Bruns, A. Bilgic, and T. Musch, "A UML Profile for the Development of IEC 61508 Compliant Embedded Software," Online Proceedings Embedded Real-Time Software and Systems, 2012.

[17] S. Bernardi, J. Merseguer, and D. C. Petriu, "A dependability profile within MARTE," Software and Systems Modeling, vol. 10, issue 3, 2011, pp. 313-336.

[18] M. A. de Miguel, J. F. Briones, J. P. Silva, and A. Alonso, "Integration of safety analysis in model-driven software development," IET Software, vol. 2, issue 3, 2008, pp. 260-280.

[19] S. Gérard, C. Dumoulin, P. Tessier, and B. Selic, "Papyrus: A UML2 Tool for Domain-Specific Language Modeling," In Model-Based Engineering of Embedded Real-Time Systems, Lecture Notes in Computer Science, Springer: Berlin Heidelberg, 2010, pp. 361-368.

[20] Z. Gregory, B. Lionel, L. Yvan, "Modeling safety and airworthiness (RTCA DO-178B) information: conceptual model and UML profile," Journal of Software & Systems Modeling, vol. 10, issue 3, SpringerVerlag, pp. 337-367, 2011.

[21] M. Wasilewski, W. Hasselbring, and D. Nowotka, "Defining requirements on domain-specific languages in model-driven software engineering of safety-critical systems," 2013, pp. 467-482.

[22] K. Berkenkötter, and U. Hannemann, "Modeling the Railway Control Domain Rigorously with a UML 2.0 Profile," Computer Safety, Reliability, and Security, Lecture Notes in Computer Science, vol. 4166, 2006, pp 398-411.

[23] D. S. Kolovos, R. F. Paige, T. Kelly, and F. A. Polack, "Requirements for domain-specific languages," Proc. of ECOOP Workshop on Domain-Specific Program Development (DSPD), vol. 2006, July, 2006.

[24] M. Rashid, M. Waseem Anwar, F. Azam and M. Kashif, "Exploring the Platform for Expressing SystemVerilog Assertions in Model Based System Engineering", 7th International Conference on Information Science and Applications, Vietnam. February 2016.