Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Validation of static properties in unified modeling language models for cyber physical systems*

Gabriela MAGUREANU†, Madalin GAVRILESCU, Dan PESCARU

(*Department of Computers, Automation and Computers Faculty, "Politehnica" University of Timisoara, Timisoara 300223, Romania*)

†E-mail: gabriela_magureanu@yahoo.com

Received Sept. 15, 2012; Revision accepted Jan. 11, 2013; Crosschecked Mar. 6, 2013

Abstract: Cyber physical systems (CPSs) can be found nowadays in various fields of activity. The increased interest for these systems as evidenced by the large number of applications led to complex research regarding the most suitable methods for design and development. A promising solution for specification, visualization, and documentation of CPSs uses the Object Management Group (OMG) unified modeling language (UML). UML models allow an intuitive approach for embedded systems design, helping end-users to specify the requirements. However, the UML models are represented in an informal language. Therefore, it is difficult to verify the correctness and completeness of a system design. The object constraint language (OCL) was defined to add constraints to UML, but it is deficient in strict notations of mathematics and logic that permits rigorous analysis and reasoning about the specifications. In this paper, we investigated how CPS applications modeled using UML deployment diagrams could be formally expressed and verified. We used Z language constructs and prototype verification system (PVS) as formal verification tools. Considering some relevant case studies presented in the literature, we investigated the opportunity of using this approach for validation of static properties in CPS UML models.

Key words: Cyber physical system (CPS), Unified modeling language (UML) design, Formal verification, Prototype verification system (PVS), Z language

doi:10.1631/jzus.C1200263 **Document code:** A **CLC number:** TP311.5

1 Introduction

Cyber physical systems (CPSs) are massively distributed heterogeneous embedded systems linked through wired and/or wireless connections (Derler *et al*., 2011). They integrate computational and physical processes, sensors, actuators, and decision modules, and thus have great economic and social potential.

Each subsystem aims to fulfill a specific task or objective. One of the main issues in developing CPS applications resides in the asynchronous intercommunication between the subsystems and the influence of the exchanged information over the controlled devices. The variety and large number of existing CPS applications suggest an increased interest in developing efficient design methods of such systems. Therefore, a well-founded formal specification dealing with both static and dynamic application aspects is required. This paper deals with static aspects to formally specify both the hardware and the network infrastructure required for the CPS applications.

The unified modeling language (UML) consists of a set of general purpose modeling elements (Object Management Group (OMG), 2010). UML is widely used for specification, visualization, and documentation in various engineering fields. UML allows an intuitive graphical approach for embedded systems design.

* Project partially supported by the Strategic Grants POSDRU/88/1.5/S/50783 Project (No. 50783, 2009), POSDRU/107/1.5/S/77265 Project (No. 77265, 2010), Romania, and the European Social Fund for Investing in People, within the Sectoral Operational Programme Human Resources Development 2007–2013

UML models can be customized by defining and using stereotypes grouped in UML profiles (OMG, 2010). Stereotypes contain tagged values and constraints, which add specific attributes and behavior to the customized UML elements. Kuzniarz *et al*. (2004) demonstrated the expressiveness of UML profiles in defining UML models. UML profiles usage can significantly improve the overall understanding of the application models in a specific field. In this work, we investigate the opportunity of using UML for CPS design and formal validation, as an intuitive and efficient approach.

In case of CPS modeling, UML profiles can be divided into profiles for hardware and software specifications. The stereotypes for hardware specification help in customizing the network topology. They are used for defining the types of nodes of which the network and their internal hardware components are composed. The stereotypes for software specification are used for customizing the system behavior. Several UML profiles were proposed to model various aspects of real-time and embedded systems (Nguyen *et al*., 2004; Andre *et al*., 2005; Andersson and Host, 2008; Riccobene *et al*., 2009). They can also be successfully used for CPS design.

Since the UML does not provide a formal syntax, the semantics of UML models cannot be formally defined and validated. The object constraint language (OCL) is a declarative language that aims to bridge the gap between the ambiguity of a natural language in which UML models are expressed and the difficulty of mathematical constructs. Invariants are used to express a range of restrictions over the UML model features. System behavior is then clarified through pre- and post-condition, describing the applicability and impact of particular operations, respectively (OMG, 2010). Various OCL checker tools like the UML-based specification environment (USE) tool (Gogolla *et al*., 2007) or ITP/OCL tool (Clavel and Egea, 2006) can then be used for system validation.

However, the semantics of OCL are not mathematically defined; therefore, OCL is considered as being a semi-formal language. Using OCL for expressing rules and constraints applied to UML models is still insufficient for formal reasoning (France *et al*., 1998). Previous research has identified weaknesses in the semi-formal approach. It has been observed that some UML modeling constructs lack precise semantics, sometimes leading to differences in interpretation, and therefore inconsistencies between design and implementation.

For example, Gogolla and Richters (1998) identified OCLAny construct as being one of the issues. As OCLAny is considered as the super type of all other OCL defined types, any equality check between two elements of the OCLAny type is considered to be valid. However, when replacing the equality with elements of specific but different types, the evaluation still returns true. Another issue is related to the three types of polymorphism supported by the OCL specification. It is not clear how exactly these concepts are applied by the OCL checker tools. Therefore, it is left to them to implement these concepts. Additionally, the flattening process is not precisely described. Regarding the OCL undefined value term, the general rule states that the expressions containing such terms are also considered as undefined. This can lead to an incomplete evaluation of the model.

Baar (2005) noticed that the OCL non-deterministic constructs are poorly understood; therefore, they tend to be ignored and unimplemented by most of the OCL tools. One of the reasons is that such constructs allow different interpretations of the same OCL semantic, which may lead to different model implementations.

Formal specification languages are intended to provide precise and complete models of the proposed software systems. Formal specification for UML models can be achieved using formal constructs. Various tools can use this rigorous specification for verifying the UML models. The purpose of using formal specification and validation constructs is to provide unambiguous descriptions of system structure and functionality and to prove before deployment that the system will function according to the expected requirements (Bondavalli *et al*., 1999).

2 Cyber physical system modeling and verification

Modeling is a key aspect of a good design for both embedded and distributed systems. As CPSs can be seen as embedded distributed systems, various modeling techniques were proposed. In this work, we concentrate on structural models of CPS applications, which represent the static information in a system

design.

Object-oriented modeling is widely spread in software design and it could be successfully adapted to CPS structural design. The main issue in this case resides in the informal nature of most object-oriented models. A solution to this problem was presented by France *et al*. (1998) and Aredo *et al*. (1999). They proposed three approaches to formalize object-oriented modeling notations: supplemental approach, object-oriented extended formal notations, and methods integration.

In the supplemental approach, formal constructs replace parts of the model expressed in informal object-oriented notations. In the object-oriented extended formal notation, existing formal notations are extended by object-oriented mechanisms, therefore becoming more compatible with object-oriented notations. The advantage of using such an approach is given by the fact that rigorous formal systems are obtained.

The methods integration approach is a more suitable solution that makes informal object-oriented modeling concepts more precise by integrating them with proper formal specification techniques. This approach is the most commonly used in the formalization of object-oriented modeling notations. Therefore, it is appropriate for analysis tools development, since the first two concepts force the user to be in contact with a great amount of complex mathematical artifacts. In this case, the designers can directly manipulate graphical artifacts in the object-oriented models and they are not required of strong formal backgrounds. The two formal specification languages adopted in this paper are part of this third category.

Bhutto and Hussain (2011) presented a complex view on the most used formal verification tools, such as OCL, prototype verification system (PVS), Z language, process or protocol meta language (PROMELA), and UPPAAL model checker. The authors compared the presented tools and presented advantages for using each modeling tool at different levels of development processes. The OCL is found to be suitable as an established language for the specification of the static properties of objects and object structures in UML models. However, OCL is not suitable for specification and verification of the dynamical aspects requested by this application type. The Z language proves to be a suitable solution for the model specification based on the standard mathematical notation and the first-order predicate logic (Spivey, 1992).

In OCL, the expressions can be undefined. The available OMG documentation does not specify how the OCL checker tools should deal with the undefined queries from the expressions (Hamie *et al*., 1999). In most of the cases, the OCL tools denote nothing in the model verification process. The Z language maintains a clear distinction between logical operators and expressions. The logical expressions are not treated as expressions within the language; therefore their truth values are unknown if they involve undefined expressions.

Shroff and France (1997) proposed a formalization based on the Z language of the primary UML constructs used to build class artifacts. In this paper, we exemplify how the UML deployment diagrams can be formally represented similar to class diagrams. A Z state schema formalizes the static aspect of a class. The attributes and object identifiers of class instances are represented by state variables. Class invariants are specified in the predicate part of a Z schema. In an operation schema, the input and output variables represent the before and after states, respectively. The relationships between these states are represented in the predicate part. We propose rules for specifying the constructs of deployment diagrams, using state and operation schemas.

A hybrid solution by merging both advantages of OCL and Z languages was described by Roe *et al*. (2002). It provides a mapping from an integrated model of UML model and OCL specifications into a Z formal specification. We use this mapping to translate the OCL constraints of the used stereotypes from our UML designed models.

Dupuy *et al*. (2000) proposed a complex approach of using the Z language and introduced a tool called RoZ. This tool allows the generation of specifications for elementary operations on classes and the generation of proof obligations to validate operation guards. It allows automatic generation of Z schemas for the UML class, object, and deployment diagrams along with the elementary operations. We use the RoZ tool as support for developing specifications for CPS applications. To validate Z theorems, several tools were proposed in the literature, and we use the Z/EVES theorem prover (Saaltink, 1997) in our work.

As an alternative to Z language, many researchers have used PVS to formally define and analyze UML models. PVS consists of a specification language, a type checker, a model checker, and a theorem prover (Crow *et al*., 1995). In this paper, we investigate the opportunity of using PVS and the Z language for specifying deployment diagrams.

Aredo *et al*. (1999) defined general formalization approaches for UML interfaces, classes, associations, generalizations, and aggregations using PVS. They defined a generic parameterized theory that becomes an efficient solution to formally represent diagram elements. However, this approach requires a rigorous definition for a single theory for all elements.

Aredo (2003) explored the advantages of using PVS formalism for checking the correctness and completeness of UML models. They proposed the formal semantic definition for UML statecharts using the PVS specification language. They developed a general framework for translating UML statecharts diagrams into PVS specifications. The benefit of using such translation resides in the capability to produce precise and analyzable specifications and to use the rigorous reasoning provided by the PVS tools. The behavioral aspects for every node of the CPS application can be modeled using the statecharts diagrams. Therefore, this research represents a suitable solution for translating the dynamic aspects into the PVS specification language.

Aredo (2002) presented the formal semantics of UML sequence diagrams using the PVS language as the underlying semantic foundation. The basic concepts of sequence diagrams must first be formally specified, and then the semantics are translated into PVS theories. We use the steps described by Aredo (2002) to formally specify the models of case studies into the PVS language.

For representing OCL constraints in PVS, we consider the method presented in Kyas *et al*. (2005). This method makes a clear separation between shallow and deep embeddings. In the case of deep embedding, the abstract syntax is represented as a data type in the logic of the PVS theorem prover. A semantic function that interprets all possible terms for the language is defined. In the case of shallow embedding, the language is represented directly in the logic of the PVS theorem prover. This is possible by using translation of the language in a verification condition, which encodes the semantic for the given model. The prototype tool defines formal semantics for UML and OCL, and enables the formal verification of models. It also solves the translation of three-valued logic from OCL into two-valued logic in PVS. Kyas *et al*. (2005)'s research is useful for our case as we use OCL constrains for modeling the behavior and restrictions of CPSs.

3 Formal specifications of cyber physical system (CPS) deployment models

The references presented use formalization for UML class, statechart, sequence, and state machine diagrams. As deployment diagrams are a key feature in the design of distributed systems, we extend the formalization approach to the CPS models designed using these diagrams.

First, we define the stereotypes and their corresponding OCL constraints, along with the main constructs, as UML nodes and components. Next, the relations such as associations, aggregations, and links are translated into the formal language. Finally, the entire application is defined as the union of specifications for all diagram elements and the relationships between them.

3.1 Z specification of CPS deployment models

We present the theoretical approach towards the formalization of UML deployment diagrams using the Z language. The constituents of deployment diagrams are represented as Z schemas, similar to the representation of UML class diagrams in Dupuy *et al*. (2000).

Each node or component can be translated into a pair of Z schemas. The first Z schema describes the types for all contained elements such as attributes, ports, and operations. The second Z schema describes the set of existing instances for the node or component.

Fig. 1 presents an example of Z schema corresponding to a 'NODE_A' type. The set of existing instances is 'NodeA_Ext', while each node instance is of the type 'NodeA'.

In UML deployment diagrams, the association construct is defined by two roles. However, in case of unidirectional constructs, only one role is required. In

Z language, a function specifies each role. The function defines the relationship by mapping instances based on the multiplicity. Therefore, it is necessary to provide the corresponding schemas of nodes and components. These schemas have to be included as variables, along with the Z domain and Z range for each role.

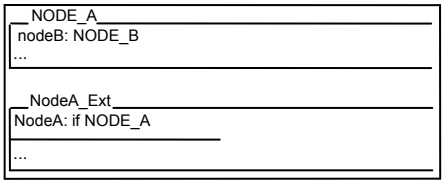

Fig. 1 An example of Z schemas corresponding to the 'Node_A' deployment node

To define the multiplicity, the theory should include an axiom regarding the number of instances of one class that can be associated to a corresponding node instance. We define the Z schema for representing a UML link differently from the one representing an association. Indeed, in the case of links, each function associates instances in a one-to-one relationship.

Fig. 2 presents an example of the Z schema for a bidirectional link between NodeA and NodeB instances. The Z schemas corresponding to NodeB deployment node type are considered similar to those already presented in Fig. 1.

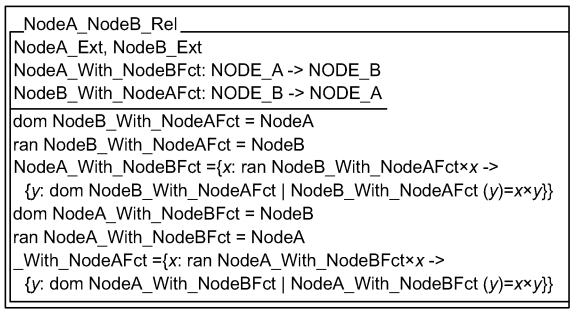

Fig. 2 An example of Z schema for a bidirectional link between 'NodeA' and 'NodeB' instances

The aggregation, generalization, dependency, and usage constructs are represented in UML as particular cases of unidirectional links. Therefore, to specify them, we state that the Z schema must contain only one Z function, which defines the relationship between the client and the supplier (NodeA_With_NodeBFct and NodeB_With_NodeAFct in Fig. 2, respectively). However, when specifying an aggregation construct, the multiplicity must also be considered, as already described above.

We define the Z schema for a stereotype by describing the types for all tagged values and OCL constraints corresponding to the stereotype. The customization of a node using a stereotype is formally represented by inserting the Z schema for the stereotype as a variable in the Z schema for that node. For stereotypes inheritance, the stereotypes that are specializations of other stereotypes must include the Z schema of the latter stereotypes in the variable part, as schema invocation.

A schema containing instances of both base and derived stereotypes states that all already created instances of the derived stereotypes are also instances of the base stereotypes. The Z schema must contain a predicate for expressing the uniqueness for object identifiers.

3.2 PVS specification of CPS deployment models

As Z language is considered to be more suitable for examples of reduced complexity, we use PVS in more complex reasoning. The UML constituents of deployment diagrams can be represented in PVS using theories and axioms, which is similar to Aredo *et al*. (1999). The PVS theories contain declarations of record types. A record type specifies the signatures for the operations, along with the declarations of the attributes and ports of a UML node or component.

If a node is a specialization of another node and it implements one or more interfaces, the defined record type must include the attributes, ports, and operations fields of all base nodes and all implemented interfaces. The resulting record type of PVS theory represents a union between the included record types in this case. These rules are also applied to components in deployment diagrams.

Here, we consider a generic parameterized theory that can be applied as a template to specify associations, links, dependency, usage, aggregations, and generalizations constructs of a deployment diagram. The list of generic parameters contains the UML elements whose instantiations are involved in the relationship. It also contains the corresponding roles of the objects and the multiplicities as a subset of the natural numbers.

To define the multiplicity in the case of association and aggregation constructs, the theory includes an axiom regarding the number of instances of one UML element that can be mapped to corresponding element instances. Links are represented by instantiations of defined generic associations. The applied rule states that links are one-to-one associations between UML elements, with the multiplicity equal to 1.

In the case of stereotyped constructs, the record type defined includes fields for the tagged values and the OCL constraints of the corresponding stereotype. The OCL constraints can be easily represented and verified in PVS tools. The OCL formulas are translated directly into the PVS specification language.

4 UML hardware profile for cyber physical system (CPS) applications

A UML approach allows high level-modeling of CPS applications. It allows static and behavioral descriptions of this kind of distributed application. The resulted UML models allow the user to define several hardware and software parts for the applications. As a result, each defined hardware component is assigned a specific behavior based on the application specifications.

Kuzniarz *et al*. (2004) showed that using predefined stereotypes in UML models helps improve the overall understanding of the UML models in question. This represents a desired goal, as it helps in a better definition and understanding of the internal configuration of each node which is part of the network and the network as a whole.

In CPS UML modeling, the user starts the design of the application by defining the static network infrastructure. The network model is specified within a deployment diagram containing a set of hardware devices connected or linked through each other. The next step represents the design for each type of node involved in the network model. Each node model is defined in its own deployment diagram consisting of the internal hardware components. For customizing the network infrastructure along with its types of nodes and their internal components, we defined a deployment diagram related UML hardware profile in Magureanu *et al*. (2011; 2012).

The behaviors of components designed at different levels of abstractions make up the UML software profile. Specific sets of software description related stereotypes are mapped to those designated for allowing the hardware description. The UML software profile has been introduced in Gavrilescu *et al*. (2012).

Next, we focus on presenting the main constructs from the hardware profile, as this paper deals with the formal specification and validation of the static aspects for designing CPS applications.

The stereotypes defined in this UML hardware profile, along with the tagged values and the OCL constraints, are used in defining the hardware components, whose instances are actually used in modeling the application. Well defined stereotypes for specifying hardware components help achieve a clear separation and grouping between families of devices. Next, we present some stereotypes which are used within the case studies presented in Section 5.

The ‘CompoundNode_PIM’ stereotype along with its inheritances is used for defining and customizing a particular type of node. The OCL constraints restrict the user in adding particular node type related hardware internal components for the corresponding deployment diagram for that node.

The ‘Can_HWST’ stereotype defines a hardware ‘CAN’ unit and its communication ports. Fig. 3 presents the tagged values and OCL constraints of the ‘Can_HWST’ stereotype used as an internal hardware component. The OCL constraints presented in Fig. 3

```
Can_HWST
bool_define_tx_enPort: BOOLEAN
self.base_Node.ownedPort -> exists(a | a.name='rx' and a.oclIsTypeOf(InputGate_HWST))
self.base_Node.ownedPort -> exists(a | a.name='tx' and a.oclIsTypeOf(OutputGate_HWST))
if self.bool_define_tx_enPort = 'true' then
  self.base_Node.ownedPort -> exists(a | a.name='tx_en' and a.oclIsTypeOf(OutputGate_HWST))
  else true
endif
```

Fig. 3 Object constraint language (OCL) constraints for the Can_HWST stereotype

specify that the models stereotype with 'CAN_HWST' must declare two different types communication ports, named 'rx' and 'tx', respectively. Optionally, the designer can design its module as having a third port used for enabling/disabling the transmit port, by setting the tagged value named 'tx_enPort' on true, a case in which the third output port named 'tx_en' is required in the model.

The 'PSOCUnit_HWST' stereotype represents a particular hardware unit named programmable system-on-chip (PSoC) and developed by Cypress Semiconductor Co. (USA). The physical hardware device integrates configurable analog and digital peripheral functions, memory, and an internal microcontroller on a single chip. Its reconfiguration capabilities allow the designer to bind internal resources on the fly, and thus to use a small number of components for each specific task (Magureanu *et al*., 2011). The stereotype presented allows customizing such a device and defining application specific hardware ports.

5 Z formal specification and validation of static aspects for a sensing node model

In Section 5.1, we illustrate Z specification at the node level, in a particular CPS application, using the specifications proposed for the deployment diagram artifacts. As a case study, we consider the system presented in Magureanu *et al*. (2011), which describes a CPS for the traffic management in an urban intersection. Section 5.2 considers the validation part for the case study already specified. Our aim is to illustrate how an operation represented by a Z schema can be formally verified and improved.

5.1 Z formal specification of a sensing node model

The traffic management application contains several types of nodes. A decision manager (DM) node computes optimal green color duration for each traffic light node. The calculations are based on the information provided by the sensor nodes. The sensor nodes from the presented CPS gather information about the number of cars waiting at the red color of the semaphores.

Each node specification is composed of hardware and software models. The hardware model is customized using stereotypes and OCL constraints of the UML profile presented in Magureanu *et al*. (2011). The communication between the hardware units of a node is realized here using wired connections. Fig. 4 presents the UML model for the hardware configuration of the sensing node used in this CPS application.

The sensing node internal architecture is expressed using deployment diagram artifacts. The blocks are instances of UML deployment nodes. These components are customized with stereotypes, which hold constraints of different hardware unit type definitions. The internal connections of the sensing node and the external connection with its corresponding traffic light node are also shown in Fig. 4.

A set of Z schemas is defined for each deployment node stereotype. Each schema is named according to the stereotype name. The variable part of the schema contains all tagged values defined by the stereotype. The predicate part contains the OCL statements expressed in Z language.

The Z schema of the sensing node's CAN unit contains the Can_HWST schema as a variable. This variable indicates the formal definition of the corresponding stereotype. The Z specification prover

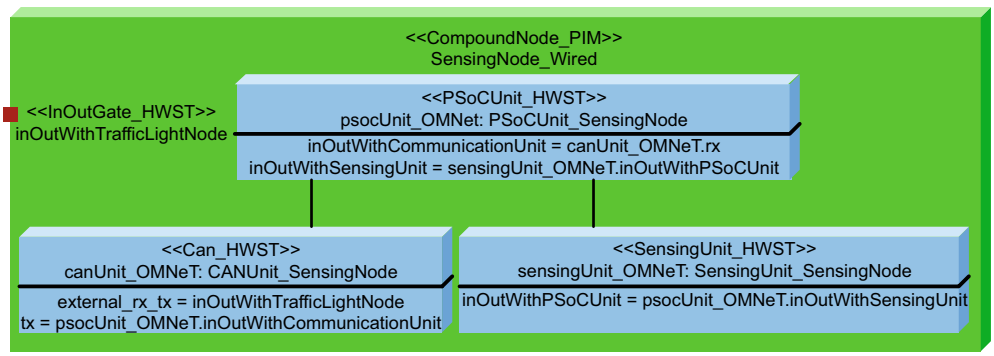

Fig. 4 Unified modeling language (UML) model for the hardware configuration of a sensing node

tool will later determine whether the stereotyped component is correctly customized with the corresponding stereotype, in terms of variables and constraints.

In Fig. 5a, the variable part of the SENSINGNODE_WIRED schema defines the hardware units of a sensing node model. For each of these components, a Z schema is provided. Fig. 5b describes the set of existing sensing nodes in the network, as instances for the node presented in Fig. 4.

As presented in Fig. 5c, SensingNode_Wired is an instance of the SensingNode set and expresses a node used in the CPS. It contains the actual internal instantiations for the hardware unit components.

Connections between units are described using Z functions for link specifications. The links between nodes in the deployment diagram or between components of a node are represented using functions defined for both roles of the association. The constraints are meant to define the domain and range of these functions. The constraints also define the fact that both functions are linked and the functions refer to the same exchanged information. In Fig. 6, we present the Z specification for the bidirectional link between a sensing node and its corresponding traffic light node.

Actual values for the domain and range of these functions are set in the network specification. The network representation at the physical level contains the schemas of node intensions and node extensions for all nodes in the traffic network, along with the links between nodes. These schemas are also represented as functions.

(a)

SENSINGNODE_WIRED

CANUnit_SensingNode: CANUNIT_SENSINGNODE_WIRED
SensingUnit_SensingNode: if SENSINGUNIT_SENSINGNODE_WIRED
PSoCUnit_SensingNode: PSOCUNIT_SENSINGNODE_WIRED
InOutWithTrafficLightNode: INOUTGATE

(b)

SensingNode_Wired_Ext

SensingNode: if SENSINGNODE_WIRED

...

(c)

SensingNode_Wired

CANUnit_SensingNode=canUnit_OMNeT
SensingUnit_SensingNode=sensingUnit_OMNeT
PSoCUnit_SensingNode=psocUnit_OMNeT
InOutWithTrafficLightNode=inOutWithTrafficLightNode
OutputToInOutGateFct(canUnit_OMNeT.tx) = psocUnit_OMNeT.inOutWithCommunicationUnit
InOutToInOutGateFct(canUnit_OMNeT.external_rx_tx) = inOutWithTrafficLightNode
InOutToInputGateFct(psocUnit_OMNeT.inOutWithCommunicationUnit) = canUnit_OMNeT.rx
InOutToInOutGateFct(psocUnit_OMNeT.inOutWithSensingUnit) = sensingUnit_OMNeT.inOutWithPSoCUnit
InOutToInOutGateFct(sensingUnit_OMNeT.inOutWithPSoCUnit) = psocUnit_OMNeT.inOutWithSensingUnit

Fig. 5 Z specification of the SENSINGNODE_WIRED node (a), the set of instances (b), and a particular sensing node type (c)

SensingNodeTrafficLightNode_Wired_Rel

SensingNode_Wired_Ext, TrafficLightNode_Wired_Ext
SensingNodeInOutWithTrafficLightNodeFct: SENSINGNODE_WIRED -> TRAFFICLIGHTNODE_WIRED
InOutWithSensingNodeFct: TRAFFICLIGHTNODE_WIRED -> SENSINGNODE_WIRED

dom InOutWithSensingNodeFct=SensingNode
ran InOutWithSensingNodeFct=TrafficLightNode
SensingNodeInOutWithTrafficLightNodeFct={x: ran InOutWithSensingNodeFct×x ->
{y: dom InOutWithSensingNodeFct | InOutWithSensingNodeFct(y)=x×y}}
dom SensingNodeInOutWithTrafficLightNodeFct=TrafficLightNode
ran SensingNodeInOutWithTrafficLightNodeFct=SensingNode
InOutWithSensingNodeFct={x: ran SensingNodeInOutWithTrafficLightNodeFct×x ->
{y: dom SensingNodeInOutWithTrafficLightNodeFct | SensingNodeInOutWithTrafficLightNodeFct(y)=x×y}}

Fig. 6 Z specification for relationship between sensing nodes and traffic light nodes

In this case study, we focus on the specification and validation of the application at the node level; therefore, the network schema is not presented in detail. However, based on presented specification steps for a CPS node, the specification of the network schema becomes intuitive.

Section 2 covers the validation process of the elementary operations generated using the RoZ tool (Dupuy *et al*., 2000). The generated elementary operations along with the evaluation are discussed in more detail.

5.2 Validation of Z static specifications for a sensing node model

Each type of the UML diagram construct can be manually transformed into formal specifications using the set of rules described in Section 3. This process can be optimized using dedicated tools, such as the RoZ tool (Dupuy *et al*., 2000), for generating complete Z specifications from annotated UML deployment diagrams. The RoZ tool is able to generate specifications of elementary operations over the UML elements and proof obligations. A common example of such elementary operations deals with modification of deployment node attributes. This is significantly helpful in validation of the UML model constraints.

When considering validation for the traffic management case study already specified, one of the application requirements states that the models must ensure the possibility of using other defined sensing units. As a method for determining the impact of defined operations over the UML model constraints, we use the computation of corresponding operations preconditions. This implies determining the conditions that must be satisfied before the operations take place. The aim is to preserve the constraints at the end of the operations (Ledru, 1998).

We discuss the ModifySensingUnit operation described by the Z operation schema (Fig. 7a). The first line specifies that the domain of the effect for this operation is restricted to objects of the type SENSINGNODE_WIRED.

The predicates state that all attributes keep their initial values except for the one representing the sensing unit. This attribute receives the value of the newSensingUnit_SensingNode input parameter. The precondition for this operation verifies the existence of values for the new types of sensing units.

Fig. 7b presents this precondition represented as Z specification. Furthermore, the information can be used to describe a theorem to validate the precondition. This theorem generated in RoZ is depicted in

(a)

SENSINGNODE_WIREDModifySensingUnit

```
Δ SENSINGNODE_WIRED
newSensingUnit_SensingNode?: SENSINGUNIT_SENSINGNODE_WIRED
---
CANUnit_SensingNode′ = CANUnit_SensingNode
SensingUnit_SensingNode′ = newSensingUnit_SensingNode?
PSoCUnit_SensingNode′ = PSoCUnit_SensingNode
InOutWithTrafficLightNode′ = InOutWithTrafficLightNode
```

(b)

```
newSensingUnit_SensingNode? ≠
newSensingUnit_SensingNode? \neq \empty
```

(c)

```
\begin{theorem}{SENSINGNODE_WIREDModifySensingUnit\_Pre}
\forallSENSINGNODE_WIRED;
newSensingUnit_SensingNode?:\finset SENSINGUNIT_SENSINGNODE_WIRED |
newSensingUnit_SensingNode?\neq \empty @
\pre SENSINGNODE_WIREDModifySensingUnit
\end{theorem}
```

(d)

```
try lemma SENSINGNODE_WIREDModifySensingUnit\_Pre;
prove by reduce;
```

Fig. 7 Z specification of the ModifySensingUnit operation (a), Z language statements for the sensing node model (b), Z language theorem for the sensing node model (c), and Z language evaluation of the sensing node theorem (d)

Fig. 7c. As mentioned before, to validate Z theorems, we use the Z/EVES theorem prover (Ledru, 1998). The commands required for proving the theorem demonstrate the validity of the defined precondition (Fig. 7d). Therefore, the initial constraints are preserved at the end of operations. This verification of the initial constraints ensures that all units from the sensing family are replaceable, as long as they satisfy the family specifications.

6 Prototype verification system formal specification and validation of static aspects for a wireless sensor network monitoring application

In this section, we consider an application for a gas-distribution monitoring system (Magureanu *et al*., 2010) as a case study. The aim is to reveal possible design problems in specification of the wireless communication in a CPS application at the network level.

6.1 PVS formal specification of a WSN monitoring application

In this section, we indicated how PVS language can be used to specify the stereotypes required for modeling a gas management system.

When deploying such nodes, it is necessary to specify the position (x, y, z), transmission, and reception parameters for the node. The application goal is to control the gas flooding in case of pipe leaks.

For an efficient monitoring, the network was logically tailored into perimeters, zones, and areas. The communication between the logically grouped layers is ensured using the beacon approach (Buratti *et al*., 2009). The beacons are represented in this case study by decision manager (DM) nodes.

For expressing this tailoring during the design process, stereotypes definitions for each modeling level are required. The OCL constraints for these stereotypes express the meanings of definitions. Next, we present these OCL constraints in a natural language.

A logical grouping stereotyped with Perimeter expresses a set of nodes containing a single instance of a node customized with DM_PerimeterCompound Node_HWST stereotype, which manages and monitors all the nodes logically grouped in a perimeter. All the other nodes are instances of nodes customized with CompoundNode_PIM stereotype, which represent sensors or actuator valves.

The Zone stereotype expresses a set of nodes containing a single instance of a stereotype named DM_ZoneCompoundNode_HWST, whereas all other nodes are stereotyped with Perimeter or are instances of CompoundNode_PIM stereotype. The node customized with DM_ZoneCompoundNode_HWST manages and monitors all the sensors, actuator valves, and its internal perimeters, logically grouped in a zone. It also handles the information received from the perimeter manager nodes.

The Area stereotype expresses a set of nodes containing a single instance of a node customized with DM_AreaCompoundNode_HWST stereotype. All other contained nodes are stereotyped with Zone or Perimeter or are instances of nodes customized with the CompoundNode_PIM stereotype. Following the same reasoning, a node stereotyped with DM_AreaCompoundNode_HWST manages a set of nodes logically grouped in an area.

To focus on relevant aspects in our discussion, we consider the simplified example of unidirectional communication, from the layer manager to the logically grouped set of nodes and from the current layer to the lower layer manager. Fig. 8 shows the UML model of this CPS. Similar to the approach presented in Aredo *et al*. (1999), we formally represent the stereotypes in the PVS specification language.

Fig. 9 depicts, as PVS theories, the stereotypes used for defining the physical nodes. For each of these stereotypes, we define a record type, in which the fields are declarations of tagged values. As we consider a specific scenario for evaluating the definitions at the area level, Fig. 9 presents only the relevant tagged values for each used stereotype.

The X, Y, and Z types are subsets of integer values expressing the bounding box of the network's topology. The PVS importing mechanism allows the already defined types to be referred to by a theory that requires them. The tagged values and operations are inherited and used along with local ones.

In case of specifying a stereotyped node, we propose the same import and inheritance mechanism for the attributes and operations defined by the stereotype. Therefore, the theory describing the

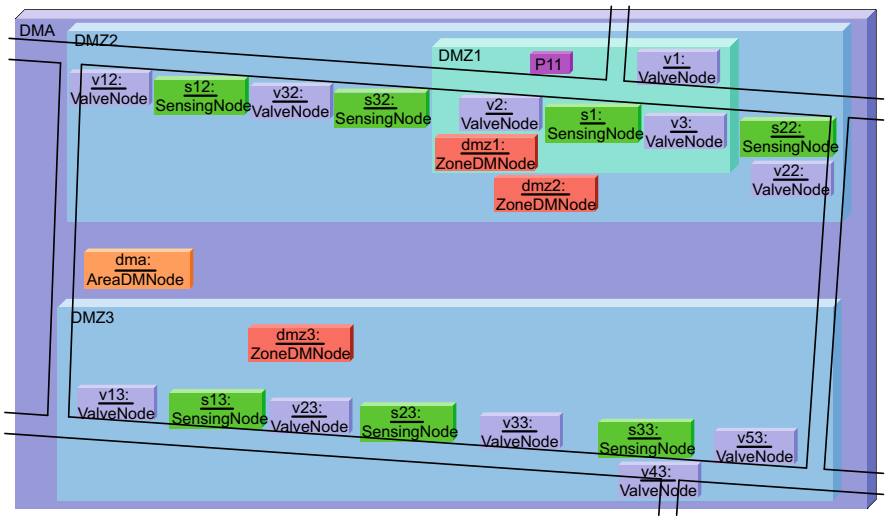

Fig. 8 Unified modeling language (UML) model for distributed gas monitoring topology

```
CompoundNode_PIM: THEORY
  BEGIN
  IMPORTING SENSITIVITY, X, Y, Z
    CompoundNode_PIM: TYPE = [# sensitivity: SENSITIVITY, x: X, y: Y, z: Z #]
  END CompoundNode_PIM
DM_PerimeterCompoundNode_HWST: THEORY
  BEGIN
  IMPORTING SENSITIVITY, X, Y, Z
    DM_PerimeterCompoundNode_HWST: TYPE = [# sensitivity: SENSITIVITY, x: X, y: Y, z: Z #]
  END DM_PerimeterCompoundNode_HWST
DM_ZoneCompoundNode_HWST: THEORY
  BEGIN
  IMPORTING SENSITIVITY, X, Y, Z
    DM_ZoneCompoundNode_HWST: TYPE = [# sensitivity: SENSITIVITY, x: X, y: Y, z: Z #]
  END DM_ZoneCompoundNode_HWST
DM_AreaCompoundNode_HWST: THEORY
  BEGIN
  IMPORTING AREA_CARRIER_FREQUENCY, PMAX, SAT, ALPHA, X, Y, Z
    DM_AreaCompoundNode_HWST: TYPE = [# carrierFrequency: CARRIER_FREQUENCY, pMax: PMAX,
      sat: SAT, alpha: ALPHA, x: X, y: Y, z: Z #]
  END DM_AreaCompoundNode_HWST
```

Fig. 9 Prototype verification system (PVS) theories for compound node, perimeter decision manager (DM), zone DM, and area DM stereotypes

stereotype is imported in the theory describing the node itself. This ensures the availability of the stereotype record type in the node theory.

Fig. 10 presents, as PVS theories, the stereotypes used for describing the logical tailoring of the network topology. The specifications derived from the associated OCL constraints are used to formally express the PVS theories.

A gas distribution system requires a large number of interconnected pipes. Therefore, to manage such a complex network, a large number of sensors are required along with local management nodes and valve actuators. However, in this case study, we try to illustrate the network as simple as possible, for a better understanding of how it is topologically structured. This allows us to reveal the benefits of using PVS tools for evaluating such massively distributed systems. We have chosen PVS tools because it is easier to specify and verify complex networks using PVS specification constructs along with the type checker and theorem prover.

The gas pipes network presented by Magureanu *et al*. (2010) is modeled using UML deployment diagrams. One particularity of this case study is the usage of wireless communication between nodes. Wireless communication operating frequency is considered placed within the industrial, scientific, and medical (ISM) radio band. As an example, we present the UML model for a sensing node. The tagged values shown in Fig. 11 are inserted by the CompoundNode_ PIM stereotype. Basically, a stereotype inserts its tagged values as mandatory attributes for the customized node.

```
PERIMETER: THEORY
  BEGIN
  IMPORTING CompoundNode_PIM, DM_PerimeterCompoundNode_HWST
    PERIMETER: TYPE=[# dm: DM_PerimeterCompoundNode_HWST,
      nodes: setof[CompoundNode_PIM] #]
  END PERIMETER
ZONE: THEORY
  BEGIN
  IMPORTING PERIMETER, DM_ZoneCompoundNode_HWST
    ZONE: TYPE=[# dm : DM_ZoneCompoundNode_HWST, perimeters : setof[PERIMETER],
      nodes: setof[CompoundNode_PIM] #]
  END ZONE
AREA: THEORY
  BEGIN
  IMPORTING ZONE, DM_AreaCompoundNode_HWST
    AREA: TYPE=[# dm: DM_AreaCompoundNode_HWST, zones : setof[ZONE],
      perimeters: setof[PERIMETER], nodes: setof[CompoundNode_PIM] #]
  END AREA
```

Fig. 10 Prototype verification system (PVS) theories for perimeter, zone, and area stereotypes

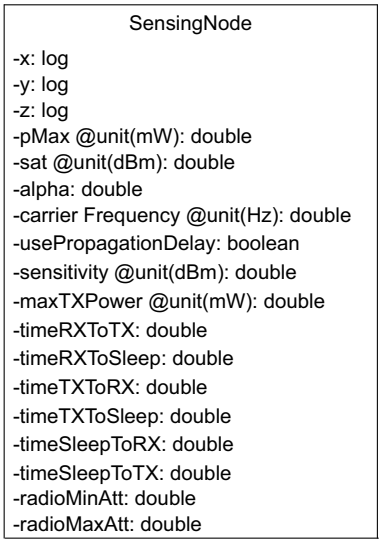

Fig. 11 UML deployment for the sensing node

6.2 Validation of PVS static specification for the wireless network area model

In this section, we discuss how formal verification can help detect lacks in the initial specification. This task is accomplished with the help of the PVS type checker and the theorem prover. The aim is to adjust UML models before starting the development process.

Here, we discuss the conditions for the nodes to be in the communication range of each other and to establish a valid route. In Fig. 9, a possible situation is when a DM node of an area (dma) is able to communicate to the DM of a zone (dmz3), the transmission is unable to reach a far located zone (dmz2).

For a wireless communication evaluation, we need to consider a proper radio model in PVS. The signal transmission in wireless networks is influenced mainly by the signal frequency, path loss, receiver sensitivity, and noise. Path loss, known as attenuation, is influenced by deployment configuration, distance between the transmitter and the receiver, height, and location of the antennas, obstacles, and weather conditions (Stoyanova *et al*., 2009). The evaluated power of the received signal is gained by multiplying the power of the transmitted signal with every attenuation mapping.

Several propagation models are proposed to estimate the radio signal propagation distance. These models rely on information specific to a considered scenario. The appropriate radio model is chosen based on the application's specific operating environment, the existing obstacles, and the technology used to implement wireless communication (Stoyanova *et al*., 2009).

We start our investigation with a simplified radio model as shown in Fig. 12a (Rousselot and Decotignie, 2009). The radio model does not take into consideration of obstacles or other environment limitations. The PVS theory for a DM node of the area level is formally specified in Fig. 12b. It makes use of the radio model theory and inherits from the PVS theory DM_AreaCompoundNode_ HWST.

Hence, subsystem parts of CPS applications based on the used stereotypes and their associated OCL constraints can be specified. Following the same reasoning, the entire system can be specified.

The validation part aims to evaluate the capability of the already specified DM node of the area to communicate with the zones included in its boundaries. Conditions for calculating the communication capacity are expressed in the form of PVS theorems. These become objectives to be verified using the PVS theorem prover. Type checking is a precondition for the evaluation of the theorems. It represents an intermediary step between a completely specified theory and theorem proving. The PVS type checker searches for semantic errors, like ambiguous types or undeclared variables in the theory.

In dma theory, the receive function determines whether an external node is in the receiving range of the considered node. The decision is based on the transmission power of the sender.

In this work, the verification determines whether the node with which the dma wants to communicate is in its range. In this case, the evaluation of the checkCommunicationWithAllNodes theorem fails.

A valid model is presented in Fig. 13. It also includes the sensitivity of the receiver in the theorem condition.

We conclude that performing formal validation of all theorems defined for the CPS model ensures a proper validation of the assumptions made in design.

7 Conclusions

CPS applications are presented nowadays in various fields of activity. Many researchers are

(a)

```
TINY_RADIO_MODEL: THEORY
  BEGIN
  IMPORTING PI_CONSTANT, LIGHT_CONSTANT, AREA
    TINY_RADIO_MODEL: TYPE=[# carrierFrequency: CARRIER_FREQUENCY, pMax: PMAX,
    sat: SAT, alpha: ALPHA #]
    calculateRange(carrierFrequency, pMax, sat, alpha) : nat =
pow((sqr(speedOfLight_value / carrierFrequency) * pMax)
        / (16 * sqr(pi_value) * pow(10, sat / 10)) , 1.0 /alpha)
  END TINY_RADIO_MODEL
```

(b)

```
DMA: THEORY
  BEGIN
  IMPORTING TINY_RADIO_MODEL
  DMA: TYPE FROM AREA
  receive(x, y, z, x_dm, y_dm, z_dm, node): bool =
    (IF (sqr(x - x_dm) + sqr(y - y_dm) + sqr(z - z_dm) <= sqr(range_receiver(node))) THEN true
       ELSE false
    ENDIF)
  inRange(x, y, z, x_dm, y_dm, z_dm, range) : bool =
    (IF  (sqr(x - x_dm) + sqr(y - y_dm) + sqr(z - z_dm) <= sqr(range)) THEN true
       ELSE false
    ENDIF)
  checkCommunicationWithAllNodes: CONJECTURE FORALL (node: zones): bool =
    (IF inRange(x(node), y(node), z(node), x(dm), y(dm), z(dm),
                  calculateRange(carrierFrequency(dm), pMax(dm), sat(dm), alpha(dm)))
      THEN receive(x(node), y(node), z(node), x(dm), y(dm), z(dm), node) == true
       ELSE false
    ENDIF)
  END DMA
```

Fig. 12 Prototype verification system (PVS) theory for a tiny radio model (a) and a DM node of an area (dma) model (b)

```
checkCommunicationWithAllNodes: CONJECTURE FORALL (node: zones): bool =
  (IF inRange(x(node), y(node), z(node), x(dm), y(dm), z(dm),
              calculateRange(carrierFrequency(dm), pMax(dm), sat(dm), alpha(dm)))
             AND inRange(x(dm), y(dm), z(dm), x(node), y(node), z(node), range_receiver(node))
             THEN receive(x(node), y(node), z(node), x(dm), y(dm), z(dm), node) == true
       ELSE false
   ENDIF)
```

Fig. 13 Corrected prototype verification system (PVS) method for validating the DM node of an area (dma) model

seeking solutions to cover the challenges of designing and implementing such applications.

High-level UML modeling is accepted as a suitable solution for the design of distributed embedded applications. However, the informal language used to express UML models constitutes a disadvantage in ensuring the correctness and completeness of the design. Although OCL offers support for UML design, it is not enough for a proper validation as it lacks the strict notations of mathematics and logic that permits rigorous analysis and reasoning about the specifications. We conclude that the CPS UML design can benefit from using OCL constraints to improve model expressiveness, but the correctness and completeness of models require formal specification, especially on the dynamic part. We argue that all the static aspects presented in this paper in particular, and in CPS design in general, can be validated using OCL constraints exclusively. However, it is deficient in providing precise and unambiguous models of proposed software systems. Formal specification and verification of UML models for CPS applications is therefore required.

In this paper, we present how deployment artifacts can be formally expressed and verified. Additionally, this paper presents a research on using Z and PVS to specify and verify static aspects like the network infrastructure and the internal hardware used in CPS applications modeled using UML deployment diagrams.

Z provides a rigorous mathematical specification. However, Z is limited in handling dynamic aspects for the systems and it has no generally accepted verification tools. Therefore, we use Z to formally specify the static part of examples presenting reduced business logic complexity.

We consider PVS to overcome the deficiency in proper verification tools for Z. PVS demonstrate its advantages in case of more complex reasoning, involving specifications for large-scale CPS models based on wireless communication. From this perspective, Z involves a strong mathematical background, while PVS is closer to programming languages and object-oriented design.

As our future work will be focused on specifying and also validating the dynamic aspects of the CPS application design, we chose PVS formal specification and validation as a homogenous solution, applicable on both static and dynamic design characteristics.

References

Andersson, P., Host, M., 2008. UML and SystemC—a comparison and mapping rules for automatic code generation. *LNEE*, **10**:199-209. [doi:10.1007/978-1-4020-8297-9_14]

Andre, C., Cuccuru, S., Dekeyser, J.L., de Simone, R., Dumoulin, C., Forget, J., Gautier, T., Gérard, S., Mallet, F., Radermacher, A., *et al.*, 2005. MARTE: a New OMG Profile RFP for the Modeling and Analysis of Real-time Embedded Systems. DAC Workshop UML for SoC Design, p.16-21.

Aredo, D.B., 2002. A framework for semantics of UML sequence diagrams in PVS. *J. Univers. Comput. Sci.*, **8**(7): 674-698. [doi:10.3217/jucs-008-07]

Aredo, D.B., 2003. Formal Semantics of UML Statecharts in PVS. Proc. 7th World Multiconf. on Systemics, Cybernetics, and Informatics, Orlando, Florida, USA.

Aredo, D.B., Traore, I., Stolen, K., 1999. Towards Formalization of UML Class Structure in PVS. Research Report No. 272, Department of Informatics, University of Oslo, Norway.

Baar, T., 2005. Non-deterministic constructs in OCL—what does any() mean. *LNCS*, **3530**:32-46. [doi:10.1007/11506843_3]

Bhutto, A., Hussain, D.M.A., 2011. Formal verification of UML profile. *Aust. J. Basic Appl. Sci.*, **5**(6):1594-1598.

Bondavalli, A., Majzik, I., Mura, I., 1999. Automated Dependability Analysis of UML Designs. Proc. 2nd IEEE Int. Symp. on Object-Oriented Real-Time Distributed Computing, p.139-144. [doi:10.1109/ISORC.1999.776367]

Buratti, C., Conti, A., Dardari, D., Verdone, R., 2009. An overview on wireless sensor networks technology and evolution. *Sensors*, **9**(9):6869-6896. [doi:10.3390/s90906869]

Clavel, M., Egea, M., 2006. ITP/OCL: a rewriting-based validation tool for UML+OCL static class diagrams. *LNCS*, **4019**:368-373. [doi:10.1007/11784180_28]

Crow, J., Owre, S., Rushby, J., Shankar, N., Srivas, M., 1995. A Tutorial Introduction to PVS. Workshop on Industrial Strength Formal Specification Techniques.

Derler, P., Lee, E.A., Vincentelli, A.S., 2011. Addressing Modeling Challenges in Cyber-Physical Systems. Technical Report No. UCB/EECS-2011-17, Electrical Engineering and Computer Science Department, University of California, Berkeley, USA.

Dupuy, S., Ledru, Y., Chabre-Peccoud, M., 2000. An Overview of RoZ: a Tool for Integrating UML and Z Specifications. Proc. 12th Conf. on Advanced Information System Engineering, p.417-430. [doi:10.1007/3-540-45140-4_28]

France, R., Evans, A., Lano, K., Rumpe, B., 1998. The UML as a formal modeling notation. *Comput. Stand. Interf.*, **19**(7):325-334. [doi:10.1016/S0920-5489(98)00020-8]

Gavrilescu, M., Magureanu, G., Pescaru, D., Jian, I., 2012. Towards UML Software Models for Cyber Physical System Applications. Proc. 20th Telecommunications

Forum, p.1701-1704. [doi:10.1109/TELFOR.2012.6419554]

Gogolla, M., Richters, M., 1998. On Constraints and Queries in UML. *In*: Schader, M., Korthaus, A. (Eds.), The Unified Modeling Language. Physica-Verlag, Heidelberg, Germany, p.109-121. [doi:10.1007/978-3-642-48673-9_8]

Gogolla, M., Buttner, F., Richters, M., 2007. USE: a UML-based specification environment for validating UML and OCL. *Sci. Comput. Program.*, **69**(1-3):27-34. [doi:10.1016/j.scico.2007.01.013]

Hamie, A., Civello, F., Howse, J., Kent, S., Mitchell, R., 1999. Reflections on the object constraints language. *LNCS*, **1618**:137-145. [doi:10.1007/978-3-540-48480-6_13]

Kuzniarz, L., Staron, M., Wohlin, C., 2004. An Empirical Study on Using Stereotypes to Improve Understanding of UML Models. Proc. 12th IEEE Int. Workshop on Program Comprehension, p.14-23. [doi:10.1109/WPC.2004.1311043]

Kyas, M., Fecher, H., de Boer, F.S., Jacob, J., Hooman, J., van der Zwaag, M., Arons, T., Kugler, H., 2005. Formalizing UML models and OCL sonstraints in PVS. *Electron. Notes Theor. Comput. Sci.*, **115**:39-47. [doi:10.1016/j.entcs.2004.09.027]

Ledru, Y., 1998. Identifying Pre-conditions with the Z/EVES Theorem Prover. Proc. 13th IEEE Int. Conf. on Automated Software Engineering, p.32-41. [doi:10.1109/ASE.1998.732566]

Magureanu, G., Gavrilescu, M., Pescaru, D., Doboli, A., 2010. Towards UML Modeling of Cyber-Physical Systems: a Case Study for Gas Distribution. Proc. 8th IEEE Int. Symp. on Intelligent Systems and Informatics, p.471-476. [doi:10.1109/SISY.2010.5647314]

Magureanu, G., Gavrilescu, M., Tal, I., Toma, A., Pescaru, D., Jian, I., 2011. Generating OMNeT++ Specifications from UML Models for PSoC Distributed Applications. Proc. 6th IEEE Int. Symp. on Applied Computational Intelligence and Informatics, p.85-90. [doi:10.1109/SACI.2011.5872977]

Magureanu, G., Gavrilescu, M., Pescaru, D., 2012. UML Profile for Cyber-Physical System Wireless Communication Specification. Proc. 7th Int. Symp. on Applied Computational Intelligence and Informatics, p.383-388. [doi:10.1109/SACI.2012.6250034]

Nguyen, K.D., Sun, Z., Thiagarajan, P.S., Wong, W.F., 2004. Model-Driven SoC Design via Executable UML to SystemC. Proc. 25th IEEE Int. Real-Time Systems Symp., p.459-468. [doi:10.1109/REAL.2004.32]

Object Management Group (OMG), 2010. Documents Associated with UML Version 2.3 Specifications. Available from http://www.omg.org/spec/UML/2.3/ [Accessed on Dec. 26, 2012].

Riccobene, E., Scandurra, P., Bocchio, S., Rosti, A., Lavazza, L., Mantellini, L., 2009. SystemC/C-based model-driven design for embedded systems. *ACM Trans. Embed. Comput. Syst.*, **8**(4):1-37. [doi:10.1145/1550987.1550993]

Roe, D., Broda, K., Russo, A., 2002. Mapping UML Models Incorporating OCL Constraints into Object-Z. Technical Report No. 9/2003, Imperial College, London, UK.

Rousselot, J., Decotignie, J.D., 2009. A High-Precision Ultra Wideband Impulse Radio Physical Layer Model for Network Simulation. Proc. 2nd Int. Conf. on Simulation Tools and Techniques, Article No. 79. [doi:10.4108/ICST.SIMUTOOLS2009.5628]

Saaltink, M., 1997. The Z/EVES system. *LNCS*, **1212**:72-85. [doi:10.1007/BFb0027284]

Shroff, M., France, R.B., 1997. Towards a Formalization of UML Class Structures in Z. Proc. 21st Int. Computer Software and Applications Conf., p.646-651. [doi:10.1109/CMPSAC.1997.625087]

Spivey, J.M., 1992. The Z Notation: a Reference Manual (2nd Ed.). Prentice Hall International Ltd., Hertfordshire, UK.

Stoyanova, T., Kerasiotis, F., Prayati, A., Papadopoulos, G., 2009. A Practical RF Propagation Model for Wireless Network Sensors. Proc. 3rd Int. Conf. on Sensor Technologies and Applications, p.194-199. [doi:10.1109/SENSORCOMM.2009.39]