Abstract – The large variety of application domains and the large number of applications developed indicate the interest given in efficient and optimal designing of Cyber Physical Systems (CPSSs). The UML Profile introduced in this paper is part of a larger project, for an intuitive and easy to use programming model for CPS application design. It is defined to help designing wireless communication part of a CPS underlying network infrastructure. The UML approach allows a better structural and behavioral description of various parts of an embedded system. Models provide support for testing and validating such systems through simulation before deploying them on the physical environment. Benefits of the proposed UML Profile are presented in this paper using an intersection traffic management application as a case study.

I. INTRODUCTION

Sensors networks and more complex Cyber Physical Systems (CPSSs) have an increased potential in various fields of activity [1]. However, designing and developing networks composed of thousand of nodes, as part of CPSSs is not an easy task due the challenges that programming massively distributed structures implies. Considering the multiple types of applications based on CPSSs, an intuitive and easy to use programming model is required.

Using already defined components, the user of such a programming methodology will be able to design the network at structure level, by specifying the selected components and the relationships between them, without considering the hardware requirements and limitations. The developer of these components has to consider these aspects. Moreover, the user should have the possibility to simply specify the goals for the corresponding subsystems and the system will handle the actual implementation of components solving the desired goals.

This programming methodology involves some well-defined steps. To accomplish that we propose two UML Profiles as one for hardware description and other for software description of the CPSS application.

The first UML Profile, which covers the basic aspects of describing the structure of different types of nodes, the connections between them, their customization and limitations, is discussed next. This UML Profile is designed for component developers. It allows them to define the library of components that regular designers can use when constructing their CPSS, based on application requirements. By transitivity, the usage of the UML Profile helps users define Platform Independent Models (PIMs) for network topology and also for the types of nodes contained in the network.

A first version of the UML Profile for hardware specification was introduced in [2]. In that version stereotypes were defined for CPS applications where the nodes are connected using wires. The defined stereotypes, along with the tagged values and constraints were used to customize the UML models corresponding to the PIMs of the applications. In this paper, we extend the UML Profile with stereotypes for wireless communication. These stereotypes were created using as model the MiXiM project for wireless sensor networks [3].

The other UML Profile, which involves defining artifacts for expressing the desired behavior of the application, is defined in conjunction with the UML Profile for nodes and network topology specification. This Profile contains a set of stereotypes which support the goal-oriented approach, as introduced in [4] and [5]. They enable specifying the levels of abstraction corresponding to the application, the reflection of the goals from higher levels of abstraction to lower levels of abstraction and the implications to higher levels of abstraction generated by goals at the lower levels.

The PIMs, which are part of the Model Driven Architecture (MDA) approach, can be transformed into several Platform Specific Models (PSMs), depending on application requirements and user specifications. Simulation of embedded systems applications at network level is desirable before deployment on hardware devices.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Section 3 introduces the new programming model for CPSSs and describes the UML Profile definition for wireless connections. Section 4 presents an example of using the proposed UML Profile. The last section presents some conclusions and proposes future work.

II. RELATED WORK

Literature presents different approaches for designing and programming CPSS networks. Some of the most well known attempts in programming wireless sensor network will be briefly presented in this chapter, underlining both their strong and weak points in regards to the MDA approach proposed by the authors.
A complex proposed model represents years of research in improving the system-on-chip and embedded system design, by unifying the capabilities of UML and SystemC/C to operate at system level [6]. The authors use MDA for reducing abstract and coarse-grained PIMs to concrete and fine-grained PSMs. They define a design methodology and development flow for the hardware part, by specifying a SystemC related UML Profile at different levels of abstraction. The hardware UML Profile describes stereotypes for SystemC components. Separately, a multithread C UML Profile for modeling software applications is defined. The software UML Profile relates to the hardware UML Profile and acts as business logic descriptor for the elements introduced in the hardware section. However, although authors give detailed explanations for the design flow, they leave out the OCL constraints, which complete stereotypes definitions. OCL constrains are of highly importance in an UML model as they allow the verification of the defined model in accordance to the used stereotypes. This verification can be made using an OCL general validation tool already defined or by defining a specific validation tool that parses only a part of the OCL language, directly involved in defining OCL constraints in a UML Profile. The programming model for CPSs defined will use the latter variant.

Transformations from UML models to SystemC have been considered in other papers also. For example, the authors propose in [7] an automatic code generation method related to hardware interconnections, based on mapping rules between UML hardware Profile and SystemC hardware descriptors. The UML hardware Profile defined in [7] is similar to the one defined in [6]. The contribution for the authors is represented by the automation of the hardware interconnection specifications, in order to minimize the design effort. However, the initiative for hardware-software co-designs in [6] already covers such aspects.

Reference [8] made an attempt to describe the communication infrastructure and timing features of SystemC using high-level UML modeling. Communication primitives from SystemC, like interfaces and channels, were stereotyped. Clock sensitivity and timing constraints were also considered. Class diagrams and state diagrams were used for capturing the internal behavior of embedded modules. The programming methodology the authors report, allows describing executable platforms at UML level and also translation of applications described in UML to SystemC level. However, a major drawback of the proposal in [8] is the fact that user must manually define a top level class for instantiating objects of already defined types. This has the effect of constraining the user behavior when specifying an application. Another drawback is that the model allows only one level of nesting in state diagrams.

The models presented above have in common the specification of hardware configurations for the types of nodes in a network and the communication between nodes in UML Profiles. The validation for the UML artifacts is made using automatic code generation for SystemC simulator. Although it is a promising solution for simulating embedded systems applications, the usage of SystemC has some disadvantages. Some relevant issues were presented in [9]. SystemC was designed for single host simulation. All threads are created in the same simulation process assigned by the operating system. The created simulation threads run concurrently with the other threads in the system: operating system threads or other applications’ threads. Synchronization and communication between modules depend on the threads priority; therefore depend directly on the processor speed. This implies an important drawback for using SystemC in simulation distributed embedded systems: the duration of the simulation cannot be estimated. A way to eventually control the duration is to increase the priority of the simulation thread. However, this implies another programming effort. The lack of performance in case of simulated distributed applications on single thread simulation kernel is obvious when simulated applications become more complex. The drawbacks for SystemC led to another choice regarding the environment in which distributed applications will be simulated. Also, the programming model based on MDA approach is not close to SystemC model.

In [10], the authors propose an UML 2.0 Profile for Embedded System Design. This Profile defines stereotypes and design rules for application, platform and mapping. The profile classifies application and platform components, at the same time enabling their parameterization. The application is seen as a set of active classes with an internal behavior, UML modeled. The platform is seen as a component library with a parameterized presentation in UML 2.0 for each library component. This approach of parameterized component library is similar to the proposed design methodology, where a middleware that recognizes the types of components used have to be developed. The middleware should ensure the correct functionality of the goals described by the users. Library functions will also be made available, in order to help defining custom applications. The library functions should ensure general application type specific requirements and constraints. The user will specify custom application goals and constraints. Then the middleware correlated with the specific application library should be able to handle them.

III. UML PROFILE FOR WIRELESS CONNECTIONS

As stated in the first section, the increased interest in CPS applications determines the need for an efficient, intuitive and easy to use programming model in which the users effort to be minimal. The best programming methodology is a visual one in order to help the users to define their applications in an easier and more intuitively way.

From the users perspective, they will be able to use different types of already defined UML components for designing the network. The users will customize the components depending on the application needs. Alternatively they can compose them into more complex components, taking into consideration communication and links between components, along with others limitations and requirements.
The components and connections between them form the hardware specifications for an application. For the software specifications, the users will declare the behavioral requirements for the applications in form of goals at a high level of abstraction in the system. Then, these high level goals will be translated to a lower abstraction levels.

The users can test and validate their defined UML model for the types of node in the network, customized based on the application requirements, the UML model for network topology and the high-level software specifications, in form of application goals. To accomplish that they will generate specifications used for a simulation environment. These specifications regards both hardware and software configuration of the network. Checking the OCL constraints will validate the correctness and completeness of the user UML model. Those constraints was previously defined and attached to UML models. In case of success, the users can test their applications in a simulation environment that will allow them to verify the defined network and goals for the applications. In case testing and validation is proven unsuccessful, the user must return to the high-level UML specifications and correct them. After several steps from UML modeling to testing in simulation environment, if necessary, when the specifications for the applications are validated, the user can deploy the network in a physical environment, having the certainty that hardware modeling errors and at least some behavioral errors have been already resolved.

This means that the visual programming model which will be proposed must be able to generate simulation code for validation of the desired application: hardware validation implies network topology, the nodes contained in the network and the communication between them; software validation implies the goals for the application, the requirements and limitations translated from UML high-level modeling. After the validation of the application in simulation environment, the programming approach must generate efficient business-logical code that can be loaded on the nodes. The physical network should be able to collaborate with the generated code, and should be as close as possible to the desired behavior. The generated code must consider the hardware limitations of the physical components that define the network. The specific application library must help the visual programming model in generating the necessary drivers for the hardware components of the network.

When discussing the components developers’ perspective, for high-level UML modeling, some UML Profiles must be first defined. An UML Profile for defining the hardware for CPS network components was introduced in [2] and extended for wireless connections in the present paper. The UML Profile allows defining particular standard elements based on stereotypes. Another set of stereotypes will define node specific requirements like localization, type of communication, and so on. These defined stereotypes contain specific tagged values and constraints, for expressing the customizations required for the further stereotyped UML elements.

Well-defined stereotypes for specifying the hardware components of a network help achieving a clear separation of devices and grouping into families of hardware devices. The wireless communication model is inspired by MIXIM project. The UML defined stereotypes for the UML hardware Profile are presented in Fig. 1 and follow the main ideas presented in MIXIM [3]. MIXIM_PredefinedUnit_HWST abstract stereotype is used for grouping the stereotypes used for customizing the components similar to the one in MIXIM project. BaseLayer_HWST stereotype is a generalization for BaseMacLayer_HWST and BaseNetwLayer_HWST stereotypes, used for customizing component units for basic nodes. ConnectionManager_HWST stereotype is used for describing the module that coordinates the connections between nodes and handles dynamic gate creation. BaseWorldUtility_HWST stereotype customizes a module that provides utility methods and information used by the entire network, as well as simulation wide black board functionality. BaseBattery_HWST stereotype is used for customizing the battery module for a node. BaseMobility_HWST stereotype is used for customizing the mobility module for a node. This module handles the physical localization for the node. BlackBoard_HWST stereotype is used for customizing the module that provides black board like information exchange between the other modules of a host. Connections_MIXIM_PIM stereotype provides the customizations for the connections for a module of a node to the upper layers, the lower layers of with the exterior of the node.

Some stereotypes regarding module interfaces were also defined: ModuleInterface_PIM stereotype is used in case compound nodes contain sub modules that implement a certain interface, instead of being an instance of a certain simple or compound unit. At the same time, simple modules can implement an interface, instead of extending other modules. INic_HWST stereotype is used for customizing an interface for network interface card. IBaseMobility_HWST stereotype is used for customizing an interface, which is implemented by modules responsible
with mobility in a node. \textit{IBaseCommunicationUnit_HWST} stereotype is used for customizing the interface that must be implemented by communication units in a network. \textit{IBaseNetwLayer_HWST} stereotype is used for customizing an interface for network layer modules. \textit{IBaseApplLayer_HWST} stereotype is used for customizing an interface for application layer modules. \textit{IBasePSoCLayer_HWST} stereotype is used for customizing an interface used for the PSoC layer for an application. \textit{IBaseGoalOriented_PSoCLayer_HWST} stereotype is used for customizing an interface used for the PSoC layer for goal-oriented applications. The stereotypes for interfaces must contain also the possible connections for modules which implement these interfaces, expressed by the following stereotypes: 
\textit{RadioDirectSendConnection_MIXIM, LowerToUpperConnections_MIXIM, UpperToLowerConnections_MIXIM}.

Another UML Profile will be defined for software specification of CPS applications. It will contain a set of stereotypes which define the goal-oriented approach, by specifying abstraction levels to which the node belongs to, the reflection of the goals from higher levels to lower levels of abstraction, and the extension of the goals from the lower levels of abstraction in a collection of goals derived from the higher levels of abstraction.

Studies have indicated that using UML defined stereotypes for specifying different requirements for applications help improving the overall understanding of the models in question. Such a study is detailed in [11].

Using the defined UML hardware Profile, it is possible to start a distributed embedded application design with clear specifications regarding network topology, the types of nodes participating in its construction and the hardware units involved in specifications for the families of nodes used. It is the task of the developers of the components to define customization for the possible types of nodes, for the network topology and other hardware units based on UML hardware Profile. These customized components are placed in a library that can be accessed by the users in order to define their applications.

In order to achieve this design methodology, a middleware that recognizes all types of choused components is required. The middleware should ensure the correct functionality of the hardware components of the network and the software components, in the form of goals to be completed by the users. The library functions are available to the users, in order to help defining custom CPS applications. The library functions should ensure general application type specific requirements and constraints. Users will insert the custom specific hardware requirements, along with application goals and constraints. They will add also the middleware correlated with the specific application library functions designed to handle them.

The developers of the predefined components should ensure the verification of the users defined applications in a simulated environment. They must provide the tools for checking the OCL constraints for the UML user defined models.

Following validation of the UML models, a code generator will ensure the generation of specifications for hardware and software specifications in the simulation environment. After validating the simulated application, the visual programming model should be able to generate executable code, which can be deployed on the hardware network.

The proposed method uses a MDA approach for the design of CPS applications. MDA was proposed as a software design approach, presented in form of a set of standards intending to conduct to a “model-based, standards-driven, and tool-supported” [12] handling of applications development. The required steps to be accomplished by the developers of the components and by the users that define the applications can be translated into the models described by the MDA approach.

A Computation Independent Model (CIM) is specified in the visual programming model. It indicates a certain system in the environment where it will evolve, without giving any details about the specific implementation. It defines the constants, simple and continuous variables of the network topology and requirements abstractions. It expresses all the values of simple and continuous variables of the network at different levels of abstraction, at a certain moment of time.

PIMs are defined using the already discussed UML Profiles indicating the network topology defined by the user, the expression of the functional and non-functional requirements, and the types of nodes required by the user. The objective of PIMs is to correlate all the desired hardware and software functionalities and to deal with them at an abstract level, without taking into consideration the hardware limitations inducted by the actual physical components. At this time, the stub should be able to be compiled by the visual programming model at a static level in order to detect the lack of correlations between goals and network topology, or between goals themselves, expressed in OCL constraints.

Several PSMs will be defined to evaluate the already defined PIM against the physical requirements and limitations of the specified hardware to be used. At this level, PSMs should be able to show the impact of physical hardware over PIM.

A possible and recommended PSM is a simulated environment, which allows testing and validation of PIMs before deployment in a real world network. Simulation must be as realistic as possible in order to obtain best results from simulating a network before deployment in a physical environment. CPSs can consist of distributed devices, each of them having its own internal clock and being able to operate at a different clock frequency than another device. Although there are clear specifications for each component of CPSs, the main issue in designing this type of systems resides in the management of complex interconnections. Dynamic aspects for the interconnections increase the difficulty of the stated problem. A recent solution for handling dynamic aspects of CPSs is based on Programmable System-on-Chip (PSoC) devices [13].
Using PSoC technology for designing CPS applications does not solve all the problems. There are still open issues regarding a correct temporal semantic for all concurrent processes involved [14]. Synchronization mechanisms must be implemented at communication level, in order to achieve cooperation between devices. Some solutions regarding issues related to simulation of PSoC based CPSs are given in [15], [16], [17] and [18].

IV. CASE STUDY

As a case study, we discuss the utility of the proposed UML Profile for wireless networks using a small CPS for an intersection traffic management.

The system depicted by Fig. 2 consists of four traffic light nodes, controlled by a decision node, which must compute optimal green color duration for each traffic light node. For simplicity, the traffic light nodes are considered to be working in pairs, and therefore, at a certain moment only two green color durations must be computed, and not all four.

Each traffic light node is connected to a sensing node and it commands the time when the sensing node starts counting the number of cars waiting at the red color for the semaphore. The resulting values for the sensing node are computed using image handling algorithms, using as inputs the video cameras placed on the sensing nodes.

Traffic light nodes forward the sensed values to the decision node. The latter interprets the received data and computes the next optimal green color periods for intersection directions, respectively.

The decisions taken on the decision node can favor a certain direction, considered having a higher priority in traffic or can be influenced on the number of cars waiting at traffic lights red color.

For designing such application at hardware level using UML high-level modeling, it is necessary first a library of possible components, from which the user will choose the most suitable components his application. Using predefined components, the user can also create customized nodes, which are valid as long as the hardware and software requirements for the nodes are completely satisfied.

All three types of nodes present some similarities. For example, communication with the exterior of the node is made using a port named radioIn, stereotyped with InputGate_HWST, which means this is an input port. All types of nodes contain a mobility component, which specifies the x, y and z coordinates of the node position in geographical space. Also, the nodes contain a BaseArp component, which is a module responsible with address resolution.

Each node contains a component responsible with the application part for that node; in particular it must implement the required behavior for the type of node. The communication for each node is ensured by a component which implements a base wireless communication unit predefined interface.

Each of the presented types of nodes is composed of several units. Each of these units is an instance of a predefined component, customized depending on application needs. In case it is required by OCL constraints that also the instances of the components to be stereotyped, on the UML representation for a certain instance will appear both used stereotypes: the one extending Node (Deployment) metaclass and the one extending Instance Specification metaclass.

Such an example, visible in Fig. 3, is the sensingUnit, which is an instance of the defined VideoSensingUnit. The stereotypes here are SensingUnit_HWST for Node (Deployment) and SensingUnit_HWST_Instance for Instance Specification.

V. CONCLUSIONS

An efficient, intuitive and easy to use programming model is desired for massively distributed embedded systems as CPS are. It is required that even designers without advanced knowledge about sensor networks design methodologies to be able to specify applications of CPSs at high level of design.
Designers will command and control such applications by specifying the network topology and component nodes, at hardware level, and by specifying the goals at software level, without taking into consideration the actual limitations imposed by the physical environment.

To accomplish that, designers will be provided with a library of predefined components, which contains different types of units for creating several types of nodes. These types of nodes are the ones required in the types of applications covered in the high level UML modeling. The predefined components can be used as they are, or can be connected into several composed nodes, with customized functionalities.

For the library of components, the developers will use the UML Profiles created for hardware and software specifications. The stereotypes proposed in the UML hardware Profile customize the physical requirements and limitations for simple and compound units, while the stereotypes in UML software Profile will customize the behavior for each component. The description of the UML hardware Profile was initiated in [2], and completed by this paper with stereotypes for wireless communication.

As a future work, we plan to continue research with automated platform code generation. Along with the extended UML Profile with artifacts for software specification, the next steps imply customization for distributed embedded applications designed with PSoC devices.

Another intention is to develop an automated code generator to help the user in generating code for simulation and for deployment in PSoC Creator.

Other future objectives refer to validating the UML artifacts defined for applications of CPSs in physical networks.

ACKNOWLEDGEMENT

This work was partially supported by the strategic grants POSDRU/88/1.5/S/50783 Project ID50783 (2009), POSDRU/107/1.5/S/77265 Project ID77265 (2010) and POSDRU 107/1.5/G/13798 (2010), co-financed by the European Social Fund – Investing in People, within the Sectoral Operational Programme Human Resources Development 2007 - 2013.

REFERENCES


