Extensions to the UML Profile for MARTE for Distributed Embedded Systems

Emad Ebeid
Dep. of Engineering
Aarhus University, Denmark
esme@eng.au.dk

Julio Medina
Dep. of Computer Science and Electronics
University of Cantabria, Spain
julio.medina@unican.es

Davide Quaglia and Franco Fummi
Dep. of Computer Science
University of Verona, Italy
{davide.quaglia, franco.fummi}@univr.it

***Abstract*—The design of distributed embedded systems is a challenging task that requires raising the level of abstraction to handle the different involved concerns. In particular, standard modeling languages and precise semantics specification are necessary to address the networking-related aspects at a high level of abstraction. The Unified Modeling Language (UML) and its MARTE profile are valid formalisms to model real-time embedded systems but they lack precise modeling elements when addressing applications and platforms forming distributed embedded systems. In this work, we formalize a coherent set of modeling elements for the design and deployment of distributed embedded systems. A novel UML profile for networking is proposed as a semantic and syntactic extension to the UML Profile for MARTE: The *Network Profile*.**

Keywords–Model-Driven Design, Network, UML Profiles, Deployment Diagram, Stereotypes, Modeling Languages, Internet of Things, Smart Systems.

I. Introduction

Distributed Embedded Systems (DESs) are finding their way into a growing range of applications such as environmental monitoring, health, sport and fitness, to cite only few. They are distributed applications of networked embedded system which are special-purpose, resource-constrained nodes interacting together through communication protocols to achieve a common goal. Today the design of DESs is becoming more and more complex, involving many different tasks spread over hundreds or thousands of heterogeneous network nodes connected through different types of channels and protocols [1]. Figure 1 shows an example of the basic entities used to describe DES at the level of abstraction used in this work. The design of DESs does not only depend on the technical specifications of their components but also on the nature of the environment in which the network will be installed. The environment can greatly influence the data transmission especially if unguided transmission media (e.g., radio links) are used. The quality-of-Service (QoS) of such transmission is the major factor to achieve a significant quality of experience and it depends on the surrounding communication environment [2].

Abstraction and Platform-Independent Models (PIM) become a solution to cope with design complexity. High-level modeling languages such as UML [3] are considered as a platform-independent modeling language and UML in particular is widely accepted in the software engineering community as a common notational framework. The UML extension mechanism (i.e., the profile) is a feature that adds more modeling elements with semantic details to its diagrams and modeling power to be closer to specific applications. MARTE [4] (Modeling and Analysis of Real-Time and Embedded systems) profile is well-known in the embedded systems community. This standard extends the features of the Schedulability, Performance and Time Analysis (SPT) [5] profile, and replaces it when used with UML 2.x. MARTE provides constructs to represent more easily the kinds of timing and performance artifacts that are needed to model real-time embedded systems [6].

This work has been partially supported by the EU large-scale integrating project CONTREX (FP7-2013-ICT-10-611146).

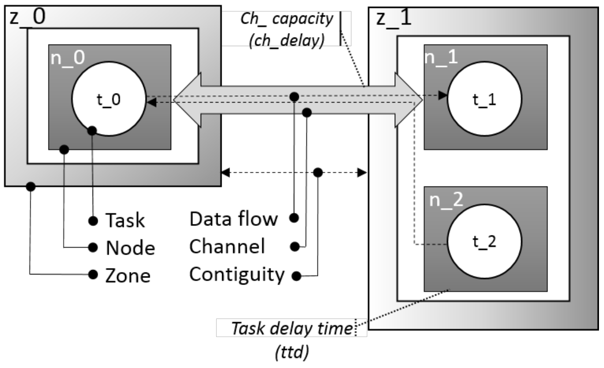

Figure 1. The structural view of a distributed embedded system.

The Quality of Service and Fault Tolerance characteristics and mechanisms profile (QoS&FT) [7] is another standard related to embedded systems. It covers real-time issues with some capabilities on communication policies and latency. But QoS&FT promotes the definition of domain specific languages, an approach to model extra-functional properties differently than the one adopted in MARTE [8].

Nevertheless, modeling of distributed embedded systems and specially the modeling of the performance of networks among embedded devices has not been supported directly by any standard UML profile. Even MARTE still lacks precise modeling elements regarding the description of general purpose networks.

Outside UML, networks have been modeled by using hybrid automata [9]. Hybrid automata are a general approach to model both discrete and continuous-time processes. Some other languages/tools go in the same direction such as SysML [10], Matlab/Simulink, Ptolemy [11], and Compositional Interchange Format [12].

This work aims at proposing a standard and complete

UML/MARTE representation of DES to be used for design space exploration. As described in the design flow in [13], the UML presentation will be used to generate configuration alternatives and, for each of them, simulation scenarios to find the best solution. This goal is achieved through the following contributions:

- The formalization of a coherent set of modeling elements for distributed embedded system;
- The enrichment of the UML MARTE profile regarding network modeling;
- A novel UML profile (*Network Profile*) used to experience in practice the proposals for enhancements of the MARTE profile.

This paper is structured as follows. Section II gives an overview about embedded systems modeling languages and the computational model which are used in this work. Section III defines the DES elements and formalizes them. Section IV explains the model views of such systems, while Section V describes our extension to MARTE and Section VI presents a home automation application modeled with it. Finally, Section VII draws some conclusions.

II. Background

This section gives a brief overview of UML, the MARTE profile, and a model of computation named UNIVERCM, which are used to model and analyze DESs in this work.

A. UML

UML [14] is a standardized general-purpose conceptual modeling language exploited mainly in the field of object-oriented SW engineering and information systems. Since its introduction it became the de-facto standard for modeling SW intensive systems.

UML 2.x includes as modeling formalisms 14 types of diagrams divided into two categories. Structural diagrams emphasize the components (and kinds of components) that are in the system (e.g., Class, Deployment and Profile diagrams). Behavioral diagrams emphasize what must happen (or how) in the system (e.g., State machines, activities, and sequence diagrams). In our modelling approach we use the Deployment Diagram, its elements are described in section IV to relate them to concrete elements of interest in the DESs entities model, which is presented in section III.

UML includes the possibility to extend its modeling power by a generic extension mechanism named Profile. Profiles are defined using stereotypes, attributes and constraints that are applied to specific model elements to modify or bring entirely new semantics to them.

B. MARTE

MARTE [4] is a UML profile standardized by the Object Management Group (OMG) [15]. It provides support for specification, design, and verification/validation stages in the development of real-time and embedded systems. It is organized in several sub-profiles (units), each with a predefined role in the set of modeling needs that its potential users may have (see sections 6.2.3 and 2.4.1 in [4]). The units of relevance for

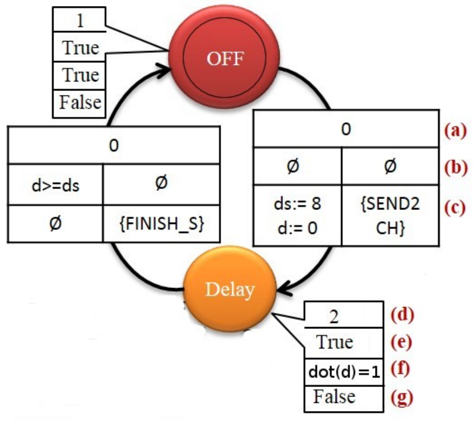

Figure 2. UNIVERCM representation of a *sensor* task.

this work are those in the Performance Analysis Compliance case and the Use Cases for the Analysis Methodology Provider actor. A particularly relevant capability in MARTE is the aid in the definition and manipulation in UML of Non-Functional-Properties (NFP), which help to describe the "fitness" of the system behavior or the amount of available resources (e.g. performance, memory usage, power consumption, etc.). The Architecture Analysis and Design Language (AADL) [16] is also extensible and may serve the purpose of simulation, but it has very few reliable tools for doing transformations. Being MARTE the official UML profile for AADL and being the OMG an open standardization body the effort presented here targets UML and MARTE as the basis for its extensions.

C. UNIVERCM

UNIVERCM [17] is a Model of Computation (MoC) capable to describe discrete and continuous systems resulting from software and both analog and digital HW components, within a unique, well defined, mathematical framework, and capable to support the integration of novel and already defined components in a mixed top-down and bottom-up flow.

Figure 2 shows an example of a UNIVERCM automaton for a sensor task of a DES which generates data periodically. The automaton has two states, e.g., OFF (initial state) and DELAY, and two transitions. Both states and transitions have a priority (denoted by (a) and (d) in the Figure) for sorting their execution. States are characterized by three predicates, i.e., invariant condition(e) to remain in this state, continuous-time evolution law (f), and atomic condition(g). Transitions are enabled by an expression on variables and a set of labels (b), e.g., the transition can be traversed only if the expression is satisfied and the labels are enabled. When the transition is traversed, the variables inside the updating function are updated and outgoing labels are activated (c). Sensor automaton starts by checking the lowest priority condition (b) first and then goes to DELAY state. During the transition, the outgoing label $\{SEND2CH\}$ is activated, the discrete variable ds is set to 8 and the continuous variable d is initialized. In the DELAY state, the delay counter d increases linearly according to time as represented by the corresponding evolution law. When the counter reaches the sensor delay time ds, the guard $d >= ds$ is satisfied, the automaton activates $\{FINISH\_S\}$ label and then it goes to OFF state. In fact, the automaton remains in state

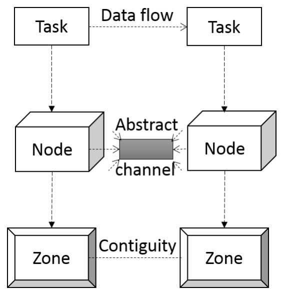

Figure 3. Network elements and relationships

DELAY to reproduce sensor processing time ds. A complete description of UNIVERCM is available in [17].

III. Formalization of Distributed Embedded System

This section provides some formal bases to describe distributed embedded systems with the intention to simulate their networking operation. It defines the necessary entities in their structural and behavioral views, as well as the notation to refer to them in this work.

Definition 1: DES structural view allows to represent the skeleton of the network, which complements its behavior. It deals with parameters that remain constant during the design process and control its behavior. A DES is mainly composed of topologies (T).

DES topology (T) is a combination of *entities* (E) and *relationships* (R). Figure 3 shows the elements which compose the network along with their relationships (detailed explanations are in the following sections).

A Distributed Embedded System (DES) consists of the following entities:

1) *Task (t)*, represents the functional part of the application which can be periodic or aperiodic and can be data producer or consumer.
2) *Node (n)*, represents the physical element of the network which will host one or more tasks to run on it. It entails a hardware component that has processing unit, memory and at least one network interface.
3) *Zone (z)*, represents a partitioning of the physical environment in which the DES is deployed. It groups nodes and defines their position. Furthermore, it captures the relevant environmental parameters such as room temperature in a temperature monitoring application.
4) *Abstract Channel (ac)*, represents the communication link between one or more entities of type node (n). An ac can be wired or wireless, and it defines the characteristics of the channel (i.e., delay, capacity. error rate).
5) *Data flow (f)*, represents the communication needs between two entities of type task (t). f synthesizes this communication in terms of throughput, maximum delay (latency) and error rate.
6) *Contiguity (c)*, represents the relationship between two entities of type zone (z). c captures the environmental characteristics between two zones which affect inter-zone communication.

Figure 1 shows an example of 3 nodes allocated in 2 zones and interacting via a channel. The so called task delay parameter (ttd) is set in this view and represents the duration of a task.

Definition 2: DES behavioral view represents the dynamic aspects of the network and complements its structure. It is fully described by a UNIVERCM automaton consisting of States (S) and Transitions (ϕ).

The global behavior of a DES is the *parallel composition* of the automata of its entities. Its state can be represented as an ordered set of the state values of each component entity. The behavioral aspects of DESs entities such as Tasks (t) and Abstract Channels (ac) can be described by UNIVERCM automata while zones, nodes, and data flows are not associated to behavioral aspects. Automata can represent periodic or aperiodic task behavior (e.g., sensors and actuators) and channel behavior. UNIVERCM hybrid automata (Section II-C) are adopted because we aim at representing both discrete events and the evolution of time as shown in the example shown in Figure 4. It starts from the *reset* state and checks the transitions conditions based on their priorities. If {*en_tx1*} label is enabled, the automaton resets the delay counter tx_delay and goes to the *delay* state. In *delay* state, the automaton remains to represent the sensor processing time ttd. When tx_delay reaches its boundary value ttd, the guard condition $tx\_delay = ttd$ is satisfied so that the automaton activates the {*intr_tx1*} label and goes to the *transmit* state. The value of ttd is set in the *structural view* (see Definition 1).

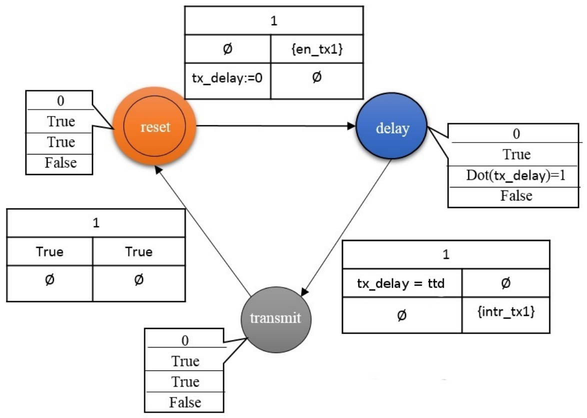

Figure 4. UNIVERCM automaton of a Sensor task.

A DES can be described as a triplet: its topology, its states, and its transition functions. The topology is a tuple itself: entity plus connection. An entity is defined by its type, which are a set of elements (i.e., zone, node, task). The connection structure consists of a set of relations between entities (i.e., channel, data flow, contiguity).

The definition is as follows:

Distributed Embedded System (DES) = ⟨T, S, ϕ⟩

$$\begin{aligned} &Topology \quad T = \langle E, R\rangle \\ &Entity \quad E = \{z, n, t\} \\ &Relation \quad R = \{f, ac, c\} \\ &Data\ flow \quad f = [QoS, task\_tx, task\_rx] \quad \in \mathscr{F} \end{aligned}$$

where: $QoS \in \mathscr{QS}$ $\quad task\_tx, task\_rx \in \mathscr{T}$

$Abstract\ Channel \quad ac = [distance, QoS, cost, mobility, N] \in \mathscr{AC}$

$Contiguity \quad c = [Rs] \quad \in \mathscr{C}$

where: $Rs \in \mathscr{R}$

$Task \quad t = [mobility, cr] \quad \in \mathscr{T}$

where: $mobility \in \mathbb{B} \quad cr \in \mathscr{CR}$

$Node \quad n = [cost, mobility, power, t] \quad \in \mathscr{N}$

where: $cost \in \mathbb{R} \quad mobility \in \mathbb{B} \quad power \in \mathbb{R} \quad t \in \mathscr{T}$

$Zone \quad z = [c, n] \quad \in \mathscr{Z}$

where: $c \in \mathscr{C} \quad n \in \mathscr{N}$

$Quality\ of\ Service \quad QoS = [delay, error\_rate, max\_throughput]$

where: $delay,\ error\_rate,\ max\_throughput, \in \mathbb{R}$

$Resistance \quad Rs = [added\_delay, added\_error\_rate, residual\_throughput]$

where: $added\_delay,\ added\_error\_rate, residual\_throughput \quad \in \mathbb{R}$

$Computational\ Requirements \quad CR = [CPU, memory\_size]$

IV. Modeling of Distributed Embedded Systems

In order to get a standardized representation of the formalization proposed for DESs we head for the conceptual modeling of its separate concerns. Model-driven design and, in particular, UML profiles such as MARTE [4] and SysML [10] have been proposed to introduce all the information required in this stage of the design process. The use of standard languages enable tool interoperability and the generation of widely understandable documentation. This section describes the UML elements and their extensions used to model DES communication needs. They are presented as a UML profile for modeling DESs together with their environment.

Modeling DESs in general involves structural as well as behavioral UML diagrams. The network configuration can be captured by using Class, Component or Deployment diagrams, while the behavior of its elements can be modeled as State Machine, Activity, or Sequence diagrams. The extensions here proposed are meant for doing simulation of communications needs. From this perspective behaviors are abstracted away, keeping only the networks usages, and expressing them as dataflows on abstract channels in UML *Deployment*. This approach results in a formalism semantically closer to our formal model. To model regular embedded systems elements as well as the precise behavioral aspects of DESs and their allocations, plain UML and eventually *MARTE* would be sufficient. In particular, using sequence charts would make it easier to transpose them into UNIVERCM automata (see section II-C). A new MARTE-based profile (named *Network Profile*) is used for modeling the simulation oriented communication environment. Details about the new profile are in the following section. The UML Deployment Diagram represents the network configuration in a static view. Figure 5 shows the syntax elements of the Deployment diagram; numerical labels are used to show the items to be explained.

The UML Deployment diagram elements used are:

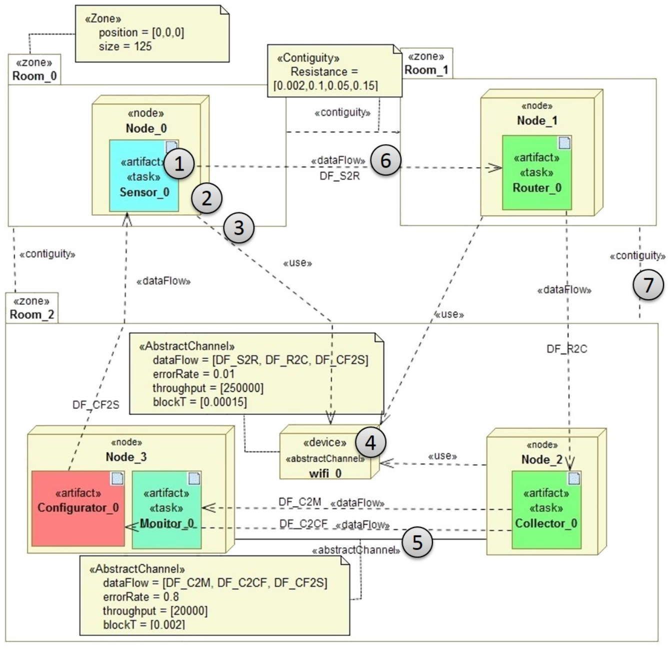

Figure 5. An example with syntax elements of Deployment Diagrams

1) **Artifact**: is a software component, such as executable SW component, files or libraries, deployed inside the Node. Artifact is used to model the `Task` element (see 1).
2) **Node**: is a physical object that represents a computational resource of the system, such as HW nodes, computers, or servers. Node is used to model the `Node` element (see 2).
3) **Package**: is a group of elements and it provides a name-space for them; it can represent groups of nodes in a given zone of the space. Package is used to model the `Zone` element (see 3).
4) **Device**: is a type of node in the system that represents a physical computational resource, such as a wireless access point. Device is used to model the `Abstract channel` element from type wireless (see 4).
5) **CommunicationPath**: defines the path between two nodes that are able to exchange signals and messages such as a wired/wireless communication channel. CommunicationPath is used to model the `Abstract channel` element from type wired (see 4).
6) **Dependency**: is a relationship between model elements that require other components for their specification or implementation; it can represent a data flow between network elements (i.e., tasks). Dependency is used to model the `Data flow` element (see 5).
7) **Association**: is a relationship between classifiers that is used to show that these classifiers are linked together or logically combined. Association is used to model the `Contiguity` element (see 6).

V. Network Profile

The proposed *Network Profile* formally extends the standard UML 2 metamodel, which is the basis for the vast majority of standard UML profiles, and specializes by inheritance the MARTE profile. The profile defines a number of stereotypes to represent DES elements with UML metaclass elements.

The main intent of this profile is to compensate the lack of specific elements in UML and MARTE with the precise semantics to describe the domain of interest for DESs elements.

A. Approach and Structure

This profile is structured around two main concerns, the *Workload*, which models the behavioral aspects of DESs elements (e.g., task), and the *Resources*, which model the DESs device library (e.g., nodes and channels). This approach is consistent with the Generic Quantitative Analysis Modeling (GQAM) sub-profile in MARTE, though the level of abstraction at which the deployments are describe includes also elements from its High Level Application Modeling (HLAM) sub-profile. The proposed extensions do not restrict the modeller from using any other valid annotation like analysis results or any other NFP. No additional NFP libraries have been needed.

B. Profile Elements Description

In this sub-section, we describe the semantics of the *Network Profile*.

1) Workload: This package offers the concepts related to the processing load of the DESs elements (Figure 6). It imports stereotypes from the HLAM MARTE sub-profile.

a) «Task»: It extends the Artifact UML meta-class and inherits the attributes and associations from the real-time unit (RtUnit) element of MARTE. It is described as:

Extensions

- Artifact (from UML::Artifact)

Generalization

- rtUnit (from MARTE::MARTE_DesignModel::HLAM)

Associations

- dataFlow: DataFlow [*]
  Data flow that is required to connect two tasks.

Attributes

- requiresMobility: Boolean[1]
  specifies the task mobility requirement (fixed or mobile task).
- CPU: Real[1]
  specifies the required CPU.
- isPeriodic: Boolean[1]
  specifies the task type (periodic or aperiodic task).

Semantics: A task represents a basic piece of functionality of the whole application. Tasks can be HW components or SW processes. For example, sensors and actuators. It may require mobility from the hosting node, and resources such as memory and CPU.

2) Resources: This package offers the concepts that are necessary to model a general platform of DESs elements. For example, nodes and channels. Figure 7 shows the elements in this package. It imports stereotypes from GQAM and GRM MARTE sub-profiles.

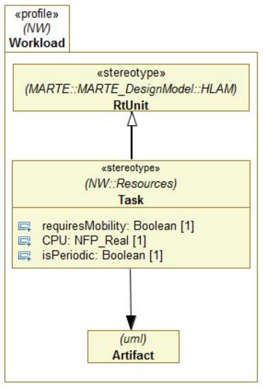

Figure 6. The Workload package of *Network Profile*

a) «Node»: It extends the Node UML meta-class and inherits attributes and associations from GaExecHost in the GRM MARTE sub-profile.

Extensions

- Node (from UML::Node)

Generalization

- GaExecHost (from MARTE:GRM::ComputingResource)

Associations

- task: Task [1..*]
  The tasks that assign to this node.

Attributes

- mobility: Boolean[1]
  specifies the node type (fixed or mobile node).
- cost: NFP_Price[1]
  specifies the economical cost of the node.
- gamma: RealVector[1]
  specifies a vector of coefficient to represent the power consumption of a node.

Semantics: A Node represents physical processing devices capable of storing and executing program code. It has attributes for economical cost and mobility.

b) «AbstractChannel»: It extends the semantics of CommunicationPath metaclass of UML standard profile and inherits attributes and implementation of GaCommHost from GQAM MARTE subprofile.

Extensions

- CommunicationPath (from UML:: CommunicationPath)

Generalization

- GaCommHost (from MARTE::GQAM)

Associations

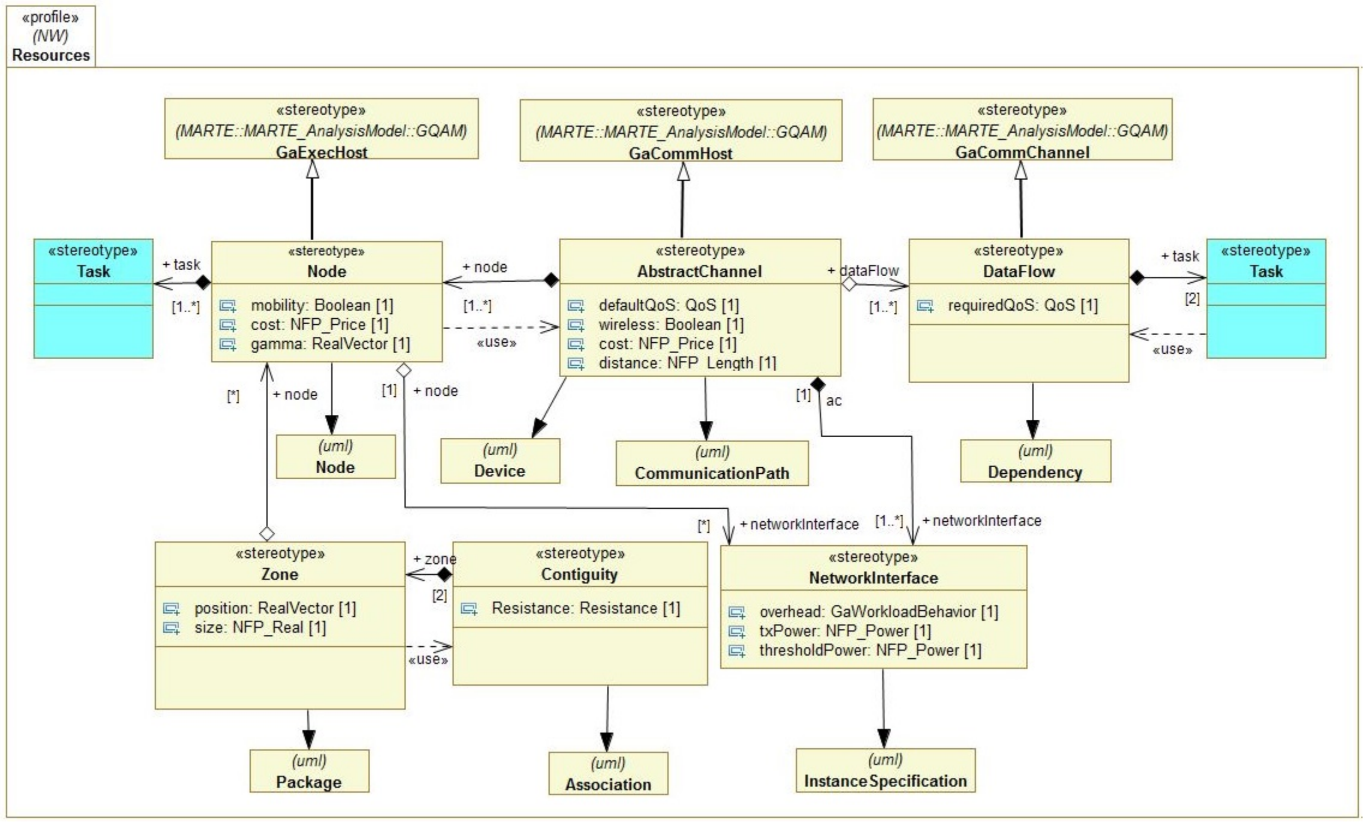

Figure 7. The Resources package of *Network Profile*

- node: Node [1..*]
  The nodes that are connected with this abstract channel.
- networkInterface: NetworkInterface [1..*]
  The network interfaces that are abstracted by the channel.
- dataFlow: DataFlow [1..*]
  The dataflows that assign to this abstract channel.

Attributes

- defaultQos: QoS [1]
  specifies the default quality of service which is provided by this channel.
- wireless: Boolean [1]
  specifies the channel type (wired or wireless).
- cost: NFP_Price [1]
  specifies the economical cost of the channel.
- distance: NFP_length [1]
  specifies the length of the channel.

Semantics: AbstractChannel is a generalization of network channels since it contains the physical channel, and all the communication protocols. It can be wire or wireless.

c) «DataFLow»: It extends the semantics of Dependency metaclass of UML standard profile and inherits attributes and implementation of GaCommChannel from GQAM MARTE subprofile.

Extensions

- Dependency (from UML::Dependency)

Generalization

- GaCommChannel (from MARTE::GQAM)

Associations

- task: Task [2]
  The source and destination task of this data flow.

Attributes

- requiredQoS: QoS[1]
  specifies the required quality of service of the data flow to perform the communication operation between two tasks.

Semantics: A data flow represents communication between two tasks; output from the source task is delivered as input to the destination task.

d) «Zone»: It extends the semantics of Package metaclass of UML standard profile.

Extensions

- Package (from UML::Package)

Associations

- node: Node [*]
  The nodes that allocate inside this zone.

Attributes

- position: RealVector[1]
  specifies the zone position in 3D space.
- size: NFP_Real[]
  specifies the size of the zone (unit m^3).

Semantics: A zone represents the environment that groups a set of node with a certain environmental information. For example, room in a building with temperature and pressure information.

e) «Contiguity»: It extends the semantics of Association metaclass of UML standard profile.

Extensions

- Association (from UML::Association)

Associations

- zone: Zone [2]
  The two zones which are linked by this contiguity.

Attributes

- Resistance: Resistance[1]
  specifies the amount of quality of service reduction which reflects the environmental effects.

Semantics: The contiguity between zones introduces to put constraints on the reachability of the corresponding nodes.

f) «NetworkInterface»: It represents an interface to connect a node to a communication channel. It extends the semantics of an InstanceSpecification that will be linked to a node or an abstract channel instance.

Extensions

- InstanceSpecification

Associations

- Node: node [1]
- AbstractChannel: abstractChannel [1..*]

Attributes

- overhead : GaWOrkloadBehavior [1]
  specifies a given load of processing flow.
- txPower: NFP_Power[1]
  specifies the transmission power of a transmitter networked node.
- thresholdPower: NFP_Power[1]
  specifies the threshold power of a receiver networked node.

Semantics: It acts as an interface to connect a physical device with a communication media. It has an attribute `overhead` which represents a given load of processing flows triggered by external (e.g., environmental events) or internal (e.g., a timer of the communication protocol) stimuli. The processing flows are modeled as a set of related steps that contend for use of processing resources and other shared resources. It may contain the communication protocol agent. It has two attributes to model the transmission power and the threshold for the receiving power.

C. Network Model Library

It is a predefined model library containing primitives and data types required to define the Network profile and also used in user models.

1) Model Library for Extended NFP Types: This sub clause defines a set of NFP Network types that use the MARTE basic NFP types library.

Figure 8 shows the internals of the concerned package (i.e., NW Basic_NFPTypes). The semantics and usage of the pre-defined data types are stated in each of the clauses that use them.

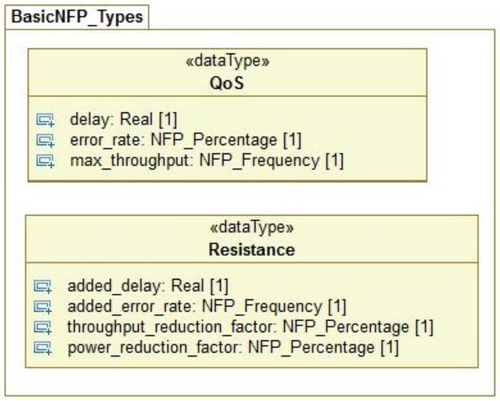
BasicNFP_Types
«dataType»
QoS
delay: Real [1]
error_rate: NFP_Percentage [1]
max_throughput: NFP_Frequency [1]
«dataType»
Resistance
added_delay: Real [1]
added_error_rate: NFP_Frequency [1]
throughput_reduction_factor: NFP_Percentage [1]
power_reduction_factor: NFP_Percentage [1]

Figure 8. Network library of general data-types

a) QoS: This is a TupleType that contains the parameters that are necessary to specify the quality of service of a communication link (see figure 8). **Attributes:**

- delay: Real [1]
  A parameter that is used to introduce a delay of a communication channel.
- error_rate: NFP_Error_Rate[1]
  A parameter that is used to introduce an error rate of a communication channel.
- max_throughput: NFP_Percentage[1]
  A parameter that is used to specify the maximum throughput/capacity of a communication channel as a percentage value.

b) Resistance: This is a TupleType that contains the parameters that are necessary to specify the environmental effects on DESs communication channels (see figure 8).

Attributes:

- added_delay: Real [1]
  A parameter that is used to introduce an added delay on a communication link.
- added_error_rate: NFP_Percentage[1]
  A parameter that is used to introduce a decreased amount of error rate of a communication channel as a percentage value.
- throughput_reduction_factor: NFP_Percentage[1]
  A parameter that is used to introduce a reduction of a throughput between two zones as a percentage value.

- power_reduction_factor: NFP_Percentage[1]
  A parameter that is used to introduce a power degradation in a channel as a percentage value.

VI. Use case

This section shows an example of the modeling phase of a home automation application. The application scenario aims to send sensing data from transmitting unit (temperature sensor unit) to receiving units (monitor and configurator units). There are two other units for routing and collecting data (router and collector nodes) which are connected via wireless and wired channels with other units.
Papyrus UML [18] editor tool, is used to model this application and show the profile annotations on its UML elements.

The application consists of three rooms, four nodes, and two communication channels. Figure 5 shows the structure of this application. Node_0 is located in Room_0 (room coordinates are x=0, y=0 and z=0 with volume = $125m^3$) and a sensor task is deployed on it. That task sends the sensing data to a Router task that is deployed on Node_1 and located in Room_1 via the wireless channel (wifi_0). The channel characteristics are: error rate =1 %, throughput = 250 kbps and delay = 150 μ sec and it serve three data flows (DF_S2R, DF_R2C, DF_CF2S). The communication between those tasks is facing the environmental effect (Resistance) which adds 2 msec of delay, 10% of error rate and throughput reduction factor by 15%. Room_2 has two nodes which are interacting via a wired channel. This channel serves the data flows (DF_C2M, DF_C2CF, DF_CF2S). Therefore, the communication between configurator task (configurator_0) and sensor_0 is performed by the data flow (DF_CF2S) which uses the wired and wireless channels to reach Node_0. These models are then transformed to SCNSL [19] code for simulation as it is described in [2].

VII. Conclusions

This paper formalizes a coherent set of distributed embedded systems elements and presents the modeling phase of them. It defines the core elements needed to model DESs and their relationships with the standard UML MARTE profile. It also addresses the lack of precise networking elements in UML and MARTE, introducing a novel profile, as a specialization of MARTE, which essays a contribution to its further standardization. It showed the use of UML deployment diagrams to model DESs. It proposes and presents in detail a new profile named as *Network Profile* and its relationship with MARTE. Finally, a home automation application is used to show the effectiveness of the modeling approach.

As future work we aim at capturing the Task functionality in UML and transforming its behavioral diagrams into the corresponding UNIVERCM automata.

References

[1] Y. Jung, L. Carloni, and M. Petracca, “Cloud-aided design for distributed embedded systems,” *Design Test, IEEE*, vol. 31, no. 3, pp. 32–40, June 2014.

[2] E. Ebeid, D. Quaglia, and F. Fummi, “UML-based Modeling and Simulation of Environmental Effects in Networked Embedded Systems,” in *16th Euromicro Conference on Digital System Design (DSD)*, 2013.

[3] Object Management Group, “UML: Unified Modeling Language,” URL: http://www.uml.org.

[4] ——, “A UML Profile for MARTE (version 1.1),” in *OMG document number: formal/2011-06-02*, Jun 2011, URL: http://www.omgmarte.org.

[5] ——, “A UML Profile For Schedulability, Performance, And Time (version 1.1),” Jan 2005, URL: http://www.omg.org/spec/SPTP/.

[6] J.-F. Le Tallec, J. DeAntoni, R. de Simone, B. Ferrero, F. Mallet, and L. Maillet-Contoz, “Combining SystemC, IP-XACT and UML/MARTE in model-based SoC design,” in *Proceedings of Workshop on Model Based Engineering for Embedded Systems Design*, 2011.

[7] OMG, *UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms Specification*, Std., 2008.

[8] H. Espinoza, H. Dubois, S. Gérard, J. Medina, D. C. Petriu, and M. Woodside, “Annotating UML Models with Non-functional Properties for Quantitative Analysis,” *Lecture Notes in Computer Science*, vol. 3844, no. 4, pp. 79–90, 2006.

[9] J. Lee, S. Bohacek, J. Hespanha, and K. Obraczka, “Modeling Communication Networks With Hybrid Systems,” *Networking, IEEE/ACM Transactions on*, vol. 15, no. 3, pp. 630 –643, Jun. 2007.

[10] Object Management Group, “SysML,” URL: http://www.sysml.org.

[11] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong, “Taming Heterogeneity - The Ptolemy Approach,” *Proceedings of the IEEE*, vol. 91, no. 1, pp. 127–144, 2003.

[12] D. A. Van Beek, M. A. Reniers, R. R. H. Schiffelers, and J. E. Rooda, “Foundations of a Compositional Interchange Format for Hybrid Systems,” in *Proc. of HSCC*, 2007, pp. 587–600.

[13] E. Ebeid, F. Fummi, and D. Quaglia, “Model-driven design of network aspects of distributed embedded systems,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 34, no. 4, pp. 603–614, April 2015.

[14] Object Management Group, “OMG Unified Modeling LanguageTM (OMG UML), Superstructure(version 2.2),” in *OMG document number: formal/2009-02-02*, February 2009, URL: http://www.omgmarte.org.

[15] Object Management Group (OMG), “OMG,” URL: http://www.omg.org/.

[16] P. Feiler, D. Gluch, and J. Hudak, “The Architecture Analysis & Design Language (AADL): An Introduction,” Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-2006-TN-011, 2006. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7879

[17] L. Di Guglielmo, F. Fummi, G. Pravadelli, F. Stefanni, and S. Vinco, “UNIVERCM: The UNIversal VERsatile Computational Model for Heterogeneous System Integration,” *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 225–241, Feb 2013.

[18] Sébastien Gérard et al., “Papyrus UML,” URL: http://www.papyrusuml.org.

[19] “SystemC Network Simulation Library – version 2,” 2013, URL: http://sourceforge.net/projects/scnsl.