Modeling, Analysis and Simulation of Ubiquitous Systems Using a MDE Approach

Amara Touil, Makhlouf Benkerrou, Lherminier Fred, Jean Vareille, Philippe Le Parc

▶ **To cite this version:**

Amara Touil, Makhlouf Benkerrou, Lherminier Fred, Jean Vareille, Philippe Le Parc. Modeling, Analysis and Simulation of Ubiquitous Systems Using a MDE Approach. International Journal On Advances in Intelligent Systems, 2011, 4 (3&4), pp.147-157. hal-00694324

HAL Id: hal-00694324

https://hal.univ-brest.fr/hal-00694324v1

Submitted on 4 May 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Modeling, Analysis and Simulation of Ubiquitous Systems Using a MDE Approach

Amara Touil*, Makhlouf Benkerrou*, Jean Vareille*, Fred Lherminier†, and Philippe Le Parc*
*Université de Brest, France
Université Européenne de Bretagne
LabSTICC - UMR CNRS 6285
20 av. Victor Le Gorgeu, BP 809, F-29285 Brest
amara.touil@univ-brest.fr, makhlouf.benkerrou@etudiant.univ-brest.fr,
jean.vareille@univ-brest.fr, philippe.le-parc@univ-brest.fr (contact author)
†Terra Nova Energy
28 rue Victor Grignard, F-29490 Guipavas
fredl@terranov.com

***Abstract*—The growth of industrial activities during the last decades and the diversity of industrial products require standards and common methodologies for building and integrating systems. It is also required that working groups use the same terminologies and concepts needed for each domain. The Model Driven Engineering approach aims to give an answer, while using a high level method based on models and transformations. In this paper, we use this approach to model ubiquitous systems. Those systems are composed of devices interconnected through various kinds of network, in order to get and provide information. We present a model for this class of systems and, its use, in terms of analysis and simulation, in the field of energy while studying real cases from our industrial partner, Terra Nova Energy.**

Keywords-Domain Specific Language; Model Driven Engineering; Ubiquitous Systems; Analysis; Simulation.

I. INTRODUCTION

Ubiquity is often defined as the property of being able to be in several places at the same time. In the field of Information Technologies, this definition can be improved in two different ways, that may look opposite. Here is the first approach: as in [2] users are surrounded with "intelligent" systems, that may deliver them needed information: in this case, computing technologies are used in order to locate users, to understand their environment, to anticipate their needs and to make it possible that any information they require is available anywhere at anytime. The second approach is to offer people to be able to get information on a system located somewhere and to be able to act safely on it: here, computing technologies are used to make it possible to be "virtually" present in several places at the same time. We place our work in the second approach, in order to model and analyze telecontrol applications as a kind of ubiquitous systems.

Terra Nova Energy (TNE) is an innovative company [3] that provides solutions for data mining in the fields of electricity, hydraulics and pulse-energy. It integrates sensing, acting and communicating devices in telecontrol systems, for collecting data from different sites using various technologies. TNE's systems allow real-time operating and provide processes for handling different data.

In order to improve its development, this company is facing two problems: how to design remote monitoring systems as automatically as possible and how to speed up their on-site deployment ?

To answer these questions, we intend to use a based model approach, relying on specific components for telecontrol domain and libraries integrating industrial parts coming from various providers. Our project is to build a framework [4] following the Model Driven Engineering (MDE) methodology [5][6] to setup a telecontrol ontology and proper transformations for building, generating, analyzing and deploying such ubiquitous telecontrol systems.

Our aim is to define a generic meta-model and to use it for various systems, as case study examples, one of them being of the TNE system.

This paper is organized as follows: first of all, the MDE approach, Domain Specific Languages (DSL) and transformations that may be conducted are briefly introduced. Then, we explain the originality of our work while describing our method. In the Section IV, we introduce the proposed generic meta-model, and in Section V this general meta-model is applied to our case study, while defining an instance for TNE. Then, we explain how to carry a static analysis on a given instance and show the first results. We then describe automatic transformation that makes it possible to simulate, using PtolemyII [7], an instance. Finally we discuss our method and we finish by further work.

II. RELATED WORKS

In this section, we introduce some basic notions about the MDE approach and DSL, and how to carry out transformations on models in order to analyse them. A focus is also made on simulation concepts and tools.

A. The MDE approach

A model is an abstracted view of a system that expresses related knowledge and information. It is defined by a set of

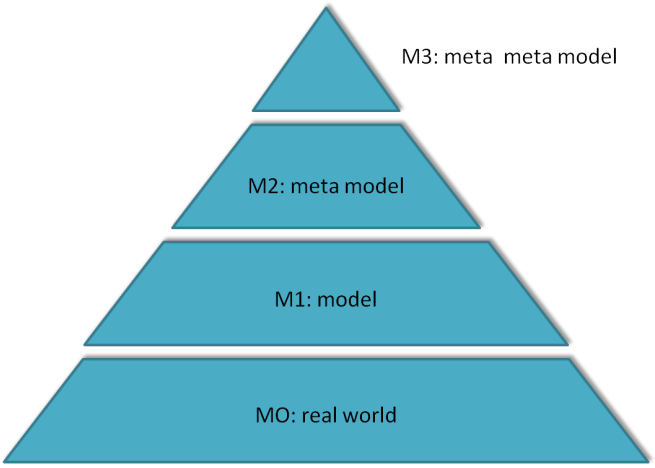

Fig. 1. The MDE pyramid

rules, used to interpret the meaning of its components [8][9]. A model is defined according to a modeling language that can give a formal or a semi-formal meaning description of a system depending on modeler's intention. Modeling languages can be textual or graphical.

The model paradigm has gained importance in the field of systems engineering since the nineties. Its breakthrough was favoured by working groups like the Object Management Group (OMG) [10] that has normalized modeling languages such as Unified Modeling Language (UML) and its profiles (such as System Modeling Language - SysML [11]- and - UML Profile for Modeling and Analysis of Real Time and Embedded Systems - MARTE [12] - for real-time systems). This group also provides the Model Driven Architecture (MDA) software design standard. The MDA is the main initiative for Model Driven Engineering (MDE) approach.

Four abstraction levels have to be considered: a meta-meta-model (M3 on Figure 1) that represents the modeling language, which is able to describe itself; a meta-model level (M2) that represents an abstraction of a domain, which is defined through the meta-meta-model; a model level (M1) that gives an abstraction of a system as an instance of the meta-model; finally, the last abstraction level is the real system (M0).

Tools have been defined to implement the MDE approach. Some have general and wide purposes, like those designed for the UML language, others have been defined for specific and reduced classes of applications, as in [13] that aims to specify wireless sensor network systems in order to generate code. Another approach is to use Domain Specific Language to specify a specific semantic and/or syntactic rules for a class of applications. One of DSL's definition is given by [14]: *"A Domain Specific Language is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain"*.

B. Model transformation

In order to breathe life into models [15], model transformations aim to exceed the contemplative model view to a more productive approach, towards code generation, analysis and test, simulation, etc.. Models are transformed into other models that may be handled by specific tools or that may be transformed again into other models. Two kinds of transformation are used, exogenous, if the source and the target of the transformation do not have the same meta-model and endogenous, if source and target have the same meta-model. We use the latter here in order to perform a static analysis of the built models.

Generally, static analysis deals with the observation of the system and the check of some properties before real system development, production or code implementation. As an example, formal verification and model checking techniques [16][17] are used to verify models. In this paper, static analysis is only carried out in simple cases, but relevant for TNE. We will focus on placement but other studies have been reralized (cost, bandwidth, etc.). As soon as the model is designed, it may provide the enfinee with answers: Are my sensors, repeaters, routers placed in a proper location? Are there better configurations?

Concerning the first question, there are several works dealing with these points like [18], which studied Wireless Network Sensors (WSN) placement for smart highways: it gives a solution based on geometric resolution algorithms in outdoor and open space. In our case, indoor deployment environments, including various fixed and mobile obstacles, have to be studied. Component placement depends on the monitored zone, on the chosen topology, on the network capacity, etc. The second question can be addressed by optimising the placement configuration.

C. Simulation concepts and tools

Simulation improves the understanding of a system without having to actually handle it, either because it is not yet defined or not available or because it can not be manipulated directly because of cost, time, resources or risk [19].

The modeling stage is the most critical phase of the simulation. To get a good modeling several issues must be addressed. It is necessary to :

- Analyze the problem to solve and set goals matching the specifications,
- Define the inputs and outputs of the system,
- Identify the interactions between the elements and build a conceptual model,
- Identify the dynamics of the system i.e., the system behavior over time in its environment (super-system)
- Study the most relevant elements of the system.

Figure 2 represents the classical simulation work flow. The notion of super-system has been added in order to express the external environment of the modeled system. The simulation can be made from a predefined scenario or interactively. This solution then allows the user to gradually build the execution trace of events.

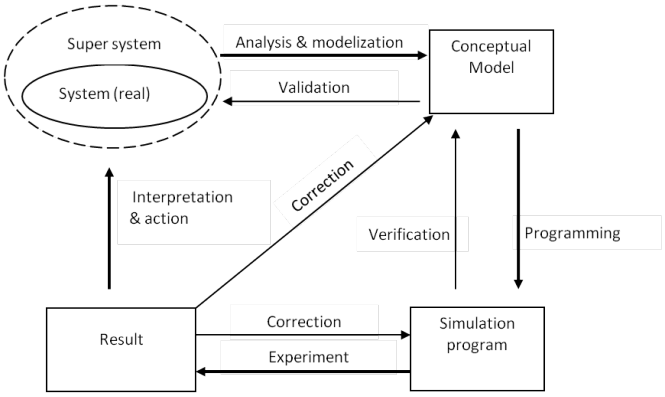

Fig. 2. Simulation scheme

There are many simulation tools such as: Labview [20], Simulink/Matlab [21], Scicos/Scilab [22], PtolemyII [7], Occam [23], etc. As in our case study, the aim is to simulate ubiquitous systems, we need a simulator that uses communicating entities. Occam and PtolemyII have been selected and studied, and the latter has been chosen. The other simulation tools are generally used for general scientific problems and mathematical calculations.

The *actor* is the key concept of PtolemyII, and the VisualSense extension [24] of this language includes generic actors designed for sensor networks such as sensor-actuator, communication channels (wired and wireless)... In addition, PtolemyII offers a wide range of calculation models known as *director*: discrete event, synchronous data flow, appointments, etc.. The graphical approach of PtolemyII is also a strength, which allows a visual monitoring of the simulation.

The next section describes the suggested meta-model for telecontrol and its derivation for TNE's remote monitoring needs.

III. OVERALL APPROACH

The aim of this work is to apply the methods and techniques defined by the MDE approach, in order to improve the development of ubiquitous applications. The main idea is to have one (and only one) high level model of the system and to use a collection of tools that may transform the model into various results, such as code to be deployed on systems, input files for simulators, 3D models etc..

Right now, in most projects, engineers use an informal description of systems and separated tools. This is enhanced in the field of ubiquitous applications, as these types of applications are quite recent, rely on components coming from various companies and use different network providers. Existing development environments are not yet ready to manage such applications, but there exists a lot of work that has been done that could be interconnected and re-used.

It is a long term work and, in this paper, we focussed only on three main aspects:

- The first one is to define a meta-model (Section IV), that will make it possible to capture the maximum information on the system to be modelized. Using existing meta-models was an opportunity [25], but, either they are too general such as UML, or too specialized such as SysML. In our case we have defined our own DSL for telecontrol, according to Eclipse Core (Ecore) meta-model [26], and we have tried to take into account the specifity of ubiquitous applications. From this meta-model, we are able to build the unique model (or instance) of a system. This instance can be seen as the main input of the transformation tools.
- The second aspect developed in this article (Section V) is the analysis of the instance. In that case, we have just "had a look" to it and we have tried to assess some properties. Transformations are used to extract interesting information and to present it in a comprehensive manner.
- The last aspect (Section VI) is the simulation of a system. Starting from an instance of the system, transformations are applied in order to produce input files for an "on the shelf" simulator. The transformation rules have to be chosen carefully to preserve equivalence between the modelled system and the simulated system.

The originality of this work relies on the definition of a meta-model for ubiquitous systems and the building of transformations within the MDE approach.

IV. SUGGESTED META-MODEL

In this section, we describe the meta-model we are suggesting to specify systems based on communicating objects. Only a high level view and the main concepts are presented. Starting from this meta-model, we derive a specific instance for TNE.

A. Generic meta-model

At a first level of abstraction, we have build our domain around systems containing entities that communicate with one another as stated in [27]: *"A system is a construct or collection of different elements that together produce results not obtainable by the elements alone"*. In the meta-model proposal (Figure 3), a system, denoted by *System*, can contain other systems according to the *ownedSystem* reference, in order to provide the "system of the system" notion. Regarding entities, they are modeled by *Entity* and its system containment is referenced by the *itsEntity* relationship. Entity paradigm in our meta-model aims to be a generic and general concept formed by extracting common features from our telecontrol domain. An entity can be logical or physical and can be also composed of other entities (*ownedEntity*).

Moreover, entities are described using several related concepts trying to keep information about their physical properties, communication facilities or behaviour. These concepts are modeled as follows:

- The *Structure* element defines the interrelation or arrangement of different physical parts or the organization of elements that provide coherence, shape and rigidity to an entity. It may describe mechanical, chemical or biological aspects.

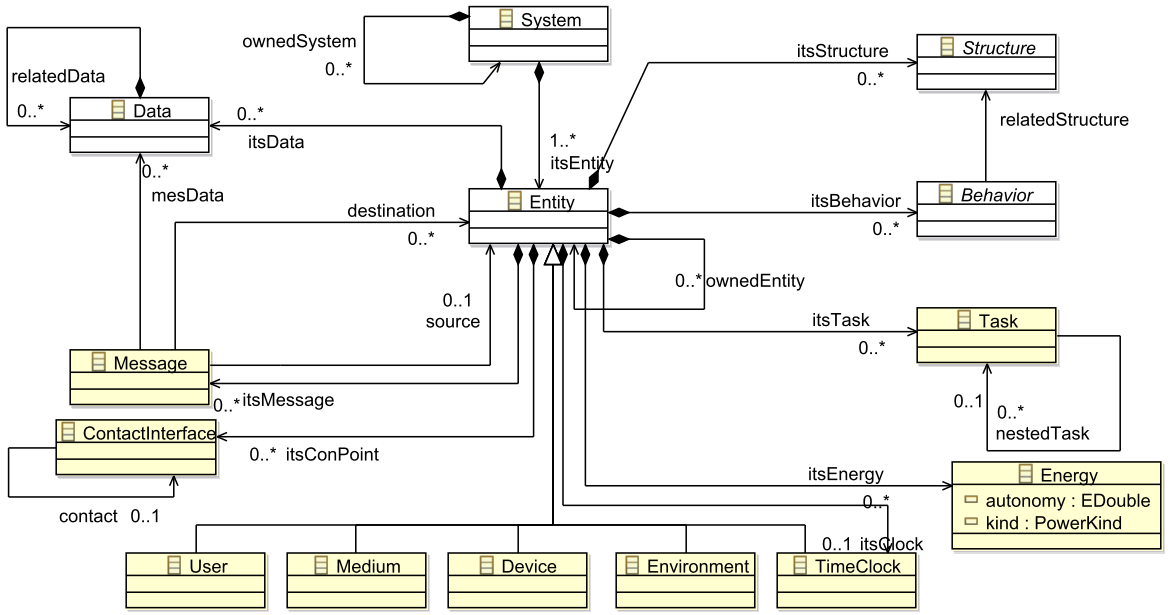

Fig. 3. A view of generic meta-model components (Basic package)

- Actions performed by entities are described by the *Task* concept. They can be internal, such as making computation, external such as sending out information and also connected to the real world such as sensing or acting on a real device.
- The *ContactInterface* element allows connection between entities. It can be a physical contact, or a logical interface depending on the specification level, on the refinement degree or on its own structure.
- An entity may contain *Data* related to itself or to its environment.
- The *Message* element provides data exchange between entities. To send (or to receive a message) from entity A to entity B, A and B must be connected through *ContactInterfaces*, directly or, it may exist a path of entities that may forward messages.
- To each entity, a *Behavior* concept is added, in order to describe the different tasks an entity may perform. From this concept, code generation or simulation may be improved.
- *Energy* is a major concern in ubiquitous applications. Entities consume energy in different manners and also may get energy in different ways.

As the *Entity* concept is very general, in order to improve our generic meta-model, different sub-entities have been defined using the concept of heritage:

- *Device* represents a physical component such as a sensor, a router, a computing unit, etc.
- *Medium* represents an entity, deployed and used between other entities to enable communication. In classical approaches, medium is often omitted and considered as a perfect link. In our work, medium has its own structure, behavior, contact interfaces, data, messages and tasks. Different types of medium may be described and classified following their own specifications, wired or wireless for example.
- *User* is used to model a person interacting with other entities, it is a kind of human avatar. Like other entities *User* can have different presentations depending on model view.
- The different entities composing the system are subjected to some physical conditions and constraints that influence the global behaviour. In our meta-model we introduce the *Environment* concept to describe such conditions. Because we are dealing with a complex system, the *Environment* is an entity of this system and not just an external element. In our modeling approach, *Environment* acts on internal elements in order to build a general framework of circumstances for the evolving system and gives context for ubiquitous entities and systems that are context-aware.
- The concept of time is crucial for telecontrol systems that is why a *TimeClock* entity is added to our generic meta-model to measure, record or indicate time.

The presented generic meta-model intends to specify any kind of system containing communicating objects. At this abstraction level, we present only a first aspect (also named *Basic package*), without getting into all the details of the meta-model. Note that environment and medium entities allow to model the super-systems previously introduced.

B. Terra Nova Energy meta-model

In the previous section, a generic meta-model at a first level of abstraction with some inherited elements from *entity* such

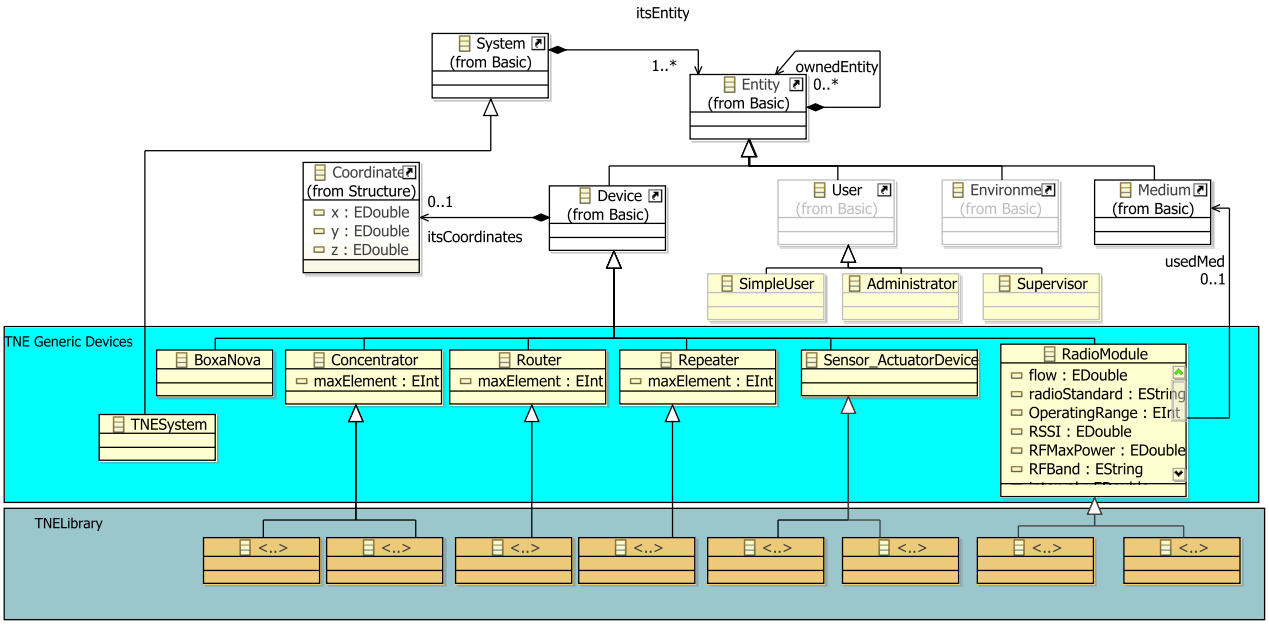

Fig. 4. A view of Terra Nova Energy meta-model

as *Device* has been presented.

According to the concept of inheritance, the TNE meta-model (Figure 4) is built as an extention and a generalization of the generic meta-model. The first element is the *TNESystem* that inherits from the generic element *System*. Other elements have been spread over two sets: generic devices and off the shelf devices that could be grouped in a specialized library.

1) Terra Nova Energy generic devices: This first set of elements represents generic devices that allow to add a new industrial component to a library or to build a specific system. All these devices inherit from the generic *Device*, imported from Basic package as shown in Figure 4. We present them successively as follows:

- the *BoxaNova* is the main device of TNE systems. It collects information, sends orders and manages the entire flow of data between different devices and users.
- the *Router* element modelizes a device that handles message transfers between different TNE elements. It handles also some other functions like controlling repeaters, sensors, actuators and some other server facilities.
- *Concentrator* is used to model a collecting place of data coming from repeaters, sensors, actuators before sending them to routers according to requests or preprogrammed sending tasks.
- *Sensor_ActuatorDevice* is used to capture measurement data and to send them or to act pursuant to an order.
- *Repeater* is used to ensure the link between a *Concentrator* and *Sensor_ActuatorDevice* elements if they are not within reach.
- *RadioModule* models the device that ensures exchanging data messages between all other devices when wireless communication links are used.

Since TNE devices are specialized from the generic element *Devices*, they inherit all its properties and references.

2) Terra Nova Energy off-the-shelf devices: This second set of elements represents the real devices, ready to be used in the framework. They are another level of specialization of the first set. They may be grouped in a kind of library (lower box in Figure 4), composed with industrial devices coming from different manufacturers and in order to be integrated into TNE systems. In this paper, we neither describe this library of devices nor the way they were designed because we focus on some other properties and aspects of model analysis.

In the next section, we propose a methodology of analysis showing the usefulness of these presented meta-models, and their examination and validation on real industrial cases.

V. MODEL ANALYSIS

In this section, we present how to exploit the MDE approach to analyse placement for a given instance, and automatic placement of repeaters and concentrators for a system, where only sensor positions are known.

The first part describes our methodological approach, based on processing properties and applied constraints for model's elements.

A. Presentation

From the TNE domain specific meta-model, presented in the previous section, an instance that describes a real system of telecontrol may be defined. According to the user's needs, this instance captures a particular concern for a system and gives the necessary elements and their relevant properties. Such a model requires some processing stages to meet the final requirements in terms of analysis. These processing stages are

done in response to expected system constraints, applied to its various elements. Figure 5 shows a model overview of the various elements used to carry out those processes.

The first analysis component (*ModelIn*) loads the instance that describes a telecontrol system according to its meta-model. This instance is composed of elements called *ElementIn*. The *Engine* scans the input model, element by element, in order to find different properties and constraints. For each property the *Analyzer* checks if it fits well the associated constraint. When properties are verified, the element will be transformed by *Transformation* and treated by *ElementOut*. The *ModelOut* automatically generates the output model formed by those transformed elements. Output model should be consistent with its meta-model, that may be the input one or another depending on the performed transformation (endogeneous or exogeneous).

Generally, input models have a tree shape structure and their components can be composed of other components. Properties and applied constraints can also have this composition criterion. In order to deal with this kind of structure and having a generic analyzer, possible compositions have been taken into account for analyzing and transforming tasks. In Figure 5, these composition aspects are represented by the *Composed* reference.

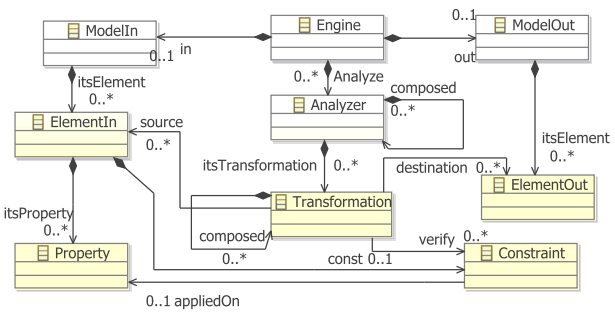

Fig. 5. A view of analyzer's engine model

The next part of this section deals with an application of this methodology to a system. First of all, analyses for a completely described given instance are conducted to verify that elements are well placed and may communicate together properly. Secondly, we show that the Model Driven Engineering approach used, may also be helpful to build instances, in the case of an uncomplete system, where only sensors and actuators are known, by adding the necessary repeaters and concentrators.

B. First case study with a complete TNE instance

In this first case study, an instance of a TNE meta-model has been created with a tree editor as shown in Figure 6. It is an abstraction of a *TNESystem* named TelecontrolSystem_1 composed of two *BoxaNova* that manage twenty *Sensor_ActuatorDevices*. Each *BoxaNova* consists in two *Concentrator* elements and a *Router*. Communication between these components uses two *Mediums*; a coaxial cable that

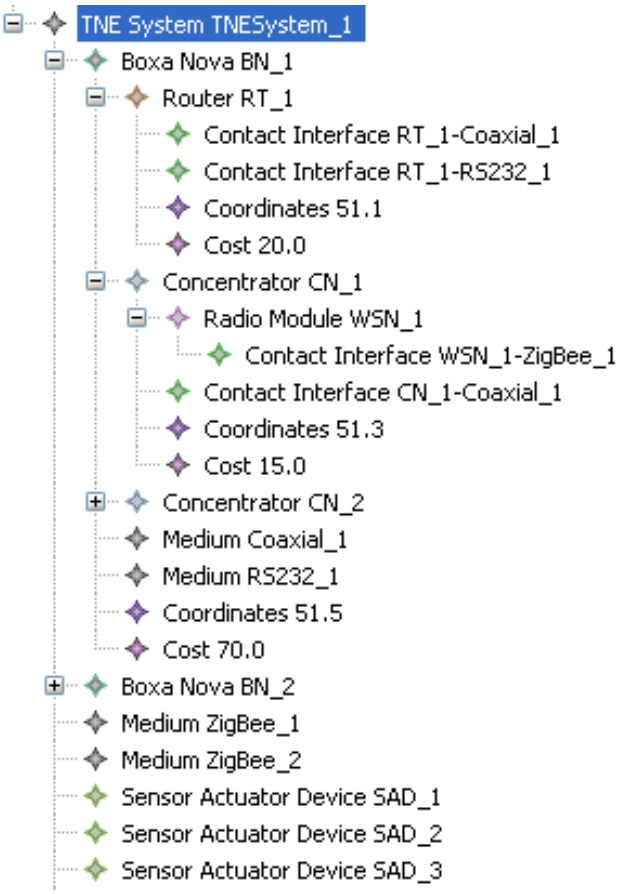

Fig. 6. A screen shot of manual instance creation

connects the first concentrator and the router and a RS232 cable that provides serial communication between the second concentrator and the router.

Sensors and actuator devices use a ZigBee protocol (IEEE_802.15.4 standard) [28] to communicate with the *BoxaNova* and the concentrators. To complete this task, they use *radioModules*. Concentrators also have their own radio modules. In general, wireless propagation environments in TNE systems may have different attenuations. Thus, each communication link between a sensor/actuator and its repeater or concentrator is defined by a medium with relevant properties and necessary (contact) interfaces. In our model, this information is captured by properties such as the attenuation value, speed of transmission, signal length, etc.. The connection with the medium and other elements is ensured by the definition of two entities as *ContactInterface*; one for the medium and the other for the associated element.

The model also captures other information like the physical location (x, y coordinates) of the different parts of the system. In this way, a *Communication vs. placement* analysis has been conducted. In fact, component's placement in TNE sytems is crucial, to ensure good communication and proper information transfer. As the model is fully known, an analyzer can be created to verify that every *Sensor_ActuatorDevice* element

according to its coordinates may communicate to its corresponding concentrator. This verification takes into account the attenuation between communicating elements , specified as a property of the medium that links them. Another analyzer can be defined and used to inspect the number of sensor/actuator devices for each concentrator and its compliance to the specified *maxElement* property.

From the information analysis of the obtained text file (see transformation model to text in [15]), manual corrections may be performed. This task is manageable for small systems but gets difficult and very expensive for large ones with many components. The next part gives an illustration of a methodology that can solve this difficulty.

C. Second case study with a partial TNE instance

In this second case study, only the number and the position of the sensor/actuator are given. The instance is partial, and the objective is to automatically find a solution for the repeater's and concentrator's placements. Finding the better placement is a complex problem and several solutions may be found in literature [18][28][29]. We used a very simple algorithm for this study.

In our case the input model is generated automatically with the required properties: as a first step, an instance of the TNE meta-model is created with a predefined number of *sensor_ActuatorDevice* and an environment that represents a factory hall. We suppose here that radio measurements were made in the environment and have identified attenuation values.

In a second step, a placement analysis engine, composed of three analysers, is used:

1) The primary analyzer performs an initial refinement of the input model. First, it divides the environment, specified in the input model, in zones where medium has the same attenuation so that the range of sensor-actuator radio modules, which are currently inside, will reach the middle. This range is calculated by applying the attenuation value imposed in the input model. Then, depending on the number of sensor/actuator devices, it creates the necessary repeaters with their radio modules, and adds them to the input model, with their coordinates around the centre of the sub-environment. Finally, it creates connections (*ContactInterface*) between repeaters and sensor/actuator elements and adds them to the model.
2) A second improvement stage of refinement with the same processing approach is made by the second analyzer. Its objective is to connect repeaters with concentrators. The input model environment is split up into several zones with new radio module repeater ranges. The number of added concentrators will depend on their capacity and also on the number of repeaters within reach.
3) The third analyzer performs an optimization of the obtained model, in order to detect the sensor/actuator devices that can communicate directly with concentrators without the use of a repeater. This analyzer checks for each sensor/actuator device if it reaches any concentrator and changes its connection from a repeater to a concentrator. Repeaters without connected sensor/actuator devices are removed.

At the end of these three analyzes, a new output model is obtained. It is composed of the initial sensor/actuator devices and added components (repeaters, concentrators and connections). Figure 7 shows a simple placement layout of TNE system components obtained with the following initial inputs: 200 sensor/actuator devices (the location of these elements has been randomly chosen for the test), an environment that represents a factory with a 800-meter length and a 600-meter width and an attenuation value of 5 decibels by meter. Rectangles represent repeaters, triangles represent concentrators and circles represent sensor/actuator devices.

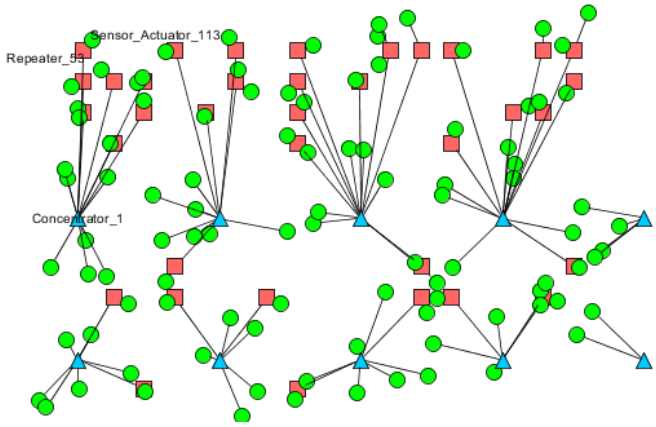

Fig. 7. Placement layout

The generated model allows us to have a first proposal for the positions of the repeaters and of the concentrators in the environment. This obtained placement is of course not optimal, as simple hypotheses have been used. Indeed, the presence of mobile obstacles in the environment and their influence on signal propagation should be considered. To optimize the placement and to ensure the highest data rate, optimal algorithms should also be used. The orientation of elements should also be taken into account.

Nevertheless, our objective is to show that using model analysis for ubiquitous systems within MDE approach, may help engineers in the design of their systems and verifying some properties at early stages without real implementation.

VI. SIMULATION

In this section, we present how an instance may be transformed into a model to be simulated, that can be used by a simulation tool like PtolemyII. The first stage is to define which actors of the PtolemyII framework correspond to the entities defined in our own model for ubiquitous systems. The second stage consists in the realisation (programming) of the different transformations needed. We conclude with some experiments made to validate the process.

A. From "entities" to "actors"

Four different entities have been defined in our model (see Figure 3): *User, Medium, Device, Environment*. In this work, the first one has not been yet described.

1) Medium: The VisualSense extension of PtolemyII provides an actor named *PowerLossChanel*, which may be used to represent the parametric model of a wireless channel. Range, power, propagation speed and power propagation factor may be set up, in order to define the medium used. Values may also be chosen in order to define a perfect medium without propagation delay and attenuation.

More, PtolemyII makes it possible to specify special components that may play an attenuation role. For example, these elements may be used to simulate walls, in order to represent real environments. The signal propagation is then completely managed by the tool.

In fact, PtolemyII includes elements external to the system and permits the simulation of super-systems.

2) Device: To modelize devices, the choice has been made, to use the *Wireless Composite* actor of PtolemyII, which contain wireless input and output ports, and the behavior of which is defined internally by a discrete event model.

In order to represent TNE systems, a specific wireless composite has to be defined for each TNE specific devices. Among these, only the sensor will be presented in the rest of this section.

At a first level, a sensor may be represented as in Figure 8. The left part corresponds to the control of the input signal in terms of power; if the power is to low, the input signal is not processed and not forwarded to the sensing part (right-hand side of the figure).

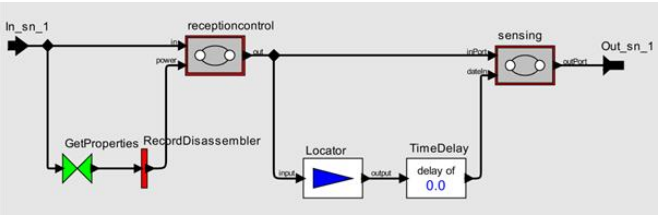

Fig. 8. Internal view of a wireless composite representing a sensor

The sensor element includes a Finite State Machine (FSM), which defines its behavior (see Figure 9). It is composed of three main parts: initialization (top of the FSM), reaction to inputs (right part of the FSM), which means forwarding values to a concentrator or a BoxaNova in this case, and reaction to control orders such as *in defect* or *repaired* (left part of the FSM).

As stated before, for each real element's behavior has to be coded with a FSM. This coding has to be made carefully, to fit as much as possible with the behavior of the real element. One may imagine that in the future, device's manufacturers will deliver their products with such a description, in order to be directly integrated into simulation tools.

3) Environment: The environment of the system can not be fully specified. In our experiments, we considered that sensors received a random value, as we are interested in value's propagation and not in value exploitation.

A *WirelessComposite* actor has also been used to represent the environment. This component is able to send information to the different sensors using a specific medium named *envMedium*. It is considered a perfect medium to deliver the values without delay.

As studying the system in case of failures is also addressed in our research, a second perfect medium (*controlMedium*) has been added in order to send orders to the different devices such as *start*, *stop*, *defect* or *repaired*. A third medium, *checkMedium* is also used to get data from the BoxaNova.

The environment entity makes it possible to express a simulation scenario where, step by step, values are sent to sensors and orders are provided to the different devices. It allows several scenarios, to be tested on the same system.

B. Transformation

As soon as the equivalent actors of PtolemyII have been chosen to express the entities of our model of ubiquitous system, the second stage may be started. It consists in transforming automatically a model of a system into a PtolemyII XML input file (eXtensible Markup Language).

It has been considered here that scenarios are seen as external artifacts of our modelization. The first step is then to add to our model the different medium defined in the previous section and the corresponding *ContactInterfaces* (see Figure 3). It corresponds to an endogenous transformation, which can be easily described by rules and implemented. Figure 10 shows the result of such a transformation, where the left-hand part described a model before the transformation, and the right-hand part after the transformation.

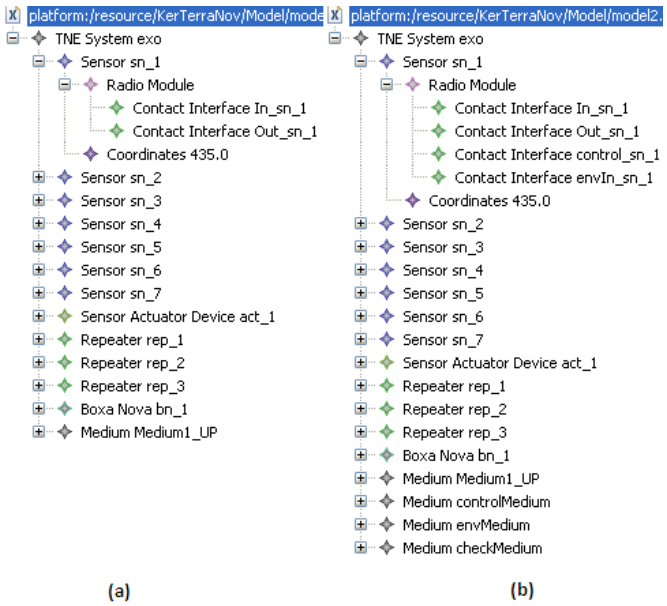

Fig. 10. First transformation

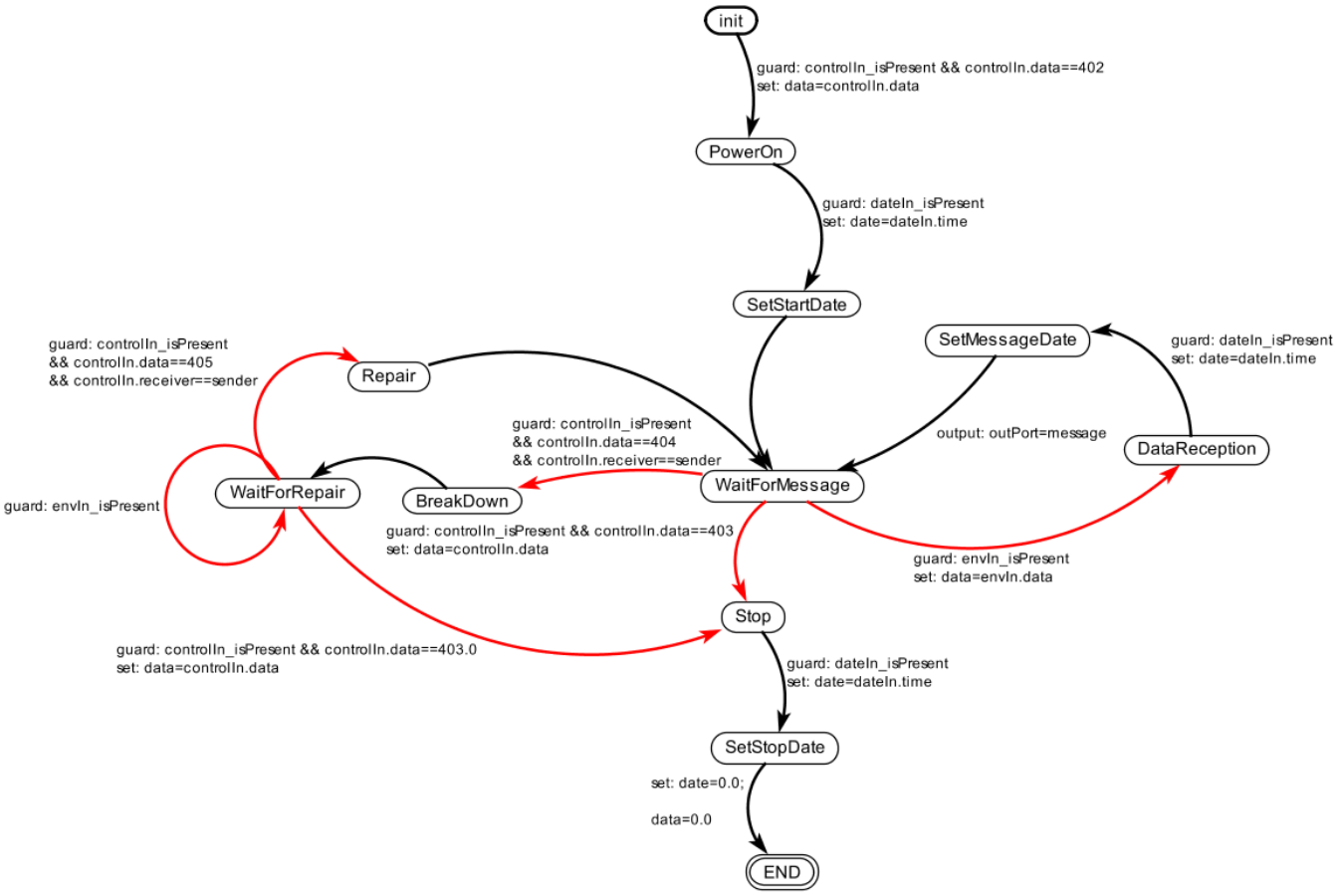

Fig. 9. Finite State Machine for a sensor

The second step consists in an exogenous transformation, to generate an XML input file for PtolemyII. The latter will contain all the needed information in order to express every entity of the system to model (Medium, Devices, Environment) with its coordinates, its specific parameters and its behavior. The transformation is performed automatically and can be used for every TNE system.

PtolemyII can then be launched to simulate the system.

C. Results

The transformation chain has been tested on a case study coming from our partner Terra Nova Energy. The modeled system is composed of 7 sensors, 1 sensor-actuator, 3 repeaters and 1 BoxaNova. In this example, only one type of medium has been used, but the distance between the different elements has been chosen in order to restrict the atteignability: each sensor may reach a repeater or the Boxa Nova, some may reach two repeaters but not more.

Figure 11 shows the different elements and their localization. This type of document corresponds to those that an engineer may use to describe where and which devices will be used for a specific application. This document is then expressed into an model under our Kermeta environment.

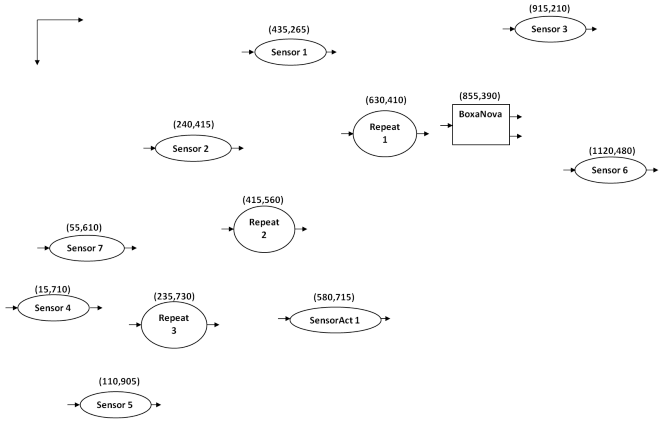

Fig. 11. Case study specification

Transformations are applied automatically and the XML file for PtolemyII is produced. Simulation can be start and the resulting interface appears on Figure 12.

The top of the figure corresponds to the different devices modeled by PtolemyII's actors. The links indicate communication between the elements. The large circles indicate the

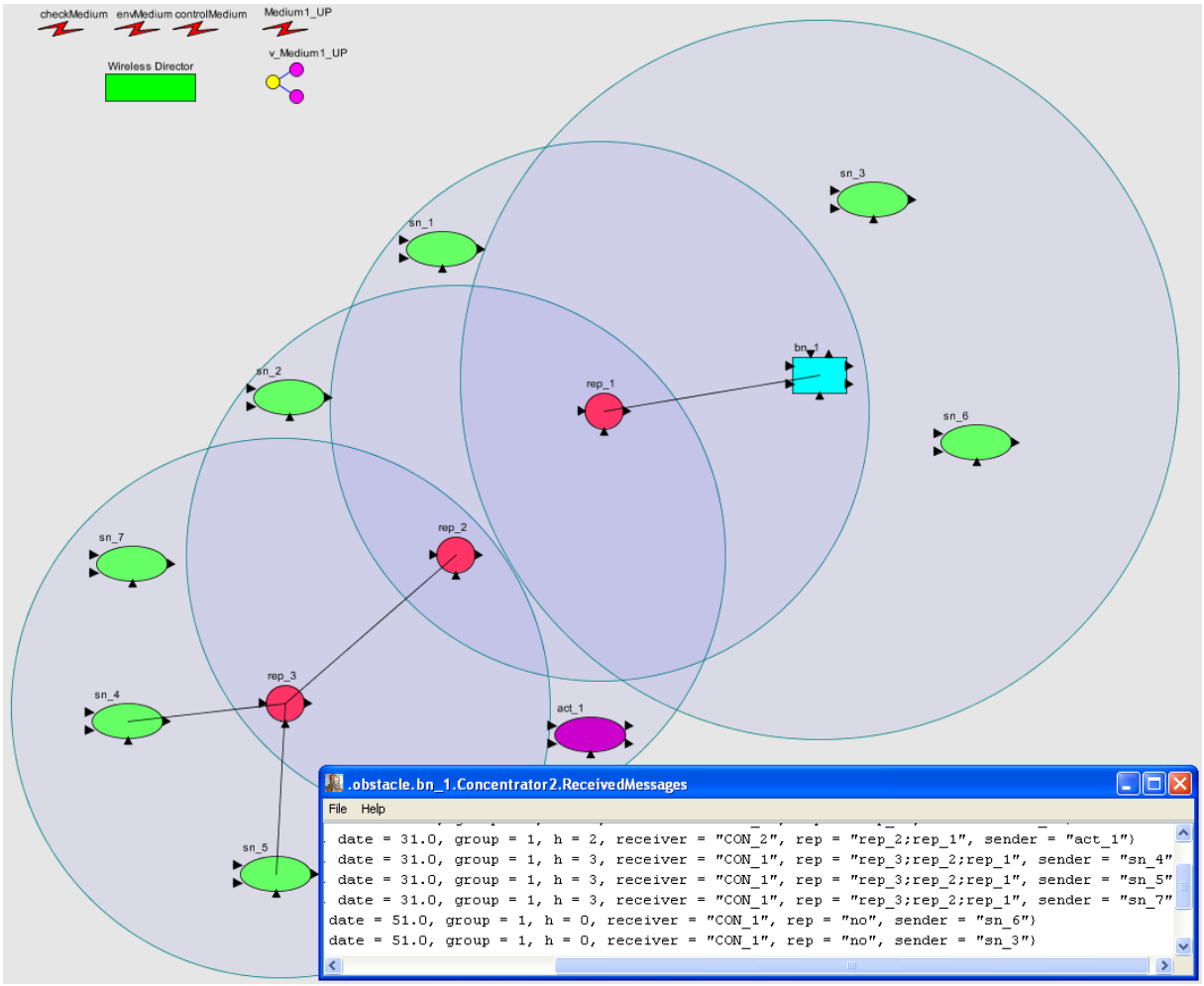

Fig. 12. Case study inside PtolemyII's simulation environment

scope of the repeaters and of the Boxa Nova. The bottom of the figure corresponds to a raw trace of the propagation of the input values, from sensors to Boxa Nova.

Several scenarios have been tested, first to verify that the systems works in normal condition, second to check what happens when a sensor or a repeater is down. More experiments are under reflexion to improve our simulation and take advantage of the benefits of this approach.

VII. CONCLUSION AND FUTURE WORKS

In this paper, a generic meta-model for modeling ubiquitous telecontrol systems and, specially telecontrol systems is proposed. A variant of this metamodel is specified to fit the needs of Terra Nova Energy.

The introduction of the Model Driven Engineering approach in this field allows us to exceed the contemplative dimension of models towards productive models. The presented case studies highlight this approach by suggesting a possible model for placement analysis for an example of TNE system, using transformations. It also shows how an instance may be transformed in order to obtain a model to be simulated.

From an unique model, we offer now two different classes of transformations. We would like to improve both of them and to validate them on various applications. We also would like to build new transformations in the direction of code generation, 3D visualization and simulation, test and verifications.

ACKNOWLEDGMENT

This work is supported by the city of Brest, "Brest Métropole Océane", and performed in partnership with Terra-Nova Energy. We thank them for their help.

REFERENCES

[1] A. Touil, J. Vareille, F. Lherminier, and P. Le Parc, "Modeling and analysing ubiquitous systems using mde approach," in *The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. UBICOMM 2010. Florence, Italy*, Oct. 2010.

[2] M. Weiser, "The computer for the 21st century," *Scientific American Special Issue on Communications, Computers, and Networks*, 1991.

[3] Terra Nova Energy, "Online Energy Intelligence," http://www.terra-nova-energy.com, 2012, [Online; accessed 09-jan-2012].

[4] P. Le Parc, A. Touil, and J. Vareille, "A model-driven approach for building ubiquitous applications," in *The Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies - UBICOMM 2009*, Oct. 2009.

[5] S. Kent, "Model driven engineering," *Lecture notes in computer science*, pp. 286–298, 2002.

[6] F. Fleurey, J. Steel, and B. Baudry, "Validation in model-driven engineering: testing model transformations," in *Workshop WS5 at the 7th International Conference on the UML, Lisbon, Portugal*, 2004.

[7] UC Berkeley, "Ptolemy Project Home Page," http://ptolemy.eecs.berkeley.edu/, 2012, [Online; accessed 09-jan-2012].

[8] S. Gerard, F. Terrier, and Y. Tanguy, "Using the model paradigm for real-time systems development: Accord/uml," *Lecture notes in computer science*, vol. 2426, pp. 260 – 269, 2002.

[9] D. Moody, "Graphical Entity Relationship Models: Towards a More User Understandable Representation of Data," *Lecture Notes in Computer Science*, vol. 1157, pp. 227–244, 1996.

[10] S. Cranefield and M. Purvis, "UML as an ontology modelling language," in *Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, vol. 212, 1999.

[11] SysML.org, "SysML Open Source Specification Project," http://www.wotug.org/occam/documentation/oc21refman.pdf, 2012, [Online; accessed 09-jan-2012].

[12] H. Espinoza, J. Medina, H. Dubois, S. Grard, and F. Terrier, "Towards a uml-based modelling standard for scheduability analysis of real-time systems," in *In proceedings of MARTES workshop at MODELS conference, Genova Italy*, 2006.

[13] F. Losilla, C. Vecente-Chicote, B. Alvarez, A. Iborra, and P. Sánchez, "Wireless sensor network application development: An architecture-centric mde approach," *Lecture Notes in Computer Science*, vol. 4758, p. 179, 2007.

[14] A. van Deursen, P. Klint, and J. Visser, "Domain-specific languages: an annotated bibliography," *SIGPLAN Notices*, vol. 35, no. 6, pp. 26–36, June 2000.

[15] K. Czarnecki and S. Helsen, "Classification of model transformation approaches," in *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, 2003.

[16] T. Alenljung and B. Lennartson, "Formal Verification of PLC Controlled Systems Using Sensor Graphs," in *Proceedings of the fifth annual IEEE international conference on Automation science and engineering*. The Institute of Electrical and Electronics Engineers Inc., 2009, pp. 164–170.

[17] A. Voronov and K. AAkesson, "Verification of process operations using model checking," in *CASE'09: Proceedings of the fifth annual IEEE international conference on Automation science and engineering*. The Institute of Electrical and Electronics Engineers Inc., 2009, pp. 415–420.

[18] S. Ghosh and S. Rao, "Sensor network design for smart highways," in *CASE'09: Proceedings of the fifth annual IEEE international conference on Automation science and engineering*. The Institute of Electrical and Electronics Engineers Inc., 2009, pp. 353–360.

[19] B. Combemale, X. Crgut, J.-P. Giacometti, P. Michel, and M. Pantel, "Introducing Simulation and Model Animation in the MDE Topcased Toolkit," in *4th European Congress EMBEDDED REAL TIME SOFTWARE (ERTS)*. Toulouse, France: SIA & SEE, Jan. 2008.

[20] National Instruments, "NI Labview - Improving the Productivity of Engineers and Scientists," http://www.ni.com/labview, 2012, [Online; accessed 09-jan-2012].

[21] MathWorks, "Simulink - imulation and Model-Based Design," http://www.mathworks.com/products/simulink/, 2012, [Online; accessed 09-jan-2012].

[22] Inria, "Scicos Homepage," http://www-rocq.inria.fr/scicos/, 2012, [Online; accessed 09-jan-2012].

[23] SGS-THOMSON Microelectronics ltd, "Occam 2.1 reference Manual," http://www.sysml.org/, 2012, [Online; accessed 09-jan-2012].

[24] P. Baldwin, S. Kohli, E. A. Lee, X. Liu, and Y. Zhao, "Visualsense: Visual modeling for wireless and sensor network systems," http://ptolemy.eecs.berkeley.edu/papers/05/visualsense/visualsense.pdf, Technical Memorandum UCB/ERL M05/25, University of California, Berkeley, USA, Tech. Rep., July 2005, [Online; accessed 09-jan-2012].

[25] M. Mernik, J. Hering, and A. Sloane, "Acm computing surveys," *When and how to develop Domain Specific Languages*, vol. 37, no. 4, pp. 316–344, 2005.

[26] R. Gronback, "Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit," *Addison-Wesley Professional*, 2009.

[27] I. C. Committee, "A consensus of the incose fellows," http://www.incose.org/practice/fellowsconsensus.aspx, International Council On Systems Engineering, Tech. Rep., 2012, [Online; accessed 09-jan-2012].

[28] P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, and Y. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards," *Computer Communications*, vol. 30, no. 7, pp. 1655–1695, 2007.

[29] X. Cheng, D. Du, L. Wang, and B. Xu, "Relay sensor placement in wireless sensor networks," *Wireless Networks*, vol. 14, no. 3, pp. 347–355, 2008.