A Modeling Approach for Marine Observatory

Charbel Geryes Aoun, Iyas Alloush, Yvon Kermarrec, Joel Champeau, Oussama Kassem Zein

Université Européenne de Bretagne, Telecom Bretagne, Institut Mines-Telecom, UMR CNRS 6285 Lab-STICC
Ecole Nationale Supérieure de Techniques Avancées, ENSTA Bretagne, Bretagne France
Lebanese University, Faculty of Sciences, Lebanon

E-mail: charbel.aoun@telecom-bretagne.eu, charbel.aoun@ensta-bretagne.fr, iyas.alloush@telecom-bretagne.eu, yvon.kermarrec@telecom-bretagne.eu, joel.champeau@ensta-bretagne.fr, oussama.zein@ul.edu.lb

Received: 14 November 2014 /Accepted: 15 January 2015 /Published: 28 February 2015

Abstract: Infrastructure of Marine Observatory (MO) is an UnderWater Sensor Networks (UW-SN) to perform collaborative monitoring tasks over a given area. This observation should take into consideration the environmental constraints since it may require specific tools, materials and devices (cables, servers, etc.). The logical and physical components that are used in these observatories provide data exchanged between the various devices of the environment (Smart Sensor, Data Fusion). These components provide new functionalities or services due to the long period running of the network. In this paper, we present our approach in extending the modeling languages to include new domain-specific concepts and constraints. Thus, we propose a meta-model that is used to generate a new design tool (ArchiMO). We illustrate our proposal with an example from the MO domain on object localization with several acoustics sensors. Additionally, we generate the corresponding simulation code for a standard network simulator using our self-developed domain-specific model compiler. Our approach helps to reduce the complexity and time of the design activity of a Marine Observatory. It provides a way to share the different viewpoints of the designers in the MO domain and obtain simulation results to estimate the network capabilities.

Keywords: Models, Verification, Simulation, Underwater Object Localization, Marine Observatories, Domain Specific Modeling Language.

1. Introduction

Sensor network is set of specialized sensors with a communications infrastructure designed to monitor data in different fields and relays this information in real-time to an Internet or specific network. Underwater sensor nodes must possess self-configuration capabilities, i.e., they must be able to coordinate their operation by exchanging configuration, location and movement information, and to relay monitored data to an onshore station. MO Based on UnderWater Acoustic Sensor Networks (UW-ASN) consists of a variable number of sensors that are deployed to gathering scientific data in collaborative monitoring missions [1]. To achieve this objective, sensors self-organize in an autonomous
network which can adapt to the characteristics of the ocean environment. Infrastructure of sensor networks is based on specific technologies (copper, fiber, wireless, etc.) to establish the connections between the different existing devices. Our research scope is in the first phase of a MO project: Marine e-Data Observatory Network (MeDON) [2]. MeDON contains different elements (Hydrophones, Fusion Servers, Object Localization Algorithms), and different communication protocols (e.g., REST, SOAP and proprietary ones) [2, 3]. The implementation of MO information system is considered as a complex distributed system [3]. We distinguish three sources of complexity: the complexity of the system itself, the specification and design activities, and the deployment. The complexity of the system under study [2, 3] is related to: (1) the architecture of the system (Distributed) which contains different elements from different sub systems (Underwater Sensor Network and the rest of the information system); (2) the interactions between the different elements of the information system and the core network that relies on standard protocols and transactions; (3) the large number of sensors (Hydrophones) and servers existing on the networks. Taking into account the environmental constraints of MO context, the deployment phase of MeDON is hard. The complexity of the process activities is related to: (1) high level services and network architecture are coexistent view-points in the system; (2) several constraints must be taken into account during the design; (3) several levels of abstraction are necessary to understand the system.

Due to these aspects, designing complex distributed systems consumes considerable time since big volume of objects (e.g., physical, logical) and elements should be related together according to a specific communication protocol or scenario. According to [2, 3], the complexity of the design is also emphasize of: (1) the different domains of experience (Business Process Modeling, Information System Modeling, and the Underlying Infrastructure Modeling) that are required from the designer(s) to be able to model and describe such system; (2) distributed Software Structure of MeDON Information System (see Fig. 1) since each component (e.g., Data Fusion Server, Smart Sensor, etc.) is responsible to perform set of specific tasks.

The deployment of a set of sensors (Sensor Network) is a costly operation due to: the necessary equipments such as specific boats, marine cables, Sensors (Hydrophones), Data Fusion Servers, and experts in diving, etc. Additionally, we cannot ignore that the deployment operation is risky and the placement of sensors and servers should be in the right position where an error in meters may cause larger bit-error rates in the communication channel (Cables). Thus, an integration between the information system (Sensors, Servers) and the communication system (e.g., IMS) [4] is needed since the IP Multimedia Services (IMS) is used to ease the integration with the internet.

The large number of sensors that are communicating with the set of fusion servers results in a more complex design [5]. The delivery scenario between the different nodes on a network is done through copper-based network or fiber-based network [6]. Bandwidths reflect the data rate (Mb/s) which means the speed of the data transferred on the network between the different devices. Reducing the data rate will result in connection delay between the source and the destination nodes. In [7], the delivered bandwidths (speed of data) decreases with the increase in the length of the copper cable. These returns to the fact that the longer the cable (copper or fiber) is, the more bit error rate in the transmission channel which means more retransmissions between the network peers or between the two ends of the connection [8]. Thus, the connection delay in a network connection is proportional to the increasing of the length of the copper cable. This affects negatively the network performance. We consider that the time to obtain useful results from the MeDON system is the resultant of: the time of the operations of deployment and the design time. Thus, our research question is: how to improve the time of the design phase, reduce the complexity of the deployment and maintenance phase, enhance the network performance, and enhance the system services and its functionalities?

So in this context, our objective is to provide a design tool to the designers of MO that helps them to model their design taking into consideration: (1) reducing the time of development process by having specific concepts, elements and relations that assist the designer to work more properly during the design phase; (2) managing the complexity that will face the designer.

In this paper, we propose a modeling design tool (ArchMo) that helps to manage the complexity and detects design modeling errors during the design time. This tool provides the designer with a set of reusable graphical elements, relations and concepts that contains ArchiMate [9] and the MO concepts. Our approach is based on the concept of domain specific modeling languages (DSMLs), which relies on Model Driven Engineers (MDE) fundamentals [10]. In order to model MO systems, we choose ArchiMate modeling language as it relies on Enterprise Architecture (EA) framework [11, 12] that allows describing a wide range of domains [13]. We use meta-models to generate the tools that are specified to different development activities using Eclipse Modeling Framework (EMF) [14].

ArchiMate is designed/defined/intended to model systems from the IT domain [9]. Our proposal extends the ArchiMate meta-model (Abstract and Concrete Syntax) to add new concepts and constraints specific for MO into ArchiMate. On one hand, a main feature of (EA) frameworks is sharing the multiple viewpoints [13]. This reduces the complexity of one view (Business layer) to a manageable size. EA frameworks introduce interoperability issues between views and their dedicated software [13]. On the other
hand, our proposed DSML is extensible, where the developers may extend it and add new concepts and standards according to the progress and needs in MO domain.

Linking our MO meta-model to the IP Multimedia Subsystem (IMS) one (proposed previously in [15]) helps to integrate the different smart sensors of the sensor network to the rest of the information system through the core network [4]. We apply our design model to a model compiler to generate simulation code that runs directly in NS-3 network simulator [16]. The resulted simulation helps to obtain an early validation of the design model before any prototype or deployment.

The paper content is organized as follow: in Section V, we present the related work that is connected to the design tools. Section II-A presents MO project. In section II-B, we present MDE fundamentals, DSMLs, ArchiMate, and our proposal meta-model for the MO/MeDON. Section III explains the abstract syntax, concrete syntax and semantics of the proposed DSML. In Section IV, we present the generated design tool and the simulation approach. In section VI, we conclude and discuss our future work.

2. Context

2.1. Marine Observatories

Underwater Sensor Networks that aim to environmental data acquisition will play an essential role in the development of future large data acquisition systems [17]. They allow the data to be exchanged and processed between the different devices (Servers, Sensors). On all these devices, we can have software components to process and store the data. An example about MO is the project Marine e-Data Observatory Network (MeDON).

In this context, the designer should be able to include acoustic sensors that are connected to the Y fusion servers as shown in (see Fig. 1). These servers analyze and processes the acoustic data acquired by the hydrophones then diffuse them on the network. Servers store their data on the same database. The Database server provides the processed and filtered data to the web server where the configuration of a web application is done. Thus, the web server broadcasts the information detected by the hydrophones such the dolphin acoustic sounds to the web clients through a graphical interface.

2.2. Model Driven Engineering (MDE) and Domain Specific Modeling Languages (DSML)

MDE [18] is a software development method which focuses on creating and exploiting domain models. It allows the exploitation of models to simulate, estimate, understand, communicate, and produce code. MDE helps to manage complexity thanks to the modeling concept and model transformations.

Modeling helps to describe the design in a high abstract way and model transformation helps to have a generated design tool.

A meta-model defines a language for describing a Specific Domain of interest [10]. In our approach, modeling tools contains the specific constraints and represents the concepts, elements and relations that are defined in the meta-model. It makes it possible to instantiate large number of models that conform to it like in programming languages [19]; numerous of programs can be implemented relying on a specific programming language (e.g., C, C++, Java, etc.).

Eclipse IDE provides a powerful environment that relies on EMF which facilitates the modeling/meta-modeling activities, it supports many model transformation languages as well. Model transformations help us to generate design tools and simulation programs directly and automatically considering meta-models and model instances. Every model transformation depends on a set of rules that describe and control the transformation process. The transformation rules may map models that conform to different meta-models, such as ATL [20], or map between different domains using one meta-model for the source model to generate texts/codes (e.g., XPAND [21]). We conclude that the main benefit or objective of the model transformation concept is the ability of changing from technological space to another technological space (e.g., from xml extension to java or c++).

In our case (see Fig. 2), the input model represents the design of highly abstract level, and the meta-model is the extended ArchiMate meta-model which represents the abstract syntax [18, 15]. Our code generation is an automated process that links directly the design model to the simulation scripts [16]. Thus, it helps to reduce the time of the implementations for large simulation programs, and it minimizes the implementation errors.
1) Domain-Specific Modeling Languages: Domain-Specific Modeling Languages (DSMLs) [22] enable designers from different domains and backgrounds to participate in software development tasks and to specify their own needs in a specific context using domain concepts, objects, classes, elements and entities. A DSML [23] is comprised of three components: abstract syntax, concrete syntax, and semantics. The abstract syntax defines modeling concepts and their relationships. There are several kinds of concrete syntaxes: visual, XML- based, textual, etc [24]. The concrete syntax is associated with a set of rules which defines the representation of the abstract syntax. Semantics describe the meaning of a model and are related to the abstract syntax. They are well-formed rules for the model and are used to constrain the concrete syntax [23]. Historically, data fusion methods were developed primarily for military applications (e.g., radars tracking a variable object) since fused data from multiple sensors provides several advantages over data from a single sensor [5]. A methodology as combining set of observations would result in an improved estimate of the target position by minimizing the error that may occur by calculating the position of a variable object. Concepts such information fusion and sensors networks have dominated the research and specially the military research. We distinguish different architecture for data fusion as follows [5, 25]: (1) centralized fusion; (2) hierarchical fusion without feedback; (3) hierarchical fusion with feedback; (4) distributed fusion. According to our context, we have selected the most complex architecture (distributed) to model it, and then simulate it as presented in section IV. During the design activity, set of constraints and restrictions should be respected by the designer in order to model distributed system architecture like in [5]. We will present them in the next section.

In general, errors caught during the design cycle are much less time consuming to identify and correct than those found during the testing phase. In order to avoid errors in the design activity, we have implemented constraints that are defined in the abstract syntax of the language (Meta-Model) (see Fig. 2). The concrete syntax that is associated with these added constraints can be implemented in the design tool such as 'ArchiMO' tool in our context. This tool is generated relying on Eclipse-EMF (Tool Generation Concept thanks to Model Transformations).

Modeling languages are used to describe a system with high level of abstraction [24]. For MeDON/MO, and in relation with our objectives, we describe distributed systems. Thus, we selected ArchiMate modeling language that can describe the systems. Fig. 2. Extending business and application layers of ArchiMate: proposal of MO Meta-Model.
from IT domain and share multiple viewpoints during the design as it relies on TOGAF framework [18].

3. Marine Observatory Meta-model

In our context we based our work on ArchiMate which relies on Enterprise Architecture (EA) framework [12, 18]. It presents the system design into three layers: business, application, and technology. In our approach, we present these layers in the following way:

1) Business layer: specifies the end-user functions and actors. It describes the service activities as perceived by the end-user, and the flows between them;

2) Application layer: specifies the functions and software components of the service. It describes the capability of the system under study, and the way of communicating the different existing elements and relations in order to perform its tasks;

3) Technology layer: specifies the functions, topology, hardware elements, and signaling protocols of the underlying platform. It describes the execution platform that offers functions to be used by the functions of the application layer.

In general, a meta-model of DSL represents the concepts/operations and constraints that belong to the domain specificities (MO in our case). In this section, we present our contribution of a new meta-model (Abstract Syntax), concrete syntax, and design tool. The concepts of ArchiMate meta-model in [26] (see Fig. 2) are extended to represent the domain specificities (MO in our case). In this section, we present these layers in the following way:

1) Business layer: specifies the end-user functions and actors. It describes the service activities as perceived by the end-user, and the flows between them;

2) Application layer: specifies the functions and software components of the service. It describes the capability of the system under study, and the way of communicating the different existing elements and relations in order to perform its tasks;

3) Technology layer: specifies the functions, topology, hardware elements, and signaling protocols of the underlying platform. It describes the execution platform that offers functions to be used by the functions of the application layer.

In general, a meta-model of DSL represents the concepts/operations and constraints that belong to the domain specificities (MO in our case). In this section, we present our contribution of a new meta-model (Abstract Syntax), concrete syntax, and design tool. The concepts of ArchiMate meta-model in [26] (see Fig. 2) are extended to represent the domain specificities of MO.

Our proposed meta-model is composed of two views: one for the business layer, and another for the application layer. Regarding to the technology layer, we rely on a meta-model for IMS underlying platform that we proposed before in (see Fig. 2). We present the proposed Meta-Model as the following: (1) Business Layer (see Fig. 2): we have extended the business actor of ArchiMate by two new concepts: the Smart Sensor, and the Data Fusion (the red classes or concepts in the Business Layer in Fig. 2). We have extended also the business function of ArchiMate by four new concepts: the Algorithm Selection, Data Transmission, Object Localization, and the Data Acquisition (the red classes or concepts in the Business Layer in Fig. 2). Smart sensor is responsible of the Data Acquisition activity, while the data fusion is responsible of the activities: Algorithm Selection (performs a procedure to select the proper algorithm), Data Transmission (to transmit the data between the different fusion components), Object Localization (to make the necessary actions that localize an object); (2) Application Layer (see Fig. 2): we have extended the application component by two elements: the Data Fusion System, and the Smart Sensor System (the red classes or concepts in the Application Layer in Fig. 2). We have extended also the application function of ArchiMate by seven new concepts: the Manage Resources, Coordinates Storage Handling, Compute Coordinates, Transmit Localization Data, Voice Streaming, Video Streaming, and Inform Server (the red classes or concepts in the Application Layer in Fig. 2). The Data Fusion System is responsible to perform the following application functions: Manage Resources (to manage the resources needed for the algorithm execution), Coordinates Storage Handling (to store the coordinates correlated with time), Compute Coordinates (to compute the coordinates according to a specific algorithm selected previously by the Data Fusion actor), and Transmit Localization Data (to exchange information between the fusion servers/systems). The Smart Sensor System is responsible to perform the following functions: Inform Server (to inform the fusion server about any detection of a specific object), Voice Streaming (this function is useful for the case of hydrophones), and Video Streaming (this function is useful for the camera sensors).

A new specific constraint (for the Smart Sensor and Data Fusion elements) is added to the abstract syntax. This constraint is a new Relationship that is used only to connect a Smart Sensor element to a Data Fusion element (see Fig. 3). We consider this new type of relationships in the business layer accompanied with the constraints as logical. According to our proposed meta-model extension in a previous section, the designer will need to connect Smart Sensors to Data Fusion servers. The designer will never be able to connect logically these two devices without giving proper values by respecting the existing constraint between Smart Sensor and Data Fusion concepts. The new meta-model enables us to generate and develop design tools that are coherent with Archi [27]. They contain additional concepts, elements, constraints and relations that are specific to the MO domain and for data fusion concepts [5].

![Fig. 3. ArchiMate Relationshíoes.](image)

Relying on the distributed fusion architecture (DFA) in [5], our meta-model (see Fig. 2 and Fig. 4), and according to [2], we distinguish the following constraints: for Smart Sensor:(1) communication between two Smart Sensor elements is not allowed; (2) communication between Smart Sensor and Data Fusion element is allowed; (3) communication
between Smart Sensor and Data Fusion is allowed according to a specific constraint; (4) Smart Sensor is only allowed to be related to the Data Acquisition function. For DataFusion: (1) communication between two Data Fusion elements is allowed; (2) Data Fusion is only allowed to be related to Algorithm Selection, Data Transmission and Object Localization functions.

Fig. 4. Communication constraint between Smart Sensor and Data Fusion.

Like in ArchiMate [12, 18] our proposed meta-model is composed of two views: one for the business layer, and another for the application layer. Regarding the technology layer, we rely on a meta-model for IMS that provides an underlying platform in [4] to integrate the information system with the core network.

For each extended concept, element or Relationship, a graphical view (belonging to the concrete syntax) should be defined [13]. Our proposed concrete syntax are shown in the palette of the business layer (the red circles in Fig. 5), application layer (the red circles in Fig. 6), and Relationship (the red circle in Fig. 7). These palettes are coherent with MO specific concepts and relations from which the designer can select, drag and drop the desired ones.

ArchiMate contains different types of relationships such as association, assignment, etc (see Fig. 3). We have specialized the definition of the relationships regarding the new added concepts. In our context, we have defined the association relationship for the smart sensor according to the constraints of DFA (e.g., smart sensor could be only associated to the data fusion). Furthermore, we have defined the assignment relationship for the smart sensor according to the constraint of MO [2] (e.g., Smart Sensor could be only assigned to the Data Acquisition). We have extended the association relationship by a new one which is the SmartSensor and DataFusion Relation. The new extended Relationship is shown in red (see Fig. 3).

Fig. 5. Generated Business Layer in Palette.

Fig. 6. Generated Application Layer in Palette.

Fig. 7. New Relationship in Palette.
For each ArchiMate element, we can define the relationship type that is allowed with this related element. The encoding ArchiMate relationship is based on an enumeration for all the possible types. The key values are: for example ‘o’ for association and ‘i’ for the assignment relationship. We have added the key value ‘x’ for the new extended SmartSensor and DataFusion Relationship. For the business and application layers, we have implemented the keys relative to the selected relationships and mainly the associated constraints in Java code regarding to our proposed extended meta-model (see Fig. 2 and Fig. 4). This implementation is the grammar of the new proposed DSML.

In order to have a graphical view for the added constraints and elements, we have generated the design tool ArchiMO relying on eclipse EMF. This design tool helps the designer to model the system in a highly abstract way by drag and drop the elements and relations from the palette. During the model edition, all the constraints specified for the MO extension are checked: (1) prevent the designer to associate two Smart Sensor elements together; (2) the designer is able to associate a Smart Sensor element to Data Fusion, Business Actor or other actors (see Fig. 8); (3) require the designer to enter a proper value in order to associate a Smart Sensor to a Data Fusion using the SmartSensor and DataFusion relationship (see Fig. 9); (4) the assignment is only allowed from SmartSensor to the DataAcquisition function (see Fig. 8). Concerning the Data Fusion element: (1) the association between two Data Fusion elements is allowed; (2) the designer is able to associate Data Fusion element to Smart Sensor element (see Fig. 8); (3) the designer is not able to associate a Data Fusion to a Smart Sensor using the using the SmartSensor and DataFusion relationship in case he didn’t answer a value between the minimum and the maximum values (see Fig. 10); (4) the designer is able only to assign the Data Fusion to the Algorithm Selection, Data Transmission and Object Localization functions (see Fig. 8).

ArchiMO tool considers different domains of experience, each domain expert works in a specific layer (Business, Application or Technology) as the model created in section VI. Our contribution replies to the concerns that we have mentioned in V as it: (C1) prevent syntax and relation errors that can be made during the design activity; (C2) provides three layers according to each domain specificity; (C3) extends an open, standard, and classical design tool to have a specific one like ArchiMO; (C4) deploys different physical components (Sensors and Servers), and logical components such acquisition/localization algorithms.

4. Object Localization Case Study

In order to validate our proposed tool, we used it to model the application of Object Localization using the different new elements that are proposed in the meta-model (see Fig. 2). Then we applied the design model to a model compiler (see Fig. 11) that we have developed to perform some error checks and generate automatically simulation code for NS-3. This simulation code runs in NS-3 tool that is a standard and classical simulator in the networking domain.
4.1. Design Model

We have modeled a system that localizes an underwater object using our generated design tool ArchiMO. In order to localize this object, sensors should be connected to data fusion servers. We have applied the distributed fusion architecture (DFA) [5] for this design.

The design model is composed of three views regarding to the layers of ArchiMate (see Fig. 12): Business, Application, and Technology. In Fig. 12 we present parts of the large model that is designed by ArchiMO. The model contains behavioral elements, in the business layer (see Fig. 12) shows the first activity of the smart sensor which is the dolphin detection1, etc. These activities are assigned to their proper smart sensors. Using the new extended SmartSensor and DataFusion Relationship these smart sensors are associated with the different data fusion servers and smart sensors that are required in the DFA.

4.2. Compilation and Simulation

The design tool ArchiMO generates an XMI file to represent the graphical design. This helps to conduct the design model to other tools. We use the XMI file as an input to our self-developed domain-specific model compiler to generate the simulation code (see Fig. 11). This hides complexity of constructing simulation programs from the designer and saves considerable time of the development process as it hides the specificities of the network simulator. The network simulator needs to be configured through a specific language and concepts using specific libraries that are not of the same abstraction level of the modeling tool and are not necessarily included in the design model. The code generator needs both the meta-model including the abstract syntax of DSML for MO, and the input model that is generated from the design tool.

The XPAND template in (Fig. 11) contains the mapping rules between the model elements and their representations in NS-3[16].

We have run the generated code in NS-3 (version 3.13), and the results of compilation and the execution of the simulation program shows no errors. Traces and logs (e.g., PCAP files) were generated to analyze the simulation outputs.

Fig. 13 shows the architecture of the system design that is generated by NS-3 for the mentioned design model. NS-3 generated hardware representations (Nodes, Interfaces, Wires) for the elements of the design model and the blue colored stream represents a message that is exchanged between two nodes in a fixed moment. This confirms that the behavioral elements were mapped as well.

We have used our approach in different application domains and network simulators (Video Conferencing System [15, 16], and MO context). The common design concept between all cases is the underlying platform (IMS) that represents the Platform Specific Model (PSM) [24].

In other words, considering using one tool (e.g., NS-3), we could change the application domain relying on ArchiMate and our extensions (DSMLs) by fixing the underlying platform that is represented in the technology layer. This confirms that our proposed design tool (ArchiMO) creates models that follow the same meta-model and domain-specific concepts/constraints.

5. Related Work

In this section, we present the related work in connection with the design tools.

In relation with the concept of Architectural Description Languages (ADLs) [28] and their design tools; we are interested in the following concerns that we shall specify and analyze in this section: (C1) preventing errors during design by invoking grammar or syntax of language; (C2) multiple viewpoints that are represented in the architectural description since a viewpoint is a basic product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns; (C3) extensibility of design tool and having specific views to tasks and roles; (C4) heterogeneity of components and communications; (C5) testing/execution platform.

According to the preventing errors concern, the design tool prevents errors during design activity that may be made by the designer, rather than correct them after deploy. This error prevention is available in [29-31]. Our approach also prevents this error by invoking the abstract syntax (Our Proposed Meta-Model) where we have defined and added our specific constraints and relations.

Concerning the multiple viewpoints concern, the design tool provides different viewpoints for the designers according to their specialties and domains of experience.
In [29-31], the design tool provides only one viewpoint in order to fit only software development tasks. This design tool does not provide the ability to share the design between different designers. This ability is essential since different domain experts can share the same design. Our approach considers this issue thanks to the different layers of EA standard that separates between perspectives since each layer is dedicated for a domain expert. Regarding the extensibility concern, the extension of a meta-model allows the extension of a design tool by adding new concepts and constraints to it [29, 30]. It’s realized in our approach by extending the ArchiMate meta-model by new elements and constraints, then generating a new design tool that contains the concrete syntax inside the palettes. These palettes contain the new added components like in [4, 13]. Concerning the heterogeneity concern, the existence of components and communications that are related to different contexts and activities. We are facing this heterogeneity concern in the software components and models in [29-31]. In our approach, we are facing this heterogeneity (e.g., Smart Sensor different than Data Fusion) but we are able to solve this issue by...
relating the different existing devices together according to a specific scenario.

According to the execution test platform concern, the de- signer in [29-31] is not able to test and verify his models or instances on an executable platform (e.g., IMS). Relying on [4] in our approach, we are able to test our proposed model (see section VI).

6. Conclusion and Future Work

In this paper, we have presented a Domain Specific Modeling Language (DSML) for MO context. Our approach is based on extending the ArchiMate meta-model relying on MDE fundamentals. We have proposed a new design tool (ArchiMO) that is generated from the extended MO meta-model which considers the domain-specific concepts and constraints.

ArchiMO helps the designer by avoiding making design errors earlier than verification and code generation activities. It assists also by representing the domain-specific concepts of MO thanks to DSML fundamentals. We rely on a standard and open tool (Archi) and developing it by extending the modeling language and Java implementations. Another advantage is the extensibility of our proposed meta-model/tool. The developers may extend it and add new concepts and standards according to the progress in MO domain. ArchiMO provides the reusability of the added MO and Data Fusion concepts (e.g., Smart Sensor, Data Fusion, etc) in different applications, activities, models or instances. ArchiMO reduces the time of the design activity as well, by having the specific elements and constraints in the palette of this tool. Additionally, we conserve the standard constraints in the abstract syntax (Meta- Model) of ArchiMate since the new added elements inherits concepts from standard ArchiMate elements. Representing the connections logically between the different smart sensors and fusion servers helps to improve the understanding of the network constraints in a business level. Thus, it improves the communication of the design between the different stake- holders. Moreover, it provides a way to link between the design constraints and softgoals [32] as the design constraints are of a highly abstract level (business layer) and platform-independent.

On the other side, representing and meta-modeling the do- main knowledge is itself a hard job that needs experience and high level of accuracy, especially when setting the grammar of the DSML according to the meta-model constraints.

As perspectives, we will extend our meta-model in order to satisfy and cover the most possible required operations, concepts and activities in the context of MO. This extension could be the addition of specific constraints and features in the context of MO during the design activity.

References

[7]. S. Morgan, Actual vs. advertised broadband speeds, InterConnect Communications Ltd, March 2011.
[16]. I. Alloush, Y. Kermarrec, and S. Rouvrais, A generalized model trans- formation approach to link


[22]. M. T. Zekai Demirezen, Barrett R. Bryant, Domain specific modeling language design space analysis, The University of Alabama at Birmingham, USA, June 2011.


___________________

2015 Copyright ©, International Frequency Sensor Association (IFSA) Publishing, S. L. All rights reserved. (http://www.sensorsportal.com)