A Domain-Specific Framework for Creating Early Trusted Underwater Systems Relying on Enterprise Architecture

Iyas Alloush*, Charbel Geryes Aoun‡*, Yvon Kermarrec*, Siegfried Rouvrais*§
*Université européenne de Bretagne
Telecom Bretagne, Institut Mines-Telecom
Email: firstname.lastname@telecom-bretagne.eu
§IRISA
‡ENSTA
Email: firstname.lastname@ensta-bretagne.fr

***Abstract*—In the context of service creation and development according to functional and non-functional requirements (NFR), Service Creation Environments facilitate the creation of complex services and play a major role for the software industry. In our scope, we are concerned with two activities: design, and verification according to NFRs related to performance and QoS requirements. To reach our objective, we aim at developing a domain-specific framework (networking domain) that uses a chain of existing "off-the-shelf" tools integrated together from the design to the verification activities. Model Driven Engineering approach facilitates our task on reducing complexity through modeling and code generation (using XPAND-Eclipse), to obtain an auto-generated simulation scenario ready to run directly in NS-3 simulator. In this paper, we propose a new meta-model that extends the Open Group modeling language ArchiMate to provide a domain-specific modeling language for the Deep Sea Observatories (DSO). We instantiate, as a case study, DSO model with identification and localization functions from this language, and apply it to our framework that relies on IMS platform to run the service model. These functions can be orchestrated with other services (e.g. military or civil reaction) or interconnected with other SOSystems. This illustrates, on one hand, our approach in relying on Enterprise Architecture (EA) framework, that respects multiple-views, perspectives of stakeholders, and domain specificities. On the other hand, it shows the reusability of our framework by changing applications from different domains: Video Conference as a Telecom Service, and Localizations for DSO.**

I. Introduction

In the context of service creation [1] and development in the domain of distributed systems, services are to be delivered on time and should respect the expected quality requirements [2]. Creating complex services consumes time and is subject to errors and quality flaws. This reflects negatively in the time to market competition factor and client intensions. Service Creation Environments (SCE) help to manage and support the different activities and phases of service creation taking into consideration the following challenges: reducing complexity, increasing reusability of tools at different abstraction levels, and relying on efficient automation of activities during the service creation process [1].

Our global objective is to provide the different stakeholders that are involved in the activities of the design phase with tools that help them to model and evaluate their design. The application scopes that we work on are in the design phase of Telecom Services (TS) ([3], [4]), and Underwater Localizations (UL) in the context of Deep Sea Observatories (DSO) [5]. Our research scope is in the first phase of an DSO project: Marine e-Data Observatory Network (MeDON)[1]. This initial phase aims at insuring the quality and experimenting the platform before it is deployed.

The DSO processes take long periods (months or years) to collect and analyze data. The mission to deploy the sensor network platform may be far from coasts, takes long time, and requires specific ships and tools to move and set the equipment with qualified crew. Additionally, the deployment of DSO equipments should take into consideration the difficult environmental constraints (depth, temperature, obstacles, etc). These constraints impact on the communication conditions where, for example, the length of networking cables influences (error rates, delays, packet drops, etc) the data transmitted. If floating access points are connected to the deep underwater sensors by cables, how to deploy these access points according to wireless channel conditions (different scenarios) to communicate with other sensors or nodes. Such concerns should be taken into consideration to insure that the data collected from the sensors are received correctly and with acceptable rates according to the communication channel conditions ([6], [7]). In this context, our research question is: **How to evaluate the design model (communication wise) earlier than the implementation phase where software and hardware elements are deployed on site?**

In the services of telecommunication and localizations (context of DSOs), the design contains two aspects: behavioral that is related to the system functionality, and structural that is related to the system components where functions should be executed. In order to detect design errors and quality flaws,

[1]MeDON official website: http://www.medon.info/

we rely on simulation approach and tools such as network simulators that enable a designer to perform simulations under different scenarios and obtain qualified outputs. We use classical tools that are widely used (NS-3 and OPNET), and they provide users with trusted results in the network domain.

Nowadays, the IP Multimedia Subsystem (IMS) [8] offers an advanced IP infrastructure that provides various added-value services that are useful for the DSOs (e.g; storing video streams and replaying them according to requests, detection data transmissions, etc) with an advanced level of provisioning. In order to enable the designer to rely on IMS platform and to take advantage from its features, we have proposed an IMS meta-model (MM) [9]. This meta-model helps to generate design tools thanks to Eclipse Modeling Framework (EMF). The design tool helps to prevent syntax errors thanks to the concrete syntax. In order to cover the rest of the errors: behavioral and performance-quality flaws, we have proposed a way to bridge the gap between design modeling and simulation activities ([10], [11]). Simulation provides us with the ability to obtain measures/traces (e.g. end-to-end delays, rates, and packet drops, etc) that are helpful to evaluate the design earlier than the implementation phase according to performance non-functional (PNFRs) and Quality of Service (QoS) requirements ([4], [12]). These measures/traces can be analyzed using many tools such: Wireshark (we use it in this paper), MatLab, etc.

Our objective in this paper is to propose a new meta-model in the context of DSOs. This meta-model will generate DSO-specific design tool. In this meta-model, we introduce elements that are needed when designing an DSO model such as data fusion functions/elements [13] to our proposed DSML. Additionally, we provide a solution for the interoperability between the different layers of a design during the simulation runtime.

We keep the technology layer extended meta-model for IMS [9], and experiment the replacement of the application (context) layer model. We have represented previously video conferencing system in the application layer ([4], [11]). In this paper, we replace it by another model from different context (DSO). This highlights the reusability of our approach for different applications.

In other words, our approach presents a way to use core-networks in the different applications that are from different domains. It provides a method to compile (syntax check, parsing, and generating executable code for NS-3) the design model [14] in order to obtain simulation scenario that is complex, and time consuming to be implemented by human efforts. The criteria to do that is to use a Service-Oriented core-networks (such as IMS in our case) that are represented in the modeling language.

We illustrate our implementations for the framework with two case-studies: a Video Conference Service relying on IMS, and a localization application (DSO context) relying on the same IMS meta-model that was proposed previously in [9]. Both of these cases are applied to the framework [11] that contains generation of simulation scenarios ready to run directly in the NS-3 simulator.

In Section II, we present the related work that is connected the Service Creation Environments. Section III presents the service creation activity and our approach to detect design errors and reduce its complexity. In section IV, we present MDE fundamentals and related tools. Section V presents the localization techniques and focuses on the relation with the design language in our approach (subsection. V-A). It also explains the DSMLs shortly and contains our proposal for the DSO/MeDON meta-model (subsection. V-B). In Section VI, we present the simulation approach. Section VII presents our method to generate simulation code. In section VIII, we present our example for the DSO/MeDON case-study and analyze the results. Then we conclude and discuss our future work in section IX.

II. Related work

In this section, we present the related work in connection with service creation environments and frameworks, in relation to the design and simulation approaches.

A service engineering frameworks [1] include different activities and phases. In our scope, we consider the design phase and we propose the addition of an early verification activity before the implementation phase ([3], [11]). In relation with concepts of: (1) software frameworks; (2) architectural description languages [15]; (3) types of requirements ([2], [16]); (4) and the requirements of service creation environments in the service engineering framework [1]; we are interested in the following concerns that we shall examine in details in this section:

1) Multiple views are represented in the architectural description;
2) Hiding complexity from the SCE user;
3) The reusability of the service creation framework in different applications and contexts;
4) The extensibility of the framework;
5) The consideration of the behavioral and structural elements in the architectural description;
6) Tools and platforms that are used.

- According to the Multiple views concern, SCEs may differ from each other according to their users. In ([17], [18], [19]), in the context of value-added TSs, the SCE is used by the end-user who can customize, provision his service from his terminal. Second case, where SCE users are designers who are not the target of the service, as in ([20], [21], [22], [23], [24]). In this case, the designer relies on modeling tools that ease the design activity and manage complexity thanks to the MDE fundamentals. However, in both two cases, only one view that fits the stakeholder tasks is provided by the design tool, as the design tool does not provide the ability to share the design between different stakeholders or designers. Our approach considers this point thanks to the different layers of EA standard, that separate between domain specificities and perspectives.
- The second concern is hiding complexity. Hiding complexity from the SCE-user's reduces the need to the

detailed specification, and minimizes the errors during the design activity. Design tools play an important role in reducing complexity, as they use modeling interfaces that are followed by code generators as in ([17], [24]). These code generators provide scripts which describe the service behaviors as in ([17], [18], [19], [23]). Then relying on a middle ware they can use APIs that are supported by the underlying platform such as IMS ([18], [24]) or an open service architecture (OSA) [23].

- Regarding to the reusability concern, the designer in ([23], [24], [18], [19], [17]) is not able to use the underlying platform (e.g. IMS) or the OSA with different applications that belong to different domains. A platform like IMS provides functions that can be used with different contexts (e.g. DSOs, value-added telecom services). For example, IMS can be used to exchange messages between terminals (e.g. cameras, hydrophones, smart sensors, and Fusion Servers, etc) and store video streams to replay them to other terminals.
  In our approach, we rely on ArchiMate language [25] for the design activity. ArchiMate[2] relies on Enterprise Architecture (EA) framework ([26], [27]) which has three layers (business, application, and technology). These layers decompose the design according to different viewpoints (e.g. business process description is different from IMS functional description). A way of using these multiple views is presented in [9] (video conference service): the business layer can be used by end-users, application layer describes the system applications (capabilities) and APIs, and the technology layer describes the underlying platform (e.g. IMS).
  Like in [21], and relying on Eclipse IDE and Model Driven Engineering (MDE) [28] discipline, we can reduce complexity by generating design tool from the syntax (abstract and concrete). As the tool provides the ability to model the service specifications in multi-layered architecture views, and it also offers help to prevent syntax errors during the design time (e.g. prevents designer from connecting function to another through assignment relationship). This design tool [3] conforms to the EA framework as it is generated from the meta-model of ArchiMate after extending it with our DSMLs [9].
- Regarding to the extensibility concern, in ([22], [20], [21]), the authors rely on meta-modeling and model transformations from MDE to generate tools and simulation codes. Extending the meta-models makes it possible to develop/extend the generated tools by adding new concepts and constraints. Relying on ArchiMate and MDE makes it possible for us as well. This is because ArchiMate is a language that respects the object oriented approach where inheritance [29] is available (e.g. specialization relationship). Relying on this concept, one can extend the tools as they are generated from models (as we did recently in [9], [12], [30]).

[2]ArchiMate: http://www.opengroup.org/subjectareas/enterprise/archimate

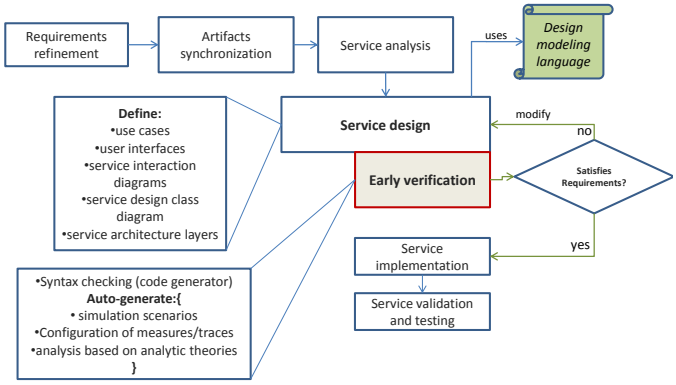

Fig. 1. Our approach: Extending the design activity by early verification in the service development framework, inspired from [1]

- Concerning elements of architectural description (fifth concern in our initial list), all of the previously mentioned related works consider only the behavioral description of the service. When the design contains an underlying platform like IMS from the networking domain, the network simulation is needed. Network simulators require topologies and nodes in the scenario of the simulation, which is needed to be considered in the design language. Therefore, in our approach, we have extended the technology layer of ArchiMate to include the nodes, protocols, functions, and topology constraints of IMS ([9], [10]). This enables us to make verifications according to performance NFRs and QoS requirements as well.
- According to the tool/platform concern, in [20], the authors rely on work-flows and model transformation to generate codes for simulators that are locally developed. While in our approach, we rely on classical tools: Archi extensions as a design tool ([31], [30]) including the proposal in this paper, and network simulators: OPNET [10], NS-3 ([11], [4]) that are widely used and provide large set of tools and utilities to obtain measures/traces, present network architectures through animations, etc.

III. Service Creation Environment

In this section, we present the service creation and development environments.

In [1], a SCE is defined as follows: ***"is a collection of software tools (together with a reuse infrastructure) used according to the service development methodology with the aim to assist the service developer(s) when applying the service development methodology by automating and simplifying as much as possible the service creation process, and facilitating consistency and verification checks"***.

In our proposal (Fig.1), we focus on the service design activity, where we propose set of tools to support it, including early verification [11] according to the functional and non-functional requirements (performance, and QoS requirements).

Extending the design activity by early verification helps the service designer to perform tests to obtain valuable feedbacks

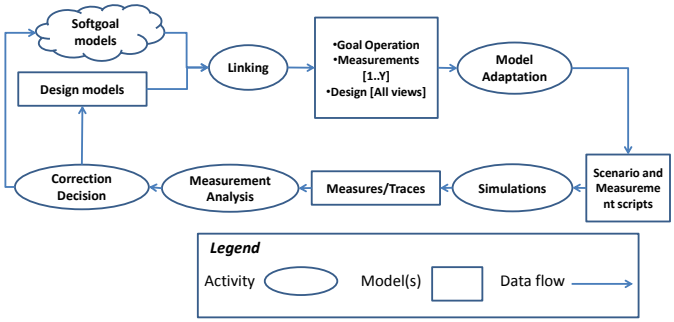

Fig. 2. Verification Activities in our approach [12]

about the design before starting the implementation phase, where the software and hardware elements are installed or deployed. Therefore, we have proposed (Fig.2) recently extensions ([11], [4]) to the design language that enable the generation of configuration for measures during the simulation time, including the automated selection of the proper simulator according to its capabilities [12]. Additionally, we make syntax and domain-specific rule checks during the code generation activity, and during the design activity through the design tool (Fig. 8) [9].

IV. Model Driven Engineering

In this section, we present the Model Driven Engineering (MDE) discipline that we rely on in all of our contributions in relation with early verification activity. MDE [3] 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. In our approach, modeling tools follow the constraints and represent the concepts that are defined in the meta-model. The meta-model [28] ***"defines by itself a language for describing a Specific Domain of interest"***. It permits to instantiate large number of models that conform to it like in programming languages; 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 helps us to generate design tools and simulation programs directly and automatically from meta-models and instance models. 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 (on the same abstraction level) such as ATL [32], or map between different domains using one meta-model for the source model to generate texts/codes (e.g. XPAND [33]). In our case (Fig. 3), the input model represents the design of high abstract level, and the meta-model is the extended ArchiMate meta-model which

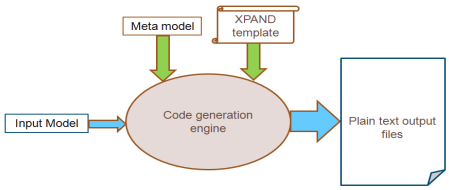

Fig. 3. XPAND workflow engine [10]

represents the domain specific modeling language (DSML) ([3], [10]). Our code generation is an automatic process that links directly the design model to the simulation script. Thus, it helps to reduce the time of the implementations for large simulation programs, and it minimizes the implementation errors.

V. Domain-Specific Modeling Languages (DSMLs)

In this section, we will present shortly the DSMLs to highlight and explain our contribution of the MeDON/DSO meta-model. The concept of DSMLs [34] relies on MDE and Model Driven Architecture (MDA) disciplines. Modeling languages are used to describe a system with high level of abstraction (e.g UML 2.0).

For DSO, and in relation to our objectives, we describe distributed systems. UML is not enough to cover our needs, as it has only one layer that contains all of the concepts of the design, and these concepts are too general and open. Thus, we selected ArchiMate modeling language that meets UML in some concepts, but it can describe the systems from IT domain and share multiple viewpoints during the design as it relies on TOGAF[3] framework [3].

ArchiMate relies on Enterprise Architecture (EA) framework ([3], [27]). It decomposes the system design into 3 layers: business, application, and technology. In our approach, we approach these layers in the following way (Fig. 4):

- Business layer: specifies the end-user functions and actors. It describes the service activities as perceived by the end-user, and the flow between them;
- Application layer: specifies the functions and software components of the service. It describes the capability of the system under study, and the way of performing its tasks;
- 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.

Additionally, ArchiMate adds many design concepts (business events, application components, technology infrastructural elements, etc) and relationships (e.g. triggering) that are very useful to describe a service from the IT domain.

In MDA [34], a meta-model provides the modeling language that is used to model and describe a system. It contains the

[3]TOGAF: is a synonym for "The Open Group Architecture Framework"

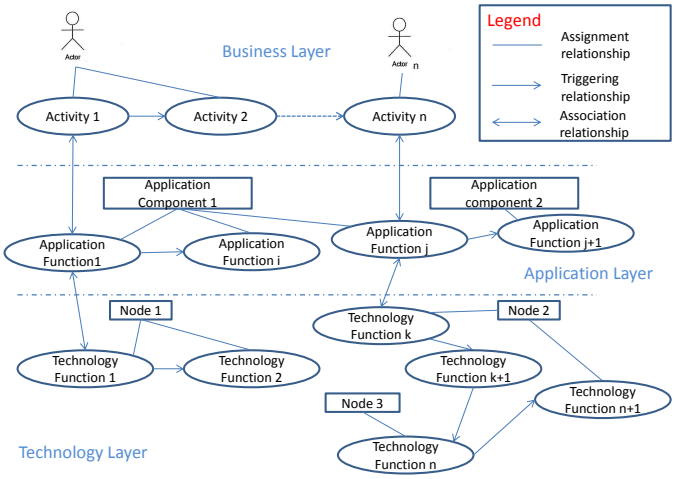

Fig. 4. A simplified example of multi-layered architecture of ArchiMate

abstract syntax of the language, where its constraints describe the concrete syntax that can be implemented in the design tool such as Archi tool[4] relying on Eclipse-EMF (tool generation concept thanks to model transformations).

A. Underwater localization in deep sea observatories

In this subsection, we present the data fusion technique that we consider in our application for the DSOs. In our context, the localization activity aims to identify the location of an object after being detected by a sensor or set of sensors in the underwater environment. The target object is not communicating through our network (e.g. IMS) thus can't be localized through the Home Subscriber Server unit (HSS) [8], and it is in the water so electromagnetic waves are not propagating for long distances. Thus, we are not using the localization services that are offered by IMS nor other systems like GPS. Acoustic hydrophones (underwater) are proper sensors for our application, where we rely on IMS platform to enable the communications (over IP) between the sensors and the other components that are included in the localization tasks. Additionally, IMS makes it possible to access the video/audio streams offered by the underwater sensors from terminals that are far away using the IP cloud.

We focus on the distributed architecture of the data fusion in order to understand:

- the importance of relying on IMS as an underlying platform for designs that use the multisensors data fusion techniques;
- usage of ArchiMate as a design language for modeling;

Reducing complexity and errors of the design to the minimum are our objectives in the design activity. ArchiMate separates between the aspects of the design (behavioral, structural) from one side, and design viewpoints (business processes, applications, topologies, etc) on the other side, this helps in reducing complexity. For example, ArchiMate separates between the function description of the localization process (Fig. 7) from the topology and architectural description of the information graph [5] (Fig. 8) of DSOs. IMS (technology layer of ArchiMate extension [9]) is a proper core-network to exchange the information (over IP) between the elements of the DSO and according to the information graph of the selected architecture. It can execute the multimedia functions that can be used for DSOs as well, such as storing and replaying video streams. In order to link between the DSO architecture and IMS, we consider the elements of the information graph as terminals (end-users) in the IMS layer.

There are different architectures that can be applied to achieve the data fusion objective [13]: centralized, hierarchical without feedback, hierarchical with feedback, and fully distributed Information graphs (Fig. 5).

The process of the data fusion [35] combines information that are obtained from sets of different sources to provide robust and complete description of an environment or process of interest. In our application, for the DSOs, these sources are sensors [5] that provide large amount of data to be fused later relying on the multisensors data fusion techniques [13].

Fusion architectures are described by information graphs. The centralized architecture represents the simplest information graph (Fig. 5), where every sensor manages independent set of measurements that are processed at a local fusion node to update the position of the object under tracking. The other architectures rely on the centralized one to develop the method of the tracking and its accuracy. In each architecture, the complexity of the information graph (Fig. 5) is different from the others where the sensors stay isolated from each other, and communicate only with the fusion unit. The difference is that more fusion nodes are operating and can communicate with each other in different topologies.

Many localization algorithms can be applied in the underwater environment such as: triangulation [36], bounding-box [36], set-membership [37], and Dive'N'Rise(DNR) [38], etc. All of these algorithms are compared to the triangulation one, and the difference is in the positioning accuracy. The accuracy of the trilateration or multilateration is acceptable. Thus, we select to represent the trilateration algorithm in our design model (Fig. 7) according to its simplicity. The localization algorithm is run in the server node (e.g. fusion node) that applies the algorithm to compute and update the location of the target.

B. Contribution: Meta-Model Proposal for MeDON/DSO

In this subsection, we present our contribution of a new meta-model that extends ArchiMate modeling language to represent the domain specifications of MeDON/DSO. This meta-model enables us to generate design tool that is coherent with Archi but contains additional concepts that are specific to the MeDON/DSO domain ([13] for data fusion concepts). The generated design tool helps the designer to model the system and avoid syntax errors that may be made during the design activity.

[4]Archi tool link: http://archi.cetis.ac.uk/

[5]***Information graphs are convenient means to understand how fusion process flows impact a network system*** [13]

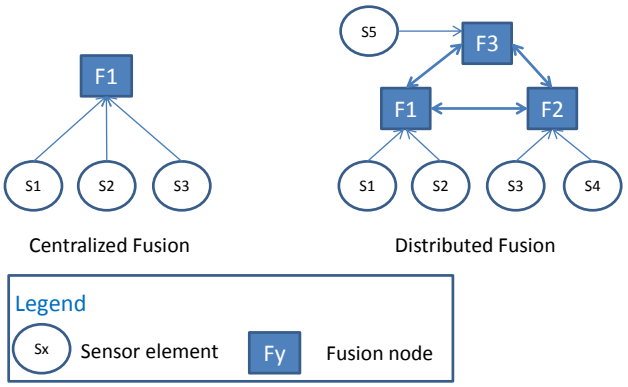

Fig. 5. Information graphs for centralized and fully distributed fusion architectures [13]

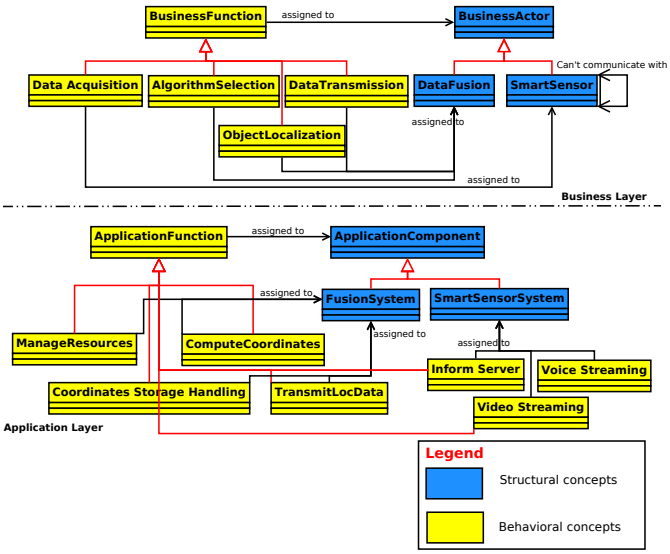

Fig. 6. Extending business and application layers of ArchiMate: proposal of DSO Meta-Model

Our proposed (contribution) 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 [9].

We present the proposed Meta-Model as the following:

- Business Layer (Fig. 6): We have extended the business actor of ArchiMate by two new concepts: the smart sensor, and the data fusion.
  Smart sensor is responsible of the Data Acquisition activity, while the data fusion is responsible of the activities: AlgorithmSelection (performs a procedure to select the proper algorithm), DataTransmission (to transmit the data between the different fusion components), ObjectLocalization (to make the necessary actions that localize an object). These functions extend the business function concept in ArchiMate;
- Application Layer (Fig. 6): We have extended the application component by two elements: the fusion system, and the smart sensor system.
  The FusionSystem is responsible to perform the following application functions: ManageResources (to manage the resources needed for the algorithm execution), CoordinatesStorageHandling (to store the coordinates correlated with time), ComputeCoordinates (to compute the coordinates according to a specific algorithm selected previously by the Data Fusion actor), and TransmitLocalizationData (to exchange information between the fusion servers/systems).
  The SmartSensorSystem is responsible to perform the following functions: InformServer (to inform the fusion server about any detection of a specific object), VoiceStreaming (this function is useful for the case of hydrophones), and VideoStreaming (this function is useful for the camera sensors).

VI. Network Simulators

Different approaches can be applied for the system experimentation [39]:

- Analytic approach: Relies on modeling the system mathematically using applied analytic theories such as (e.g. Queuing Theory);
- Simulation approach: Discrete event simulation (DES) is a powerful research technique to investigate the protocol designs, interaction, and the large scale performance issues. Network simulators (NS-3 in our case) contain compiler to check the syntax of the simulation scenario and apply verification rules that conform to the networking concepts (e.g. IP network conflicts). This helps the designer to detect errors in relation with networking concepts and observe the interactions (behaviors) between the different nodes of the design. Additionally, they provide the service designer with measures that help to detect the quality flaws.

Network simulators allow a service designer to experiment the design models with lower cost and implementation time than dealing directly with real network elements. They provide a suitable way to execute the service design and evaluate it according to networking domain. In our approach, we rely on the network simulation activity to detect domain-specific errors of the service design and to obtain measures that help the designer to improve the performance and QoS. NS-3 is configured through a C++ program that controls the simulation flow. Additionally, using C++ to implement the simulation program gives us the ability to fine-tune it, thanks to the object oriented approach (class concept) that is applied in C++ language.

On the other side, implementing an NS-3 simulation scenario for a complex model such as TS or MeDON/DSO case studies is not a simple task. It consumes time to analyze the mapping between the technical spaces in the different domains (high abstract modeling, and network simulation modeling), then to implement the C++ code correctly without

making errors and relying on NS-3 libraries and specifications. To give an example, the generated code (in our automatic code generator) for a design (multiple layers) instance of MeDON/DSO application requires 3203 lines of C++ code that is ready to be run directly in NS-3 simulator.

This shows the importance of having a compiler to transform the design high abstract model into a simulation scenario. The transformation and syntax checks (e.g. reject the associations between two layers that are not consequent) rely on the constraints and abstract syntax that are described by the meta-models: ArchiMate [25], IMS [9], and our new contribution for MeDON/DSO in (Section. V-B).

VII. Code generation using Xpand-Eclipse

This section presents the model-to-text (M2T) transformation that we rely on to generate the simulation scenario directly from the design model.

In section IV, we have presented the MDE discipline that is helpful to manage the complexity problem [3]. Model transformations are conceptual activities when one needs to exchange intrinsic information (concepts included in the design scenario) between the different tools (e.g. from design tool to network simulator), where implementation languages are totally different.

XPAND [33]: "***is a statically-typed template language featuring polymorphic template invocation, aspect oriented programming, functional extensions, a flexible type system abstraction, model transformation and validation. It includes an editor which provides features like syntax coloring, error highlighting, navigation, refactoring and code completion***". We rely on Eclipse IDE to use XPAND, this also enables us to benefit from the environment of Eclipse that is rich of tools for modeling thanks to the EMF.

Most of the network simulators are configured through text file (configuration scripts). For example, OPNET simulator can be configured by an XML file ([10], [9]). This is the first motivation for using XPAND. The second motivation is that XPAND, thanks to EMF, can rely on the abstract syntax that is provided by the meta-model of the modeling language to perform syntax checks before and during the code generation process. Another motivation is that it permits us to add static code side to side with the dynamic one which makes it possible to generate codes of a new syntax that is not described in the meta-model of the modeling language, and the proper formalism that suits the target tool grammar of configuration.

We have recently described the method to map between design models and the NS-3 simulator in [4]. The XPAND template (Fig.3) contains the rules that are structured sequentially to perform the mapping algorithm correctly.

The intrinsic information that are exchanged between the design model and simulator are the information of the design scenario, but they take different formalisms and are implemented in different languages. Thus, it is important that the template of the code generation should cover all of the elements of the design models, and adds new elements (when necessary) to adapt the model to the target tool (e.g. using concept of helpers in NS-3).

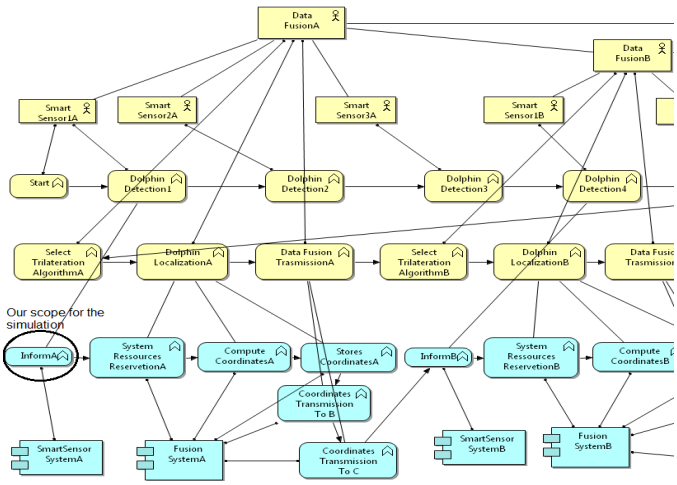

Fig. 7. Part of the model of an underwater localization, done by design tool that extends Archi. Layers are: Business and Application

VIII. Case-Study: MeDON/DSO Application

In this section, we present an example of MeDON/DSO design model to illustrate our approach according to two points: (1) the ability of instantiation of the proposed new meta-model (Section. V-B) and generate code to simulation to evaluate it; (2) To show the ability of change applications from different domains: video conference TS before in [4], and localization application for MeDON/DSO in this paper, while fixing the IMS meta-model in the technology layer [9] and the code generation rules in both cases.

A. DSO Localization Model

In the context of DSOs, the objective of the object-localization system is to identify and localize an object that enters in the range of a sensor or a set of sensors. Sensors are connected to fusion servers that apply a distributed algorithm to compute the position of that object thanks to the data collected from these sensors.

In our approach, we concentrate on the interactions between the different nodes that are included in the DSO model. Internal actions can be implemented by extending the modules of the simulator. Our interest is to show our ability of modeling the DSO scenario relying on IMS core-network and generate simulation codes to be run directly in NS-3. This helps to evaluate the design according to the networking concepts and the constraints that are defined in the meta-model (DSML).

The design model is composed of 3 views according to the layers of ArchiMate: Business, Application, and Technology (Fig. 4). In Fig. 7, we present a part of the large model that is designed by a design tool that extends Archi, and relies on the proposed meta-model (section V-B). The model contains behavioral elements, e.g. the InformA Application Function. This function aims to inform the fusion server A that an object is detected by the sensor 1A. A large series of functions are

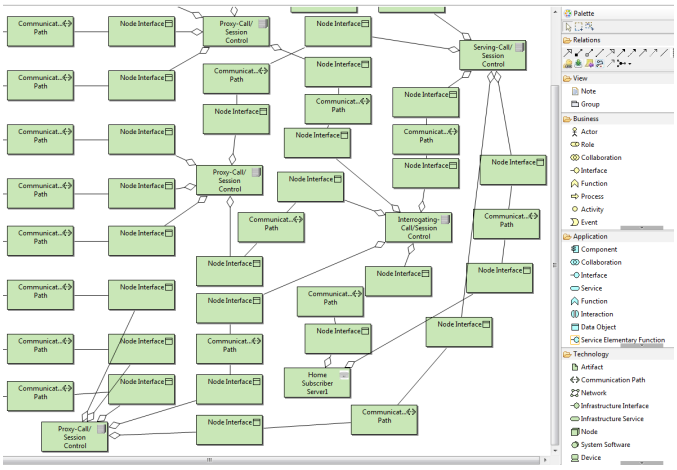

Fig. 8. Part of the model of an underwater localization, done by design tool in [3]. Represented in the technology layer-Structural view

associated in the technology layer (e.g. *sendto*) to execute this application function. The *sendto* function forwards/sends a message of type SIP or Diameter from one node to another.

Regarding to the technology layer, we have relied on the IMS meta-model and ArchiMate to extend the Archi tool in ([9], [3]). We designed the model of DSO according to the IMS specifications [8] and following the sequences in the application layer (Fig. 8), thanks to the association relationship that connects the application functions to the technology ones.

For all layers, the design tool prevents syntax errors thanks to the meta-models, their constraints, and the grammar of the DSML. It generates XMI file that represents the graphical design model. This XMI file is fed to the code generator to produce directly the simulation scenario for NS-3. The code is run through the command in Linux *"sudo ./waf –run file-name"*, no errors or warnings appeared: none for the compilation neither during the runtime or debugging through the *ddd* debugger. An XML animation file is automatically generated by NS-3 (animation configuration is generated automatically by the code generator) to be run using the NetAnim tool. The animator can visualize the topology of the design and show the messages/packets (interactions) exchanged between the different nodes according to the simulation scenario (Fig. 10)[6].

B. Observations and Analysis after Simulation using NS-3

The design model starts by the function start in the business layer which is of high abstract level (represents an activity). It is represented as a function in the class (SmartSensor1A). The generated code contains the instantiation of this object and calling this function automatically (Fig. 9). The function itself calls the next one, etc until calling the last function in the technology layer. This results in a better fine-tuning level of the simulation auto-generated scenarios thanks to ArchiMate architecture. This shows also that our code generation does

[6]The names of the nodes were added to the figure manually according to the node ids in the auto-generated simulation scenario for better clarity

```
        //=======

        NodeContainer N_CommunicationPath_PCSCF2_SCSCF1;
        //******Here, we can define the predefined general attributes of the P2P link between the two nodes*******
        NetDeviceContainer dev_CommunicationPath_PCSCF2_SCSCF1;
        N_CommunicationPath_PCSCF2_SCSCF1 = NodeContainer (NC.Get(3),NC.Get(7));
        PointToPointHelper p2p_CommunicationPath_PCSCF2_SCSCF1;
// configuring the attributes of the Point 2 Point helper thus to configure the device and channel attributes
        dev_CommunicationPath_PCSCF2_SCSCF1 = p2p_CommunicationPath_PCSCF2_SCSCF1_Install(N_CommunicationPath_PCSCF2_SCSCF1);
        //      define the 3rd Layer issues for the interfaces that form the 2 ends of a communication path Communication Path
        //TEST=TRUE
                        //PCSCF2_Interface4

                //Assign Ipv4 from the specific Ipv4Helper that is set to a specific subnet
                        CommunicationPath_PCSCF2_SCSCF1_IpContainer = Network13.Assign(dev_CommunicationPath_PCSCF2_SCSCF1);
                        //


        //=======

        //****** Structural Configuration End
// Routing Protocol, we fix it as it is not configured in the higher level of abstraction.. But it can be configured later in
the routing protocol in the network entity
NS_LOG_INFO("Enable static global routing.");
//Turning on the global static routing across the network
Ipv4GlobalRoutingHelper::PopulateRoutingTables();// by this we avoid to include routers in our design, assuming that the Simu
necessary routing procedures

//** Here is the call of the first function from the behavioral part, that will call by its definition another function and t
due to the message flow chart
//The first function is from the business layer which represents according to the EA framework the control layer in our appro

        Ptr<SmartSensor1A> StartFunction = CreateObject<SmartSensor1A>();
        StartFunction->Start();
```

Fig. 9. NS-3 code (auto-generated by XPAND): a partial snapshot

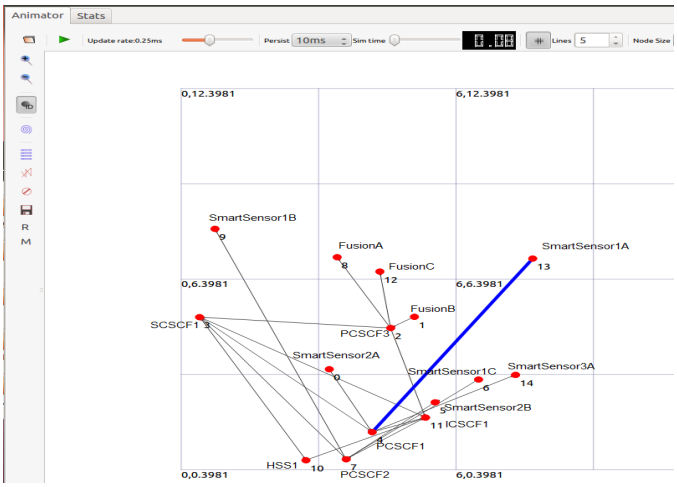

Fig. 10. Animation of the simulation scenario, snapshot from NetAnim tool

not change the architectural specifications of the original design model, thus it is transparent regarding to the intrinsic information of the design.

Using the Wireshark [7] analysis tool, we have analyzed the packet information that are generated by NS-3 in the (pcap) format. The chart in (Fig. 11) shows the number of the SIP messages that are passed through every node interface to execute the application function InformA (Fig. 7). The results for the signaling packets are relative to the *sendto* functions of the technology layer. This means that the code generator didn't miss any information from the design, because exchanging data between nodes relies on the configuration of both structural and behavioral elements of the network. This confirms the transparency of the code generator in keeping the intrinsic information of the design and representing them in the simulation scenario.

IX. Conclusion and Future Work

In this paper, we have presented our approach of a domain-specific service creation framework that assists the service designers to create and evaluate the design models, relying

[7]Wireshark network-traffic analyzer: http://www.wireshark.org/

on classical tools (widely-used network simulators) and code generations. Extensibility is confirmed by extending ArchiMate with different DSMLs (IMS, MeDON/DSO), where we could generate design tools from these extensions thanks to the Eclipse Modeling Framework. Our approach includes a method and a work-flow from the requirement analysis activity to the measurement analysis one in the context of service creation environments. Relying on MDE and model transformations enables us to automate considerable parts of our work-flow (default behavior in the context of software frameworks), especially the model compiler that is able to obtain NS-3 (and OPNET previously [10]) simulation scenarios from high abstract models. These transformations are transparent in conveying information, and they are not subject to change by the designer, only domain experts are allowed to improve/develop them. Our framework defines the roles of the different actors that deal with it. The user of the framework is the service designer, and he is not supposed to change the design language or the code generation templates, only domain experts are allowed to develop these elements. It relies on design tools to assist the designer, these design tools are themselves generated from the extended design languages.

Additionally, in this paper, we have proposed a new meta-model in the context of MeDON/DSOs. This meta-model provides the syntax of the DSML that is specific for the MeDON/DSO project, and conforms to the ArchiMate/EA standard. The modeling language that we use in our approach (ArchiMate) is extensible, and is suitable for the usage in the IT domain. ArchiMate, thanks to EA, shares the different viewpoints in a project through its multiple layered architecture. Our recent contribution, IMS meta-model, is reused in this paper to run the localization application model through simulation, replacing a previous case-study of video conference TS. Relying on our code generator that works with the ArchiMate meta-model and is implemented in XPAND-Eclipse, we have compiled the DSO model to obtain a simulation scenario that is run directly in NS-3 network simulator. We have shown, in this paper and [10], the transparency of our code generator in conducting the intrinsic information of the design model to the simulation scenario.

The design tool assists the designer to avoid syntax errors thanks to the meta-model and the concrete syntax constraints. This is expressed to detect the errors that are related to network domain. Simulation makes it possible to detect domain-specific (network specific) errors and generate measures to be analyzed in order to improve the qualities (performance, and QoS). In the context of MeDON/DSO, simulation helps the designer to evaluate the design before DSO equipments are deployed.

On the first hand, our code generator provides a fine-grained simulation scenario, as the structural elements of our interest are represented by classes which contain functions. Additionally, it separates the control (signaling) and user planes from each other, thanks to the multiple layers of ArchiMate/EA. On the other hand, the implementation time of the code generation template consumes considerable time, and needs domain experience in both domains: modeling and network

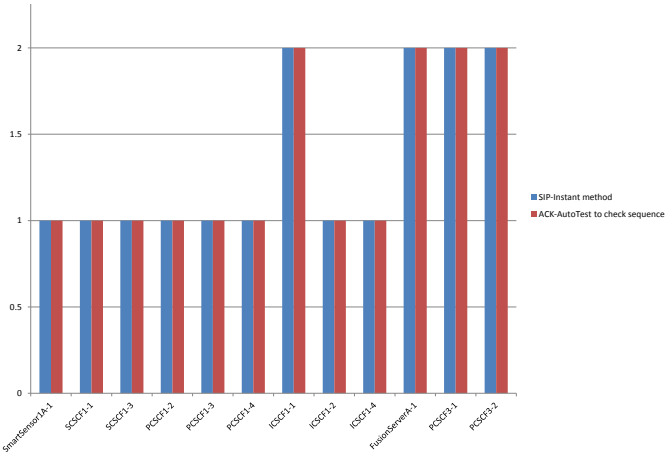

Fig. 11. SIP signaling messages (P2P) per node-interface, that are exchanged to execute InformA application function. Results are collected by Wireshark and the figure is drawn using Excel

simulation. Running our code generation on both cases (video conferencing and localization application) generated scenarios that are applied directly to the NS-3 simulator without any problem in the compilation neither in the runtime.

In the future, we intend to obtain real measures from the DSO network that relies on our design and compare them to the measures that we have obtained during the simulation. Additionally, we intend to generate analysis scripts for tools (e.g. MATLAB) automatically using model transformations, and relying on the non-functional requirements to obtain feedbacks that help to decide wither to modify the design or not.

ACKNOWLEDGMENT

The authors would like to thank Dr. Vanea Chiprianov for his help by providing the design tool that enables us to use the DSML (extends ArchiMate and includes IMS specifications) [3].

REFERENCES

[1] D. Adamopoulos, G. Pavlou, and C. Papandreou, "Advanced service creation using distributed object technology," *Communications Magazine, IEEE*, vol. 40, no. 3, pp. 146 –154, march 2002.

[2] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, ser. II, V. R. Basili, Ed. KLUWER ACADEMIC Publishers, 1999, vol. I, no. 65 1999.

[3] V. Chiprianov, "Collaborative construction of telecommunications services. An enterprise architecture and model driven engineering method," Ph.D. dissertation, Telecom Bretagne, France, 2012.

[4] I. Alloush, Y. Kermarrec, and S. Rouvrais, "A transversal alignment between measurements and enterprise architecture for early verification of telecom service design," in *Advances in Communication Networking*, ser. Lecture Notes in Computer Science, T. Bauschert, Ed., vol. 8115, IFIP International Federation for Information Processing. Springer Berlin Heidelberg, August 2013, pp. 245–256.

[5] O. Zein, J. Champeau, D. Kerjean, and Y. Auffret, "Smart sensor metamodel for deep sea observatory," in *OCEANS 2009 - EUROPE*, May 2009, pp. 1–6.

[6] A. Tengberg, C. Waldmann, P. Hall, D. Atamanchuk, and M. Kononets, "Multi-parameter observations from coastal waters to the deep sea: Focus on quality control and sensor stability," in *OCEANS - Bergen, 2013 MTS/IEEE*, June 2013, pp. 1–5.

[7] J. Aguzzi, C. Costa, J. Company, Y. Fujiwhara, P. Favali, V. Tunnicliffe, M. Matabos, M. Canals, and P. Menesatti, "The new synthesis of cabled observatory science: Technology meets deep-sea ecology," in *Underwater Technology Symposium (UT), 2013 IEEE International*, March 2013, pp. 1–8.

[8] G. Camarillo and M. A. García-Martín, *"The 3G IP Multimedia Subsystem (IMS) Merging the Internet and the Cellular Worlds"*, third edition ed. A John Wiley and Sons, Ltd, Publication, 2008.

[9] V. Chiprianov, I. Alloush, Y. Kermarrec, and S. Rouvrais, "Telecommunications service creation: Towards extensions for enterprise architecture modeling languages," in *6th Intl. Conf. on Software and Data Technologies (ICSOFT)*, vol. 1, Seville, Spain, 2011, pp. 23–29.

[10] I. Alloush, V. Chiprianov, Y. Kermarrec, and S. Rouvrais, "Linking telecom service high-level abstract models to simulators based on model transformations: The IMS case study," in *Information and Communication Technologies (EUNICE 2012)*, ser. Lecture Notes in Computer Science, R. Szabó and A. Vidócs, Eds., vol. 7479. Springer Berlin Heidelberg, August 2012, pp. 100–111.

[11] I. Alloush, Y. Kermarrec, and S. Rouvrais, "A generalized model transformation approach to link design models to network simulators: Ns-3 case study," in *International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2013)*. SciTePress Digital Library, July 2013, pp. 337–344.

[12] ——, "An Automated Tool Selection Method based on Model Transformation: OPNET and NS-3 Case Study," in *International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, M. S. Obaidat, Ed., vol. 45, no. 9. The Society for Modeling and Simulation International (SCS), IEEE CommunicationsSociety, July 2013, pp. 10–17.

[13] M. E. Liggins, D. L. Hall, and J. Llinas, *Multisensor Data Fusion, Theory and Practice*. Taylor & Francis Group, LLC, 2009.

[14] A. Kleppe, "Mcc: A model transformation environment," in *Model Driven Architecture – Foundations and Applications*, ser. Lecture Notes in Computer Science, A. Rensink and J. Warmer, Eds. Springer Berlin Heidelberg, 2006, vol. 4066, pp. 173–187.

[15] N. Medvidovic and R. Taylor, "A classification and comparison framework for software architecture description languages," vol. 26, no. 1, Jan 2000, pp. 70–93.

[16] I. Sommerville, *Software Engineering*, 9th ed., M. Horton, Ed. PEARSON, 2011.

[17] Y. Shin, C. Yu, S. Chung, and S. Kim, "End-user driven service creation for converged service of telecom and internet," in *AICT '08. Fourth Advanced International Conference on Telecommunications, 2008.*, 2008, pp. 71–76.

[18] J. Yelmo, J. del Alamo, R. Trapero, P. Falcarm, J. Yi, B. Cairo, and C. Baladron', "A user-centric service creation approach for next generation networks," in *Innovations in NGN: Future Network and Services, 2008. K-INGN 2008. First ITU-T Kaleidoscope Academic Conference*, 2008, pp. 211–218.

[19] R. Glitho, F. Khendek, and A. De Marco, "Creating value added services in internet telephony: an overview and a case study on a high-level service creation environment," vol. 33, no. 4, Nov 2003, pp. 446–457.

[20] L. Touraille, M. K. Traoré, and D. R. C. Hill, "A model-driven software environment for modeling, simulation and analysis of complex systems," in *Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, ser. TMS-DEVS '11. San Diego, CA, USA: Society for Computer Simulation International, 2011, pp. 229–237.

[21] A. Achilleos, K. Yang, and N. Georgalas, "Context modelling and a context-aware framework for pervasive service creation: A model-driven approach," *Pervasive and Mobile Computing*, vol. 6, no. 2, pp. 281 – 296, 2010.

[22] ——, "A model driven approach to generate service creation environments," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, 30 2008-dec. 4 2008, pp. 1 –6.

[23] J.-L. Bakker and R. Jain, "Next generation service creation using xml scripting languages," in *Communications, 2002. ICC 2002. IEEE International Conference on*, vol. 4, 2002, pp. 2001–2007 vol.4.

[24] A. Hartman, M. Keren, S. Kremer-Davidson, and D. Pikus, "Model-based design and generation of telecom services," 2007. [Online]. Available: https://www.research.ibm.com/haifa/projects/services/sce/papers.shtml

[25] The Open Group, *ArchiMate 1.0 Specification*, The Open Group Std., 2009.

[26] O. Noran, "An analysis of the zachman framework for enterprise architecture from the GERAM perspective," *Annual Reviews in Control*, vol. 27, no. 2, pp. 163 – 183, 2003.

[27] D. Quartel, W. Engelsmanb, H. Jonkersb, and M. van Sinderenc, "A goal-oriented requirements modelling language for enterprise architecture," in *Enterprise Distributed Object Computing Conference, 2009. EDOC '09. IEEE*. University of Twente, 2009, pp. 3 – 13.

[28] J. Bezivin, "In search of a basic principle for model driven engineering," *Novatica Journal*, vol. 2, pp. 21–24, 2004.

[29] B. Henderson-Sellers, *Book of Object-Oriented Knowledge, Object-Oriented Analysis, Design and Implementation: A new approach to software engineering*, A. Binnie, Ed. PRENTICE-HALL, INC., Englewood Cliffs, N.J. 07632, 1992, iSBN: 0-13-059445-8.

[30] V. Chiprianov, Y. Kermarrec, and S. Rouvrais, "On the Extensibility of Plug-ins," in *ICSEA 2011, 6th Intl. Conf. on Software Engineering Advances*, Bercelona, Spain, October 23-28 2011.

[31] ——, "Meta-tools for Software Language Engineering: A Flexible Collaborative Modeling Language for Efficient Telecommunications Service Design," in *FlexiTools2010: Workshop on Flexible Modeling Tools (in conjonction with 32nd ACM/IEEE ICSE Intl. Conf. on Software Engineering)*. Cape Town, South Africa: ACM/IEEE, May 2010.

[32] Atlas transformation language. The Eclipse Foundation. [Online]. Available: http://www.eclipse.org/atl/

[33] Eclipse modeling. Eclipse. [Online]. Available: http://www.eclipse.org/modeling/

[34] I. Kurtev, J. Bézivin, F. Jouault, and P. Valduriez, "Model-based DSL frameworks," in *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, ser. OOPSLA '06. New York, NY, USA: ACM, 2006, pp. 602–616.

[35] H. Durrant-Whyte, *Multi Sensor Data Fusion*, 1st ed., Australian Centre for Field Robotics, Australia, January 2001.

[36] M. Erol, F. Vieira, and M. Gerla, "Auv-aided localization for underwater sensor networks," in *Wireless Algorithms, Systems and Applications, 2007. WASA 2007. International Conference on*, Aug 2007, pp. 44–54.

[37] A. Caiti, A. Garulli, F. Livide, and D. Prattichizzo, "Localization of autonomous underwater vehicles by floating acoustic buoys: a set-membership approach," *Oceanic Engineering, IEEE Journal of*, vol. 30, no. 1, pp. 140–152, Jan 2005.

[38] M. Erol, L. F. M. Vieira, and M. Gerla, "Localization with dive'n'rise (dnr) beacons for underwater acoustic sensor networks," in *Proceedings of the Second Workshop on Underwater Networks*, ser. WuWNet '07. New York, NY, USA: ACM, 2007, pp. 97–100.

[39] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*. Springer verlag, 2009.