Transforming Viewpoints of Distributed Designs to Support Simulation Scenarios

Iyas Alloush[1,2], Yvon Kermarrec[1,2] and Siegfried Rouvrais[1,3]
[1]*Telecom Bretagne, Institut Mines-Telecom, Université européenne de Bretagne, Bretagne, France*
[2]*UMR CNRS 6285 Lab-STICC, Bretagne, France*
[3]*IRISA, Campus Universitaire de Beaulieu, Rennes, France*

Keywords: Enterprise Architecture, Viewpoint, Model Driven Engineering, Code Generation, Network Simulation, IMS.

Abstract: In order to reduce the time to market and improve the qualities during the service construction activities, we provide the designers with integrated tools that help them to construct services. They are able to evaluate their designs earlier according to functional, performance non-functional, and QoS requirements. The contribution of this paper is concerned with representing the different viewpoints of a design model in the simulation scenario through code generation. The main benefits of our approach based on separation of concerns are to better manage the complexity of the design and to improve the fine-tuning level of simulation scenarios. We illustrate our approach with a video conference service relying on the IP Multimedia Subsystem platform.

1 INTRODUCTION

In the context of Service Creation Environments (SCE)s (Adamopoulos et al., 2002), the TS design is a major activity where any mistake results in major consequences in later phases. It may lead to installation/deployment of the hardware elements in the wrong way. The Telecom Service (TS) designs contain process behaviors and structural elements that represent the underlying networks on which they will operate. Thus, the TS design is complex due to the diversity of different domains and perspectives that are used. This results in increasing the time of the design phase and the errors that can be made by TS designers.

In order to improve the time to market and cost factors, we are interested in the evaluation of the TS design earlier than the implementation phase according to the functional, performance non-functional, and QoS requirements. We consider reduction of complexity and facilitation of error detection in the TS design to be important add-ons in the tools that we aim to provide, where they form the main motivation behind the contribution of this paper.

Network simulation makes it possible to obtain valuable measures/traces from the TS design and detect errors or quality flaws by early analysis of these results. The contribution in this paper is connected directly to the network simulation scenarios. It is a continuation to our recent results in (Alloush et al., 2013). In this former proposal we have proposed an approach to bridge the language gap between modeling and network simulation activities relying on model transformations and the Enterprise Architecture (EA) standard (Noran, 2003; Quartel et al., 2009). This is achieved by generating simulation scenarios that can be run directly by classical simulators: e.g. OPNET, NS-3. The modeling language (ArchiMate) relies on the EA standard, thus it has an architecture that shares the different ***viewpoints***[1] in the design thanks to the multiple layers that it has (business, application, technology).

Representing these highly abstract layers in the simulation program helps, at least, to check the data flow between the design elements (from cause to the result). Based on that context and problematic, the research question is: ***How to transform the multiple viewpoints of ArchiMate from the TS highly abstract design to the scenario of classical network simulators?***.

In this paper we present our contribution in conveying the viewpoints represented in ArchiMate/EA to the simulation scenario. We rely on our code gen-

[1]According to the IEEE 1471-2000: A viewpoint is considered as the central concept for organizing software architectures. Its main goal is to facilitate the comprehension of complex systems by providing separation of concerns.

Alloush I., Kermarrec Y. and Rouvrais S..
Transforming Viewpoints of Distributed Designs to Support Simulation Scenarios.
DOI: 10.5220/0004997103210328
In *Proceedings of the 9th International Conference on Software Engineering and Applications* (ICSOFT-EA-2014), pages 321-328
ISBN: 978-989-758-036-9

erator presented in (Alloush et al., 2013) where we select NS-3 as a target tool.

As benefits and in addition to the rapid and automated code generation of the simulation program (Alloush et al., 2013), our contribution helps the TS designer during the evaluation of the design model by: (1) improving the clarity of the generated code as it applies some object oriented "class concept"; (2) improving the reconfigurability of the simulation parameters and service functions as they are abstracted; (3) separating the control from the user plane descriptions thanks to the separation between the application and technology layers in the design model.

This paper is structured as follows: In section 2, we present the related work by analyzing the similarities and differences from our work. Section 3 highlights the service creation environments and points to our approach. In section 4, we explain the EA and ArchiMate concepts that directly affect our contribution. In section 5, we present the main concept of code generation in the picture of the Model Driven Engineering (MDE) discipline. We present our mapping method (contribution) to transform the multiple viewpoints of ArchiMate to the scenarios of network simulation. In section 6, we illustrate our approach by an example from the TS domain: a Video Conference service and we analyze the results. Finally, section 7 contains the conclusion and the future work.

2 RELATED WORK

Design architecture and composition, tool capabilities and domains, are important points to be considered as they are related to the transformation of the design architecture (design tool) to the simulation program (evaluation tool). We are interested in the following concerns: **(C1)** the verification tool capabilities that are used in the evaluation process and their domain, as the code generation process follows the target verification tool in our approach; **(C2)** the consideration of the behavioral and structural aspects in the design model. This concern is related to the modeling language (see section 4); **(C3)** the consideration of the different viewpoints inside the design model. This concern is related to the modeling language too (see section 4). There are different approaches and SCEs that help to make early verification before the implementation phase. In the following, we present some of them and we analyze according to the stated concerns.

MDE helps to manage complexity thanks to the modeling and model transformation fundamentals. All of the following related works rely on the MDE discipline.

- The usage of the verification tools **(C1)** is guided by their capabilities to produce measures that help to evaluate the design. Some of these tools are specific to an analytic domain like performance analysis. Cheddar[2] simulator (Dissaux and Singhoff, 2008) is an example of such a domain specific tool, it can help the designer to make performance analysis such as feasibility tests through real-time scheduling simulation. Verification tools can differ according to their type of analysis, such as model checkers. In (Berthomieu et al., 2010), the authors rely on the FIACRE pivot language and Tina tool-chain to make behavioral verifications depending on Timed Transition Systems (TTS). The difference from our work is that we rely on classical tools (or COTS[3]): NS-3 and OPNET. Additionally, these tools cover a wide range of domains and applications (e.g. WIFI, LTE, etc), and different traces/logs too. These tools (network simulators) can make behavioral and performance verifications in the networking domain at the same time;
- Regarding the design aspects **(C2)**, they are mainly divided into behavioral and structural concepts. The behavioral aspects correspond to the functional description of the system. The structural ones are used to identify the place where the behavior(s) should be executed. In the TS creation domain, value-added services are designed relying on the APIs that are provided by the underlying platform as in IP Multimedia Platform (IMS) in (Shin et al., 2008), SIP transactions to control services as in (Hartman et al., 2007), or Open Service Architectures (OSA) and Parlay in (Bakker and Jain, 2002; Glitho et al., 2003). However, all of these approaches do not consider the performance evaluation of the hardware elements in the system design, where they concentrate on implementing the functions and protocols only. In our approach, we consider both the behavioral and structural (including hardware elements) aspects thanks to the concepts of the modeling language (ArchiMate).
- Regarding the multiple viewpoints **(C3)** concern, the authors in (Berthomieu et al., 2010) rely on the AADL language to model their design. AADL enables the designer to separate software components of the system (e.g. process, thread, etc) from

[2]Cheddar framework: http://beru.univ-brest.fr/∼singhoff/cheddar/#Ref6

[3]COTS is a synonym for "Commercial Off-The-Shelf" existing tools.

the components of the execution platform (e.g. processor, bus, memory, device, etc). This manner of describing the system is close to our approach thanks to the separation between the design aspects in AADL, but the difference is that their approach is proper for real-time systems (RTS) and that AADL does not provide business modeling capability as in ArchiMate.

In (Shin et al., 2008; Yelmo et al., 2008), the authors propose SCEs in the domain of value-added services (TSs). Both of the approaches consider one viewpoint during the design phase: the end-user as a designer to involve him in the service creation activities. Furthermore, in both of (Agarwal et al., 2005; Achilleos et al., 2010), the authors consider the service designer who is not the benefiter of the service. These approaches do not consider multiple viewpoints in the design phase of a TS. The difference is that our approach relies on the IMS platform specifications as represented in the technology layer of ArchiMate. Therefore, the difference is in the underlying technology representations and the nature of the descriptions of the elements in the application layer. ArchiMate provides a proper abstraction in all layers, and makes it possible to extend its concepts according to the needs of different domains (to produce DSMLs) under the ceiling of the IT domain.

3 SERVICE CREATION ENVIRONMENTS

Service creation environments (SCE)s rely on software tools that are used to achieve the service development methodology. The aim of the SCEs is to assist the service developers by automating and simplifying the service creation process. In our approach

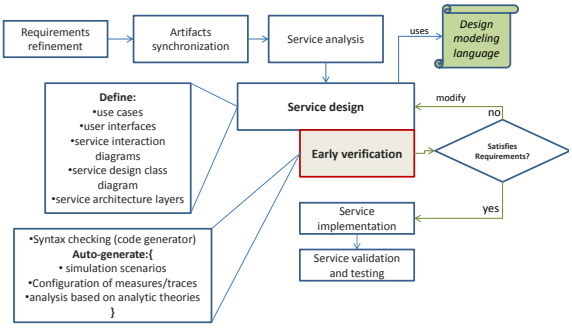

Figure 1: Extending the design activities by early verification in the service development framework, inspired from (Adamopoulos et al., 2002).

(Fig. 1), we contribute by extending the design phase to include our proposed early verification activity (Alloush et al., 2013). In order to assist the service designer, we generate tools (Chiprianov et al., 2011) that help to avoid syntax errors and errors in the relationships between the elements of the design. Additionally, we generate simulation scenarios automatically to be run directly in network simulators: OPNET and NS-3 (Alloush et al., 2013).

4 ENTERPRISE ARCHITECTURE AND ARCHIMATE

In this section, we present the EA and ArchiMate modeling language, as our contribution relies directly on ArchiMate concepts.

According to (Jonkers et al., 2006), the enterprise architecture (EA) is defined as: "***a coherent whole of principles, methods, and models that are used in the design and realization of the enterprise's organizational structure, business processes, information systems, and infrastructure***". The EA objective is to define frameworks that provide a way to structure concepts and activities necessary for designing and building systems. The Open Group Architecture Framework (TOGAF) covers the activities that are related to the IT domain. The closest EA modeling language to TOGAF is ArchiMate. This makes ArchiMate suitable for modeling telecommunication applications and development.

ArchiMate includes the three phases of TOGAF (Fig. 2): Business Architecture, Information Systems Architecture, and Technology Architecture. It represents these architectures using three corresponding layers. Furthermore, it can provide interoperability between these three layers thanks to the different dependencies and the passive structural elements that ArchiMate contains.

Like other modeling languages ArchiMate has both (abstract and concrete) syntaxes and semantics. The multi-layered architecture is represented in the meta-model of ArchiMate that represents the abstract syntax of the modeling language. Every layer contains different concepts: structural, behavioral, and passive structural (C2 in section 2). This helps the designer of the service to represent the system from different viewpoints (multi-layers) (C3 in section 2), and provides the ability to represent software and hardware concepts (behavioral and structural aspects). Regarding the business layer, it contains the concepts that can describe any business process through modeling (e.g. business actor, business function, process, etc). Our approach considers that the concepts of this layer are proper to the usage of the end-user in the TS

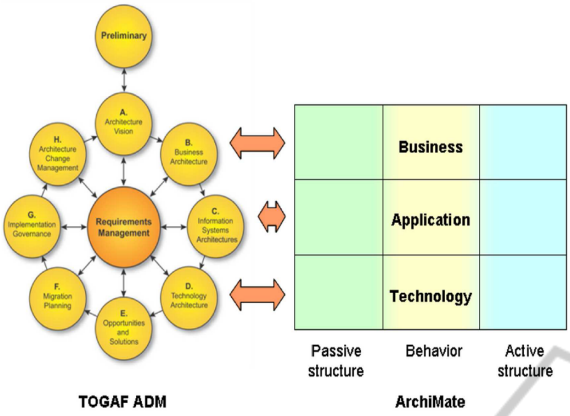

Figure 2: The correspondence between TOGAF and ArchiMate (Chiprianov, 2012).

domain. With regard to the application layer, it represents the service applications and the different systems (e.g. video conferencing system, push-to-talk system etc) and how they interact with each other. It contains several important elements (e.g. application components, application interfaces, functions, etc). Finally, the technology layer represents the underlying platform that is able to run/execute the applications of the service. We have contributed to this layer by extending it to include IMS core-network concepts (Chiprianov et al., 2011) through a domain specific modeling language (DSML).

Including of IMS concepts and constraints in ArchiMate gives the designer the ability to benefit from the different functions of IMS. The other advantage is that these new concepts provide us with syntax that enables us to generate network simulation scenarios directly from the design models, for OPNET and for NS-3.

5 EXPRESSING THE VIEWPOINTS OF ARCHIMATE IN THE NETWORK SIMULATION SCRIPTS

In this section, we present our method to transform the multi-layered architecture of the ArchiMate modeling language (see section 4) to the NS-3 simulation scenario through the code generation process.

The objective from the viewpoints concept is to organize the different activities between the designers from different domains and backgrounds. The point of transforming this concept to the network simulation is to represent the different actors, roles, functions, components in order to: (1) trace the flow of execution of the design; (2) implement functions even if abstracted for override (reconfiguration) later by the developers according to their needs; (3) separate the concepts of the TS design (user and control planes) in the simulation program. On one hand, ArchiMate provides a language to describe the EA through its multiple layers and the different domains of concepts. On the other hand, NS-3 (Henderson et al., 2006) relies on class concepts as it accepts the C++ language for configuration. The C++ program can control the simulation scenario through different APIs that are provided by the libraries of the NS-3 simulator. Our method in this section relies on these features of both the ArchiMate language and the NS-3 simulator to express the architecture of ArchiMate in the simulation C++ program.

The Eclipse IDE[4] is a powerful IDE rich in plug-ins and tools to start with research and development. The Eclipse modeling framework (EMF)[5] provides us with a powerful means to model and generate codes in the Model Driven Engineering discipline. There are many languages that are used for model transformations (e.g. ATL as a model-to-model transformation, and XPAND as a model-to-text transformation). We chose the XPAND language to build templates that are used to generate simulation code directly from design models relying on the abstract syntax of the modeling language. XPAND is used because it contains means to make syntax checks, and to analyze and manage the replacement of variables with their values from the design model. This replacement occurs side by side with the static implementations that respects the language of the target tool.

In the following we present the mapping rules that we implement through the XPAND model transformation: (1) Every Business Actor or Role, and Application Component in the design model is mapped as a class in NS-3; (2) Every Business Function or Activity, and Application Function in the design model is mapped as a function that is implemented in the corresponding class in NS-3 (correspondence is represented by an assignment relationship in the design model); (3) The relationship between the functions/activities of the different layers is accepted to be an association relationship while it is excluded (through the model transformation analysis) when it occurs between elements that do not belong to two consequent layers; (4) The calls between the functions in the simulation scenario are initiated by triggering and cross-layer association relationships in the design model; (5) Regarding the technology layer mapping, it stays as implemented in (Alloush et al., 2013); (6) The initialization behavior that starts the

[4]Eclipse IDE website: https://www.eclipse.org

[5]EMF website: https://www.eclipse.org/modeling/emf/

sequence of function calls in NS-3 is by default the starting function of the business layer.

We implement these rules using XPAND language. The first function that initiates the call sequence is the start function and should be in the business layer. The instantiation of its class (assigned business actor) and the function call are done in the main function of the simulation program (C++ program). In order to implement all of the classes in the same simulation program (one file), we start by implementing the application layer elements then we go up to the business ones. This is because the application classes are to be instantiated in the business ones and so they should be declared beforehand. Figure. 3 presents an algorithm to list the application components of the application layer (in ArchiMate) in reverse order from that of the function calls. The same algorithm is used to order the elements of the business layer.

After having listed the model elements in the proper order for the simulation program, we iterate over the elements of these lists (application then business layers) to create an application class in NS-3[6] for each element. The assignment relationship helps to establish the correspondence between functions and structural elements (in Fig. 4, the assignment relationship is shown between 'Conference System' and the function 'response checking'). This helps the code generator to correctly implement the functions in the corresponding application class of the simulation program.

With regard to the calls between the different functions (Fig. 6), there are two cases according to the triggering relationship **(intra-layer callings)**: (1) the source and target functions belong to the same class/component - then there is no need to instantiate a class (call is direct); (2) the source and target functions belong to different classes/components - in this case there is a need to instantiate the target class in the source one. Another case is considered: the association relationship between two functions that belong to different layers **(inter-layer calls)**. In this case, we instantiate the class of the target function and then call the destination function (except for the case when the target belongs to the technology layer where we call the function directly because all of the technology functions are declared at the beginning of the simulation program).

The intra and inter layer calls contribute to the concern (C3) in section 2 through achieving the interoperability between the different layers and representing that in the simulation program. Mapping the design aspects (behavioral, structural) using the application class concept helps to improve the fine-grained level of the simulation program - this contribution is related to the concern (C2) in section 2. Additionally, relying on NS-3 enables us to verify and test designs of different distributed systems (different applications and objectives) according to the networking domain and performance non-functional requirements - this contributes to the concern (C1) in section 2.

[6]In NS-3, an application class is a class that has start and stop application functions for overriding purpose. One to many application classes can be assigned to the node of NS-3.

6 VIDEO CONFERENCE EXAMPLE

In order to illustrate our approach on transforming the different viewpoints (concepts from different layers) of the ArchiMate language to the scenario of the network simulator (NS-3 v3.13), we present an example of a TS - video conference - (VCTS).

6.1 Design Models

The VCTS design model is divided into 3 views: business, application, and technology. In our approach, we consider that the business layer model includes the end-user activities represented in business processes which is a high abstract level process. Furthermore, we consider the application layer view as representing the system components (Fig. 4), in this case study it is video conferencing system (Chiprianov, 2012; Chiprianov et al., 2011). Application-layer system functions call technology-layer functions thanks to the association relationship that is defined in ArchiMate standards. The sequence of the technology functions is designed according to the video conference sequence diagram that is mentioned in (Camarillo and García-Martín, 2008). This diagram represents the message flow and the domain-specific activities that are related to the IMS core-network. The design tool in (Chiprianov, 2012) insures the conformance between the specifications of the modeling language (DSML) and the design model instance created by the service designer.

6.2 Observations and Analysis

Our contribution in this paper is to represent the different viewpoints of the design (distributed system) in the simulation scenario. In order to insure that this representation is accurate, it is important to insure: (1) the correct transformation of the flow of execution to the simulation program (call continuity from

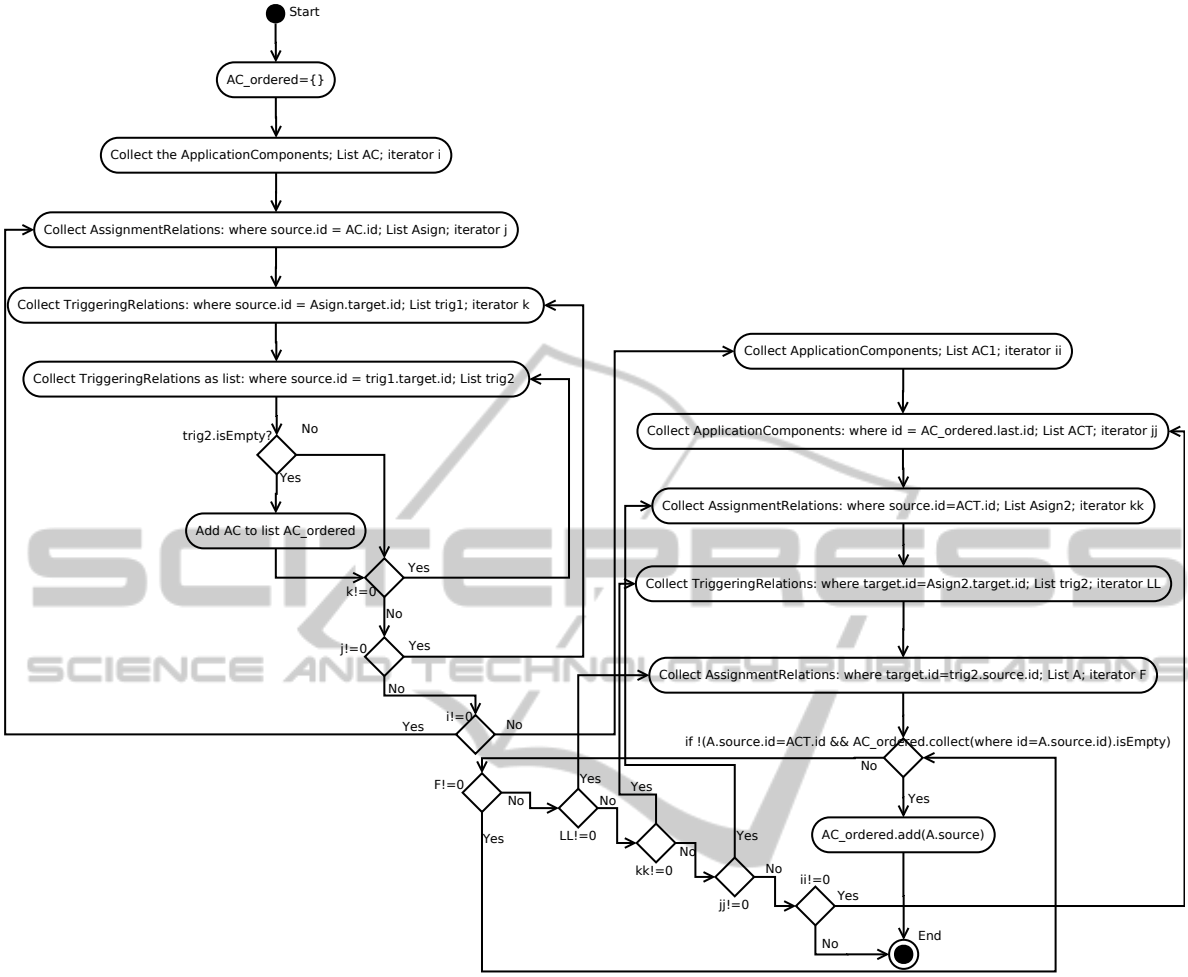

Figure 3: Reverse ordering of structural elements in Application Layer/ArchiMate.

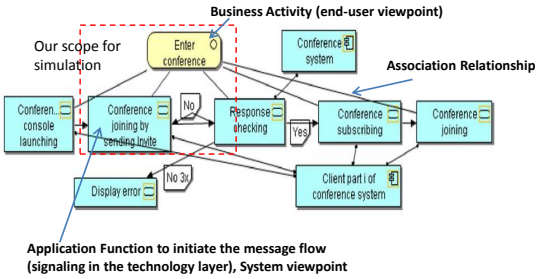

Figure 4: Application layer view from the design model of the video conference TS using DSML (Chiprianov, 2012).

cause/event triggered by the business actor to the execution in the node); (2) the correct implementation according to the target tool language (C++).

NS-3 compiles the simulation code according to the C++ language and to the constraints of the networking domain. Executing the the command (./waf –run scratch/scenario-name) leads to compilation of the simulation scenario and then execution of the network simulation. The compilation and network simulation (through NS-3) show no errors or warnings. All of the trace and capturing (.pcap) files were generated too. This means that the classes and functions are all declared in the right order thanks to the algorithm in (Fig. 3), and that both the class instantiation and function calls are correct according to the C++ language rules.

Figure 6 shows the correspondence between the different layers of the design and the generated code for NS-3. One notices that calls between functions in the C++ code can be intra- or inter- layer. This insures the continuity of calls between the different functions in the same layer and the interoperability between the consecutive layers. Thus, causality of behavior (flow of execution) is guarantied to be correctly transformed (when) using our approach. In the same figure, it is clear that the application configuration is realized in the functions and classes of the applica-

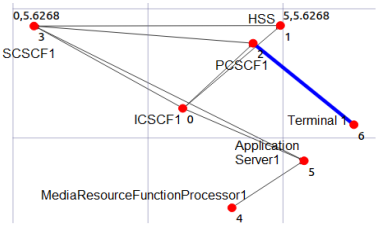

Figure 5: Snapshot of the NetAnim animator

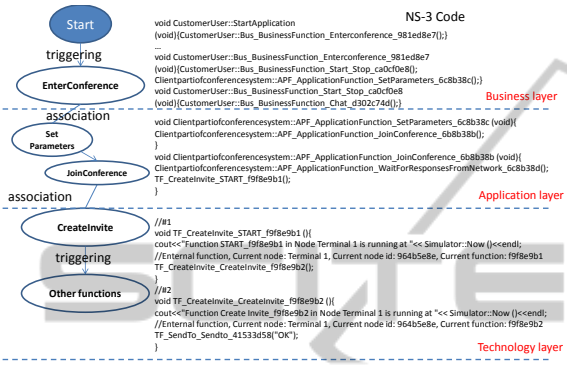

Figure 6: Interoperability between the EA-layers in the Simulation Scenario

tion layer where the signaling actions (e.g. exchanging SIP messages) and hardware internal functions of IMS are represented in the technology layer elements. This confirms the ability to separate the control- and user- planes in both the design and simulation tools.

In order to check the correspondence of the message flow with the design, we used the NetAnim tool that reads the auto-generated animation scenario file and displays the packet flow (Fig. 5). Additionally, we have used wireshark[7] in order to analyze the signaling traffic and check its correspondence with the design model.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we have presented our contribution to transforming the viewpoints of Enterprise Architecture standard from a design model of a telecom service (as a case study of a distributed system) to the network simulation technical space (NS-3 simulator). We have proposed model transformation rules that insure the transparent mapping between the design model and the simulation technical spaces.

Our approach reduces the time of the verification during the design phase earlier before the implementation or deployment ones. This corresponds to the automated actions in the model transformation. According to our approach in extending design activities, we are now able to provide a tool chain (Mellor, 2002) that integrates support for development tools useful for the SCEs (Adamopoulos, 2009). These tools are extensible as they can be generated from meta-models (e.g. Archi tool) and they rely on the Eclipse Modeling Framework that is well supported and widely used too. Additionally, as far as we know, there is not yet any state of the art code generation technique to link between models and classical (or COTS) tools like NS-3 simulator. We make use of the architecture of ArchiMate to generate a fine-grained simulation program. This enables the designer to reconfigure the part of the code that is related to his domain experience. Our model transformation enables the designer to generate complex and large simulation scenarios directly from the design model in a very short time (few seconds). Additionally, it helps to separate the user and control planes in the simulation program. On the other hand, the implementation of the transformation template consumes considerable time and needs accuracy and domain experience in both the modeling and network simulation activities.

In the future, we expect to generate executable analysis scripts from the constraints that belong to the performance and QoS requirements. These scripts can be run directly in COTS such as MATLAB. Analysis tools helps to obtain valuable feedbacks that help the designer to make decisions to improve the design quality. Additionally, we intend to provide a checker for the domain-specific constraints that are related to the language (DSML) through the code generation process. This checker should generate reports that help the designer to modify the model efficiently.

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

REFERENCES

Achilleos, A., Yang, K., and Georgalas, N. (2010). Context modelling and a context-aware framework for pervasive service creation: A model-driven approach. *Pervasive and Mobile Computing*, 6(2):281 – 296.

Adamopoulos, D. (2009). A service-centric approach for exploiting network intelligence. In *Second International Conference on the Applications of Digital Information and Web Technologies, 2009. ICADIWT '09.*, pages 145 –150.

Adamopoulos, D., Pavlou, G., and Papandreou, C. (2002). Advanced service creation using distributed object technology. *Communications Magazine, IEEE*, 40(3):146 –154.

Agarwal, V., Dasgupta, K., Karnik, N., Kumar, A., Kundu, A., Mittal, S., and Srivastava, B. (2005). A service

creation environment based on end to end composition of web services. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 128–137, New York, NY, USA. ACM.

Alloush, I., Kermarrec, Y., and Rouvrais, S. (2013). 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)*, pages 337–344. SciTePress Digital Library.

Bakker, J.-L. and Jain, R. (2002). Next generation service creation using xml scripting languages. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 4, pages 2001–2007.

Berthomieu, B., Bodeveix, J.-P., Zilio, S. D., P.Dissaux, Filali, M., Gaufillet, P., Heim, S., and Vernadat, F. (2010). Formal verification of AADL models with FIACRE and TINA. In *Embedded Real Time Software and Systems (ERTS) 2010*.

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

Chiprianov, V. (2012). *Collaborative Construction of Telecommunications Services. An Enterprise Architecture and Model Driven Engineering Method*. PhD thesis, Telecom Bretagne, France.

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

Dissaux, P. and Singhoff, F. (2008). Stood and cheddar: Aadl as a pivot language for analysing performances of real time architectures. In *Proceedings of the European Real Time System conference*, Toulouse, France.

Glitho, R., Khendek, F., and De Marco, A. (2003). Creating value added services in internet telephony: an overview and a case study on a high-level service creation environment. volume 33, pages 446–457.

Hartman, A., Keren, M., Kremer-Davidson, S., and Pikus, D. (2007). Model-based design and generation of telecom services.

Henderson, T. R., Roy, S., Floyd, S., and Riley, G. F. (2006). ns-3 project goals. In *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, WNS2 '06, New York, USA. ACM.

Jonkers, H., Lankhorst, M., ter Doest, H., Arbab, F., Bosma, H., and Wieringa, R. (2006). Enterprise architecture: Management tool and blueprint for the organisation. *Information Systems Frontiers*, 8(2):63–66.

Mellor, S. J. (2002). Make models be assets. *Commun. ACM*, 45(11):76–78.

Noran, O. (2003). An analysis of the zachman framework for enterprise architecture from the GERAM perspective. *Annual Reviews in Control*, 27(2):163 – 183.

Quartel, D., Engelsmanb, W., Jonkersb, H., and van Sinderenc, M. (2009). A goal-oriented requirements modelling language for enterprise architecture. In *Enterprise Distributed Object Computing Conference, 2009. EDOC '09. IEEE*, pages 3 – 13. University of Twente.

Shin, Y., Yu, C., Chung, S., and Kim, S. (2008). End-user driven service creation for converged service of telecom and internet. In *AICT '08. Fourth Advanced International Conference on Telecommunications*, pages 71–76.

Yelmo, J., del Alamo, J., Trapero, R., Falcarm, P., Yi, J., Cairo, B., and Baladron', C. (2008). 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*, pages 211–218.