Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 36 (2014) 80 – 86

Procedia Computer Science

Complex Adaptive Systems, Publication 4
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2014-Philadelphia, PA

Develop an Executable Architecture for a System of Systems: A Teaching Management Model

Darío J. Delgado[a]*, Rodrigo Torres-Sáez[b], Ricardo Llamosa-Villalba[a],

[a] *School of Electrical and Electronic Engineering - Universidad Industrial de Santander, Bucaramanga 680002 , Colombia*
[b] *School of Chemistry - Universidad Industrial de Santander, Bucaramanga 680002, Colombia*

Abstract

The architectural frameworks are increasingly being used to design and document many kinds of complex systems. This paper focuses at DODAF 2.0 to aid in the development of an executable architecture from a teaching management model (TMM) represented by a System of Systems (SoS). The principal goal is the using of a standard set of DoDAF artifacts (the Operational Activity Model OV-5, the Operational Rules and Operational State OV-6a and OV-6b, Logical Data Model OV-7) and the associated data to generate an executable model based on Petri Nets. With the TMM executable architecture develop from a particular teaching model that we call Agile School (AS) designed to function as a SoS. It seeks to demonstrate through simulation the correct logic of AS and the feasibility of the application of this model. And also, demonstrate how an architectural framework like DoDAF, could be used as a benchmark in the modeling and simulation of a SoS and the study of their feasibility.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology

Keywords: Executable architecture; Petri nets, System of Systems,Teaching management model

* Corresponding author. Tel.: +57 6344000 ext 2676 .
E-mail address: dario.delgado@correo.uis.edu.co

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2014.09.041

1. Introduction

An architectural representation from any architectural framework (FEAF, TOGAF, DODAF, Zachman, etc.) is a model. D.T Ross defines a model as "*M is a model of A with respect to question set Q if and only if M may be used to answer questions about A in Q within tolerance T*"[1]. This definition implies that an architecture represented by an architectural framework could be a model for a System or a System of Systems (SoS) if the architecture can be used to answer questions about the system with some specified tolerance. However, the architectural frameworks outputs are typically called artifacts or views of a system or SoS. Those artifacts are created, as a result, of a pictorial representation and those artifacts are used as a model for a system. In many cases, a pictorial representation to a System or SoS could be enough[2,3]. However, in other cases an executable simulation must be associated to the architectural artifact to be an appropriate model to answer a set of questions of a particular system. The executable architectures are commonly defined as executable dynamic simulations that can be generated automatically or semi-automatically from an architectural artifact[4,5,6]. In the case where simulation is necessary, the executable architecture can be used to create a link between the modelling of architectural artifacts and some modelling and simulation environments. In this work be shown how to do the mapping between the architectural artifacts (UML or SysML models) and the executable architectures. The paper is organized as follows. Section II discusses the importance of the architectural frameworks and its relationship with the executable architectures. Section III discusses the methodology to mapping the static models into an executable architectural artifact, using in this case a Petri Net approach. Finally in section IV introduces the teaching management model based in System of Systems and its description.

2. Methodology

2.1. Executable architecture

UML and SysML dominate the modelling languages for development architectural descriptions. However, UML and SysML are weak in a formal semantic to build numerical and executable simulations models[2,7]. An approach to the application of executable architectures and their use to bridge the gap between architectural frameworks and modelling and simulation (M&S) technologies is the conversion of static architectural models, specified by different modelling languages, to executable models based on various executable formalisms [8, 9], see Fig. 1.

Executable architectures and the architectural frameworks: There are many models to analyse the dynamic and static aspects to a System or a System of Systems (SoS). Some of them are FEAF, Zachman, Goethals, Shekkerman, DODAF, TOGAF, between others. Each architectural framework has the necessary specifications and characteristics to create models or architectural artifacts.

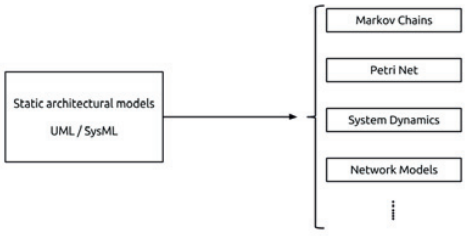

Fig. 1. (Conversion of static architectural models, specified by different modeling languages, to executable models based on various executable formalisms.

According to authors like (Bingfeng Ge et al, 2012) [9], (Kelly Griendling andDimitri N. Mavris, 2011) [10], (Garcia, J.

2007) [11], (Erik Baumgarten., 2007) [12], (BAI Xiao-lil et al, 2008) [13], (Edwin A. Shuman., 2010) [14], between others. DODAF provides more flexibility and adaptability to the automated construction of executable models directly from the architectural data. This paper focuses at DODAF 2.0 to aid in the development of an executable architecture. The principal goal is the using of a standard set of DoDAF artifacts and the associated data to generate an executable model (Petri Nets).

DoDAF and executable architectures: DoDAF 2.0 includes 52 views, divided into eight-category views, but the artifacts are broad, diverse and beyond the scope of an executable architecture focus. A subset of DoDAF artifacts is enough to model an EA.[10,13,14,15] The most relevant artifacts reported to create executable architectures are shown in Table 1.

Table 1. Mapping Between DoDAF Product Models and Modeling and Simulation Tools

Architectural Artifact	Description	Simulation Tool
OV-5	Operational activity model	Markov chains, Petri nets, System dynamic[10,16,12]
OV-$6_{a,b}$	Operational rules and operational state	Markov chains, Petri nets, System dynamic[10,17]
OV-7	Logical model data	Petri nets [16,17]
SV-1	System interface description	Network models [12]
SV-2	Systems communications description	Network models [12]
SV-4	Systems functionality description	System dynamic [12]
Sv-5	Operational activity to systems function traceability	System dynamic [12]

According to the Table 1, to develop a petri net as an executable system model it is only necessary the artifacts OV-5, OV-6a, OV-6b and OV-7. Each architectural artifact used to create the executable architecture has a set of specific models; see Fig 2, in a specific modelling language. The most common languages used to modelling the architectural artifacts are UML and SysML. However, the focus of interest is the mapping between the models in a specific modelling language and an executable element as a Petri net.

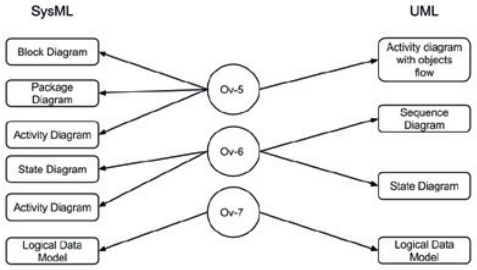

Fig. 2. Modeling languages and DoDAF artifacts.

3. Modeling and simulation

Modeling and simulation (M&S) is used today for both industry and academy. However, modeling and simulation are two different areas, and each approach has different specific objectives. But most of the times these two separate areas are taken together. In the area of systems, according to (Jose L. Risco et al, 2007) "*Modeling is more art than science and the level of abstraction plays a crucial role in model's performance*"[18]. It is difficult to develop a system model because it needs both; domain knowledge and simulation techniques. UML and SysML are widely accepted in the industry as modeling tools, but those modeling languages lacks simulation power. A modeling formalism for

executable architectures should holistically describe the elements of an executable architecture using standard mathematical notation. Petri Nets are a mathematically precise model, and so both the structure and the behavior of Petri net models can be described using mathematical concepts[19].

The mapping process: Transform UML/SysML models into a formal simulation tool need a semantic for model mapping. Petri Nets (PNs) are a formal modeling method with a mathematical approach based on network models and good graphical representation of systems. With the PNs approach is possible analyze the dynamic aspects of systems structure through mathematical and computational simulations. PNs and UML/SysML describe in a different way the operational objects and their associations in the systems models. However, PNs are a good approach to extend the UML/SysML static models to a dynamic environment[16].

Petri Nets[20] is a *5-tuple*, $PN = \{P,T,F,W,Mo\}$ where:

a) $P = \{p_1, p_2, ..., p_m\}$ is a finite set of places,
b) $T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions,
c) $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation),
d) $W:F \rightarrow \{1,2,3,...\}$ is a weight function,
e) $M_o : P \rightarrow \{1, 2, 3, ...\}$ is the initial marking,
f) $P \cap T = \varnothing$ *and* $P \cup T \neq \varnothing$

Exist many approaches for mapping UML/SySML models into PNs, Zhaoxia Hu and Sol Shatz (2004) [21], proposed use the state-chart models and Colored Petri Nets (CPNs) with an Object Net Models (ONMs) and an Internal Linking Place (ILP) approaches. Tony Spiteri (2008) [22] proposed the use of activity diagrams as a natural model to mapping from UML/SysML models into a CPNs. John Anil Saldhana and Sol M. Shatz (2000) [23] proposed the use of collaboration and state-chart diagrams for mapping into Object Petri net Models (OPMs) and CPNs. BAI Xiao-lil et al (2008) [16] emphasizes in the object modeling approach; in the use of class diagrams; and the Extended Colored Petri Nets (ECPNs) as an approach to developing the executable components. All the approaches try to solve the synchronization, no determinism and parallelism aspects that the PNs and the UML/SysML cannot address by themselves. In this work, we will use the approach proposed by John Anil Saldhana and Sol M. Shatz (2000) [22] that uses the activity diagrams for mapping into the CPNs. According to the methodology in [22] the steps suggested for mapping the activity diagram to CPNS are three: i) select from the model repository the activity diagram (OV-5 or OV-6), and select the logical data model (OV-7) as complementary information, ii) transform the activity diagram and the supplementary information in a model that is easier to understand and identify patterns that are important for the executable model 24, and iii) take the first-modeling approximation (second step) and transform it in a CPN.

4. Case study

The system architectures try to structure its components, its relationships and how the systems evolve in the time (Sage, 2005). A generic SoS architecture shows how are the structural and administrative relationships between the constituent systems. According to (Davendralingam & DeLaurentis) [25], a SoS architecture has a layered hierarchy in which is possible organizes the constituent systems. Thinking in a SoS as a set of operationally independent layers that works together to achieve a common goal. A first layer called α−level, exist the systems as individual nodes or fundamental blocks to build a SoS. In a second layer called β − level are represented by systems clusters that come from the α − level which perform collaborative work between them to achieve a particular task. A third layer called γ − level represents the common goals that are subject to the capabilities and requirements from the systems and clusters of systems in the β − level and α − level. Finally, there is a fourth layer called δ − level in which are found the command and control policies for development, operation, and evolution of the SoS. Using the structural scheme to the SoS architecture, we present the teaching management system[26, 27] specifications that are planned to operate as a SoS, see Fig 3 a. In α − level, is possible find the set of available systems to achieve a set of teaching processes, those systems have a particular scheme of work in which the teaching processes are separated into a number of independent tasks (Authorship, Training, Evaluation, Learning). Each of those tasks is assigned its management and operation to a particular set of the sociotechnical system, and a set of support systems as a communication system,

document management system, and measurement system. In the δ − level, is possible found the control and command scheme, the management scheme, communications scheme, and evolution scheme to the systems in the α − level. The management, evolution, and communications schemes are determined according a maturity model for educative processes called MEMORIA-PE © [26], and a comprehensive model of organizational systems architectures ARQUEOTIPOS [27], and organizational management model called POLO© [27]. Those models allow to give meaning and order to the particular activities of each system in the α − level. With those models is possible to model the constituent systems organization and its relationships, this organizational relationships are called β − level, those systems relationships and partnerships work cooperatively to achieve the goals in the γ − level.

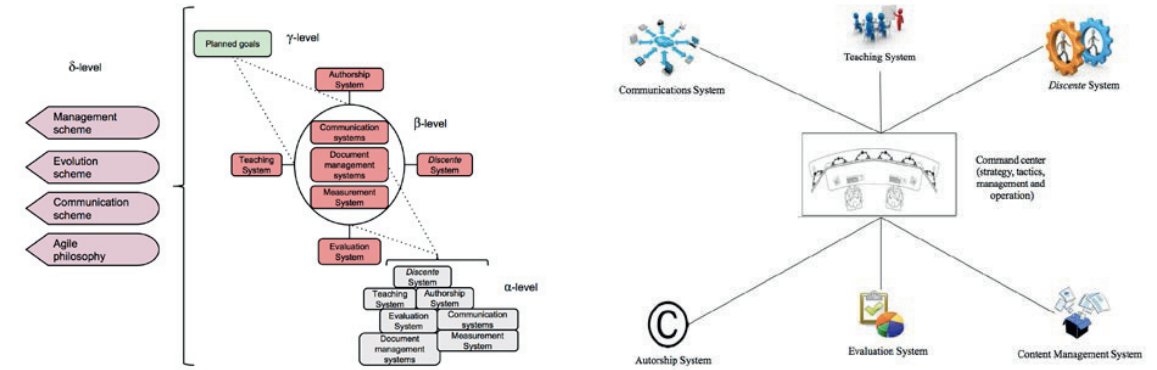

Fig. 3. a) Hierarchical levels in the architectural design of SoS, b) General Scheme of operation

The goal is to manage a teaching process set using a SoS approach. The traditional approaches are supported in the teachers as the principal support in the teaching process manage. With this model the work is distributed in a set of support systems that provide robustness, see Fig 3 b. The SoS operates on a specific operating cycle, see Fig 4. This cycle works according the operation rules defined in δ -level. For this work, we only consider the first stage of Feasibility. However, the SoS consists in six stages immersed in an operation cycle (Feasibility, Initiation, Planning, Transition, Iteration and Accreditation).

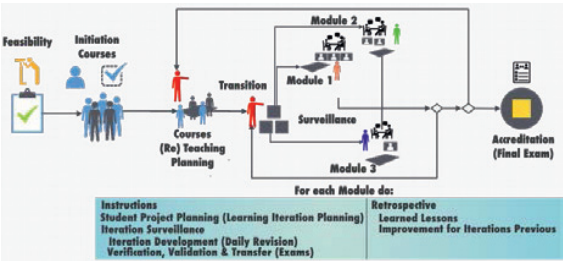

Fig. 4. General operation scheme

The feasibility stage: According to operational aspects in the SoS extracted from the architectural artifacts (OV-5, OV-6a,b and OV-7) and taking only the activity diagram from the feasibility stage, see Fig 5, is possible see how the interaction between the systems is fundamental to achieving the goal of the stage of feasibility. This stage consists in determining if the SoS has the capability to take more courses. The diagram shows the set of activities in which each system take part to achieve the goal for a particular stage.

Building a color petri net: According to the methodology exposed in [22] create a CPN is possible following three

particular steps. i) Select the activity diagram, see Fig 5, ii) With the activity diagram create the FMC (fundamental modeling concepts) an intermediate model, Fig 6 a, iii) and finally build the CPN with the support of FMC. The FMC provides structural information for the next step, and the CPN develop. The petri net build for the feasibility model is shown in the Fig 6 b. The dynamic model shows the feasibility process in which the SoS tries to verify if the capacity to supports additional tasks. The process begins with a new petition to the SoS and continues with the confirmation of command system with the other systems (Teaching system, Evaluation system, Authorship system and Content management system) the resources available to take on new challenges. In the model is possible see an executable specification from a static architectural diagram in which is possible determine additional facts that the static diagram cannot.

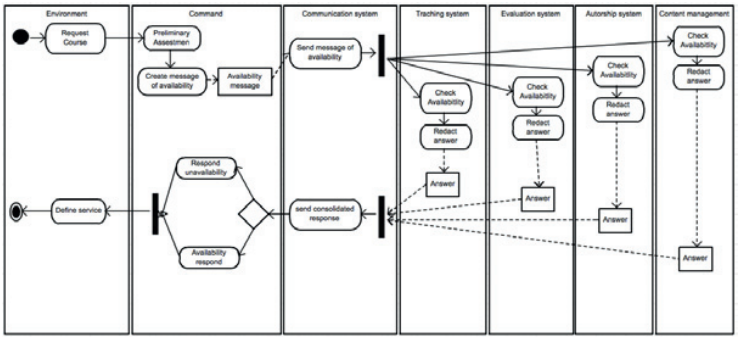

Fig. 5. Feasibility activity diagram

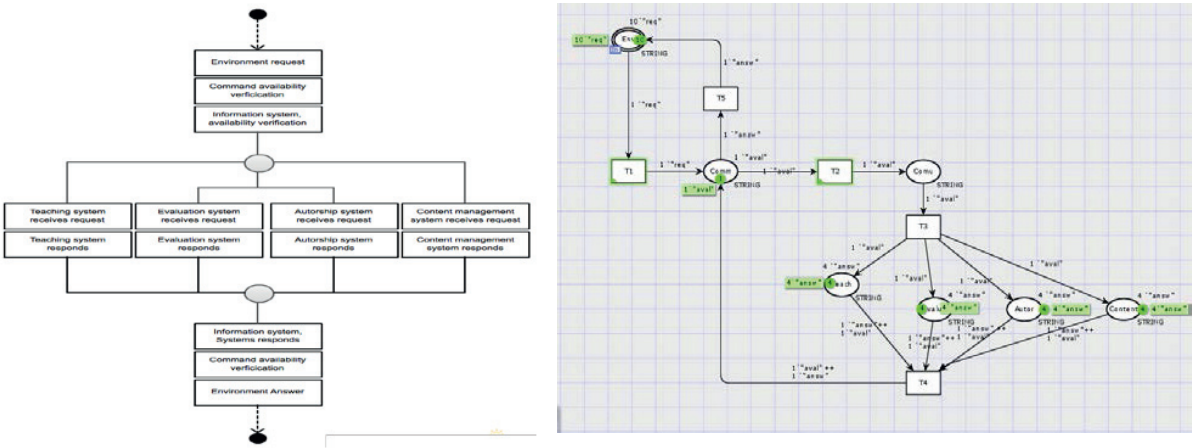

Fig. 6. a) Fundamental modeling concepts, b) Feasibility petri net model

5. Discussion and conclusions

CPNs as a formal modeling approach to verify the dynamic aspects from a System of Systems, is a good tool to validate and testing the logic and the performance from static modeling artifacts. The system complexity is not a difficulty in terms of high-level interpretations of its dynamic, if and only if exist a good operational approximation model of the system in terms of its architecture, this architecture facilitates the dynamic analysis by creating executable architectures from the modeling documents (architectural artifacts). In this case using CPNs, a dynamic model was created from the activity diagram. The approach used in [22] who used the activity diagram as a basis for the CPNs creation, results in a natural transition between static model (activity diagram) and the dynamic model (CPNs), in contrast with other authors[16,20,22] who use the statechart diagram or the class diagram whose transition to

CPNs is less natural.

Acknowledgements

In this work we would like to thank to the Colombian Science and Technology Department (COLCIENCIAS), who through the Francisco José de Caldas Scholarship (scholarship program number 511) for national doctoral programs funded this research.

References

1. Ross, D. T., Structured Analysis (SA): a language for communi- cating ideas. IEEE Transactions on Software Engineering SE-3, 1977
2. R. Wang and C. H. Dagli., Executable system architecting using systems modeling language in conjunction with colored petri nets in a model-driven systems development process, Syst. Eng., vol. 14, no. 4, pp. 383409, Oct. 2011.
3. Garcia, Johnny. Executable Architecture Analysis Modeling. . Vol. 2, pp. 177-184, 2007.
4. TOGAF, The Open Group Architectural Framework. Welcome to TOGAF. Available: www.opengroup.org/architecture/togaf7-doc/arch/, 2005.
5. DoDAF 2.0., The Open Group, The Open Group Architecture Framework (TOGAFTM 9) and theUS Department of Defense Architecture Framework 2.0 (DoDAF 2.0), 2010.
6. Borshchev, Andrei and Filippov, Alexei., From System Dynamics to Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools. . Oxford, England, UK : System Dynamics Society, Proceedings of the 22nd International Conference. 2004.
7. I. Khan, Methodology for the development of executable system architecture, in Proc. 8th Int. Conf. on Frontiers of Information Technology, Islamabad, Pakistan, Dec. 2010.
8. L. W. Wagenhals, S. W. Liles, and A. H. Levis., Toward exe- cutable architectures to support evaluation, in Proc. 2009 Int. Symp. on Collaborative Technologies and Systems, pp.502-511, May 2009.
9. Griendling, K., Drive, F., & Mavris, D. N., Development of a Dodaf-Based Executable Architecting Approach to Analyze System-of-Systems Alternatives. 2011.
10. Griendling, K., Drive, F., & Mavris, D. N., Development of a Dodaf-Based Executable Architecting Approach to Analyze System-of-Systems Alternatives. 2011.
11. Garcia, J. Executable Architecture Analysis Modeling, SpringSim 07 Vol. 2, ISBN 1-56555-314-4. 2007.
12. Baumgarten, E., & Silverman, S. J. Dynamic DoDAF and Exe- cutable Architectures. MILCOM 2007 - IEEE Military Communications Conference, 15. doi:10.1109/MILCOM.2007.4455237. 2007.
13. Xiao-lil, B. A. I., Xue-shan, L. U. O., Xiao-hui, B. A. I., Xian-qing, Y. I., Hong-huil, C., & De-ke, G. U. O., Study of DoD Architecture Simulation Validation based on UML and Extended Colored Petri Nets, 1, 6166. 2008
14. Shuman, E. A., Understanding Executable Architectures Through An Examination of Language Model Elements. SCSC '10 Proceedings of the 2010 Summer Computer Simulation Conference
15. Andrew W. Zinn, Captain, USAF., The Use of Integrated Archi- tectures to Support Agent Based Simulation: An Initial Invesigation PhD THESIS, DEPARTMENT OF THE AIR FORCE AIR UNIVERSITY, 2004.
16. Xiao-lil, B. A. I., Xue-shan, L. U. O., Xiao-hui, B. A. I., Xian-qing, Y. I., Hong-huil, C., & De-ke, G. U. O., Study of DoD Architecture Simulation Validation based on UML and Extended Colored Petri Nets, 1, 6166. 2008.
17. Li, L., Dou, Y., Ge, B., Yang, K., & Chen, Y., Executable System-of-Systems architecting based on DoDAF meta-model. 2012 7th International Conference on System of Systems Engineering (SoSE), 362367. doi:10.1109/SYSoSE.2012.6384204
18. Risco-mart, L. . From UML State Charts to DEVS State Machines using XML. Published by Elsevier Science B. V. 2007
19. T. Murata,. Petri Nets: Properties Analysis and Applications. In Proc. the IEEE, 77(4), pp. 541-580. 1998.
20. Tadao Murata, Petri Nets, Properties, Analysis and Aplications, Proceeedings of the IEEE, Vol. 77. No 4. 1989.
21. Hu, Z., And Shatz, S. M. Mapping UML Diagrams to a Petri Net Notation for System Simulation.SEKE, page 213-219. 2004.
22. Staines, T. S. Intuitive mapping of UML 2 activity diagrams into fundamental modeling concept Petri net diagrams and colored Petri nets. In Engineering of Computer Based Systems, 2008. ECBS 2008. 15th Annual IEEE International Conference and Workshop on the (pp. 191-200). IEEE.
23. Saldhana, J., and Sol M. Shatz. Uml diagrams to object petri net models: An approach for modeling and analysis. International Conference on Software Engineering and Knowledge Engineering. 2000.
24. A. Knöpfel, B. Gröne, P. Tabeling, Fundamental Modelling Concepts, Wiley, West Sussex UK, 2005, pp. 1- 321.
25. Davendralingam, N., And DeLaurentis, D., A Robust Optimiza- tion Framework to Architecting System of Systems. Procedia Computer Science, 16, 255264. doi:10.1016/j.procs.2013.01.027, 2013.
26. R. Llamosa-Villalba, H. Camacho, R. F. Valdivieso, and D. J. Delgado- Quintero., Modelo e Madurez de Procesos Educativos - MEMORIA/PE- , WEEF 2013, Cartagena, Colombia Asociacion Colombiana de Facultades de Ingenieria. 2013.
27. R. Llamosa-Villalba, and L. M. Gomez-Mendoza., Arquitecturas Integrales de Sistemas Organizacionales - ARQUETIPOS-, CIDLIS, Universidad Industrial de Santander, Bucaramanga, Colombia. 2013.