# A Conceptual Map of Model-Driven Development for Systems-of-Systems*

**Valdemar Vicente Graciano Neto[1,2], Milena Guessi[2,3], Lucas Bueno Ruas de Oliveira[2,3], Flavio Oquendo[3], Lina Garcés[2], Elisa Yumi Nakagawa[2]**

[1]Instituto de Informática (INF/UFG), Universidade Federal de Goiás, Goiânia, Brazil
[2]ICMC, University of São Paulo, São Carlos, Brazil
[3]IRISA-UMR CNRS/Université de Bretagne Sud, Vannes, France

valdemarneto@inf.ufg.br, {milena, oliveira, linamgr, elisa}@icmc.usp.br, flavio.oquendo@irisa.fr

***Abstract.*** *SoS development involves difficult and error prone activities related to the constituents runtime integration, interoperability configuration, and deployment. A Model-Driven Development (MDD) approach for SoS can automate the aforementioned activities. However, there is no consensus about which terminology, tools, or models are more suitable for representing SoS in MDD approaches. This paper brings to light a conceptual map that exposes the main concepts and relations of MDD approaches for developing SoS. We designed our model with on the top of a systematic literature review. The main contribution of this paper is the presentation of the proposed map to the community, sharing the state-of-the-art about MDD for SoS.*

## 1. Introduction

Systems-of-Systems (SoS) are systems engineered from a set of pre-existing independent systems, so-called *constituents*. Constituents usually are heterogeneous and accomplish missions by means of interoperability [Fitzgerald et al. 2012]. They have a diversity of technologies, communication mechanisms, operation systems, and data representation [DeLaurentis 2007]. In this perspective, software engineering for SoS faces new challenges, such as (i) providing interoperability among constituent systems, (ii) fitting constituents into a cohesive set of to deliver emergent behaviors [Maier 1998], and (iii) correctly deploying SoS [Barbi et al. 2012].

Model-Driven Development (MDD) has been applied to deal with the aforementioned issues imposed by SoS [Farcas et al. 2010]. MDD has been reported in a variety of domains [Farcas et al. 2010]. However, there is still a lack of consensus on the adoption of languages and models for representing constituents with all their particularities, and about the best technologies and practices to use. Then, there is a necessity for a systematization of the relevant knowledge regarding MDD for SoS in order to support decisions about the best MDD practices, tools, and approaches to apply in SoS engineering [Graciano Neto et al. 2014]. Knowledge about MDD for SoS is spread in literature.

Conceptual maps can support SoS engineers in this endeavor. They represent knowledge captured from a diversity of sources, exposing the main concepts of a subject

*This research is supported by FAPEG (grant number 09/2013, ID 2013.009.97100854), and FAPESP (grants 2014/01646-4, 2011/06022-0, and 2013/20317-9).

area and the relations that link them. This paper presents a conceptual map of MDD for SoS. The main contribution of this paper is to propose an artifact that represents the state-of-the-art in MDD for SoS. In particular, the main concepts documented in this model were identified in a Systematic Literature Review (SLR) previously reported [Graciano Neto et al. 2014]. Remainder of this paper is structured as follows. Section 2 describes the background on MDD and SoS. Section 3 presents the conceptual map, explaining the concepts and how they are related to each other, and discusses our findings. Section 4 presents final remarks.

## 2. Model-Driven Development for Systems-of-Systems

SoS are developed as a modern class of systems to match emerging society's demands. They are formed by pre-existing systems called "constituents", and many of them can participate in an SoS [DeLaurentis 2007]. SoS are engineered to deliver emergent behaviors, i.e., functionalities that can not be individually performed by any of its constituents in isolate which is a result of collaboration among constituents [Fitzgerald et al. 2012]. SoS are classified according to level of managerial independence of constituents, i.e., the autonomy given to a particular system for managing its own resources. They are classified as directed, collaborative, acknowledged, and virtual [Maier 1998]. Distinguishing features are inherent to SoS, such as evolutionary development, dynamic behavior, operational independence, and geographical distribution [Maier 1998]. SoS development is often driven by particular missions (e.g., goal, functionality, or set of tasks) that can only be accomplished by the SoS as a whole [Silva et al. 2014].

MDD has been considered for SoS development due to the facilities it offers. It uses models, metamodels, and model transformations to automatically generate code. A *model* is, essentially, an abstraction of reality represented in a textual or visual language (e.g., a Domain-Specific Language, DSL), and models must conform to their respective metamodels. *Metamodels* are special models that guide and restrict how to construct other models. Models and their respective metamodels are submitted as input to model transformers and, through the use of *model transformations*, they transform the source models into target models or code.

MDD approach contributes to SoS development since it [France and Rumpe 2007]: (i) supports the visualization of the whole SoS, mastering complexity related to its large dimensions issues; (ii) faces problems related to large configuration files used for middleware configuration and constituents deployment, converting this error-prone task into a modeling task (more abstract); (iii) transforms models in correspondent software code and configuration files, automating such task, reducing errors, and maximizing quality, productivity, and traceability; and (iv) supports an adequate deployment, providing also maintainability. For this scope, we use MDD as the acronym to designate all model-driven approaches.

## 3. A Conceptual Map of MDD for SoS

Conceptual Maps represent knowledge [Novak and Ca?as 2006]. They facilitate the understanding on a topic, in a simple graphical format. They can be used for a diversity of purposes, and have been introduced into work environments for problem solving purposes. Aiming at obtaining knowledge related to the development of SoS based on MDD

approaches, we previously performed an SLR [Graciano Neto et al. 2014]. As result, we identified 12 studies related to MDD for SoS. After that, a conceptual map was established according to the following steps:

***Step 1:*** Definition of a *core conceptual map*;

***Step 2:*** Analysis of each study and identification of important concepts. Those concepts were associated to the core conceptual map, creating thus, a conceptual map for each study;

***Step 3:*** Combination of the 12 conceptual maps into a large conceptual map, using the core conceptual map as a reference. This is the main artifact proposed by this research.

A complete version of the conceptual map is externally available[1]. According to our map, a *MDD Approach for SoS* can solve one or more *problems*, offer one or more *advantages*, and it can be applied to one or more *domains*. Such an approach comprises *models*, *metamodels*, *transformations*, and *tools*. A model must necessarily conform to exactly one metamodel, and a model represents one or more *SoS features*. A MDD Approach is supported by adequate *tools*.

According to the complete version of the conceptual map:

An SoS can be classified as Directed, Acknowledged, and Collaborative (Virtual SoS did not appear in the SLR). SoS have received the following denominations in the literature: NetCentric SoS (which requires a Virtual Machine to run), Large-Scale Network-Centric Embedded SoS, Large-Scale Distributed Real-Time Embedded System, Interconnected IT Landscape, and Federation of Constituents.

An MDD Approach for SoS can be considered as a Systems Engineering Approach. MDD Approaches for SoS can provide an important support for realizing tasks such as: composing constituents on COTS with middleware support, and handling text files (configuration and deployment files). Furthermore, MDD can offer manageable approaches for dealing with the diversity of technologies, data representation, operating systems, and languages of constituents; the independent function of constituents; the increasingly size and complexity presented by configuration files that are demanded for configuring and deploying SoS; and considerable complexity of large-scale SoS. MDD Approaches have already been applied for the SoS conception in the following domains: Water Management Policies Systems, Air and Ground Traffic in Airport, Flight Booking, Air Force, Flight Control Systems, and Avionics.

A Model can represent one or more SoS features such as Interoperability, Architecture, Missions, Self-Management, Emergent Behavior, and Constituents. An SoS can execute one or more missions. As observed in the included studies, a mission can be represented by a Mission Scenario (or View), and Colored Petri Nets. A Mission is composed by Mission Parts which are structured in software code (usually in the constituents). A Constituent can be represented as an Agent, and each constituent presents a Behavior. The set of constituents' behaviors can form an Emergent Behavior at SoS level. Self-Management is an SoS feature that can be modeled using SelfMML. Constituents are realized by a Middleware Configuration, Mission Code, and COTS. These COTS can be hardware or software. An Architecture can be expressed as a set of views. A view can be a Deployment view, a Structure View, an Activity View, and a Protocol Definition View.

Transformations required by MDD Approach can be accomplished with tool support. Transformations can be written using a Transformation Language. SoS are modeled using SoS Modeling Language. Examples of SoS Modeling Languages include AADL (Architecture Analysis and Design Language), BPMN, CML (COMPASS Research Group Modeling Language, a formal language), COMPASS (Composable Adaptive Software Systems), DEVSML, MATLAB, OPL and OPD (Object-Process Language and Diagram), SelfMML, SySML (a recurrent language), Simulink, UML, WSDL, and XML. DEVSML is a part of DEVS framework supported by Dunip, which is capable to model simulation environments. As transformation languages, we can highlight oAW, XText, XSL, and XSLT, which are part of EMF (Eclipse Modelling Fra-

[1]Conceptual Map, http://goo.gl/3mW4D5

mework). Tools include ACTUAL [Barbi et al. 2012] (Automation of the Configuration and deploymenT of distribUted AppLications), (a Middleware Platform), CoSMIC, GME/GMF, and INGENME.

Advantages of using MDD for SoS engineering include: Analysis, abstraction of constituents and interfaces, automation, design precision, communication between stakeholders is facilitated, high-configurability, high-confidence code generation, interoperability among models, knowledge capture, maintainability, productivity, raising abstraction level, reuse, reduced development risk, simulation, traceability, validation.

## 4. Final Remarks

This paper presented a conceptual map covering MDD approaches for SoS, offering a panorama of the area of MDD for SoS. It captures a collection of relevant concepts and relations among them. Such map was conceived as a result of an SLR previously carried out. We expect to contribute to the SoS community by offering an starting point from where new researches could be conducted. Contributions include (i) a list of languages currently used or recommended to model SoS, (ii) a collection of the main denominations SoS have received, (iii) a catalog with the main technologies used to engineer SoS with MDD approaches, (iv) a list of the main problems reported by studies as recurrent for SoS development, (v) the main advantages which motivates the adoption of MDD in an SoS development effort, and (iv) prominent domains where MDD have been successfully applied for SoS engineering. Future works include (i) construction of other conceptual artifacts from this map, (ii) consolidating such a model as representative for the area, and (iii) conceiving metamodels for MDD approaches using this conceptual map as a starting point.

## Referências


Barbi, E., Cantone, G., Falessi, D., Morciano, F., Rizzuto, M., Sabbatino, V., and Scarrone, S. (2012). A model-driven approach for configuring and deploying systems of systems. In *SoSE*, Genoa, Italy.

DeLaurentis, D. (2007). System of systems definition and vocabulary. Technical report, School of Aeronautics and Astronautics, Purdue University, West Lafayette.

Farcas, C., Farcas, E., Krueger, I., and Menarini, M. (2010). Addressing the integration challenge for avionics and automotive systems from components to rich services. *Proceedings of the IEEE*, 98(4):562–583.

Fitzgerald, J., Bryans, J., and Payne, R. (2012). A formal model-based approach to engineering systems-of-systems. In Camarinha-Matos, L. M., Xu, L., and Afsarmanesh, H., editors, *Collaborative Networks in the Internet of Services*, volume 380 of *IFIP AICT*, pages 53–62. Springer Berlin Heidelberg.

France, R. and Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. In *FOSE 2007*, pages 37–54, Minneapolis, MN, USA.

Graciano Neto, V. V., Guessi, M., Oliveira, L. B. R., Oquendo, F., and Nakagawa, E. Y. (2014). Investigating the model-driven development for systems-of-systems. In *SESoS*, ECSAW '14, pages 22:1–22:8, Vienna, Austria. ACM.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284.

Novak, J. D. and Ca?as, A. J. (2006). The theory underlying concept maps and how to construct them. Technical report, Technical Report IHMC CmapTools 2006-01.

Silva, E., Cavalcante, E., Batista, T., Oquendo, F., Delicato, F. C., and Pires, P. F. (2014). On the characterization of missions of systems-of-systems. In *Proc. of the ECSAW*, pages 26:1–26:8, Vienna, Austria. ACM.