The Journal of Systems and Software 98 (2014) 155–171

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

On building a consistent framework for executable systems architecture

Imran Khan, Sajjad Haider*

Faculty of Computer Science, Institute of Business Administration, Karachi, Pakistan

ARTICLE INFO

Article history:
Received 14 October 2013
Received in revised form 27 August 2014
Accepted 27 August 2014
Available online 6 September 2014

Keywords:
Systems engineering
Executable systems architecture
Colored Petri Nets

ABSTRACT

The paper presents a framework for executable systems architecture. Termed as Consistent Systems Architecture Description and Behavior Framework (CSADBF), the framework shows how consistency can be maintained while modeling architectural description of systems as well as their behavior. Convergence of three established modeling techniques: ontology, UML, and Colored Petri Nets (CPN), is used to develop this framework. Each tool complements others in accomplishing the goal of consistency maintenance for the executable systems architecture. The framework suggests various mapping schemes that help in establishing strong concordance among different artifacts of these modeling techniques and maintaining consistency of overall system architecture. The first scheme maps OWL ontology to UML and is responsible for maintaining consistency of the architectural description. The second scheme maps combination of OWL ontology and UML to CPN and is responsible for maintaining consistency between static and dynamic views. The third scheme ensures the behavioral consistency of the architecture by providing mapping between Semantic Web Rule Language (SWRL) and CPN Guard conditions. Thus, the framework allows architects to model the systems architecture requirements in OWL ontology and UML and to analyze the behavior and performance of systems architecture in CPN. The paper demonstrates the framework with the help of a case study and also compares it with the existing frameworks.

1. Introduction

Systems architecture can best be thought of as a representation of an existent (or to be created) system, and the process and discipline for effectively implementing the design(s) for such a system that defines the structure and/or behavior of a system.[1] One of the major challenges within the systems architecture domain is the maintenance of consistency between the different diagrams of a system's architecture. As the number of diagrams, used to model a system's architecture, increases, the task of maintaining consistency also gets more complex. It becomes even more challenging when a system's architecture has multiple views such as static (architectural description) and dynamic (architectural behavior) and each view has multiple diagrams in it. In such situations, the validation and verification of the overall system's architecture requires strong concordance between both views.

Several efforts have been reported in the literature (Wagenhals et al., 2000, 2003; Shin et al., 2003; Noguera et al., 2009; Wang and Dagli, 2011) that aim to establish concordance between static and dynamic views. Wagenhals et al. have used dataflow (Wagenhals et al., 2000) and UML (Wagenhals et al., 2003) to define the static view of a system's architecture and then mapped the corresponding diagrams into Colored Petri Nets (CPN) to build an executable system architecture. They give preference to their object orientation approach over structure analysis approach due to better expressive power of UML as well as due to familiarity of new engineers and architects with object orientation techniques. They have proposed a process for creating executable architectures by defining mapping among UML and CPN constructs. A subset of UML diagrams are used to represent the static view of an architecture. One of the main challenges in their work is the maintenance of concordance among different UML diagrams. They have addressed it by developing an integrated data dictionary manually. The data dictionary contains definition and description of each and every elements of all the diagrams used in the architecture. This manual maintenance of concordance, however, could also be regarded as a major limitation of their approach as it is prone to human error. The other major weakness in their approach is the manual addition of operational rules in the dynamic view of the architecture. In the context of this paper, operational rules mean constraints in a particular domain. Weak semantics of UML diagrams also raises the issue of consistency

* Corresponding author. Tel.: +92 3343737606.
E-mail addresses: sajjad.haider@khi.iba.edu.pk, sahaider@iba.edu.pk (S. Haider).

[1] http://en.wikipedia.org/wiki/System_architecture.

http://dx.doi.org/10.1016/j.jss.2014.08.049
0164-1212/

(Vanderperren and Dehaene, 2005; Bahill and Daniels, 2003; Evans, 1998; Whittle, 2000; Yi-zhi et al., 2004; OMG, 2007; Costal et al., 2011). Liles (2008) automated the manual transformation proposed by Wagenhals et al. (2003). Liles (2008) created a code in Rational System Developer tool that generated a file from UML that can be opened in CPN Tools. The transformation by Liles is limited to the mapping of an activity diagram to a CPN model.

Wang and Dagli (2011) have developed a framework that implements an executable system architecture by mapping SysML notations to CPN. For this purpose, they have introduced a new transformation scheme based on SysML Sequence diagram and have established concordance among Sequence diagrams, Activity diagrams and Block diagrams. They use the principles of model driven architecture (MDA) in their framework. For architecture evaluation purposes, Wang and Dagli (2011) have used BRITNeY Suite, a java application that runs on top of CPN that controls the simulation and generates graphical output such as message sequence charts (MSC) and state space graphs. They use these outputs to verify and validate the system architecture. Concordance between the diagrams in static view and the transformation between static and dynamic views is not completely automatic. In addition to this, the operational rules are added to CPN manually which leaves the consistency issues open.

Noguera et al. (2009) propose a UML-based framework in combination with Web Ontology Language (OWL) ontology and CPN. They use combination of UML activity diagram and OWL ontology for static view modeling and CPN for dynamic view modeling. They have developed ontology for UML activity diagram that helps to preserve the consistency of an Activity diagram. The framework also contains a mapping scheme that maps OWL ontology to CPN. All the constraints/rules are defined using decision boxes in the Activity diagram which limits the expressiveness of the modeled operational rules as Activity diagrams has very limited support for rules. In addition, the use of built-in CPN functions for rules mapping also restricts the rules mapping functionality. Moreover, the semi-automated process of transformation between OWL and CPN also raises the issue of consistency.

The Department of Defense Architecture Framework (DoDAF), a system engineering tool mainly used by the U.S. engineering and acquisition communities (DoDAF, 2010) also provides an organized way to model a system into a series of architecture products, and groups them into different "views". In 2009, DoDAF was revamped with a new data model, the DoDAF Meta Model (DM2). The introduction of DM2 shifts the focus of DoDAF from output products or documentation, to the collection of the underlying data that represent the architecture. DM2 introduces standard vocabulary in DoDAF that helps in defining the purpose, scope and information requirements of the architecture up-front and also provides a mechanism to store that information (McDaniel, 2009). The products in previous versions of DoDAF are renamed as models; models can be documents, spreadsheets, or other graphical representations, and serve as templates for organizing and displaying data in an easily understood format. These models, populated with architectural data, become the architectural view and the collections of these views are referred to as viewpoints. Use of multiple views provides depth in architecture but at the same time it also makes the model building process complex particularly for simple scenarios.

Grady (2009) used a combination of UML, SysML and four artifacts from traditional structured analysis and developed a Universal Architecture Description Framework. His proposed framework covers the modeling of software and hardware specification under one umbrella. According to Grady, his proposed framework is efficient but the framework can further be improved by forming a stronger linkage between analytical and synthesis process from problem space to solution space that help to strengthen the verification process of requirement through design. In his proposed framework Grady did not say much about model execution. Guangsheng et al. (2006) suggests a manual approach for transforming OWL DL and rules defined in Semantic Web Rule Language (SWRL) into Predicate Transition net (PrT-nets). The transformation allows rule inference using Petri nets. Zou et al. (2010) proposes accountability in business services through internet using combination of OWL and SWRL. They validate the OWL ontology in combination with SWRL rules through Pellet reasoner[2] and model action sequence of concepts in CPN. They, however, do not suggest any mapping between SWRL and CPN and rather use rules modeled in SWRL for reasoning.

The key issues that all the above mentioned approaches aim to solve are (a) a seamless transformation between different views of system architecture and (b) maintenance of consistency. Both issues are interrelated. Incomplete transformation between different views of a system's architecture forces the architect to manually model the leftover architectural components. These manual activities become the main source of introducing inconsistency in the system's architecture. Besides consistency maintenance, the existing approaches are also unable to verify the behavioral aspect in the static view which is due to insufficient support for modeling behavioral element, particularly operational rules at the abstract level within the problem domain's architectural description. Even the definition of operational rules in static view alone cannot solve the problem of behavior verification unless these rules are verified through dynamic view using a suitable simulation tool. The verification of operational rules through a simulation tool also requires mapping of these rules from the static view to the dynamic view.

The paper aims to address the issue of consistency maintenance across multiple views of systems architecture using the proposed Framework named Consistent Systems Architecture Description and Behavior Framework (CSADBF). The framework aids architects in validating the architectural description as well as in verifying the behavioral aspect of a system's architecture. Overall, CSADBF is comprised of two views: static and dynamic views. The framework enables an architect to model the static view via OWL ontology. The reason for using OWL ontology is to overcome the issue of static view modeling with languages having weak semantics. Even though ontology is good at defining a domain model but it does not support ordering or sequencing of activities within a problem domain and may not be the best choice when it comes to modeling interactions and behaviors. In the proposed framework, this short fall is overcome via UML activity diagram. To model the architectural behavior using UML activity diagram, the framework provides an automated mapping of OWL ontology constructs to UML class diagram constructs which in turn are used to generate UML activity diagram constructs. Even though certain elements (such as sequence of activities) of the activity diagram have to be built manually, the framework restrict the architects from introducing any new class (or methods) other than the classes that are available in the class diagram created from OWL ontology.

The dynamic view of the proposed framework uses Colored Petri Nets (CPN). To maintain consistency in the dynamic view and to establish concordance with given architectural description, the framework provides a set of mapping steps. These steps map (a) the architectural description from ontology to CPN global declaration (b) the process flow arrows from activity diagram to the process flow arrows in the CPN graphical models and (c) operation flow defined via SWRL to Guard conditions in CPN. In this way the framework supports automatic creation of dynamic view from static view. An architect can verify the desired behavior of

[2] Pellet is an OWL 2 reasoner, provides reasoning services for OWL ontologies. http://clarkparsia.com/pellet/.