A Technique of DEVS-Driven Validation

Vadym Shkarupylo

Abstract - **A technique of DEVS-driven (Discrete Event System Specification) distributed software systems validation has been proposed. As a case study the Composite Web Service usage scenario has been considered. There are have been considered two approaches to Composite Web Service validity checking – testing-driven one and the approach based on DEVS-driven simulation. Experimental results obtained have shown the reasonability of proposed technique practical usage.**

Keywords – **Validation, Simulation, Composite Web Service, DEVS, WS-BPEL, TLA+, Kripke Structure.**

I. Introduction

Today we have a plethora of different distributed software systems involved in the diverse business processes execution scenarios. The most prominent ones are actually based on Service Oriented Architecture (SOA) principles, e.g. composability, loose coupling, etc. [1]. Good demonstrative example of such systems are Amazon Web Services, where intensive formal methods usage is deeply involved in engineering process [2].

It should be mentioned though that formal methods usage (model checking for instance) can guarantee us the correspondence between obtained formal specification and given requirements but it will not guarantee us that the system will be actually applicable to the practical business-scenarios it is intended for. To make a leap ahead in the later direction the validity checking procedure has to be conducted. It can be done in two opposite/complementary ways – by way of testing or/and by way of simulation.

In this paper the opposite-scenario will be considered. The main idea here is to use simulation for validation purposes during engineering process when time costs are critical. It takes place in fact during iterative engineering [3]. At the same time the validity checking procedure is not considered to be apart – it is supposed to be one of the final stages of engineering process, a complementary one to verification procedure, implemented by way of model checking.

As a case study the Composite Web Service (CWS) usage scenario has been considered [4]. To conduct the validation by way of simulation the DEVS-formalism has been chosen (Discrete Event System Specification, by Bernard P. Zeigler) [5]. Its concepts of atomic and coupled models provide the convenient way for building scalable and easy reconfigurable models of distributed systems (Web Services in particular) and their components.

To maintain CWS functioning the centralized orchestration model described in WS-BPEL-specification (Web Services Business Process Execution Language) has been chosen [6]. In this paper the CWS WS-BPEL-description will be considered as service's functional property representation. Because CWS functioning itself is the sequence of atomic web services invocations, complemented with switch/case and flow constructs. So here and further atomic web services will be considered to be the CWS components. The soundness of such components orchestration-driven coordination can be checked in an automated manner with formal methods usage. For this purpose the TLA+ specification (Temporal Logic of Actions, by L. Lamport) has been chosen [7]. This formalism provides the rigid and convenient way for CWS functional properties representation. Such representation is called a formal model (specification) intended to be verified with model checking method – TLC (TLA Checker).

II. Problem Statement

Let Kripke structure on a given set of atomic prepositions AP be the representation of CWS TLA+ specification given [8]:

$$\langle S, \{s_0\}, R, L \rangle, \quad (1)$$

where S – specified system's states set, $s_0 \in S$ – initial state, $R \subseteq S^2$ – transitions set, $L: S \to 2^{AP}$ – states labeling function.

Let AP set is obtained from predefined $V \times D$ set, where $V = \{v_i | i = 1,2,...,m\}$ – state variables set, $D = \{0,1,2\}$ – set of allowable state variables values: 0 – atomic web service involved in composition is passive, 1 – its conducting the computations, 2 – the computation is already finished.

The task to be solved is to synthesize the following DEVS-structure from the Kripke structure (1) and to successfully conduct the DEVS-driven CWS validation with corresponding time costs reduction (in comparison with testing-driven alternative) [4]:

$$\langle IP, OP, Atomics, set, break \rangle, \quad (2)$$

where IP – resulting CWS coupled DEVS-model input ports set, OP – output ports set, $Atomics$ – total set of atomic DEVS-models representing the atomic web services, $set: OP \times Atomics \to IP \times Atomics$ – function for setting the connections between atomic models' ports, $break: 2^{Atomics} \to Atomics$ – function for breaking the connections between atomic models' ports.

Vadym Shkarupylo – Zaporizhzhya National Technical University, Zhukovsky Str., 64, Zaporizhzhya, 69063, UKRAINE, E-mail: vadshkar@yandex.ua

III. Technique Description and the Case Study

The place of the proposed technique in engineering process is shown on Fig. 1.

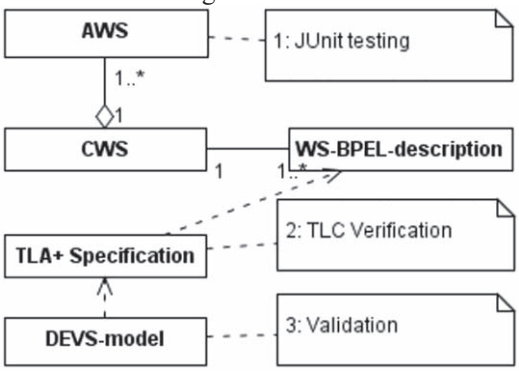

Fig. 1. The proposed technique specifics

On Fig. 1 AWS is a set of atomic web services used as CWS components. The CWS can be characterized with multiple WS-BPEL-descriptions: each description is the specification of the appropriate functional property. For such description (descriptions) the formal TLA+ specification then has to be synthesized. TLA+ specification is intended to be the basis for automated TLC-verification. When TLC-verification has been successfully accomplished the appropriate DEVS-model should be synthesized to conduct the simulation-driven validation.

The approach to the proposed technique correctness checking is as follows:

- conduct the testing-driven validation. Atomic web services here should be implemented with JAX-WS-technology usage. The CWS functioning should be driven through centralized WS-BPEL-orchestration;
- conduct the simulation-driven validation, following the way represented on Fig. 1;
- compare the results obtained (Fig. 2).

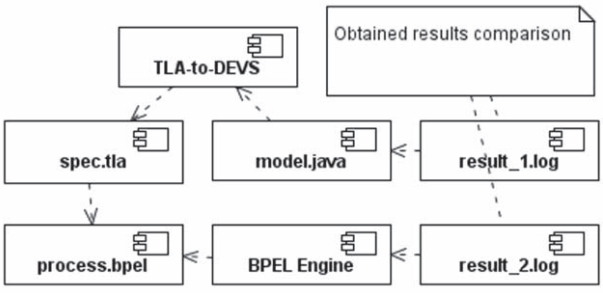

Fig. 2. The approach to technique checking

On Fig. 2 both mentioned ways of validity checking procedure implementation are shown. The upper one (ends with result_1.log artifact) is a DEVS-driven simulation, the lower one (ends with result_2.log artifact) is an alternative testing-driven validation. Both ways are based on a given WS-BPEL-description usage.

The synthesized DEVS-model is characterized as an adequate one and the proposed technique of validation – as the correct one if the result of CWS functional property implementation obtained by way of simulation is equal to the result obtained by way of testing. The proposed technique to simulation-driven validation practical usage will be considered to be reasonable if its correct and the simulation-driven validation time costs are significantly lower than testing-driven ones.

As a case study the CWS usage scenario has been considered. Let assume that such system is intended to perform the π-value calculation in a distributed manner (Fig. 3).

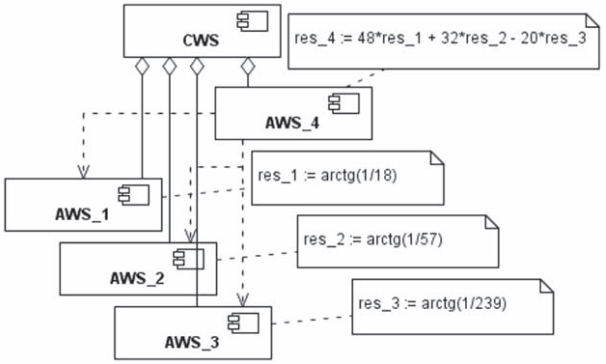

Fig. 3. Composite Web Service architecture

On Fig. 3 AWS_1,...,AWS_3 – atomic web services (the components) invoked concurrently and intended to retrieve intermediate calculations results, AWS_4 – gatherer-component for the resulting π-value obtaining. So there are can be distinguished two distinct steps of CWS functioning.

The implementation of the appropriate CWS based on orchestration model is given on Fig. 4.

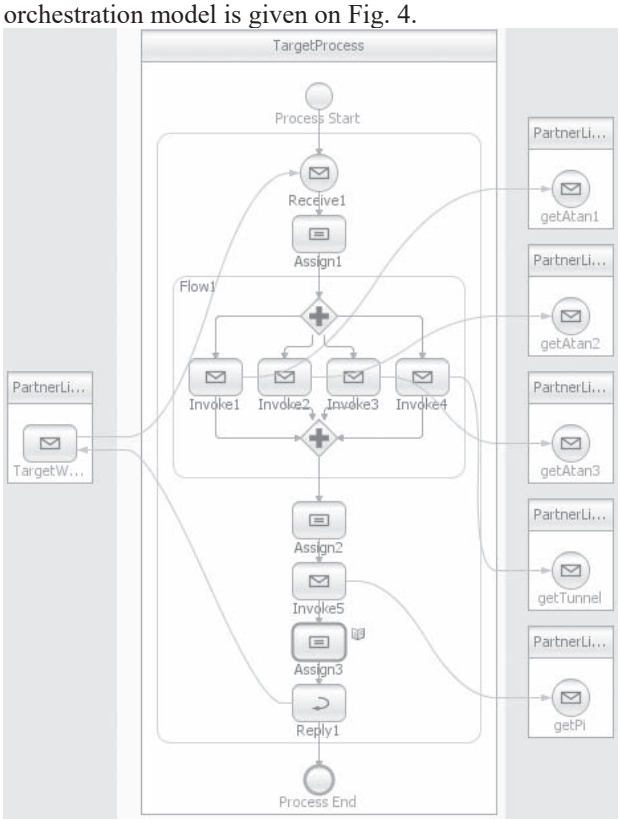

Fig. 4. Composite Web Service WS-BPEL-implementation

The generalized architecture of the appropriate coupled DEVS-model is given on Fig. 5.

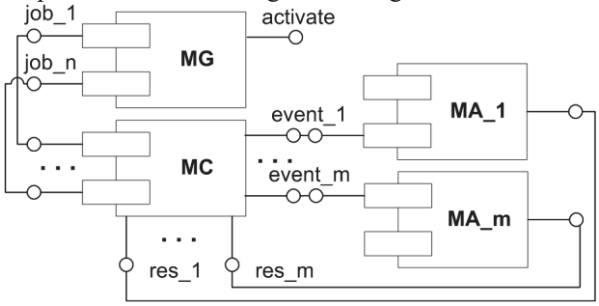

Fig. 5. Resulting coupled DEVS-model generalized architecture

On Fig. 5 MG – CWS client atomic DEVS-model, MC – BPEL-engine atomic DEVS-model, $MA_1,...,MA_m(m=4)$ – atomic DEVS-models of atomic web services involved.

IV. Results Analysis

The experiments have been conducted on the following platform: CPU – AMD K10, 3 GHz; RAM – DDR3, 2 GB, dual channel; OS – MS Windows 7; Java Runtime Environment – version 1.7.

Experimental results obtained are given on Fig. 6.

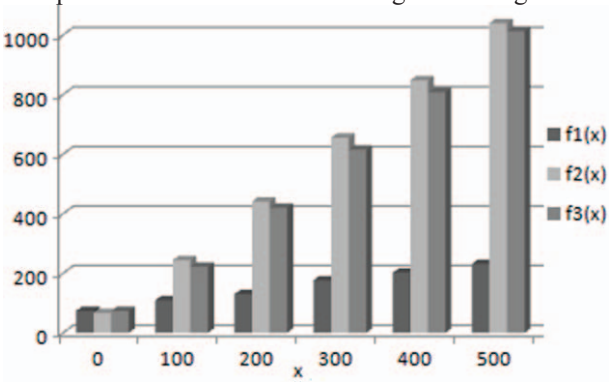

Fig. 6. Time costs comparison between testing-driven and DEVS-driven validations

On Fig. 6 x – predefined response time for each atomic web service (ms), $f_1(x)$ – time spent on DEVS-driven validation by way of simulation (ms), $f_2(x)$ – estimated aggregated CWS response time values obtained while DEVS-simulation (ms), $f_3(x)$ – actual time spent on testing-driven validation (ms).

An argument for proposed CWS DEVS-model adequacy is the equality of π-value calculation results, obtained with real CWS usage, and the ones, obtained with corresponding DEVS-model by way of simulation.

The adequacy of the resulting synthesized DEVS-model has been proofed by the equality of the obtained π-value with the π-value retrieved by the alternative testing-driven validation way. Which is also the proof of the proposed technique correctness.

The reasonability of the proposed validation technique practical usage is demonstrated on Fig. 6: on a given set of testing data values ($x = 0,10^2,...,5\cdot10^2$ ms) the DEVS-driven validation based on the proposed technique is about 3,4 times faster on average in comparison with alternative testing-driven validation. Such time costs reduction for validity checking procedure conduction can be characterized as significant.

V. Conclusion

Thus in given paper the technique of simulation-driven validation based on DEVS-formalism has been proposed. The proposed technique has been considered as a step of iterative engineering process, preceded by the automated formal verification. As analytical representation of formal TLA+ specification the Kripke structure has been used. Technique correctness has been checked.

The experimental results obtained have shown the reasonability of the proposed technique practical usage: on a given set of testing data values it has been shown that proposed technique requires significantly lower time costs (about 3,4 times faster on average) in comparison with alternative testing-driven validation.

References

[1] M.P. Papazoglou, P. Traverso, S. Dustdar and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges", *IEEE Computer*, Vol. 40, No. 11, pp. 38–45, 2007. DOI: 10.1109/MC.2007.400

[2] C. Newcombe, T. Rath, F. Zhang [et al.], "How Amazon Web Services Uses Formal Methods", *Communications of the ACM*, Vol. 58, pp. 66–73, 2015. DOI: 10.1145/2699417

[3] C. Larman, "*Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*", Addison Wesley Professional, Massachusetts, October 2004.

[4] V.V. Shkarupylo, R.K. Kudermetov and O.V. Polska, "DEVS-oriented Technique for Composite Web Services Validity Checking", *Radio Electronics, Computer Science, Control*, Vol. 35, No. 4, pp. 79-86. DOI 10.15588/1607-3274-2015-4-12

[5] G.A. Wainer and P.J. Mosterman, "*Discrete-Event Modeling and Simulation: Theory and Applications*", CRC Press, NY, 2010.

[6] Y. Vasiliev, "*SOA and WS-BPEL: Composing Service-Oriented Solutions with PHP and ActiveBPEL*", Packt Publishing Ltd., Birmingham, UK, 2007.

[7] L. Lamport, "*Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*", Addison-Wesley, Boston, 2002.

[8] E.M. Clarke, O. Grumberg and D. Peled, "*Model Checking*", The MIT Press, Massachusetts, 2001.