BPEL VERIFICATION: THE BACK-ANNOTATION PROBLEM

Ábel HEGEDÜS
Advisor: Dániel VARRÓ

I. Introduction

Complex distributed systems including numerous business processes are widely employed nowadays. Business processes implemented in BPEL [1] (Business Process Execution Language) are often used for providing autonomous services. Since the quality of these processes are frequently critical for its users, their design-time verification is essential. Numerous approaches, techniques and tools were developed to support verification, usually by modeling the behaviour of BPEL processes using formal languages.

Modeling the BPEL process, however, is only part of the solution. The results of the verification carried out on the formal model have to be described using the formalism of BPEL. This reverse projection of information is known generally as the *back-annotation* problem.

Given that the verification results describe an execution of the BPEL process, modeling the structure and behaviour of BPEL (*static model*) is not sufficient for providing back-annotation support. Therefore a *dynamic model* is defined, which describes the actual state of the process (e.g. activities currently executed) together with a *simulation trace model*, to represent the information regarding the history of changes in the dynamic model.

In this paper, we describe how the simulation trace models can be defined for a given BPEL verification approach [2]. Furthermore a brief description of the back-annotation transformation implemented in the VIATRA2 framework [3] is given.

Several different simulation models have been proposed, for instance [4] uses models which include the dynamic behaviour information as well as the static and defines transformation rules for the simulation of these models. However, the dynamic model doesn't use any generic metamodel and the simulation trace is not stored. This approach was implemented in the AToM^3 [5] framework, which provides meta-modeling and model transformation support. [6] introduces change history models which are similar to simulation models although defined on a lower level. Model changes are recorded incrementally in order to enable model synchronization.

II. BPEL modeling with Labeled Transition Systems

The BPEL standard does not define formal semantics for business processes. However, design-time verification of BPEL processes demands approaches which provide semantics through modeling business processes using formal languages. In the selected method, labeled transition systems defined in the Symbolic Analysis Laboratory (SAL) [7] were chosen for this purpose, and requirements against the BPEL processes are verified using model checking.

The SAL model of a given BPEL process is generated by a complex model transformation (BPEL2SAL). The various elements of the BPEL process are represented as stateful SAL variables and the process execution is modeled with transitions that alter the state of variables if the transition system is in a given state. Note that the BPEL2SAL transformation abstracts the behaviour of processes [2]. A sequence of state changes (transition firings) correspond with the execution of the BPEL process. While model checking of the SAL system may return a counter-example, deriving the corresponding BPEL execution is a non-trivial back-annotation problem (see Fig. 1).

III. Back-annotation of the counter-example

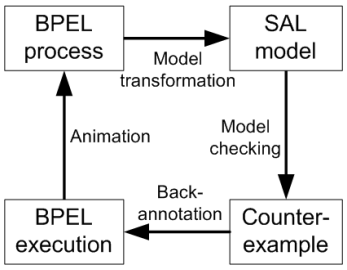

Figure 1: Approach

The BPEL execution (target trace) can be derived automatically from the SAL counter-example (source trace) by means of a model transformation, though this transformation is not the reverse of the original BPEL2SAL one. Instead of altering the static models, runtime information is represented in dynamic models and the traces are modeled separately using simulation metamodels. The SAL counter-example is modeled as a series of *steps* which represent the *firing transition* and the *variable assignments* it invokes. The assignments refer to *runtime variables* (in the dynamic model) that store the actual value of a variable in the static model and the values of the variable before and after the assignment. By storing the earlier value, backwards stepping in the trace is possible.

The simulation metamodel for the BPEL execution is similar to the SAL semantics given that the purpose of the original BPEL2SAL transformation was to model BPEL semantics as a transition system. Thus the various elements of the BPEL process are modeled as stateful entities and the steps of the execution represent state changes of the different entities. It is important to note that the simulation metamodels did not exist before and were devised in order to make the back-annotation possible.

The back-annotation transformation is implemented as an interface in the sense that it provides the following functions for handling the BPEL trace: (a) *initialize* for creating the dynamic BPEL model and the empty trace; (b) *forward step* for updating the dynamic model according to the next step in the trace, the step is generated based on the corresponding SAL trace step if it does not exist yet; (c) *backwards step* for reverting the dynamic model to the state before the actual step; and (d) *reset* for returning to the start of the trace and the initial state of the dynamic model. After the execution of each function, the state changes of the BPEL elements are exported so that they can be used outside the model transformation framework (e.g. to drive the animation of the BPEL process).

IV. Conclusion

In this paper, we presented the design-time verification of BPEL processes as a challenging problem where automated back-annotation is essential. In order to provide transformation-based back-annotation of the counter-example returned by model checking, we defined dynamic and simulation trace models for both SAL and BPEL. The implemented tranformation can be used for animating the execution of the BPEL process.

Current research directions include exploring the possibilities of generalizing the simulation trace model and creating a domain-independent simulation interface providing similar functions as the described back-annotation transformation.

References

[1] OASIS, "Web services business process execution language version 2.0 (OASIS standard)," 2007.

[2] M. Kovács, D. Varró, and L. Gönczy, "Formal Analysis of BPEL Workflows with Compensation by Model Checking," *IJCSSE*, 23(5), November 2008.

[3] Fault Tolerant System Research Group, "VIsual Automated model TRAnsformations," http://www.eclipse.org/gmt/VIATRA2/.

[4] J. de Lara, "Meta-modelling and graph transformation for the simulation of systems," *Bulletin of the EATCS*, 81, 2003.

[5] J. de Lara and H. Vangheluwe, "AToM^3: A tool for multi-formalism and meta-modelling," in *FASE*, 2002.

[6] I. Ráth, G. Varró, and D. Varró, "Change-driven model transformations," in *Proc. of MODELS'09, CM/IEEE 12th International Conference On Model Driven Engineering Languages And Systems*, 2009, Accepted.

[7] N. Shankar, "Symbolic Analysis of Transition Systems," in *ASM 2000*, Y. Gurevich, P. W. Kutter, M. Odersky, and L. Thiele, Eds., number 1912 in LNCS, pp. 287–302, Monte Verità, Switzerland, mar 2000. Springer-Verlag.