Constraint-Based Runtime Prediction of SLA Violations in Service Orchestrations (extended abstract)

Dragan Ivanović[1], Manuel Carro[2], Manuel Hermenegildo[3]

[1] idragan@clip.dia.fi.upm.es Universidad Politécnica de Madrid
[2] mcarro@fi.upm.es, manuel.carro@imdea.org
[3]herme@fi.upm.es, manuel.hermenegildo@imdea.org
Universidad Politécnica de Madrid and IMDEA Software Institute

Abstract: Quality of Service (QoS) attributes, such as execution time, availability, or cost, are critical for the usability of Web services. This in particular applies to service compositions, which are commonly used for implementing more complex, higher level, and/or cross-organizational tasks by assembling loosely-coupled individual service components (often provided and controlled by third parties). The QoS attributes of service compositions depend on the QoS attributes of the service components, as well as on environmental factors and the actual data being handled, and are usually regulated by means of Service-Level Agreements (SLAs), which define the permissible boundaries for the values of the related properties. Predicting whether an SLA will be violated for a given executing instance of a service composition is therefore very important. Such a prediction can be used for preventing or mitigating the consequences of SLA violations ahead of time.

We propose a method whereby constraints that model SLA conformance and violation are derived at any given point of the execution of a service composition. These constraints are generated using the structure of the composition and properties of the component services, which can be either known or measured empirically. Violation of these constraints means that the corresponding scenario is unfeasible, while satisfaction gives values for the constrained variables (start / end times for activities, or number of loop iterations) which make the scenario possible. These results can be used to perform optimized service matching or trigger preventive adaptation or healing.

The derivation of the constraints that model SLA conformance and violation is based on two key information sources. The first one (called *continuation*) describes the processing that remains to be performed until the end of execution of a given orchestration instance. In general, the continuation is either provided by an orchestration engine, or extracted from its internal state and/or external events. The second information source is the set of assumptions on QoS for the service components used in the orchestration, which are normally empirically collected. The component QoS is described with upper and lower bounds (under some level of confidence), while the prediction is based on (crisp) logical reasoning about the possibility of SLA violation and compliance under the given component bounds.

The constraint-based prediction can be performed at each point of execution for

M.M. Gallardo, M. Villaret, L. Iribarne (Eds.): PROLE'2012, pp. 195-196, ISBN:978-84-15487-27-2.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

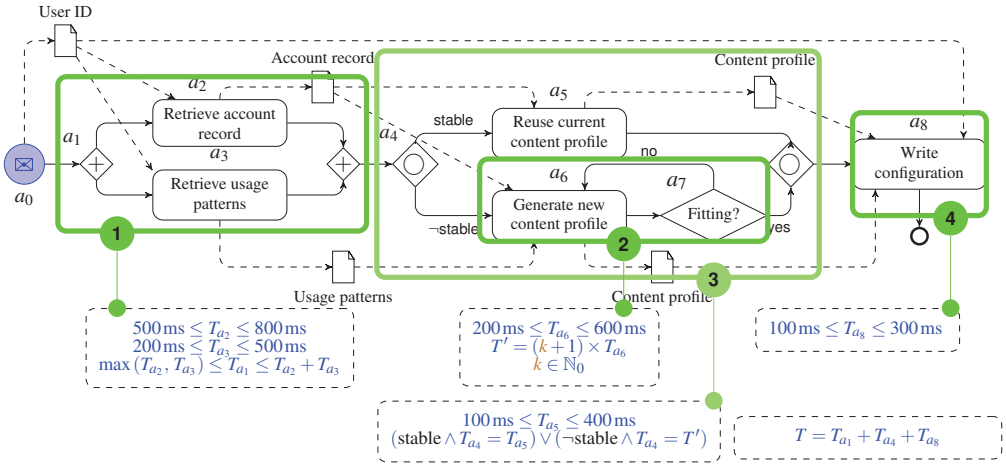

Figure 1: Generation of constraints to predict SLA violations.

which we have the continuation. Unlike data-mining approaches, it does not depend on a historic state of the environment, and can adapt instantly to a dynamic changes in the orchestration code. The first step in prediction is the formulation of the constraint model for the continuation of the running instance (Figure 1), which is dynamically built from the structure of the continuation and the (expected) component QoS bounds. This model expresses the possible range of the remaining QoS for the executing instance (execution time T in case of Figure 1). As an example, the constraints for the structure marked **1** (an and-split) include bounds for the activities and for the structure itself: the components may in fact run in parallel or, if only one thread and one CPU are available, sequentially. The constraints for nested control structures are combined upwards to give the total QoS, T. By constraining T to meet or not the SLA conditions ($T \le T_{\max}$) and $T > T_{\max}$, resp.), we try to rule out one of the two cases, and thus to predict the other (e.g., *SLA compliance ruled out* $\Rightarrow$ *SLA failure predicted*).

Additionally, and since constraints are generated dynamically, we get progressively simpler (and more accurate) systems as the execution proceeds. In addition, we use techniques derived from automatic complexity analysis to derive bounds on the number of loop iterations (k in Figure 1) as functions of the input data, which greatly improves prediction accuracy.

Keywords: Service Orchestrations, Quality of Service, Service Level Agreements, Monitoring, Prediction, Constraints.

A full version of this paper has been published in the Proceedings of ICSOC 2011 [ICH11].

Bibliography

[ICH11] D. Ivanović, M. Carro, M. Hermenegildo. Constraint-Based Runtime Prediction of SLA Violations in Service Orchestrations. In Kappel et al. (eds.), *Service-Oriented Computing – ICSOC 2011*. LNCS 7084, pp. 62–76. Springer Verlag, December 2011. Best paper award.

PTL: A Prolog-based Model Transformation Language

Jesús M. Almendros-Jiménez[1] **and Luis Iribarne**[2]

[1] jalmen@ual.es
[2] luis.iribarne@ual.es
Dpto. de Lenguajes y Computación
Universidad de Almería
04120-Almería (Spain)

Abstract: In this paper we present a model transformation language based on logic programming. The language, called PTL (*Prolog-based Transformation Language*), can be considered as an hybrid language in which ATL-style rules are combined with logic rules for defining transformations. ATL-style rules are used to define mappings from source models to target models while logic rules are used as helpers. The proposal has been implemented so that a Prolog program is automatically obtained from a PTL program. We have equipped our language with debugging and tracing capabilities which help developers to detect programming errors in PTL rules.

Keywords: MDD; Logic Programming; Software Engineering

1 Introduction

Model Driven Engineering (MDE) is an emerging approach for software development. MDE emphasizes the construction of models from which the implementation is derived by applying model transformations. Therefore, *Model Transformation* [Tra05] is a key technology of MDE. According to the *Model Driven Architecture (MDA)* initiative of the *Object Management Group (OMG)* [OMG03], model transformation provides a framework to developers for transforming their models.

MDE proposes (at least) three elements in order to describe a model transformation: the first one is the so-called *meta-meta-model* which is the language for describing meta-models. The second one consists in the *meta-models* of the models to be transformed. Source and target models must conform to the corresponding meta-model. Such meta-models are modeled according to the meta-meta-model. The third one consists in the source and target models. Source and target models are instances of the corresponding meta-models. Furthermore, source and target meta-models are instances of the meta-meta-model. In order to define a model transformation the source and target models are modeled with respect to the meta-meta-model, and source and target meta-models are mapped.

In this context, model transformation needs formal techniques for specifying the transformation. In most of the cases transformations can be expressed with some kinds of *rules*. The rules have to express how source models can be transformed into another. Several transformation languages and tools have been proposed in the literature (see [CH06] for a survey). The most relevant one is the language *ATL (Atlas Transformation Language)* [JABK08] a *domain-specific language* for specifying model-to-model transformations. ATL is a hybrid language, and pro-

M.M. Gallardo, M. Villaret, L. Iribarne (Eds.): PROLE'2012, pp. 197-211, ISBN:978-84-15487-27-2.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.