Pattern-Based Translation of BPMN Process Models to BPEL Web Services

Chun Ouyang, Queensland University of Technology, Australia
Marlon Dumas, Queensland University of Technology, Australia and University of Tartu, Estonia
Arthur H.M. ter Hofstede, Queensland University of Technology, Australia
Wil M.P. van der Aalst, Queensland University of Technology, Australia and Eindhoven University of Technology, The Netherlands

ABSTRACT

The business process modeling notation (BPMN) is a graph-oriented language primarily targeted at domain analysts and supported by many modeling tools. The business process execution language for Web services (BPEL) on the other hand is a mainly block-structured language targeted at software developers and supported by several execution platforms. Translating BPMN models into BPEL code is a necessary step towards standards-based business process development environments. This translation is challenging since BPMN and BPEL represent two fundamentally different classes of languages. Existing BPMN-to-BPEL translations rely on the identification of block-structured patterns in BPMN models that are mapped onto structured BPEL constructs. This article advances the state of the art in BPMN-to-BPEL translation by defining methods for identifying not only perfectly block-structured fragments in BPMN models, but quasi-structured fragments that can be turned into perfectly structured ones and flow-based acyclic fragments that can be mapped onto a combination of structured constructs and control links. Beyond its direct relevance in the context of BPMN and BPEL, this article addresses issues that arise generally when translating between graph-oriented and block-structured flow definition languages.

Keywords: BPMN; BPEL; business process modeling; Web services

INTRODUCTION

The business process execution language for Web services (BPEL) (OASIS, 2006) is emerging as a de facto standard for implementing business processes on top of Web services technology. Numerous platforms support the execution of BPEL processes. Some of these platforms also provide graphical editing tools for defining BPEL processes. However, these tools directly follow the syntax of BPEL without elevating the level of abstraction to make them usable during the analysis and design phases of the development cycle. On the other hand, the
Business process modeling notation (BPMN) (Object Management Group, 2006) has attained some level of adoption among business analysts and system architects as a language for defining business process blueprints for subsequent implementation. Despite being a recent proposal, BPMN is already supported by more than 30 tools.\(^2\) Consistent with the level of abstraction targeted by BPMN, none of these tools supports the execution of BPMN models directly. Instead, some of them support the translation of BPMN to BPEL.

Close inspection of existing translations from BPMN to BPEL, for example the one sketched in (Object Management Group, 2006), shows that these translations fail to fulfill the following key requirements: (i) completeness, that is applicable to BPMN model with arbitrary topology; (ii) automation, that is capable of producing target code without requiring human intervention to identify patterns in the source model; and (iii) readability, that is consistently producing target code that is understandable by humans. The latter requirement is important since the BPEL definitions produced by the translation are likely to require refinement (e.g., to specify partner links and data manipulation expressions) as well as testing and debugging. If BPEL was only intended as a language for machine consumption and not for human use, it could be replaced by mainstream programming languages or even (virtual) machine languages, but this would defeat the purpose of BPEL as a language for service composition.

The limitations of existing BPMN-to-BPEL translations are not surprising given that BPMN and BPEL belong to two fundamentally different classes of languages. BPMN is graph-oriented while BPEL is mainly block-structured (albeit providing graph-oriented constructs with syntactical limitations). Mapping between graph-oriented and block-structured process definition languages is notoriously challenging. In the case of flowcharts, mapping unstructured charts to structured ones is a well-understood problem. However, graph-oriented process definition languages extend flowcharts with parallelism (i.e., AND-splits and AND-joins) and other constructs such as deferred choice (Van der Aalst, ter Hofstede, Kiepuszewski, & Barros, 2003).

In prior work (Ouyang, Dumas, Breutel, & ter Hofstede, 2006), we proposed a translation that achieves the completeness and automation requirements outlined above for a core subset of BPMN models. However, the code produced by this translation lacks readability. Essentially, the BPMN process model is translated into a set of event-condition-action rules, and these rules are then encoded using BPEL event handlers. Thus, the translation does not exploit the block-structured constructs of BPEL, which would clearly lead to more readable code.

This article presents a complementary technique to translate BPMN to BPEL that emphasizes the readability requirement. The proposal is based on the identification of structural patterns of BPMN models which can be translated into block-structured BPEL code. The patterns are divided into: (i) well-structured patterns, which can be directly mapped onto block-structured BPEL constructs; (ii) quasi-structured patterns, which can be re-written into perfectly structured ones and then mapped onto block-structured BPEL constructs; and (iii) flow-based acyclic fragments which can be mapped onto combinations of block-structured BPEL constructs and additional control links to capture dependencies between activities located in different blocks.

Beyond their direct relevance in the context of BPMN-to-BPEL mapping, the translation patterns and algorithm presented in this paper address issues that arise generally when translating from graph-oriented process languages (e.g., UML Activity Diagrams, EPCs, YAWL, or Petri nets) to block-structured ones.

The rest of the article is structured as follows. The second and third sections provide an overview of BPMN and BPEL respectively. Next, the fourth section presents the identification of patterns in a given BPMN process model and their mappings onto BPEL. The overall translation approach is then illustrated through a case study in the fifth section. Finally, the sixth...
Related Content

**Service Evolution and Maintainability**
[www.igi-global.com/chapter/service-evolution-maintainability/26086?camid=4v1a](www.igi-global.com/chapter/service-evolution-maintainability/26086?camid=4v1a)

**Extensible Architecture for High-Performance, Scalable, Reliable Publish-Subscribe Eventing and Notification**
[www.igi-global.com/article/extensible-architecture-high-performance-scalable/3108?camid=4v1a](www.igi-global.com/article/extensible-architecture-high-performance-scalable/3108?camid=4v1a)

**AND/OR Graph and Search Algorithm for Discovering Composite Web Services**
[www.igi-global.com/article/graph-search-algorithm-discovering-composite/3069?camid=4v1a](www.igi-global.com/article/graph-search-algorithm-discovering-composite/3069?camid=4v1a)
Adaptive Ensemble Multi-Agent Based Intrusion Detection Model
www.igi-global.com/chapter/adaptive-ensemble-multi-agent-based/43646?camid=4v1a