Chapter 7
Validating Component-Based Implementations of Business Processes

Jens Lemcke
SAP Research Karlsruhe, Germany

Andreas Friesen
SAP Research Karlsruhe, Germany

Tirdad Rahmani
SAP Research Karlsruhe, Germany

ABSTRACT

This chapter provides a formal specification of non-atomic, relaxed action refinement suited for component-based business process engineering. Engineering a business process involves multiple process models created by different people on different levels of abstractions. Keeping the models consistent during the engineering procedure—refinement validation—is one objective of this chapter. In component-based software engineering, the lowest abstraction of a business process is mapped on existing components that have a description of their behaviors. Checking the consistency of process and component behavior—grounding validation—is the second objective. Both refinement and grounding validation increase the robustness of business process implementations and the productivity of process engineers. Technically, the specification given in this chapter is in terms of deadlock analysis in safe Petri nets. The evaluation of this straight-forward implementation underlines the exponential complexity of deadlock analysis in safe Petri nets. For use cases with more than 30 activities per process or heavy parallelism, optimized implementations are needed.

DOI: 10.4018/978-1-60960-485-1.ch007
INTRODUCTION

Electronic business interoperability critically depends on proper integration methods as well as their usable embedding in the software development process. We apply this general statement in the domain of component-based standard business software: A set of generic components from different vendors is customized and integrated by a team of consultants to implement the specific business process of an end-user organization. That classical enterprise application integration (EAI) activity consists of data integration, data flow integration, and control flow integration.

We restrict to the challenge of control flow integration. We address control flow integration in a new way by linking business to IT (information technology). The business view is addressed by validating arbitrary refinement steps of business process models, ranging from abstract business-oriented process models to more specific IT-oriented process models. The IT view is additionally addressed by validating the grounding of process models to existing IT components which pose control flow requirements themselves. The validations are performed using Petri nets. As an assumption, we follow the semantic Web services and component-based systems argumentation that information on IT components’ behavior, expressed in a component model, is mandatory to mechanically support control flow integration—see for example, (Heineman & Councill, 2001) and (Pistore, Spalazzi, & Traverso, 2006).

The main question of this chapter is:

How to support the consultants creating a control flow that both integrates the component models and implements a specific business process?

A lot of research has been performed for control flow integration. The majority of which proposes a method to generate an integration either fully automatic with a certain degree of confidence—for example, (Dijkman, Dumas, García-Banuelos, & Käärik, 2009)—or semi-automatic from a formal specification—for example, (Berardi, Calvanese, Giacomo, Hull, & Mecella, 2005) and (Lemcke, 2009). Both approaches have their downsides. In the fully automated case, the generated integration comes with a degree of confidence which is always below certainty. Therefore, no matter how good the machine’s guess was, the integration has to be entirely checked by the consultant. The semi-automatic case can be seen as an intentional approach. The consultant expresses their business intention of the integration, for example in the form of policies, and the system generates an according integration. However, expressing one’s intention can be assumed to be as hard as creating the integration directly: Policies could conflict each other, miss desired cases, or cover unwanted cases. The result is a natural human dislike against automatic wizards operating on behalf of the consultant because in the end, it is the consultant’s personal responsibility that components are integrated correctly and do not cause costly business disruptions at run time.

Due to the downsides of integration approaches that automatically “guess” or convert business intentions, we propose an approach that lets the consultant model the integration manually, but provide an automatic verification that determines technical mismatches between the control flow of the integration and the components’ behaviors, called grounding validation. In addition, we support the consultant transforming their business process intention to an integration by a means of verifying correctness of process model refinement: A sketch of the integration idea can be modeled as an abstract process model $A$. A finer process $B$ can be modeled together with a relation stating for each step $b$ of $B$ which step $a$ of $A$ it refines. The refinement relation can be seen as covering the intention of each $b$, namely, operationalizing the corresponding abstract step $a$. Vice versa, the refinement relation covers the semantics of each step $a$, namely, to consist of all corresponding $b$. 

