Chapter 7

Light-Weight Semantic Integration of Generic Behavioral Component Descriptions

Jens Lemcke
SAP Research, Germany

ABSTRACT

As stated by the Aberdeen group, integration costs 40% of a company’s IT budget today. The difficulty of integration arises because the description of software components—in service-oriented architectures (SOA) predominantly done by XML schema definition (XSD) and the Web service definition language (WSDL)—is on the technical level. The technical description neglects important detail of the intended usage of software components, which is also referred to as their semantics. Particularly, semantics needs to be considered in two major integration tasks. First, semantically corresponding data types that can be used for communication between components need to be identified. Second, natural language documentation needs to be studied today in order to understand component behavior; that is, dependencies between operation invocations and how semantically different outcomes of operation calls are represented in the technical output format. The approach presented in this chapter supports the two tasks as follows. First, closed frequent itemset mining (CFIM) is employed to help identifying semantically corresponding data types. Second, a formal representation for component behavior is introduced. However, as component behavior is specified during component development, but used during integration—two distinct phases involving distinct teams—we provide model transformations to ensure the consistent transfer of generic behavioral information to specific integration constraints before automated integration techniques can be applied. We applied the CFIM on the message types exposed by SAP’s standard software components and show that we are able to find semantically relevant correspondences. Furthermore, we demonstrate the practical applicability of our behavioral model transformations on the basis of an SAP best practice business scenario. With the little more effort to specify behavioral information at development time in a formal way instead of in natural language, our approach facilitates the reuse of behavioral component descriptions in multiple integration projects and eases the construction of correct integrations.

DOI: 10.4018/978-1-60566-804-8.ch007
INTRODUCTION

The integration of different business partners’ IT systems is an essential task to improve the agility of a company and thus facilitate its success in a more and more dynamic business environment. Unfortunately, the task of business integration is extremely expensive. As stated by the Aberdeen group, it consumes about 40% of a company’s IT budget (Kastner and Saia, 2006). We address two main cost drivers in this chapter.

1. An inherent prerequisite for complex workflow integration is enumerating the potential communications between the IT systems that were independently developed and are now to be integrated. This task is based on the semantic similarity of the participant’s message types. The identification of semantic correspondences is one of the two by far most work-intense tasks in IT system integration besides the actual design of the collaborative business process (Küster et al., 2007, Pistore et al., 2005d).

2. The behavioral capabilities of IT systems – in this chapter we focus on dependencies of operation invocations – are usually documented in natural language. The integration of IT systems into complex workflows must properly utilize the IT systems with respect to their capabilities. This means that a team of consultants has to interpret the natural language documentation in order to properly create the complex workflow. Interpretation of natural language is ambiguous and cannot be stored for the repeated use of the IT systems in subsequent integration projects.

We base the solution of the above mentioned challenges in the context of the Web services architecture (Kraßig et al., 2004). That means we assume the existence of WSDL-based Web service descriptions and XSD-based message type descriptions. We address the challenges in the following ways.

1. We employ closed frequent itemset mining (CFIM) to identify semantically similar message types. This task serves two purposes. First, identified semantically similar types are not necessarily technically identical. Thus, types being technically incompatible require either to be changed to become interoperable or an appropriate adapter needs to be constructed if the identified types are later used for communication. Second, identified semantically similar types can be used for communication between multiple components or IT systems. This information is one prerequisite for process integration in the following step.

2. We propose a methodology to manage behavioral information throughout a component’s lifecycle: When a component is built, writing a technical documentation in natural language is replaced by formally describing the generic behavior of components – this is in our approach done by drawing a state transition system (STS). When a component is integrated with others, the specific role the component plays in that integration is derived in a model-driven way from its generic behavior. As no interpretation of natural language is necessary any more, ambiguity is gone and the unambiguous component’s capability description can be reused in subsequent integration projects.

As connotated, we instantiate the methodology by introducing one possible syntax and semantics of behavioral models and by defining the necessary transformations of behavioral models for each step in the methodology.

3. As an integration of software components must be executable to be effective, we define the execution semantics of behavioral