Chapter IV

Specification of Business Components Using Temporal OCL

Stefan Conrad
University of Düsseldorf, Germany

Klaus Turowski
University of Augsburg, Germany

ABSTRACT
Compositional, plug-and-play-like reuse of black box components requires sophisticated techniques to specify components, especially when combined with third party components that are traded on component markets to individual customer’s business application systems. As in established engineering disciplines, like mechanical engineering or electrical engineering, formal documentation of business components that become part of contractual agreements are needed. With this initial problem, we explain the general, layered structure of software contracts for business components and show shortcomings of common specification approaches. Furthermore, we introduce a formal notation for the specification of business components that extends the Object Constraint Language (OCL) and allows for a broader use of the Unified Modeling Language (UML) with respect to the layered structure of software contracts for business components.

INTRODUCTION
Compositional, plug-and-play-like reuse of black box components requires sophisticated techniques to specify components, especially when combined with third party components that are traded on component markets to individual customer’s business application systems. As in established engineering disciplines, like mechanical engi-
neering or electrical engineering, formal documentation of business components that become part of contractual agreements are needed. With this initial problem, we explain the general, layered structure of software contracts for business components and show shortcomings of common specification approaches. Furthermore, we introduce a formal notation for the specification of business components that extends the Object Constraint Language (OCL) and allows for a broader use of the Unified Modeling Language (UML) with respect to the layered structure of software contracts for business components.

The remainder of the chapter remains as follows. After providing background information in the next section, we discuss the necessity of a multi-level notation standard. Thereafter, we explain how the OCL can be used to specify business components. Using this as a basis, we proceed to the main thrust of our chapter - the temporal extension of OCL. Finally, we present our conclusions and provide an outlook.

SOFTWARE CONTRACTS FOR BUSINESS COMPONENTS

Combining off-the-shelf software components that are offered by different vendors to individual customer’s business application systems is a goal that is followed up for a lengthy time. By achieving this goal, advantages of individually-programmed software that is combined with standardized, off-the-shelf software could come together. In this context, we are speaking about compositional reuse techniques. Compositional reuse represents special kinds of reuse techniques as generative techniques or code and design scavenging (Sametinger, 1997, pp. 25-28). The emphasis on compositional reuse stems from our guiding model, which is the compositional, plug-and-play-like reuse of black box components that are traded on a component market. In general, a guiding model is an ideal future state that might not be reached completely.

Corresponding with our guiding model, a company that (e.g., needs new software for stock keeping), could buy a suitable software component on the component market and further integrate it into its business application system with little effort. Brown and Wallnau (1996, pp. 11-14) explain the steps that are generally necessary to do so (e.g., technical and semantic adaptation or composition). Expected improvements, which should come along with using software components, concern cost efficiency, quality, productivity, market penetration, market share, performance, interoperability, reliability, or software complexity, cf., e.g., Orfali, Harkey, and Edwards (1996 S. pp. 29-32).

According to Fellner and Turowski (2000, pp. 3-4), the term component is defined as follows: A component consists of different (software-) artifacts. It is reusable, self-contained, and marketable; provides services through well-defined interfaces; hides its implementation, and can be deployed in configurations unknown at the time of development. A business component is a component that implements a certain set of services out of a given business domain. Refer to Szyperski (1998, pp. 164-168) and Fellner and Turowski (2000) for an in-depth discussion of various other component approaches given in literature.

To use business components according to our guiding model, it is necessary to standardize them as detailed in a discussion on standardizing business components (cf., Turowski, 2000). Furthermore, we must describe the component’s interface and behavior in a consistent and unequivocal way. In short, we have to specify them. Specification
UML- and XML-Based Change Process and Data Model Definition for Product Evolution
Ping Jiang, Quentin Mair, Julian Newman and Josie Huang (2005). Software Evolution with UML and XML (pp. 190-221).
www.igi-global.com/chapter/uml-xml-based-change-process/29614?camid=4v1a