Abstract

This dissertation addresses the problem of connecting high-level, executable business rules with existing object-oriented applications.

State-of-the-art research on developing object-oriented software applications with rule-based knowledge advocates making rules explicit and separate from the object-oriented core functionality. Although many approaches target this goal and are a considerable improvement on the solution that embeds rules as conditional statements in object-oriented applications, these approaches still suffer from three major inherent problems.

First, even when business rules are successfully decoupled, the rule connection code is still tangled with and scattered in the implementation of the core application functionality. Therefore, when existing business rules need to be integrated differently, or when new business rules need to be connected at unanticipated events, the source code of the core application must be adapted manually at different places. Consequently, it becomes difficult to localize, add, change or remove rule connections.

Secondly, regardless of the approach taken to decouple the business rules, executable rules are ultimately low-level. This makes rules not understandable to domain experts who are not adept at programming. A third, and closely related problem is a tight coupling between the business rules and the existing implementation of the core application. This causes rules to be fragile and not reusable, and prohibits the non-invasive realization of rules in terms of unanticipated implementation elements. As a consequence, business rules cannot be deployed by the domain experts without the intervention of a developer.

This dissertation presents a comprehensive solution to these problems, enabling existing applications to integrate business rules at the domain level. The first problem is addressed by encapsulating the rule connection in a separate module, decoupled from both the core application functionality and the rules. This decoupling is not straightforward because rule connections *crosscut* the core application functionality. *Aspect-Oriented Programming* (AOP) provides new modularization mechanisms, i.e. aspects, for the encapsulation of crosscutting code while ensuring *dependency inversion* between the core application and the aspects. These properties make AOP suitable for encapsulating crosscutting rule connections. This dissertation identifies commonalities and variabilities in the implementation of rule connection aspects and proposes abstracting these recurrent issues as elements of *aspect patterns*.

The second and third problems are addressed by building a layer of abstraction, a *domain model*, which allows for the expression of business rules in terms of domain concepts. This domain layer is able to represent domain concepts explicitly. A dedicated high-level business rule language is provided which enables the expression of high-level rules in terms of the domain concepts. Consequently, the coupling between the existing implementation of the core application and the rules is loosened. Moreover, the domain model is evolvable, which allows for the realization of unanticipated domain concepts and business rules that appear as a result of domain evolution.

This dissertation observes that, although aspects are a suitable solution to the problem of decoupling crosscutting rule connections, they exclude domain experts, as these aspects reside at the implementation level. Moreover, rule connection aspects need to take into account several recurrent issues, which renders the task of implementing these aspects difficult for developers. This dissertation supports the expression of rule connections at the domain level. A second dedicated high-level language is provided for this purpose.

The solution presented in this dissertation incorporates ideas from *Model-Driven Engineering (MDE)* in order to achieve the automatic generation of executable implementations for the high-level rules and rule connections. High-level rules and rule connections are automatically transformed to rule objects and rule connection aspects respectively.

The approach presented in this dissertation is evaluated in the domain of *Service-Oriented Architecture (SOA)*. Service-oriented applications are very volatile: new services appear, services become unavailable, non-functional properties of services vary (even at run-time), and applications need to cope with all these changes. Moreover, clients also change their requirements with respect to the selection and integration of services. This dissertation shows how high-level business rules can automate the customization of service-oriented applications. A Web services management layer, the WSML, is used as case study. Two scenarios are presented: an *evolution scenario*, which shows that it is possible to add new rules to the existing management framework, and a *refactoring scenario*, which shows that existing rules in the core WSML implementation can be refactored and externalized as high-level business rules.