Towards Automatic Integration of High-Level Business Rules Using Aspect-Oriented Programming

María Agustina Cibrán - Maja D'Hondt
System and Software Engineering Lab
Vrije Universiteit Brussel
Belgium

{mcibran, mjdhondt}@vub.ac.be

1. Context

The real-world domains of many software applications are inherently knowledge intensive. Part of this knowledge is rule-based, representing knowledge about policies, recommendations and decisions based on business information. For instance, e-commerce applications contain discount and personalization policies, medical applications contain health care decisions, etc. Typically, rule-based knowledge is represented implicitly and thus tangled in the application's code, with negative effects on reusability and maintainability. Nowadays rule-based knowledge is gaining more and more importance and it is also referred as business rules [1, 2, 3]. A business rule is a statement that defines or constraints some aspect of the business, either structure or behavior [4]. The main characteristic of business rules is that they tend to evolve more frequently than the core application functionality. Therefore, it is becoming more important to consider business rules explicitly and decoupled from the core applications. Moreover, besides writing separate and explicit rules, it is desired to do so using an expressive lightweight language: full-fledged rule-based languages imply a technological overhead and require the business analysts to have low-level programming language skills to write the rules, which may result overkill for some applications. Other approaches suggest the use of object-oriented patterns [5], which are not a declarative format and thus not very suitable for the expression of the rules.
Moreover, we observe that even though some approaches allow rules to be defined separately at the implementation level, they fail to separate and encapsulate their connection, i.e. the code that links the business rules to the core application. This code still results crosscutting in the core application, as identified in previous work [6, 7].
A third problem appears as a consequence of evolution: domains evolve, new rules are defined that talk about domain concepts that were unforeseen in the current application. It is desired to allow the definition of new domain vocabulary and to transparently adapt the current implementation in order to realize it.

2. Our approach

In order to overcome the identified problems, we propose the definition of a high-level lightweight rule language. This would allow us to express the rules and the details of their connection with the core application in a language that is closer to the domain, independent of implementation details. Our approach proposes: a) the definition of a domain model b) the automatic translation of the domain model to the implementation.

2.1. Definition of a Domain Model

We propose a domain model that consists of three components: the domain entities, the high level rules and the specification of the connection of the rules with the core application. The proposed model is a) high-level and declarative; b) loosely coupled from implementation.
The domain entities represent the vocabulary of the domain of interest. They constitute the building blocks used in the definition of the high-level rules and their connection. Considering an application already developed using an object-oriented (OO) approach, the domain entities are used for: a) extracting the domain knowledge present in the current implementation; b) defining new domain vocabulary which is unanticipated in the current implementation.
The high-level rules express relations between domain entities in the domain model. They have the form of an *if condition then action* statement, meaning that the condition has to evaluate to true in order the action to be performed. They are defined in terms of domain entities and thus are independent of implementation details.

The connection of the rules specifies how the rules need to be integrated with the core application. The connection typically denotes an event which defines when the application of the rule needs to be triggered. In case the application of the rule requires information unavailable at that event, the connection also specifies how to capture it and make it available to the rule.
The high-level nature of this domain model allows the reusability of the business logic among different applications on the same domain or among different versions of an evolving application.

2.2. Realization of the Domain Model: suitability of AOP

The ultimate goal is the automatic translation from the domain model to the implementation. This translation has two characteristics: (1) it is transparent for the business analysts who define the domain vocabulary and the business rules (2) it happens non-invasively for the core application where the rules are to be integrated.
We analyze how each component of the domain model can be mapped to the implementation. Consider the core application developed in OO. Given a *domain entity* that represents a domain term, two situations can occur: the domain entity is anticipated in the core OO software application, in which case it is mapped to the existing OO entities that implement it; for instance, the domain entity "customer" maps to the OO class "Customer" present in the current implementation. The other possibility is that the domain entity is unanticipated in the current implementation, and thus adaptations or extensions to the core application might be needed. For instance, a domain term that refers to the "average amount spent by a customer in the last month" might not be anticipated in the current implementation. Moreover, the implementation of such a domain term might even result crosscutting in the core application. We want to non-invasively adapt the core application, since it is not desired to invasively modify the implementation each time the domain vocabulary changes. Thus AOP seems ideal in this regard and its use is being explored.
Business rules are implemented using OO entities. Each rule is represented by a class that defines two methods, for the condition and action of the rule.
As the *connection* of the rules crosscuts the core application [8], Aspect-Oriented Programming (AOP) is suitable for its implementation, as identified in previous work [6, 7, 9]. The specification of the connection of the rules involves different parts, namely the specification of the application time, the information needed for its application, how to combine the rules that are applied at the same time, etc. Thus the translation of the connection to implementation might result in a different AOP pattern solution, as the ones identified in [8].
To conclude, we are working on the definition of this domain model and its automatic translation to implementation. In doing so we are analyzing the usefulness of AOP, in particular for: obliviously achieving the implementation of unanticipated domain entities; non-invasively implement the connection of the rules with the core application.

3. References

[1] Date C.: What not How: The Business Rules Approach to Application Development. Addison-Wesley Publishing Company (2000)
[2] Ross R. G.: Principles of the Business Rule Approach. Addison-Wesley (2003)
[3] Von Halle B.: Business Rules Applied. Wiley (2001)
[4] The Business Rules Group. Defining Business Rules: What Are They Really? http://www.businessrulesgroup.org/, July 2000.
[5] Arsanjani A.: Rule object 2001: A Pattern Language for Adaptive and Scalable Business Rule Construction (2001)
[6] Cibrán M. A., D'Hondt M., Jonckers V.: Aspect-Oriented Programming for Connecting Business Rules. In Proceedings BIS, Colorado Springs, USA (2003)
[7] Cibrán M. A., D'Hondt M., Suvée D., Vanderperren W., Jonckers V.: JAsCo for Linking Business Rules to Object-Oriented Software. In Proceedings CSITeA, Rio de Janeiro, Brazil (2003)
[8] Cibrán M. A., "Using aspect-oriented programming for connecting and configuring decoupled business rules in object-oriented applications", Master Thesis, Vrije Universiteit Brussel, Belgium, 2002.
[9] Cibrán M. A., Suvée D., D'Hondt M., Vanderperren W. and Jonckers V., "Integrating Rules with Object-Oriented Software Applications using Aspect-Oriented Programming", Proceedings of ASSE'04, Argentine Conference on Computer Science anbd Operational Research, Córdoba, Argentina, September 2004.