Workshop Report — ECOOP'98 Workshop 7 Tools and Environments for Business Rules

Kim Mens[1], Roel Wuyts[1], Dirk Bontridder[2], and Alain Grijseels[2]

[1] Vrije Universiteit Brussel, Programming Technology Lab
Pleinlaan 2, B-1050 Brussels, Belgium
kimmens@vub.ac.be rwuyts@vub.ac.be
http://progwww.vub.ac.be/
[2] Wang Global Belgium,
System Integration & Services Division, Application Engineering
Madouplein 1 box 8, B-1210 Brussels, Belgium
dirk.bontridder@wang.com alain.grijseels@wang.com

Abstract. This workshop focussed on the requirements for tools and environments that support business rules in an object-oriented setting and attempted to provide an overview of possible techniques and tools for the handling, definition and checking of these rules and the constraints expressed by them during analysis, design and development of object-oriented software.

1 Workshop Goal

Business rules are nothing new. They are used by every organisation to state their practices and policies. Every business application contains many business rules. One of the current problems with business rules is that code, analysis and design models specify business rules only implicitly and that current software engineering tools provide inadequate support to explicitly and automatically deal with business rules when building object-oriented business applications. With this workshop we intended to investigate which tool and environmental support for handling business rules during software development and evolution is needed and/or desired and which is already available. To come to a categorisation of available support tools and techniques, position papers were solicited from both academia and industry. In our call for contributions, we asked the participants to focus on the following topics:

1. What tools and environments currently exist to handle business rules?
2. Which extra support is needed or desired?
3. How can business rules be made explicit during the different phases of the software life cycle?
4. How to specify business rules and the constraints they express in a rigorous way?
5. How can we deal with enforcement of business rules, compliance checking, reuse and evolution of business rules?

6. How can code be generated automatically from business rules or how can business rules be extracted from existing software systems?

We agree with Margaret Thorpe that tool support for business rules is important: "Giving the business community the tools to quickly define, modify and check business rules for consistency as well as get those changes quickly implemented in the production environment has made them able to respond much more quickly to changes in the business environment." [6]

2 About the Organisers

We were interested in organising this workshop for several reasons:

- Kim Mens' [4] current research interests go out to intentional annotations and the semantics thereof. He thinks that business rules could provide some insights in this matter: what extra "intentional" information do business rules provide and how?
- Roel Wuyts' research focusses on the use of declarative systems to reason about the structure of object-oriented systems [10]. He is particularly interested in business rules that can be expressed declaratively and then used to extract or view particular information from an object-oriented system.
- Dirk Bontridder and Alain Grijseels wanted to validate their current experiences on business rules in object-oriented framework-based software development projects with the insights of other people working on or with business rules.

3 About the Workshop

About twenty persons actually participated in the workshop, with an equal amount of participants from industry and the academic world. Eight of them were invited to present a position statement during the first part of the workshop. We categorised the different presentations based on the topics on which they focussed. Bruno Jouhier [2], Paul Mallens and Leo Hermans [1] reported on existing *tools* and environments for dealing with business rules, and on the qualities and shortcomings of these tools and the applied techniques. Michel Tilman [7], Hei Chia Wang [9] and Stefan Van Baelen [8] discussed some *techniques* for dealing with business rules (based on their experience or inspired by their research interests). Gerhard Knolmayer [3] and Brian Spencer [5] *related business rules to data base rules.* These presentations certainly gave an inside in some of the topics enumerated in section 1. For more details we refer to the position statements which are included at the end of this chapter.

Because we wanted to come to understand what characteristics and properties tools and environments for handling business rules should have, we took the following approach in the remainder of the workshop. The goal was to obtain a list of requirements for tools and environments. To this extent, we assigned

the participants to different working groups, composed of both industrial participants and researchers, to construct such lists from different perspectives. The perspectives adopted by the different working groups were:

Function of the person. Different kinds of persons may have a different perspective on the kinds of requirements that are essential for tools and environments for business rules. We asked the members of this working group to assume the role of a particular kind of person (e.g. project manager, problem domain expert, application developer, end user, business manager, ...), and to reason about requirements from that perspective.

Nature of the application. We assumed that the particular nature of an application might have an impact on the requirements for tools and environments for business rules. Therefore, we asked the different members of this working group to reason about such requirements from the perspective of particular kinds of applications (e.g. administrative, financial, telecom, ...)

Software life cycle. This workshop group focussed on finding the requirements for tools and environments to support business rules throughout the entire software life cycle.

In a concluding session, the results of the different working groups were merged and discussed with the other working groups.

4 Requirements for Tools and Environments for Business Rules

4.1 Initial List of Requirements

This initial list of requirements for tools and environments was intended to serve as the basic input for discussion in the different working groups. It was extracted from the position papers submitted by the participants. First of all we wanted to know whether this list was complete or not (if not, participants were encouraged to add to this list). Secondly, we were interested in the participants' motivations why (or why not) the listed requirements were deemed necessary in tools and environments supporting business rules. The initial list of tentative requirements is given below:

(Centralised?) repository: There should be a (centralised?) repository of business rules.

Adaptable: Allow for easy changing, refining and removing of existing rules.

Conflict detection: Support for detecting conflicting rules is needed.

Debugging facilities: Provide support for debugging systems containing lots of business rules.

Declarative language: Use a declarative language to express business rules.

Dedicated browsers: Use dedicated browsers for "querying" business rules.

Efficiency: Achieve an "acceptable" efficiency in tools and environments for business rules.

Explicit business rules: Make business rules explicit in the software.
First class business rules: Business rules should be first class.
Formal and rigorous foundation: Need for formal and rigorous foundation of business rules.
Identification and extraction: Support for identification and extraction of business rules from the real world.
Maintain integrity and consistency: Support for maintaining software integrity and consistency.
Representation: Use of metaphors of business rule representation that or more interesting than "if-then-else" constructs.
Open: Easily allow new rules as well as rules about new items.
Reasoning engine (or mechanism): Use a reasoning engine to allow inferencing about rules (rather than having "stand-alone" rules).
Scope/views of rules: Support different scopes or views of business rules (e.g. application dependent as well as independent).
Integration with existing tools: Provide a symbiosis between business rule tools and environments and integrated development environments for managing the rest of the software.

Every working group took these initial requirements and separately discussed them according to their perspective. This resulted in an annotated requirements list, containing extra comments or considerations made by some groups according to their viewpoint. The working groups also added new requirements they deemed important from their perspective. The next two subsections present the annotated requirement list everybody agreed on and a list of added requirements.

4.2 Annotated Requirement List

(Centralised?) repository: Every group agreed on this, without much discussion. The "Function of the Person" working group (FoP) stated that the repository should not necessarily be physically centralised, but certainly virtually. The other two groups explicitly mentioned that the repository should contain all business rules, about software components at any phase of the software life cycle.
Adaptable: Everybody agreed on this obvious requirement.
Conflict detection: The "Software Life Cycle" working group (SLC) argued that conflict detection is important, but further investigation should make clear what kinds of conflicts are interesting or important to be detected. The FoP group mentioned that they currently see two different levels where rules can conflict: at the business level or at the technological level.
Debugging facilities: Everybody agreed that there certainly is need for debugging support, for example for prototyping business rules, or for tracing facilities.
Declarative language: Was considered important by all working groups.
Dedicated browsers: Dedicated browsers for several types of users should be available. The FoP group related this to the scope/view requirement: browsers should support different scopes or views of business rules as well.

Efficiency: The FoP group mentioned that efficiency should at least be "reasonable", but more importantly, things should remain efficient when scaling the system. The SLC group recognised that there is a subtle trade-off between efficiency versus flexibility (e.g. adaptability). Building an application by generating code for the business rules could make it more efficient, but limits its flexibility. On the other hand, an application that accesses and uses the rule-base at run-time is very flexible (open, adaptable, ...) but is probably much less efficient.

Explicit business rules: Everyone agreed.

First class business rules: Everyone agreed.

Formal and rigorous foundation: Two important remarks were made here. First, the FoP group mentioned that maybe rules could be informal in an initial phase during requirements elicitation, but in the end they should be declared in an explicit, formal and declarative way. The "Nature of the Application" working group (NoA) noted that it is important to have a formal notation language, but preferably, it should be a standard one. A number of potential candidates are: KIF, OCL, UML, ...

Identification and extraction: According to the NoA group, there is a need for knowledge elicitation tools to extract business rules from human sources, examples (cases) and electronic sources (reverse engineering). The FoP group agreed that fully automated extraction of business rules is a beautiful goal, but seems unrealistic.

Maintain integrity and consistency: To achieve integration and consistency, only the proper tools should have access to the rule base. It is not allowed to access the rule base from the outside. A question is what kinds of tools and techniques are currently available to support consistency maintenance?

Representation: Appropriate representations should be supplied when accessing the rule base (e.g., different representations for different users). These representations should not necessarily be completely formal, but should best suit specific users (FoP). Possible alternative representations could be tables, trees or graphs (NoA). One way of allowing different representation schemes could be to use a single language for internal representation of and reasoning about business rules, but many different external languages (SLC).

Open: Everyone agreed.

Reasoning engine (or mechanism): It should be investigated which kind of reasoning mechanism is most appropriate to deal with business rules. (Inferencing, constraint solving, event handling, forward or backward chaining, etc...).

Scope/views of rules: The FoP group stated that mechanisms are needed to classify the business rules according to different views. This will facilitate in browsing the rule base and finding particular rules in the rule base. The NoA group elaborated further on this by proposing a notion of contexts or scopes that should allow the classification of business rules in conceptual groups.

Integration with existing tools: All tools should be integrated, and consistent at all times. For example, when changing a view in some tool, it should be changed automatically in the other tools. It is also important to integrate

the tools and environments for business rules with existing object-oriented methods, techniques, and notations.

4.3 Additional Requirements

Some of the working groups proposed some additional requirements to be added to the initial list. For example, the FoP group was able to formulate some extra requirements by looking at the requirements from a managerial perspective.

The list with all extra requirements formulated by the different working groups is presented below:

Life cycle support: The SLC group claimed that support for business rules is needed throughout the entire software life-cycle. The other working groups agreed on this.

Management decision support : The FoP group mentioned the need for support for workload assignment, progress management, and decision management.

Traceability : There is a need for traceability between a business rule and the source from which it was extracted, at different phases of the software life cycle. Furthermore, traceability is not only important within a single phase of the software life-cycle, but also throughout the different phases. Traceability is important because it facilitates reasoning about the business application.

Code generation : The application developer could be supported by generating code, templates or components from the business rules. But although generating code from business rules seems an interesting issue, some questions immediately spring to mind: when should code be generated (only at end?), and what should be generated?

Team development : Tools should provide team development support such as system configuration management, multi-user support etc. This additional requirement was mentioned by several working groups.

Evolution : Support for tracking the evolution of business rules, in and throughout different phases of the life cycle. (SLC group)

Completeness : Being able to check completeness, i.e. is the business described completely by the business rules, seems like an interesting requirement but might not always be feasible (e.g. how to check completeness?), wanted (e.g. not important in an initial phase) or relevant.

Regression testing : How to build tools for regression testing in the context of business rules?

Impact analysis : Techniques are needed for analysing the impact of changing a business rule on the rest of the system, again at all phases of the software life cycle. (SLC group) Note that this requirement is somewhat related to the requirements of evolution and conflict checking.

4.4 Further Remarks

A conclusion of the working group NoA group was that the requirements of business rule tools and environments seem rather independent of the application domain. However, there was some disagreement with this conclusion by the

other working groups. For example, they mentioned the example of real-time applications which seem likely to give rise to more specific requirements.

The FoP group not only formulated extra requirements by reasoning (for example) from a managerial perspective but also remarked that new jobs (such as an auditing or rule manager) may need to be defined when adopting a business rules.

The SLC group effectively identified several additional requirements by focussing on the use of tools for business rules throughout the software life cycle: support for traceability, evolution, conflict checking, impact analysis, ... not only at a single phase of the software life cycle, but also between different phases in the life cycle.

There was some unresolved discussion about the internal language that should be used for representing business rules. One viewpoint was that a standard language or notation (such as UML) should be used in which it is possible to declare as many (kinds of) business rules as possible. The opponents of this approach preferred the complete openness of a meta-approach.

5 Conclusion

During the workshop, there seemed to be a lot of agreement regarding the constructed list of requirements for business rule tools and environments. This is a hopeful sign indicating that there is a clear feeling of what characteristics such tools and environments should have, despite of the fact that there still is no precise and generally accepted definition of what a business rule is.

6 Acknowledgements

Thanks to all workshop participants for making this a great and successful workshop.

References

1. Hermans, L., van Stokkum, W.: How business rules should be modeled and implemented in OO. Position paper at the ECOOP'98 Workshop on Tools and Environments for Business Rules. Published in this workshop reader (same chapter).
2. Jouhier, B., Serrano-Morale, C., Kintzer, E.: Elements Advisor by Neuron Data. Position paper at the ECOOP'98 Workshop on Tools and Environments for Business Rules. Published in this workshop reader (same chapter).
3. Knolmayer, G. F.: Business Rules Layers Between Process and Workflow Modeling: An Object-Oriented Perspective. Position paper at the ECOOP'98 Workshop on Tools and Environments for Business Rules. Published in this workshop reader (same chapter).
4. Mens, K.: Towards an Explicit Intentional Semantics for Evolving Software. Research abstract submitted to the ASE'98 Doctoral Symposium. To be published in the Proceedings of Automated Software Engineering 1998 (ASE'98).

5. Spencer, B.: Business Rules vs. Database Rules — A Position Statement. Position paper at the ECOOP'98 Workshop on Tools and Environments for Business Rules. Published in this workshop reader (same chapter).
6. Gottesdiener, E.: Business Rules show Power, Promise. Cover story on software engineering, Application Development Trends (ADT Magazine), March 1997.
7. Tilman, M.: A Reflective Environment for Configurable Business Rules and Tools. Position paper at the ECOOP'98 Workshop on Tools and Environments for Business Rules. Published in this workshop reader (same chapter).
8. Van Baelen, S.: Enriching Constraints and Business Rules in Object Oriented Analysis Models with Trigger Specifications. Position paper at the ECOOP'98 Workshop on Tools and Environments for Business Rules. Published in this workshop reader (same chapter).
9. Wang, H.-C., Karakostas, V.: Business-Object Semantics Communication Model in Distributed Environment. Position paper at the ECOOP'98 Workshop on Tools and Environments for Business Rules. Published in this workshop reader (same chapter).
10. Wuyts, R.: Declarative Reasoning about the Structure of Object-Oriented Systems". Proceedings of Technology of Object-Oriented Languages and Systems (TOOLS'98), 1998, pp. 112-124.