

SOFTWARE ADAPTIVITY THROUGH XML-BASED BUSINESS RULES AND AGENTS

L Xiao and D Greer

School of Computer Science, Queen's University of Belfast, Belfast

Key words to describe this work: software adaptivity, executable requirements, agent, business rule, UML, XML

Key Results: The combination of agent and business rule provides adaptivity in software systems. Business rules capture requirements and are made executable by agents. Through three developed editors, rules can be changed on demand very easily. Up-to-date requirements are implemented immediately by agents, without the need for frequent system rebuilds.

How does the work advance the state-of-the-art? The work advances knowledge and practice on agents for adaptive systems. By making requirements executable as rules, a shortcut development process that goes directly from requirements to deployed system is demonstrated as possible. Maintenance becomes the process where business experts change text-based rules (low risk) as opposed to code change by developers (high risk).

Motivation (Problems addressed): Software requirements are very difficult and expensive to adapt after the software has been developed, chiefly because they are embedded in code and new requirements are difficult to determine/predict.

Introduction

It is a long established fact that maintenance and evolution accounts for the majority of software lifecycle costs [1]. To reduce costs much effort has been made to make systems more adaptive, where systems change their behaviour according to their context [2]. We propose an approach that makes changes to requirements reflected in deployed software immediately, at run-time and on demand, according to business people's configuration.

Prior Approach to Adaptivity

One influence to this research is from the use of the 'TypeObject', 'Property' and 'Strategy' patterns to achieve the Adaptive Object Model (AOM) [3]. In the AOM framework, business units are modelled with metadata, later interpreted at run-time. Thus, unpredicted classes can be created at run-time from generic ones, and dynamic behaviours can be configured for classes. Further, behaviours can be adapted if they can be predicted. However, the AOM approach has several weaknesses. Firstly, since classes are created dynamically in the AOM, if they are to be persistent, their definition must be stored in a database. The hard-coded original classes therefore do not represent business abstractions because most information about the business is in the database. Hence, developers may find the architectures required in this approach hard to understand and maintain. Secondly, AOM can only customize behaviour within the limits of what has been predicted. Thirdly, there is no easily accessible central model, so that analysts and clients

alike may find it difficult to visualise the current system.

Solution Overview

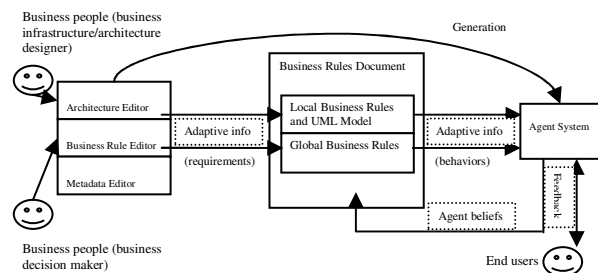


Figure 1: Solution Overview

We propose an agent-oriented approach. Unlike objects, agents are active. Instead of using static methods which are to be invoked and have the same effects all the time, agents are granted the flexibility to choose how to react. In different contexts, they will have different behaviours according to pre-defined and easy to change rules, thus achieving easier adaptivity. A business rule is a compact statement about some aspect of a business. It is a constraint in the sense that a rule lays down what must or must not be the case [4]. Business rules are the ideal companion with agents as they capture requirements and can be used to govern agent behaviours.

Our approach is to generate agent systems and let them execute business rules specified in editors by business people. Code is not changed. Rather, various editors are used to easily change the rules which govern the agent behaviours. New Behaviours take effect immediately by the running agent system. Ultimately, agents will be self-learning, self-reasoning, and self-adaptive. Their knowledge is not only from business

world's demand, but also from their beliefs from the actual execution environment. The approach is summarised in Figure 1.

Business Rules Editor

A typical global business rule generated with the Business Rules Editor is shown in Figure 2.

```

- <global-rule>
  <id>1</id>
  <condition>
    customer.type = premium
  </condition>
  <action>
    customer.discount := 0.05
  </action>
  <priority>1</priority>
</global-rule>

```

Figure 2: Sample Business Rule in XML

A Business Rules Manager Agent (BRMA) is used to govern the application of rules that represent global business policies. These rules are deployed by agents as executable requirements and take effect immediately because agents retrieve, translate and execute them at run-time. Global rules evolve as business policies change.

Metadata Editor

New business terms can be registered through the Metadata Editor and stored in XML format. E.g. "credit" as a new attribute of "customer" is registered in the editor and stored as in Figure 3.

```

- <object>
  <name>customer</name>
  <attributes>
    <attribute>name</attribute>
    <attribute>type</attribute>
    <!-- more attributes -->
    <attribute>credit</attribute>
  </attributes>
  <behaviour/>
</object>

```

Figure 3: Updated Metadata

After this, new global business rules on new business terms can be defined through the Business Rules Editor, as shown in Figure 4. Without rebuilding the system, agents can start to 'speak' these new vocabularies immediately during their sending and receiving messages and this can reflect new business concepts/policies.

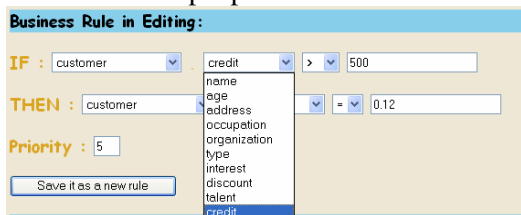


Figure 4: Business Rule Editor using a New Business Concept

Architecture Editor

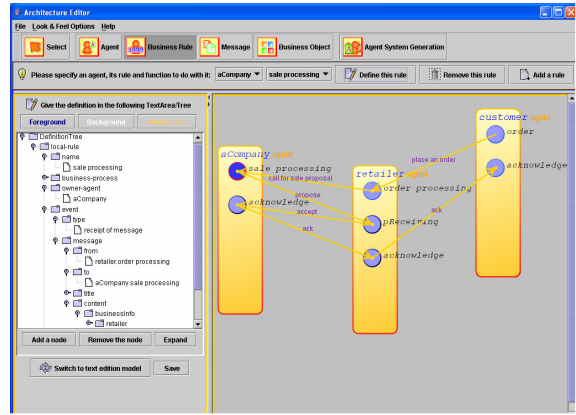


Figure 5: Architecture Editor

Local rules guide behaviours of individual agents and their collaboration in various conditions, and so describe business processes. These rules are defined through the Architecture Editor, as demonstrated in Figure 5. They have the structure {event, condition, action, belief}. Specification of them through the editor allows adjustment of agent communication structure dynamically. Thus agents in the system can have their partners changed in order to accomplish various business tasks, according to business people's specification. Like the global rules, agents will not execute the local rules until it is time for the agent to participate in a certain business process. Hence, new requirements on the system architecture and tasks to be done at each system unit can be configured easily and deployed without the delay of a rebuild.

Conclusion

Agents can be used in combination with business rules to achieve adaptivity. Three editors have been developed to capture business knowledge at three levels: business policy, business concept and business process. Agent systems running on Jade platform [5] can be generated from the Architecture Editor. The agent behaviours are governed by business rules which adapt according to the editors so that the generated agent systems are adaptive.

References

- [1] Manna, M., "Maintenance Burden Begging for a Remedy", *Datamation*, pp53-63, April, 1993.
- [2] Lieberherr, K, "Workshop on Adaptable and Adaptive Software", Proc. 10th Conference on Object Oriented Programming Systems Languages and Applications proceedings, pp149-154, 1995.
- [3] Yoder, J.W., Balaguer, F., Johnson, R., "Architecture and Design of Adaptive Object-Models", *ACM Sigplan Notices*, 36(12), pp50-60, 2001.
- [4] Morgan, T., "Business Rules and Information Systems", Addison-Wesley, 2002.
- [5] JADE platform, <http://sharon.csel.it/projects/jade/>.