Using rules to adapt applications for business models with high evolutionary rates

Juan Fuente, A. A.[1], López Pérez, B.[1], Infante Hernández, G.[2], Cases Fernández, L. J.[2]
[1]*Computer Science Department, University of Oviedo, Asturias, Spain*
[2]*Laboratory of Software Architecture, University of Oviedo, Asturias, Spain*

***Abstract* — Nowadays, business models are in permanent evolution since the requirements belongs to a rapidly evolving world. In a context where communications all around the world travel so fast the business models need to be adapted permanently to the information the managers receive. In such world, traditional software development, needed for adapting software to changes, do not work properly since business changes need to be in exploitation in shorter times. In that situation, it is needed to go quicker from the business idea to the exploitation environment. This issue can be solved accelerating the development speed: from the expert to the customer, with no –or few, technical intervention. This paper proposes an approach to empower domain experts in developing adaptability solutions by using automated sets of production rules in a friendly way. Furthermore, a use case that implements this kind of development was used in a real problem prototype.**

***Keywords* — business rules, domain experts, software adaptability, software architecture**

I. INTRODUCTION

BUSINESS environments and business needs are changing rapidly, thus a progressive change and adaptation of the systems development is unavoidable in order to maintain the customer satisfaction. Even though, it is an expensive and difficult task for software engineers and developers to align the changing business requirements with actual software systems to keep them working properly [1]. Software adaptability must therefore be taken into account throughout the full software life cycle. Systems adaptation may be undertaken using two different levels in most of the cases: simple adaptations usually performed by using configuration files and complex adaptations where solutions are commonly structural ones. This paper focuses on the latter type of system adaptation, the complex or logical systems. These systems can be modified by using rules that solves first order logical issues over the predicates in order to assist decision making process [2].

The difficulty in ensuring systems adaptability is highly related with the software development lifecycle. Usually, human knowledge is transformed into software systems by the mediation of requirements documents and design models. These documents and models provide a high level view of the system and guide developers in producing running systems from the specification. Even though, the original requirements textual descriptions of system functions are separated from the developed design models, which lack the capability to capture the exact behavioral semantics from what is stated in the functional requirements [3].

Hence, these behavioral semantics need to be expressed in a more flexible and abstract manner to avoid coupling with actual developed systems and at the same time ease the adaptability process. The domain expert's role in defining the behavior of the systems expressed in comprehensible business rules is then a matter to take into account since their business knowledge can be transformed in adaptability solutions. There are a number of proposals that include tools with interfaces for non-skilled users, more than personalization, they either allow rapid development of prototypes [4], [5] or provide for the visual expression of simple rules, which, although powerful enough in certain cases, is somewhat limited in the application domain. Therefore, it would be beneficial to explicitly involve the final users and allow them to provide part of the desired configuration for system adaptation, since they will be familiar with their own environment and their requirements.

II. BACKGROUND

There are some methods used usually to adapt applications to existent business models. One of such methods include the Model-Driven Architecture (MDA) [3], [6] and [7] which promotes the production of business models with sufficient detail so that they can be used to generate or be transformed into executable software, running on target systems [8].

MDA proposes a Platform-Independent Model (PIM), a highly abstracted model, independent of any implementation technology. This is translated to one or more Platform-specific Models (PSM). The translation is based on a particular technological implementation including specific constructs and features of the implementation [9]. PSM is translated into code in a similar pattern.

The transformation process of PIM to PSM and finally code starts from the design products rather than requirements models. Hence, it requires highly creative work [6] to build a PIM from narrative requirements documents. This results in high costs in requirements change because of the need of

DOI: 10.9781/ijimai.2013.227

skilled software engineers. Furthermore, as stated in [13], UML alone is not able to capture some semantics in its diagrams and a combination of UML and OCL [7] is used in MDA. However, OCL constraints are static and used in the design stages rather than the requirements stages. Moreover, MDA relies heavily on the tools which are supposed to have strong transformation capabilities from PIM to PSM and then to code.

MDA can reproduce object oriented OO systems despite the intrinsic static nature of object structure and behavior, code being regenerated from models. However, changes cannot be made to systems at runtime without interruption. Another important issue is that some business representation cannot be directly formed as objects, such as business rules. Additional maintenance problems would be otherwise added to systems if business rules were hard-coded [10]. These weaknesses MDA have led to the exploration of an alternative component technology at a higher level abstraction, being capable to retrieve, understand, as well as interpret business knowledge directly and dynamically.

There are a number of different technologies that may be used to express this sort of information. Almost any language that supports some form of rule-based inference can be used; this includes rule engines such as Drools [11], Jena [12] and Jess [13]. The Java specification request JSR-94 [14] covers the definition of a Java rule engine API, and most commercial rule engines are implementations of this standard. Drools, is an open source business rule management system and inference rule engine implemented in Java [11]. Inference rules are evaluated using an enhanced implementation of the Rete algorithm [15]. Drools natively provides an expressive textual language for defining inference rules, but also supports the integration of a custom rule DSL to improve the productivity of defining rules within certain domains. The underlying model that Drools operates within is simple plain old java objects (POJOs), making it easy to integrate into an existing Java-based software system. The structure of inserted POJOs does not need to be defined as part of the rule base; this means that all metamodel properties and operations are always accessible to a Drools rule. These are the main reasons why to choose Drools in this approach in order to build and execute the rule sets.

III. Related Work

There are some significant studies aimed at providing a mechanism for not skilled users to specify the rules needed by the system in order to be better adapted to their needs. Authors in [4] present an application prototyping tool which does not require coding and instead uses a graphical interface based on controls, which allows context and devices to be collected and rules to be constructed from them by taking only logical–relational operators and restrictions on types of complex conditions. The technique might not be considered suitable for domain users to modify the applications since is actually a prototyping support intended for developers.

In [16] the authors present a programming prototyping environment intended for domain users. The system provides a series of data flows from different inputs where users can select the input flows required for the behavior they want to express. It also specifies the actions to be executed. This proposal makes use of machine learning algorithms to interpret the annotated flows in order to determine the user's intention. Since domain users are familiar with their own activities and environments they are able to tell the environment how it should behave, but they might have

The work described in [17] sketches a visual interface that specifically targets non-expert users based on a drag-and-drop metaphor. It relies on a rule grammar for expressing conditions and rule alternatives. This tool is intended to be implemented in future with an emphasis on providing visual hints and suggestions to facilitate incremental rule construction by end-users, but has not yet been tested with real non-skilled users.

Although these systems are intended to be used by domain experts, they fail in the way to represent the information in a comprehensible way for not skilled domain experts. Some of them use programming languages or domain specific languages that are more suitable for developers in order to express the adaptation rules. The solution proposed in this paper can be more suitable for domain experts since the representation of the rules is done graphically with no special knowledge of the technology used. Moreover, the rules predicates are expressed with a very simple way close to natural language, avoiding complex logical structures.

IV. Materials And Methods

A target application developed to be in permanent adaptation by the use of rules needs a previous architecture design where the modules that will be affected by the adaptation process along with invariant ones need to be defined. Moreover, these rules need to be edited and manipulated by domain experts instead of undertake the development of new features by software engineers.

In order to develop such applications using rules, an engineering model able to support integration [18] have been used. The method is composed by the following steps:

Problem statement

The study of the business issues and those exposed to high evolutionary rates is addressed in this step. This step represents an important task within the whole process since the accuracy in identifying these issues will impact future developments or avoid them if possible, saving money and gaining efficiency in a long term.

Domain experts' knowledge represents a valuable source of information in this step. This information describes the general business features and the most frequently scenarios to take into account by software engineers. The business knowledge is taken by software engineers to identify the main core of the system (which is more immutable and thus less subject to change), along with the more dynamic elements that may vary the most in exploitation time.

From the identified elements, those which bring the possibility to be adapted with simpler techniques (e.g. configuration files) are separated. The remaining dynamic elements are classified whether they are sensitive to be adapted with rule-based systems or require new developments.

The final number of dynamic elements sensitive to be adapted by rules, resultant from this classification is big enough to justify the use of rule-based systems.

The elements that require new development in order to be adapted are studied apart. These elements must be designed with architectural patterns that minimize the interdependencies among them and simplify the systems evolution.

Architecture design

This step follows the attribute driven design (ADD) used in [18]. The most dynamic modules are incorporated with a "modifiability" quality attribute and quality scenarios are designed to check this feature. Each of the modules that are adapted by rules receives the classification of rule-based architectural style modules within the architecture context [2].

Rule-based system design

This step describes the design of an architecture module that will be adapted by rules. The rules to be applied along with their attributes and predicates are identified. The component that enables domain experts to interact with the rules' management in an intuitive way is also designed. There are four basic elements needed to address systems adaptability with rule-based systems:

Attributes. Bring the possibility to query any object feature in the system.

Predicates. Code elements that perform complex queries to system elements, evaluates them and return a value that can be processed by rules.

Actions. Situated in the rules consequent, they have effects over the system since they can modify its behavior, reason why it is important not to extra limit their scope. To properly establish actions scope, a set of services are defined (i.e. façade, web services, etc.) this way actions can only trigger these services and do not affect other parts of the system.

Rules. These are the most dynamic aspect of the system. They are intended to be dynamically inserted in the system. Domain experts use the set of rules to design new actions to be executed by the system.

In order to ease domain expert in designing new rules, a rule editor is constructed following the business vocabulary. It is only necessary that the expert have a little notion of logic to interact with the editor. Even though, the editor is intuitive enough to assist the expert in creating syntactically correct rules.

The rules' structural modifiability is restricted only to the set of attributes, predicates and possible actions. In the case that this modifiability has a broader scope, then its structural significance gets bigger than the logical adaptation and the module can be classified as complex system where the adaptations which solutions are commonly structural ones, therefore, they out of this paper scope. In order to solve future structural issues, the intervention of technological teams aided by domain experts is necessary. This roles' combination enables the construction of modules (which insertion in the system is previously set) typed as: new attributes, new rules and new actions. All modules must be created with the same previously defined constraints.

V. USE CASE

This work presents a design model to adapt a traceability management that is able to handle, in sufficiently short times, the alarms triggered by user actions that do not follow established procedures. The use case address the establishment of a traceability system integrated in the enterprise applications of a transportations company. This company has several quality procedures that require certain records in some specific moments over time within its activities. In particular, the activity studied is the personnel hiring for driver positions.

The process under investigation is composed by the activities performed by the actors involved in hiring a new driver. Every activity generates a corresponding record that is stored in a record control system (see Fig. 1)

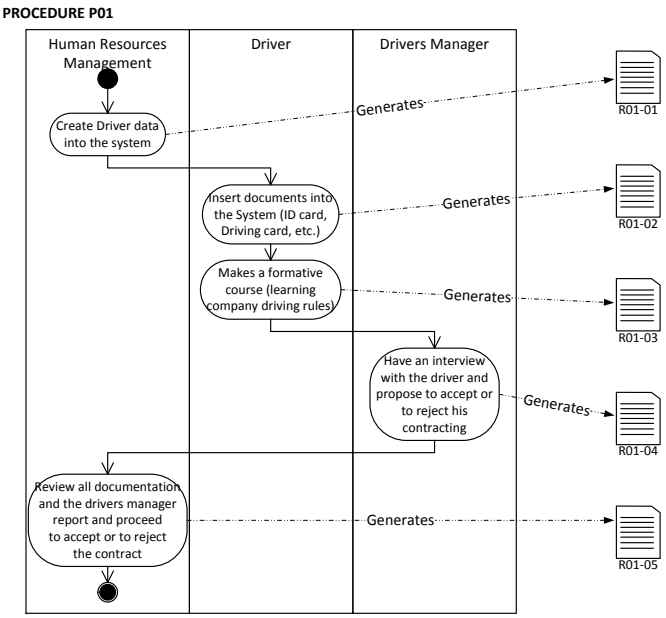

Fig. 1. Description of the procedure for contracting drivers

The company's product manager (PM) needs to know whether the procedures are performed properly or not and most important if the drivers' interviews are carried out. Furthermore, he/she requires that the information queries be recorded in order to later be aware of the time spent between the interviews and the actual hiring. It is worth to mention that these queries have a very short lifetime, e.g. the PM might need to have this data during a month with a special increment in accidents, while the next month this information is no longer required. For this reason, to undertake and ad-hoc development in order to achieve the adaptation of queries' information needs results expensive and repetitive. The solution proposed in this work has a reasonable low cost to make viable these kinds of adaptations.

This kind of queries can only be executed against the record database. This represents an issue for the PM since he/she needs this information available all the time without expressly perform the query. Hence, the system needs to be proactive and inform the PM in an autonomous way. This situation cannot be solved with a traditional development technique since every query needs to be adapted to fulfill PM information needs. This context fosters the implementation of rule-based architectural style for this module.

VI. Proposed Solution

There are several reasons to decouple traceability systems from business modeling systems, the following are some of the most significant:

- The procedures, in general, are items subject to changes which cannot be executed in monolithic systems that require new developments constantly.
- Actual legacy systems are not integrated with traceability systems. This situation hinders to perform changes in their behavior in order to avoid unordered activities.
- Highly dynamic systems may violate the strict path predetermined for the activities execution within a procedure.

The evolution of business models brings the adaptation to new standards which demand a change in the rules that handle the traceability of the products in a reasonable time. This scenario may not be suitable for traditional software development.

Once identified the main changing points, the system proposed was enriched with a rule-based adaptable module easy to modify by domain experts with no technical skills. This enables the performance of business adaptations in a very short period of time and with a very low cost. In order to achieve such adaptation, a graphical rule editor was implemented.

General system context

The proposed system is formed by three main components:

1. Legacy systems component (green) to model the business without traceability integrated.
2. Record systems component (blue) to interact with legacy systems in order to produce records at the right moment.
3. Information exploitation systems component (orange) to apply the rule-based architectural style in the architecture.

Fig 2. shows the general context of the proposed architecture.

The Information exploitation systems component retrieves the information from the records warehouse and processes it to obtain reports, indicators, etc. that enables the responsible actors to have data about the events and situations being gathered by the traceability system.

As can be shown in Fig. 2, the Information Exploitation Component has two basic modules:

- **Rules Manager**. Manages the rules that respond to the events and explores the available system information.

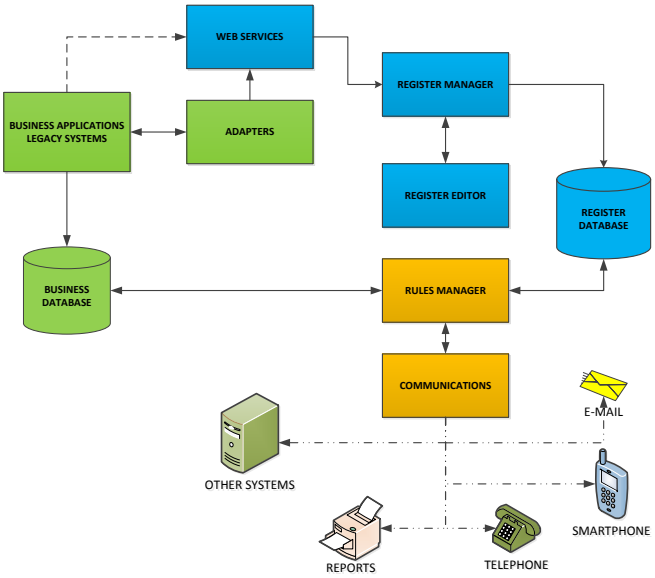

Fig. 2. Architecture context for the entire system

- **Communications.** Manages the users' information communication systems.

A more detailed description of the Information Exploitation Component is depicted in Fig. 3.

Basically, the process consists of the application of a set of rules that identifies different situations that need to be documented in some specific way. These rules are executed in two different ways:

- **On-line rules**. Executed in the precise moment of recording the registration in the database.
- **On demand rules**. Executed on demand by the users with privileges.

Once the rules are executed, if the conditions are met the associated actions are triggered.

The data mining system retrieves the business information needed to fire a rule or to complement the communication of an alarm action. There are four types of actions:

- **Just in time information (Alarms)**. Information that detects situations where some users wants to take just in time information from. These alarms can be configured by non-technical users.
- **Configurable reports**. Generic reports that can be configured to multiple purposes by non-technical personnel.
- **On demand reports**. Specialized reports that require technical personnel intervention.
- **On demand alarms**. Alarms version that cannot be handled by the rule configurable system and requires a technical service intervention.

On demand components perform solutions to more complex design that could not be designed at the development time.

Reports are sent to specific people or to another system by using different information channels; this requires the information adaptation depending on the channel used to send the data. This adaptation task is performed by the Communications Component.

Rules

The rules used follow the predicates logic format. The

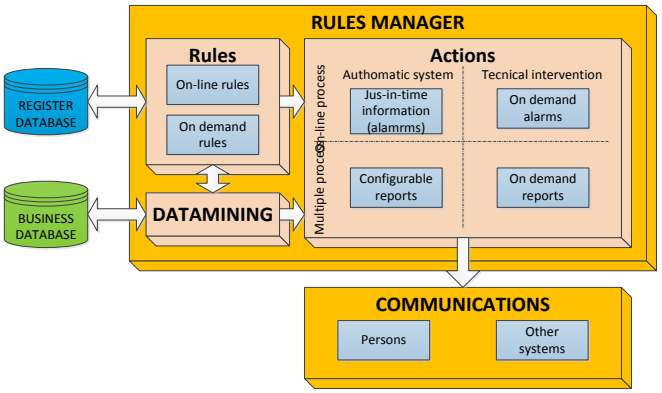

Fig. 3. Basic architecture for the Information Exploitation Component

quantifiers are eliminated since they can be included in the predicates. The rules are fired by triggered events; the basic functioning can be described as (1):

event: evaluation(rules_set) (1)

For each rule, if its evaluation is true, the associated actions are triggered as in (2):

IF Evaluation (Predicates) = TRUE →Execute (Actions) (2)

The general appearance of the rules used is the following (3):

P_1 (x,…) ∧P_2 (y,…)…→Action (Info, Stakeholders) (3)

Where *Info* represents the information required for the report, which will be composed by the *Action* itself along with the data mining component that seeks the information in the database. *Stakeholder* is the user that will receive this information. The parentheses are allowed in order to establish the priorities and associations when evaluating the predicates. The connectors are ∧: Connective "AND", ∨: Connective "OR", ¬: Denial and →: Implication.

As a further constraint every predicate must be an object that allows a Boolean evaluation as in (4):

Record_written_at_DB_event:
IsProcedure(P1)∧IsRegistry(R1.5)∧¬(ExistsRegistry(R1.3)∨ExistsRegistry(R1.4)→Alarm(Info,Stakeholders) (4)

Previous rule formulates the following predicate: IF the procedure is P1, the registration in course is registry R1.5 from the mentioned procedure and the registries R1.3 or el R1.4 does not exist then the alarm is raised.

Every time an alarm or report is created, rules that triggers actions related to them are generated. These rules are in a rule set that has different subsets which are evaluated against the rule engine depending on events detected. For example, there is a subset of temporal events that contains all the rules to be executed when a time-out is reached. When this event takes place, the rules related with it are evaluated. Another important subset is the one that is evaluated every time the system receives a new registry. The objective is to optimize system responses in order to enable the growing of rules number since not all of them are evaluated in every event.

The identified predicates list by default is depicted in Table II.

All the predicates should be in context. The possible contexts are:

- **Default Procedure**. It is the procedure that is generating

TABLE II
PREDICATES CLASSIFICATION

Predicate	Description
IsProcedure(string)	Returns TRUE if the procedure name **matches** with the given string.
IsRegistry(string)	Returns TRUE if the registry name **matches** with the given string.
ExistsRegistry(string)	Returns TRUE if the registry named by string **exists** in the execution of the procedure
RegistryNumberGreaterEqual(string, num)	Returns TRUE if during the execution of the current procedure, the registry name (string) has been repeated **equal or more** times than the number in num.
RegistryNumberGreater(string, num)	Returns TRUE if during the execution of the current procedure, the registry name (string) has been repeated **more** times than the number in num.
RegistryNumberEqual(string, num)	Returns TRUE if during the execution of the current procedure, the registry name (string) has been repeated **equal** times than the number in num.
RegistryNumberLess (string, num)	Returns TRUE if during the execution of the current procedure, the registry name (string) has been repeated **less** times than the number in num.
RegistryNumberLessEqual (string, num)	Returns TRUE if during the execution of the current procedure, the registry name (string) has been repeated **equal or less** times than the number in num.
TimeBetweenTwoRegistriesGreaterEqual (string1, string2, num)	Returns TRUE if the time consumed between two registries named as string1 and string2 is greater or **equal** to num in milliseconds.
TimeBetweenTwoRegistriesGreater (string1, string2, num)	Returns TRUE if the time consumed between string1 and string2 is **greater** than the num in milliseconds.
TimeBetweenTwoRegistriesEqual(string1, string2, num)	Returns TRUE if the time consumed between string1 and string2 is **equal** to the num in milliseconds.
TimeBetweenTwoRegistriesLess(string1, string2, num)	Returns TRUE if the time consumed between string1 and string2 is **less** than the num in milliseconds.
TimeBetweenTwoRegistriesLessEqual(string1, string2, num)	Returns TRUE if the time consumed between string1 and string2 is **equal or less** than the num in milliseconds.
LastRegistryTimeGreaterEqual(num)	Returns TRUE if the time consumed from the last registry of the studied procedure is **greater or equal** to num in milliseconds.

the current register to be stored. It refers to the procedure template.

- **Instance procedure**. It refers to the current execution of the procedure.

The predicates could be added to the system by means of new development processes, for improving the configuration possibilities.

Actions identification

Actions associated with system are basically those that generate reports. Generally, actions' information processing goes across three stages:

- Information recovery
- Information treatment
- Presentation to users

These stages can be executed on-line, just like alarms, or require elements to store the information temporally (recovered and/or treated) in order to compose the report when a period ends.

Alarms

Alarms are information from specific identified events usually on-line. When an event or an unexpected situation that has been programmed as an alarm appears, the traceability system sends the configured information to the defined users.

For example, when registering the information in the database, if a rule detects that some procedure step has been ignored and that step is important enough to inform some person then an alarm is programmed as in (5).

$$IF\ R1.3 \wedge \neg R1.2 \rightarrow Alarm\ (Info,\ list\ (Stakeholder)) \quad (5)$$

A more complex example can be the following: when R1.5 (last registry that closes the procedure) arrives and some of the previous registries are missing (i.e. R1.0, R1.1, R1.2, R1.3 or R1.4), then the corresponding alarm is raised as in (6).

$$IF\ R1.5 \wedge \neg\ (R1.0 \vee R1.1 \vee R1.2 \vee R1.3 \vee R1.4) \rightarrow Alarm\ (Info,\ list\ (Stakeholder)) \quad (6)$$

Configurable reports

These reports can be configured by domain experts with no technical skills and may be used in different scenarios. A report can be received periodically containing a set of completed procedures, i.e. those which has completed the last registry. These reports may also contain information about procedures opened but not completed, or lacking of some step registration. Generic reports like these enable to obtain information about some processes execution, this information represent a business report for domain experts. When adapting these reports to be sent by email, e.g. to the personnel manager, they contain the last month hiring processes summaries, the hiring that did not followed the process correctly, the people hired and the ongoing hiring.

On demand reports

This is the component that tries to solve the structural modifications the system allows. These reports or alarms are adapted to a specific situation and created with software development processes by software engineers. They are integrated in the system by means of plug-ins and behave like basic actions. On the other hand the events the system responds to are:

- **New registry event.** Launched when a new registry is stored in the database.
- **Error event.** Launched when some of the next errors is generated: *incomplete registry error*, *repeated registry error* or *login error*.

Rules editor

The last step is the construction of a rules editor in order to simplify domain expert's work. The editor enables experts to model the conditions as a predicates tree and the actions as lists that can be configured. Fig. 4 depicts the rules editor interface where the left panel shows the available predicates

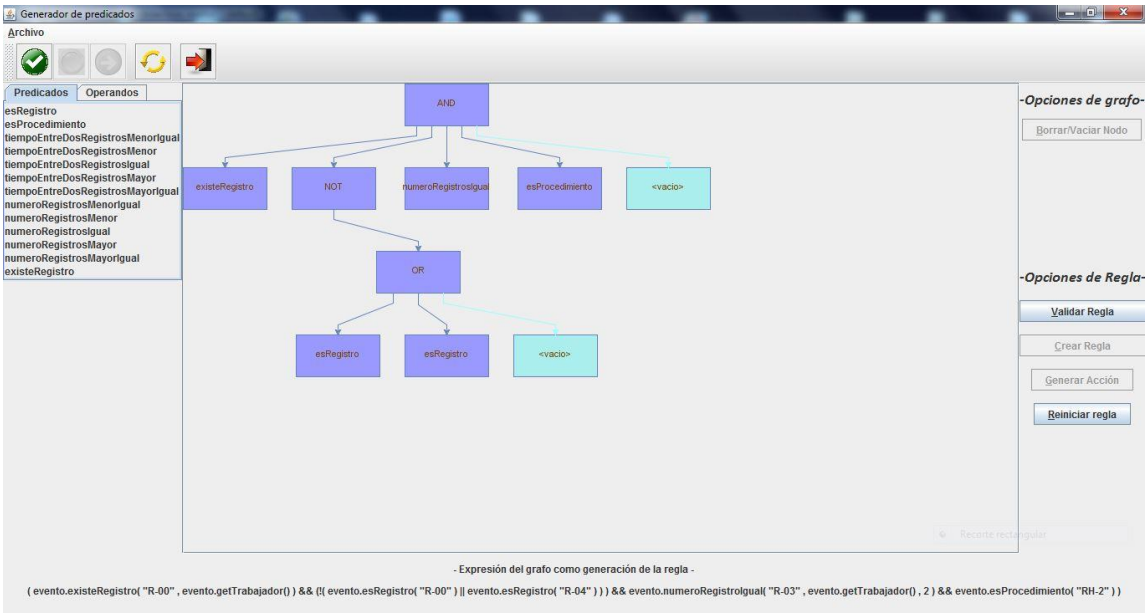

Fig. 4. Rules editor interface

and the logical operators. These options allow the graphical composition of the rules in the central panel with a tree style.

This prototype enables the configuration of the events that fires the rules' evaluation along with the configuration of each predicate and action. It also brings the visualization of the rule list defined for the system. In Fig. 4, the composed rule is being performed at the bottom of the window editor.

VII. Conclusions

This work proposed a solution to software adaptability reducing the development time with no, or few, technical intervention. A set of automated production rules has been used to achieve software adaptability. Furthermore, a use case that implements this kind of development was used in a real scenario and a prototype developed. It can be said the development time needed to adapt new software solutions to business model is reduced by using the approach presented in this work. Also, this proposal increases the possibilities of domain experts in modeling the most frequent adaptations by using the graphical rules editor developed. As far as the concern of the authors, evaluating the results of this work, rule systems have shown a high suitability for the adaptation of very dynamic systems in reducing the time and cost of putting those systems into exploitation. Furthermore, the rule-based architectural style has been implemented in order achieve systems adaptation to frequent changes with minimal effort.

References

[1] M. Fayad and M. P. Cline, "Aspects of software adaptability," Communications of the ACM, vol. 39, no. 10, pp. 58–59, 1996.

[2] R. . Gamble, P. . Stiger, and R. . Plant, "Rule-based systems formalized within a software architectural style," Knowledge-Based Systems, vol. 12, no. 1–2, pp. 13–26, Apr. 1999.

[3] M. Fowler, UML Distilled. Addison-Wesley, 2004, p. 192.

[4] T. Sohn and A. Dey, "iCAP: an informal tool for interactive prototyping of context-aware applications," in Human-Computer Interaction, 2003, vol. 2, pp. 974–975.

[5] Y. Li, J. I. Hong, and J. A. Landay, "Topiary: a tool for prototyping location-enhanced applications," in Proceedings of the 17th annual ACM symposium on User interface software and technology, 2004, vol. 6, no. 2, pp. 217–226.

[6] A. G. Kleppe, J. Warmer, and W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise. 2003.

[7] Omg, "OMG Model Driven Architecture," Object Management Group, vol. 2009, no. Marh 13th. pp. 1–5, 2009.

[8] S. J. Mellor and M. J. Balcer, Executable UML: A Foundation for Model-Driven Architecture. Addison-Wesley Professional, 2002, p. 416.

[9] J. Huamonte and K. Smith, The use of roles to model agent behaviors for model driven architecture. IEEE, 2005, pp. 594–598.

[10] T. Morgan, Brought to you by Team-Fly Business Rules and Information Systems : Aligning IT with Business Goals. Addison-Wesley Professional, 2002, p. 384.

[11] "Drools Expert," 2013. [Online]. Available: http://www.jboss.org/drools/drools-expert.html. [Accessed: 21-Mar-2013].

[12] B. McBride, Jena: a semantic Web toolkit, vol. 6, no. 6. IEEE Computer Society, 2002, pp. 55–59.

[13] E. F. Hill, Jess in Action: Java Rule-Based Systems. Manning Publications Co., 2003.

[14] D. Selman, "JSR 94: Java Rule Engine API," Java Community Process, 2004. [Online]. Available: http://www.jcp.org/en/jsr/detail?id=94. [Accessed: 13-Apr-2013].

[15] D. Sottara, P. Mello, and M. Proctor, A Configurable Rete-OO Engine for Reasoning with Different Types of Imperfect Information, vol. 22, no. 11. 2010, pp. 1535–1548.

[16] A. K. Dey, R. Hamid, C. Beckmann, I. Li, and D. Hsu, "a CAPpella: programming by demonstration of context-aware applications," Proceedings of the 2004 conference on Human factors in computing systems CHI 04, vol. 6, no. 1, pp. 33–40, 2004.

[17] L. De Russis, F. Corno, and D. Bonino, "A User-Friendly Interface for Rules Composition in Intelligent Environments," in Ambient Intelligence Software and Applications, 2011, vol. 92, pp. 213–217.

[18] R. Hilliard, "IEEE-Std-1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems," IEEE httpstandards ieee org, no. IEEE-Std-1471–2000, 2000.

Juan Fuente, A. A. is a lecturer of Computer Sciences at the University of Oviedo (Spain). He received PhD in Computer Engineering in 2002. His research interests are Software Architecture, Web engineering and e-learning Architectures. He has published more than 30 books and papers in refereed scientific journals and conferences, and he has taken part in more than 20 research projects.

López Pérez, B. has a Ph.D. on Computer Science Engineering from the University of Oviedo. He's actually a tenured Associate Professor at the Computer Science Department of the University of Oviedo. Research interests are Computational reflection, Aspect Oriented Software Development, Meta-level systems and meta-object protocols, Web Engineering and software architecture applied to e-government domain. He has participated in several research projects funding for Microsoft Research, Spanish Department of Science and Innovation, and Regional Government. He has held various positions. He is currently Director of the Computer Science Engineering School at University of Oviedo.

Infante Hernández G. has a Web Engineering Master degree 2010, from University of Oviedo. Has worked as research fellow in University of Holguin, Cuba from 2006 to 2010 as part of the Computer aided Design and Computer aided Manufacture (CAD/CAM) research department team. From 2010 to the present he has been working on his doctoral thesis at the Computer Science Department of University of Oviedo. Some of the research interests he is working on include Software Architectures, Model Driven Engineering, Rule driven software development and Web Engineering among others.

Cases Fernández L. J. has a computer engineering degree from University of Oviedo. Has been collaborating in the development of several research projects related with the use of rule-based systems for business models management included the one described in this paper. He is currently pursuing the master degree in Web Engineering from the University of Oviedo.