Achieving business process flexibility with business rules

Tim van Eijndhoven
University of Twente
t.e.vaneijndhoven@alumnus.utwente.nl

Maria-Eugenia Iacob
University of Twente
m.e.iacob@utwente.nl

María Laura Ponisio
University of Twente
m.l.ponisio@utwente.nl

Abstract

Business processes used in networked business are often large and complex, which makes them difficult to manage and change. In this paper we address this lack of flexibility by proposing a solution that uses business rules and workflow patterns to model the variable parts of process flow, thus facilitating dynamic pattern composition in these areas. We argue that the increase in flexibility is justified by the fact that changes in a business process can be confined to the variable isolated parts of the process.

Keywords: business rules, business processes, networked business, service oriented architecture

1 Introduction

Business networks and their impact on business processes have gained a lot of attention lately. With the adoption of networked business, automation and flexibility of business processes in organizations have become critical. This is confirmed by the list of requirements for successful adoption of networked business proposed in [18], in which organizational and operational flexibility and transformation of the organization into a process-centric one are indicated as being necessary. In particular, the dynamic character of networked business requires organizations to quickly respond to the rapidly changing business structures and multitude of upcoming technologies. This demonstrates that business processes flexibility is an important factor influencing networked business success. In this context, business rules are rapidly gaining popularity as a means to separate the business logic from the operational processes and applications. They allow the specification of business knowledge in a way that is understandable by 'the business', but also executable by rule engines, thus bridging the gap between business and technology. Although rule-based systems and rule engines have been in existence for a long time, their application at the business level is of a much later date.

In the context of the above-mentioned developments, we propose a rule-based approach to support the specification of variable parts of service oriented business processes. Hence, our aim is to develop a method and technical solution that facilitates the customization of a business process to a particular usage context by isolating the variable parts from the reusable parts of business process models and combining the reusable parts with business rules that model the variable parts. Such an approach is likely to be of direct relevance and applicability for Dutch governmental organisations since they aim to integrate their processes and deliver together a variety of services for citizens and since their existing business processes have proved to be too rigid and inflexible to support new services. This application area supplied us with the case used throughout the paper to illustrate our approach.

The paper is organised as follows: in Section 2 we give a brief introduction to some basic theoretical concepts used throughout the paper. In Section 3 we present our solution in a nutshell. In Section 4 we illustrate the application of our method in detail and provide a brief description of the implementation of a prototype (Section 4.3). We discuss related work in Section 5 and we draw conclusions and discuss directions for future research in Section 6.

2 Preliminaries

In this section we give a brief introduction to process flexibility, workflow patterns and business rules. Furthermore the running example we use in the remainder of the paper is introduced and motivated.

2.1 Process flexibility

Although business process flexibility is abundantly mentioned in the literature, it is difficult to express in concrete quantifiable terms what the flexibility of a

1541-7719/08 $25.00 © 2008 IEEE
DOI 10.1109/EDOC.2008.23

business process entails. There are few explicit criteria that can be used to measure process flexibility.

Kasi and Tang [9] proposed a framework for the comparison of business processes in which they express the flexibility according to three dimensions:

- Time – the process should adapt to change more quickly;
- Cost – the process should adapt to change with less cost;
- Ease – the process should adapt to change with maximum ease.

This suggests that an increase in flexibility can be achieved when a process can be changed in shorter time, with less costs and easily. While time and cost can be measured relatively easy the 'ease' dimension is not easy to quantify. In this research we assume that ease is expressed in terms of:

- less items to change,
- having the items that have to be changed in one place and
- being able to make the translation from the new requirements for process change to executable workflow in less stages (e.g., specifying new requirements in a manner that is closely related to the executable workflow).

2.2 Variability

Variability is the property of an object of being changeable. The capacity of systems to be tailored (commonly known as 'system variability') has been extensively researched in relation with software engineering [26], [27] and ERP systems [19]. In contrast, limited attention has been paid to variability in relation with behavioural aspects of process-oriented systems (see [12], [7], [14]). However, in [26] an analogy is established between the product family engineering paradigm (also known as software product line engineering [27]) and a Process Family Engineering approach. In [26] the *variability* in the process family is modelled by means of *variation points* to which *variants* can be bound by means of *variability mechanisms*. Furthermore, the following four types of basic variability mechanisms are identified:

- encapsulation of varying sub-processes;
- parameterization;
- addition, omission and replacement of single elements and
- data type variability.

[7] also suggests a general - variable relation between process parts, in which the differences between alternative specialized process parts are made explicit at the so-called *points of variability* where parts that are likely to change will be externalized. We adhere to this line of thinking motivated by the argument that by the use of this approach the number of different business processes can be significantly reduced since it is no longer necessary to create a whole new process flow for each of the variants of the variability points.

2.3 Workflow patterns

A second analogy between software design and process design, which is of relevance in the context of this paper, is the parallel drawn between design patterns (as defined by Gamma et al. [6]) and workflow patterns [23]. The idea is that complex code structures can be broken into pieces for which certain design patterns are reused throughout the code. This can also be done for business processes. Research by Van der Aalst et al. [1] supplied us with a set of workflow patterns which provide common flow functionality for workflows. These flows also occur in the business processes, which makes them applicable in the context of this research as well.

2.4 Business rules

A business rule is "a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behaviour of the business" [2].

Business rules can often be specified by using near natural language. Thus they allow specifying business knowledge in a way that is understandable by 'the business', but also executable by rule engines, thus bridging the gap between business and technology. The above definition of business rules is general enough to cover a wide range of business rule types. We identify two main categories:

- Rules that influence the operational process: *Derivation rules* (such as deduction rules and computation rules) that are used to establish information that is used in a process and *Action rules* that establish when certain activities should take place. Two variants of action rules can be distinguished: condition-action rules (production rules) and event-condition-action (ECA) rules.
- Constraints, which impose certain limitations to the structure, behaviour or information of an organisation or system (i.e., *deontic assertions, state constraints, process constraints*).

The focus in this paper is on action rules.

Work is currently done to finalise standards for business rule specification languages in all MDA layers [14] of models (e.g., SBVR [16], PRR [15]).

Also results have been reported with respect to the definition of model transformations between business rule specification languages positioned in the different MDA abstraction layers (e.g., [12]).

2.5 Problem statement

The general problem addressed in this paper is the need to develop a rule-based technique to improve the change management and maintainability of business processes. A number of requirements that had to be met by a possible solution have been identified at the beginning of this research. Such a method should allow process architects to predict and mange the impact of change, while diminishing the time needed to implement changes. In other words, in the event of a change such a method should allow us to isolate the parts of the process that are likely to be changed and, thus, maximize the process parts that are stable. Because business rules change at a different speed than the implementation part of the process, a desirable characteristic is that the method will explicitly keep business rules separated from the implementation of process. Finally, the method has to be realizable with tools and platforms that are currently available.

3 Our solution in a nutshell

The method we propose comprises three steps:

Step 1. Identify the variable and non-variable segments in a process. Such an analysis must result in a generally stable high-level process flow that contains a number of variation points (i.e., one for each variable part of the process) that can thus be isolated from the rest of the process. Henceforth we assume that possible changes will only affect the variable parts.

Step 2. Identify an appropriate (combination of) workflow pattern(s) that model the behaviour of each variant in a variation point. In this step we use a dynamic composition of workflow pattern instances. Experts link the parts of the process flow to general instances of patterns. Our solution does not constrain the choice for a specification language used to describe the patterns. Consequently, any specification language can be used.

Step 3. Implement workflow patterns using business rules. In this step experts select a business rule specification formalism and use it in a structured way to implement the workflow patterns. For each variant, all the identified variation points in the overall process model must be resolved. In addition this step may include the testing, deployment and execution of the resulting process.

3.1 Example

Step 1. An example could be a process in which applications can be accepted or rejected, when an application is accepted this should be published via various channels. The decision on which of these channels the decision should be published varies for each application. In this process the variation point is the point where the publication activities are to be invoked.

Step 2. We implemented a number of simple workflow patterns from the work of Russell et al. [23] using Event-Condition-Action (ECA) rules [10]. For the example from step 1 we use the "choice" pattern. An informal description of this pattern is "the selection of branches in which the preceding branch can diverge, based on logical expressions associated with each of the branches. According to its refinement the thread of control is transferred to a single or multiple branches."

We model this pattern using Business Process Modelling Notation (BPMN). Figure 1 depicts the choice pattern.

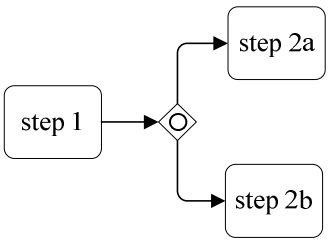


Figure 1. The choice pattern

Step 3. This pattern can be expressed by means of business rules, using the business rule notation proposed in [10], as follows:

Listing 1.Choice pattern: step 1

```
(Rule 1)
ON application accepted
DO prepare publication
RAISE publication ready
```

Listing 2.Choice pattern: step 2a

```
(Rule 2)
ON publication ready
IF publication for website
DO publish to website
RAISE publication finished
```

Listing 3. Choice pattern: step 2b

```
(Rule 3)
ON publication ready
IF publication for newspaper
DO publish to newspaper
RAISE publication finished
```

The rule in Listing 1 raises an event 'publication ready' when it is finished. This event triggers the rules in Listing 2 and Listing 3 in no particular order. These rules fire only if their respective conditions are satisfied. In terms of business rules, a new parallel branch corresponds to adding another rule that links the fired 'publication ready' event with the desired conditions.

3.2 Results

We implemented a prototype to support our method and to demonstrate its feasibility. Using our method in practice revealed some advantages for the maintainability of the process.

First, changes were isolated. Business rules were stored and maintained separately from process models, facilitating the detection of the parts of the process that had to change. A smaller part of the process had to be changed, namely the workflow patterns related to certain rules.

Moreover, the process was easier to comprehend than with traditional methods because activities were more easily readable and writable. Business process diagrams are very suitable for giving the overall view of the process and for seeing where in the process a specific activity is used but not when looking at the level of individual activities. In general, many process changes revolve around the conditions in which a certain activity has to be carried out. Using business rules for expressing the conditions under which specific activities occur makes them easier to comprehend. Moreover, when the rule that corresponds with a given activity has to be retrieved from a library of rules and altered according to the required change, the main process and the other rules remain *unchanged*. Furthermore, business rules can be written in a nearly natural language, which makes them comprehensible for business people, which actually have the business knowledge required to write and maintain them.

Finally, business rules can be stored and maintained separately from process models. When using business process diagrams, the same activity occurring in different processes is often duplicated amongst those processes, allowing for little reuse and requiring changes to be made in various places when that activity is modified; which in turn increases the risk of adding an inconsistent change. Using business rules, processes may invoke/retrieve (and reuse) rules controlling specific activities from a central repository. In fact, most business rule management systems are actually aimed at organizing large sets of business rules and at enhancing their reuse. As such they fulfil the role of such a repository, thus making rules available and consistent across different systems and organisations.

To summarise, we argue that our approach has two major advantages: it facilitates the reuse of the general part of the process and it allows us to isolate and externalise those parts of the process that are likely to change, thus limiting the impact of such changes. This will reduce the time and cost required for incorporating and carrying out changes and will increase the business process flexibility [24].

4 Method in detail

We applied the method described in the previous section to an example. Our running example (which we call "The environmental permit request" case) describes the process to get an environmental permit. This process is currently under development in the Dutch Ministry of Housing, Spatial Planning and Environment. The example is interesting because it describes a real and complex process requiring high process flexibility under change. The following subsection describes our running example in detail.

4.1 The environmental permit

The new process of requesting an environmental permit combines numerous separate permit requests into one all-embracing permit request. For each of the request parts different advisory parties are involved and expected to assess whether the permit should be granted. As these advisers differ for each municipality the number of different advisers is very large. In addition to this, the fact that a large number of combinations of permit request parts are possible leads to an even larger number of possible activity combinations in an instance of such a permit request process. This makes the detailed process model very large and complex, while the basic high-level process is relatively simple and always having the same three steps:

1. receive permit application;
2. request advice on the permit application;
3. inform the requestor on the permit request outcome.

The complexity of this business process is mainly caused by the multitude of possible implementations of the "request advice" step in the process. Using "traditional" modelling techniques for each of the possible process flows in step 2 will result in a very large and complex diagram or set of diagrams that are difficult to comprehend. This means that if a change of the process is necessary it will take a lot of time/money to incorporate the change into all the possible

alternative flows. Also in terms of "ease", it is very likely that carrying out such changes will not only require having the overview of the whole process with all its possible alternatives and exceptions but also knowing how and where they have been implemented in the supporting applications. Especially in such an environment in which regulations and laws regarding this type of requests may change quite often, the above considerations show that using classical diagramming techniques will very quickly result in difficult, time consuming and expensive business process maintenance. Furthermore, a supplementary argument for choosing this application area is the fact that especially such regulations are very suitable to be modelled as business rules.

4.2 Applying the method

We assume that a business process execution engine and a business rules engine are available such that the specification and execution of both processes and rules is possible. Consequently, at each variation point the process engine is invoking the business rules engine, which in turn executes the part of the workflow specified by means of business rules before returning its result to the business process. Thus, the process engine and the business rules engine are independent but communicate directly with each other and the control of the execution is shared between the process engine (when the main process flow is active) and the business rules engine (when its service is invoked).

In order to demonstrate the feasibility of the approach a prototype (briefly described in Section 4.3) implementing this interoperable architecture has been built, using the Aqualogic BPM Studio (BEA systems) process engine and the ILOG business rules engine.

Step 1. We refine the first step by dividing it into the following sub-activities:

The identification of the high level process. In this first step a 'generic' high-level process description has to be defined. At this level the process parts that are likely to be variable must be encapsulated into individual activities.

For the simplified Environmental Permit Request this consists of the following abstract activities (see the model below):
1. Receive a permit request,
2. Request a number (x) of advice on the permit request,
3. Inform the requestor with respect to the decision.

The identification of variability points in the process. After creating the generic process the points of variability in the process have to be identified. These points have to be made explicit in order to be able to isolate them from the parts of the process that are static and further specify them. These are usually those activities in the generic process that can not directly be operationalised.

In our generic process example the 'receive request' and the 'provide answer' activities are stable and can directly be made executable, for example by invoking appropriate supporting software services. The 'request advice' is an activity that can not directly be converted into an executable workflow. This activity is built up from other activities and flows that vary according to the contents of the permit request. Therefore the variability point of this generic process is the 'Request advice' activity.

Estimate the level of variability. For all of the identified points of variability an estimation of the frequency of changes that may occur in the future at that point in the process has to be made. Based on these estimations decisions have to be made on which variability points will be modelled using business rules and which will be treated as 'static' parts of the process. For example in the case of a variability point that has a small number of small variants that are unlikely to change in the future it might be overkill to model them in business rules.

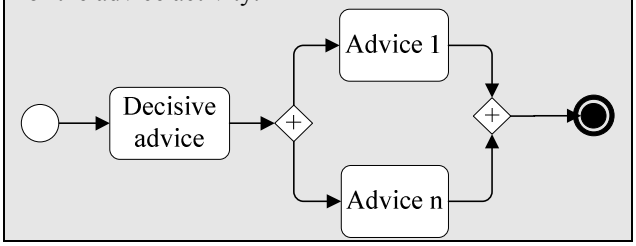
For the 'Request advice' activity the actors involved may vary according to the contents of the request and according to the location. Also the flow of the advice activities depends on the contents of the permit request. This means that there is a large number of different process flows possible. Also it might be very well possible that due to new or modified regulations in the future new elements will be added to the permit request, possibly requiring new advisers. This makes the estimated level of change high and justifies the workflow specification using of business rules.

Step 2. Identify the workflow patterns of the variability points. For all of the variability points that were chosen to be modelled using business rules, first the workflow patterns used to model the possible variants have to be identified.

To illustrate this step for our example we choose to use the following three patterns for a particular variant:
- Parallel flow (Advice can be given in parallel)
- Sequential flow (Advice can be given in sequence)
- Synchronization (Since there can only be one decision the parallel advice will have to be re-

joined using synchronization)

At an abstract level the following process is used for the advice activity.



Step 3. We refine the third step by dividing it into the following sub-activities:

Write rules to implement the workflow patterns. When the workflow patterns have been identified rule-sets will have to be written that implement them. In this step decisions have to be made on the events and conditions used to structure the flow of the business rules. The choice on what events to use is important because it determines the impact that a future change of business rules has on the whole set of business rules. When for example introducing an intermediate step in a sequence flow this will have impact on either the events that subsequent business rules react on, or on the events that previous business rules trigger.

The advice activities that have to be performed are determined by the contents of the permit request so it is important to know how this request is defined. In this case study the permit request is a simple string using XML-like tags to delimit the various elements. As an example the following elements are used:

<city> The city in which the activities will take place,
<soil> Request part for soil activities,
<cutting> Request part for cutting activities,
<building> Request part of building activities,
<province> Request part used for organizations that have a special status requiring the province to advice on a permit request.

All of these elements contain descriptions of their respective request part. Based on the existence of a <soil>, <cutting> or <building> element, activities for requesting advice to the responsible advisers will be inserted into the flow. For the <city> and <province> element the adviser is selected based on the name of the city or province that is contained in the element.

The <province> element is considered to have the highest priority and therefore province-related advice activities will be executed before other advice, if present. The other elements are equally important and therefore the corresponding necessary advice activities will be executed in parallel.

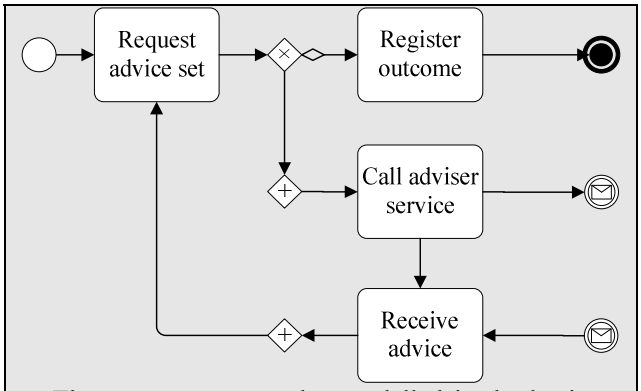
Due to space limitations we only give here one example of rule specification written in the intellirule syntax of iLOG:

```
definitions
  set 'the url of the advisor' to "http://vm_bea:9000/
    fuegoServices/ws/ProvideAdviceServiceListener" ;
  set 'the advisor' to "overijssel" ;
  set 'the indicator of the advice' to "<province>
    overijssel</province>" ;
  set 'the event trigger' to "preferrent_advice" ;
  set 'the triggered event' to "normal_advice" ;
if
  'the current event' is 'the event trigger'
  and { 'the given advices' } do not contain 'the advisor
    '
  and 'the request contents' contains 'the indicator of
    the advice'
  and it is not true that 'the request set' contains a
    request to 'the advisor'
then
  set 'the selected advisor' to 'the advisor' ;
  set 'an advisor is selected' to true ;
  add request to 'the advisor' at 'the url of the advisor
    ' to 'the request set' ;
  set the event that 'the request set' triggers to 'the
    triggered event' ;
  set 'the current event' to "preferrent_advice_selected"
    ;
```

Model the concrete process. Now that the business rules sets have been defined, the parts of the high-level process that are not modelled using business rules have to be modelled using a business process language. This includes the modelling of the interaction points between the business process engine and the business rules engine.

In this case the process is modelled as indicated below:

In this process the 'request advice' is a collapsed sub-process activity that is modelled in a separate sub-process model. In this sub-process the process engine first executes an activity 'Request advice set' in which it performs a request to the business rule engine providing the permit request. The business rule engine in turn executes its business rules based on the provided permit request and context and constructs the next 'RequestSet' containing the information for the invocation of the required advice services. The business process engine now checks if there are more requests to perform and if so calls the required advice services. After the answers are received the process loops back and requests the next request set. If there are no more requests to perform the result is checked and registered after which the process ends. Graphically the process looks like:



These processes are also modelled in the business process software that is used for the prototype. Those models contain some implementation specific elements such as intermediate activities linking to sets of screens that a user has to go through to fill in or view data.

Most commercial business rules management systems offer templates and editing support functionality that makes business rule writing even easier. In addition, the translation step from rules specified in nearly natural specification languages to executable specifications can be automated by means of model transformations.

Test, Deploy and Execute the process. When the process is fully modelled it will have to be tested and debugged to ensure the correct working. After this phase the process can be deployed to the business process engine and the business rule engine such that it can be used.

In the case of the prototype the business process models have been modelled, tested and deployed in the Aqualogic software and the business rules have been modelled, tested and deployed in the iLOG JRules software. After their deployment they can be used to execute the permit request process.

Update process. During the lifetime of the business process changes will occur at the variability points that were identified. In order to support these changes the business rule sets will have to be updated such that they can reflect the new requirements regarding the process flows. If changes may affect a part of the workflow that is currently modelled using business process models careful consideration has to go into deciding whether to alter only the process model or to consider from now on that part of the process as a point of variability that has to be modelled using business rules.

There are two major scenarios for change in the environmental permit case:

- A change of adviser for a specific element
- An additional element in the permit request

These can easily be solved by changing the specific business rule for the adviser or adding a business rule that is triggered by the new element. With the rule structure used in the prototype this would require changing the appropriate variables in the case of a change of adviser or filling in the various variables for a new rule (based on a specific rule template) in the case of an additional element.

Using this method careful consideration goes into the possible changes to the business process in the future. Based on this the process is set up in such a way that most changes can be made faster, cheaper and easier compared to traditional methods.

With the application of the method in the case it can be seen that using this method allows for easier specification of all the business process variances. This offers a more flexible process specification. It can also be seen that the functionality that is offered by our solution is similar to using business process diagrams, the same business process and its variances can be expressed using business rules.

4.3 Prototype

In order to demonstrate the feasibility of the approach a prototype has been built. In this prototype the Aqualogic BPM Studio (BEA systems) process engine interoperates with the ILOG business rules engine. The Aqualogic software is also used to generate the user interface for the end user interaction in the case. The ILOG business rules engine is used to model and execute the business rules that provide the flow for the variable parts of the business processes from the case. Although the JRules engine is not a specialized ECA rules engine, it can execute such rules using the Rete algorithm. Thus, ECA rules can be modelled by using a number of variables as events. When these variables change (i.e., when an event is triggered) the engine will re-evaluate the conditions of the rules that involve these variables (the rule is fired).

The Aqualogic process engine runs within the Aqualogic studio environment. The ILOG business rules engine runs in a JBOSS application server environment. The interaction between the two engines takes place using web services.

When developing the prototype it became evident that the software that was used has not been designed for the way in which the prototype for this research tries to use them (e.g., the business process engine could not use complex type arguments for the web services and failed on parameters with an underscore in their name). These limitations have also influenced the design of the interaction between the process engine and rule engine described below:

1. *The process engine invokes the rule service with the data of the request.* The input data for the

business process is passed on to the rule engine for evaluation.

2. *The rule engine creates an empty context object*. A "RequestContext" object is created which contains the current event and a sequence of "RequestSet" objects which basically is the next set of parallel actions to execute, the RequestSet in turn contains "RequestItem" objects which contain the specific data for each action and the event that is to be triggered after its execution.
3. *The rule engine evaluates its rules based on the data*. Based on the input data and the current context the rule engine evaluates its business rules. In the actions of the business rules the various "RequestItem" objects are created and a new "RequestSet" is added to the context.
4. The rule engine returns the updated "RequestContext". The context is serialized into a JSON string for the exchange and then returned to the business process engine.
5. *The business process engine deserializes the context and executes the "RequestSet"* The business process reconstructs the "RequestContext" object from the serialized representation it received from the rule engine. The current "RequestSet" is divided into its "RequestItem" objects, thus creating new execution threads for each item and allowing for parallel execution. The request is performed and the results are recorded into the "RequestItem". When all items are executed the updated "RequestItem" objects are saved to the "RequestSet" which in turn is saved to the "RequestContext". The current event property of the "RequestContext" is updated according to the event that the items have triggered.
6. *The business process engine re-invokes the rule service with the new context*. The business process engine serializes the "RequestContext" again and sends it to the rule service.
7. *The business rules engine deserializes the "Request-Context" and re-evaluates*. The business rules engine reconstructs the "RequestContext" and evaluates its rules again to construct the next "RequestSet".
8. Steps 3 through 7 are repeated until there is no next "RequestSet" When the evaluation of the business rules leads to the conclusion that there are no new actions to execute the context object is returned to the business process engine as is.
9. The business process engine processes the result and notices that no request set have to be executed. It will therefore query the "RequestContext" for the results of the variability part of the process.

5 Related work

Charfi and Mezini [4] offer a comparable solution by using a BPEL dialect: AO4BPEL. In this dialect BPEL is extended to support aspect oriented constructs like before, after and around advice [3]. The business rules actions and results are translated to business process constructs and to so-called "point-cuts" (statements to relate the aspect to specific points in the code such as every assign activity). This requires a modified BPEL engine to be able to cope with the additional aspects. This differs from our approach which makes use only of existing software and does not need a specialised engine.

In research by Cibran and Verheecke [5] the idea of using aspects to relate business rules to business processes is presented in a more generic way. They do not limit their approach to BPEL and consider the activities in the process description as points to place point-cuts. The business rules are translated to aspects, which in turn contain business process constructs to change behaviour at the place of a point-cut. This approach differs from our approach because it uses business rules as a means to alter business process models, while our approach replaces the models in the variable parts with business rule specifications implementing the corresponding workflows.

Rosenberg and Dustdar use a business rules service to intercept all incoming and outgoing messages and to apply business rules on them [19]. In addition they implemented a business rules broker to enable the use of different rules engines with a pluggable interface [21]. This approach is different because the business rules do not directly alter the business process, they function more like a filter while our approach allows for interaction between the business process and the business rules.

Orriëns et al. [17] generate process specifications by using a composition engine that takes the process elements from a component element repository and the business rules from a composition rule repository. The rules contain facts on the process elements and their required flows, based on which the composition engine is able to construct the flows and the elements into a process description that can be then executed.

In research by Lee et al. [11] a method is proposed without a direct implementation. Similar to the solution proposed by Orriens et al. [17], Lee et al. [11] consider the activities as process elements and construct flows based on ECA rule constructs. By chaining these rules a flow is constructed that can be translated to an executable process specification.

In research by Knolmayer et al. [10] business rules are related to workflow patterns with the use of ECA

rules. With these relations they show that a business process can be expressed in terms of business rules. In their approach they still use a generator to actually generate the process specification. These approaches differ from our approach because they are used for generating process specifications while our approach composes the process at runtime.

Research by Rosenberg and Dustdar [22] suggests an approach using distributed rules engines that communicate with each other in order to handover work between the different systems. This is done by using a rules engine wrapper called ViDRE [13]. This solution addresses the distribution of the control flow over different actors. In this approach the execution of the business rules forms the actual process flow.

The same issue of distributed flow execution is also addressed by Schmidt [25], which uses the structure of SOAP messages to execute business rules at different locations. This is done by recording the business rules in the SOAP headers and then sending this message through a set of intermediaries. These intermediaries then execute the rules applicable to them. This allows for distributed execution of the business rules [25]. These approaches base the whole process on business rules where our approach uses a mix between business process models and business rule specifications.

6 Conclusions and future work

In this paper we have proposed a solution to increase the flexibility of service oriented business processes by using ECA rules to execute parts of the process at variability points. By explicitly identifying and isolating the variability points in the business process the ease of incorporating changes in the process increases since changes are localized.

ECA rules can be used to model the flow in a business process because they can implement the same workflow patterns as "traditional" business process languages. Furthermore they have the advantage of being modelled using near natural language, thus allowing non-IT people to maintain them.

We have also demonstrated that this approach is practically feasible by implementing a prototype that was used for method testing purposes. The selected case shows that incorporating the changes that are most likely to occur become easy when using the approach suggested in this research. Furthermore, although the prototype implementation was tuned to the selected software platforms, the concepts are generic enough to be applied in combination with other software packages.

This research also raised a number of questions that could be researched in the future. In particular we believe more methodological support is needed for the identification of variability points in business processes and for finding the right balance between the use of business rules and business processes. Another issue open to research is the maturity of the tools. It has been suggested that current tools can be used to implement the solution presented in this paper. However, during the development of our prototype, the selected platforms have proved to have certain limitations. Therefore, it has to be investigated to what extent business rules engines must be transformed from decision support tools into business process composers to allow a better integration with process engines. Finally we mention the issue of business rule performance and manageability in relation with large scale business rule sets. More precisely, we are of the opinion that more research is necessary concerning the impact the size of rule collection has on manageability and on performance. Especially when this solution must be implemented in large scale complex business processes this might become an issue.

Acknowledgments

We would like to thank TNO, and in particular Menno Holtkamp and Wout Hofman for supporting this research. The work presented in this paper was also supported by:

- the Freeband A-MUSE project (http://a-muse.freeband.nl), which is sponsored by the Dutch government under contract BSIK 03025.
- the project Quality-Driven Requirements Engineering and Architectural Design (QuadREAD), which is part of the Dutch Jacquard program.

References

[1] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros, "Workflow patterns", *Distributed and Parallel Databases* 14(3) (2003) 5-51.

[2] BRG, "Defining business rules what are they really?" *White paper*, July 2000, http://www.businessrulesgroup.org/first_paper/br01c0.htm.

[3] A. Charfi and M. Mezini, "Aspect-oriented web service composition with AO4BPEL". In *Proceedings of the 2nd European Conference on Web Services (ECOWS)*, volume 3250 of LNCS, pages 168–182, September 2004.

[4] A. Charfi and M. Mezini, "Hybrid web service composition: business processes meet business rules". In *ICSOC '04: Proceedings of the 2nd international*

conference on Service oriented computing, pages 30–38, New York, NY, USA, 2004. ACM Press.

[5] M. A. Cibrán and B. Verheecke, "Dynamic business rules for web service composition". In *R. E. Filman, M. Haupt, and R. Hirschfeld (eds), Proc. of the Second Dynamic Aspects Workshop (DAW05)*, p. 13–18, 2005.

[6] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Element of Reusable Object-Oriented Software*. Published by Addison-Wesley, 1995. ISBN 0201633612. 27th printing, November 2003.

[7] G. Goldszmidt and C. Osipov, "Make composite business services adaptable with points of variability, part 1: Choosing the right implementation". *IBM developerWorks*, April 2007.

[8] W.J. van den Heuvel and M. Jeusfeld, "Model tranformation with Reference Models", in *Proc. 3rd International Conference Interoperability for Enterprise Software and Applications*, Funchal, Portugal, March 2007, pp. 63-75.

[9] V. Kasi and X. Tang, "Design attributes and performance outcomes: A framework for comparing business processes". In *Proceedings of the Eighth Annual Conference of the Southern Association of Information Systems (SAIS)*, pages 226 – 232, Savannah, Georgia, USA, 02 2005.

[10] G. Knolmayer, R. Endl, and M. Pfahrer, "Modelling processes and workflows by business rules". In *Business Process Management*, pages 16–29, 2000.

[11] S. Lee, T.-Y. Kim, D. Kang, K. Kim, and J. Y. Lee, "Composition of executable business process models by combining business rules and process flows". *Expert Syst. Appl.*, 33(1):221–229, 2007.

[12] M.H. Linehan, *Semantics in model-driven business design*, IBM T.J. Watson Research Center, 2006.

[13] C. Nagl,, F. Rosenberg, and S. Dustdar, "Vidre - a distributed service-oriented business rule engine based on ruleml". In *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06)*, pages 35–44, Washington, DC, USA, 2006. IEEE Computer Society.

[14] Object Management Group, *MDA Guide version 1.0.1*, Document Nr: omg/2003-06-01, June 2003.

[15] Object Management Group, *Production Rule Representation: Request for Proposal*, br/2003-09-03, Sept. 2003. http://www.omg.org/docs/br/03-09-03.pdf.

[16] Object Management Group, *Semantics of Business Vocabulary and Business Rules Specification*, OMG Adopted Specification, 2006.

[17] B. Orriëns, J. Yang, and M.P. Papazoglou, "A framework for business rule driven service composition". In *TES*, pages 14–27, 2003.

[18] M.P. Papazoglou and P.M. Ribbers, *e-Business: organizational and technical foundations*. John Wiley & Sons Ltd, Chichester, 2006.

[19] M. Rosemann and W.M.P. van der Aalst, "A configurable reference modelling language", *Information Systems,* vol. 32, no. 1, 2006, pp. 1-23.

[20] F. Rosenberg and S. Dustdar, "Business rules integration in bpel -a service-oriented approach". In *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*, pages 476–479, Washington, DC, USA, 2005. IEEE Computer Society.

[21] F. Rosenberg and S. Dustdar, "Design and implementation of a service-oriented business rules broker". In *CECW '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology Workshops*, pages 55–63, Washington, DC, USA, 2005. IEEE Computer Society.

[22] F. Rosenberg and S. Dustdar, "Towards a distributed service-oriented business rules system". In *ECOWS '05: Proceedings of the Third European Conference on Web Services*, page 14, Washington, DC, USA, 2005. IEEE Computer Society.

[23] N. Russell, A. ter Hofstede, W. M. P. van der Aalst, and N. Mulyar, "Workflow control-flow patterns: A revised view". Technical report, BPMcenter.org, 2006.

[24] S. W. Sadiq, W. Sadiq, and M. E. Orlowska, "Pockets of flexibility in workflow specification". In *ER '01: Proceedings of the 20th International Conference on Conceptual Modelling*, pages 513–526, London, UK, 2001. Springer-Verlag.

[25] R. Schmidt, "Web services based execution of business rules". In *RuleML*, 2002.

[26] A. Schnieders and F. Puhlmann, "Variability mechanisms in e-business process families", in *Abramowicz & Mayr (eds.), Proc. 9th International Conference on Business Information Systems (BIS 2006)*, volume P-85 of LNI, Bonn, Gesellschaft für Informatik, 2006, pp. 583 601.

[27] T. Ziadi and J.M. Jézéquel, "Software product line engineering with the UML: Deriving products", In *Software Product Lines*, LNCS, 2006, p. 557-588.