Towards a dynamic rule-based business process

Jose Felipe Mejia Bernal* and Maurizio Morisio

Politecnico di Torino,
Dipartimento di Automatica e Informatica,
Turin, Italy
Fax: +39 0115647099
E-mail: jose.mejiabernal@polito.it
E-mail: maurizio.morisio@polito.it

Abstract: Making a business process more dynamic is an open issue, and we think it is feasible if we decompose the business process transitions and activities in a set of rules defined as Event Condition Action (ECA) rules. The goal of a dynamic rule-based business process is to change the implementation of a process instance at runtime. We are proposing a way for representing a Dynamic Business Process in terms of rules based on pattern identification. In this paper, we also discuss characteristics related to business process execution time and effective business process modification support. In particular, we analyse YAWL and Bonita Workflow in order to compare them with our approach and discuss their strengths and weaknesses.

Keywords: business process; context awareness; web based services; XML.

Biographical notes: Jose Felipe Mejia Bernal is a Research Assistant and PhD student in the Software Engineering Group at the Department of Computer Science and Automation of Politecnico di Torino, one of the leading engineering universities in Italy. He received his MS Degree in Computer Science Engineering in 2006. He is involved in Italian projects mainly related to mobile and Web-based technologies. His current research interests include dynamic business processes, automated software engineering and mobile programming. He has published research papers at national and international conference proceedings as well as chapters of books.

Maurizio Morisio is an Associate Professor in the Department of Automation and Computer Science of Politecnico di Torino. He works in the area of Software Engineering. Within this broad domain, his goal is to understand how software is produced and maintained in real life settings. In the last years, his research has focused on technology and tools – object oriented analysis, design and programming, modular petri nets for specifying and simulating CIM systems, product lines and framework based development.

1 Introduction

A business process defines the context and the logical relationships between activities, and also specifies both the order of invocations and the rules for data transfer. Such activities are configured to produce a specific output and associated with specific objectives. Business processes can be described from many points of view, for example the process designer, the person who is responsible for scripting how the application is supposed to interface with users and software components.

Nowadays, many business process solutions are often Web-based. Web Services Business Process Execution Language (WS-BPEL) is widely used for specifying and executing composite business processes (Kareliotis et al., 2009). Some of them are made for long-lived processes, others for short-lived ones. A loan application, for example, is a long-lived process. A short-lived process is often related to user-interfaces. For example, the flow of screens that users see in one interaction with a Web application. This one might never have a process that lives more than a few minutes.

Business logic often changes, that is exactly the place in which a rule engine can add value. A rule engine is used to externalise and ease access the business logic in terms of a language that can be close to plain english. A business rule is a statement that defines or constrains some aspect of the business (Business Rules, 2000). They provide the means to express, manage and update different business components. Such rules have to be expressed and integrated in terms of the defined activities.

A rule engine improves the translation layer from business logic to technical language. Many approaches are not completely dynamic, flexible and effective when we need to automatically modify the business process instance, by adding or deleting an activity, according to changes of business process context. An important aspect in business process execution is runtime process adaptability. Processes have to be flexible to react to changes in their environment. The goal of runtime adaptability is to change the process while it is running, without having to either remodel or redeploy it.

A specific change might be needed, because someone wants to select the most appropriate service considering different kind of properties (e.g., network capabilities, system security). To address these issues, we propose an approach focused on providing a more dynamic and flexible adaptation of business processes. The main purposes of our approach is to increase the adaptability of the system, by decomposing and representing a business process structure in terms of rules through patterns identification; and a reliable method for executing dynamic business process adaptation, according to eventual modifications in the context information.

2 Related works

In this section, we describe and discuss some commercial and academic solutions that provide support for adaptability and flexibility related to dynamic business process modifications.

According to authors in Schonenberg et al. (2008), there are different types of flexibility:

1 Flexibility by design incorporates alternative execution paths within a process definition at design time. The selection of the most appropriate execution path can be made at runtime.

2 Flexibility by deviation provides the ability to deviate at runtime from the execution path prescribed by the original process without altering its process definition.

3 Flexibility by under-specification provides the ability to execute an incomplete process specification at runtime.

4 Flexibility by change provides the ability to modify a process definition at runtime.

One or all of the currently executing process instances are migrated to a new process definition.

There is a vast body of prior work on dynamic modification of business process at runtime. For example, Ellis et al. (1995) propose concepts to manage the continuous changes in the processes. In another work (Reichert and Dadam, 1998), dynamic changes are possible in the process instance but the main problem is related to some restrictions that have to be applied on the operations, in order to maintain the required consistency.

EROICA is a framework (Akhil and Zhao, 2002) that extends the syntax of the ECA rules, but it does not provide an organisational rule modelling and enforcement framework for dynamic business processes.

Distributed workflow execution is characterised by separating one integrated workflow model into small partitions and allocate them to different servers to be executed. Tan and Fan (2006) proposes a Petri net-based approach for dynamic fragmentation of a workflow model. Their approach divides the centralised process model while the process is executed. The fragments created can migrate to proper servers, where tasks are performed and new fragments are created.

Analysis of workflow dynamic changes (Sun and Jiang, 2008) is being considered one of the major issues of business process adaptability. Since migrating a process instance (Khriss and Levesque, 2008), from the original schema to a new one, involves a set of complex steps, we consider that a solution based on schema evolution, could be very expensive in terms of memory, time and resources.

In Xia and Wei (2008), authors introduce an approach for enabling ad-hoc modifications of process instances. They propose a way for perceiving and understanding the business process environment through message communication and monitoring. Some researches propose adaptation based on the logic and the content. Several researchers focus on the logic adaptation of services (Marquet et al., 2002); the service is represented by components; the adaptation is characterised by adding or replacing a component. According to Stantchev and Schröpfer (2009), web-services-based systems present challenges mainly related to service-level enforcement. In this way, it is necessary providing proper self-adapting mechanisms.

Web services have become an emerging and promising technology to design and build complex business applications. To enforce dynamic adaptation of service composition, authors (Eid et al., 2008) propose a reference model for describing the

functional structure of dynamic web service composition systems based on existing platforms and prototypes.

On the other hand, other researchers (Boszormenyi et al., 2003) are focused on content adaptation; a typical example of content adaptation is changing the service presentation depending on the context data. The data properties can be modified in order to adapt the service according to terminal capabilities, network capabilities and/or even user preferences.

To sum things up, existing approaches are mainly based on schema evolution and instance adaptation. Adaptation in schema evolution approaches, may affect all process instances. That is a good practice if the process schema is inadequate. The main problem is how to handle process changes when they are needed only in a single instance.

The main contributions of our approach are both the mapping from a business process definition to a set of rules, and the easier dynamic adaptation of a process instance structure, by replacing or removing the rules that define the transitions among activities. In our approach, adaptation is only required when context information causes workflow changes. For instance, let us suppose that in a process instance, a new activity A3 is inserted between activities A1 and A2, because of new context information.

3 Business context rules

The representation of user's context data is usually composed by many variables related to user's device and data (e.g., location, time, contacts, presence, agenda, type of used device, network status). In order to flexible insert and withdraw context-related rules in the business process, we propose to use a rule-based notation (Rosenberg and Dustdar, 2005) to represent the business process structure, and thus a rule engine to execute the business process defined.

The possibility given by a rule engine of adding and removing rules at run-time in an easy and safe way makes dynamic business processes a more feasible option. Adaptation of a context-rule business process involves the ability to perceive the status, attributes and changes of relevant elements in the environment, and execute an action according to the actual context. These actions involve the adaptation of the business process depending on different factors (e.g., user situation, network capabilities, device performance). Dynamic business process must provide proper logic to apply reasoning based on the activities state.

In mobile and Web-based systems, important adaptation aspects are mainly related to network variations (e.g., bandwidth, latency), hardware variation (e.g., screen size, buttons) and software variations (e.g., memory capacity, installed applications). According to Schou (2008), business process adaptation can be based on: Technology: (e.g., display size and resolution, memory, CPU power, network bandwidth), activity behaviour (e.g. the user's location or task), user interface, user's conditions and presentation (e.g., contents presented depending on user's profile (Nivala and Sarjakoski, 2004)).

Web services are usually selected and composed based on their reputation. In general, the reputation of a web service is computed using the information provided by the user (Atrey et al., 2008). In business process adaptation, inferred

information is useful to apply modification policies in the business process instances, like adding, deleting or just modifying a specific business process block. By identifying the current user activity (e.g., working, cooking or running), the system can adapt the business process instance according to the context retrieved. A rule engine, like JESS (Friedman-Hill, 2008), may detect that a specific situation has occurred and raise an event or create higher knowledge level at runtime.

4 Dynamic rule-based business process

Decomposing the initial business process structure in a set of rules is a procedure based on pattern identification. This approach consists of the next phases: mapping of business process to rules and adaptation of the business process according to the context data.

The first phase is executed to provide a representation of the initial business process definition in terms of rules. The second phase is applied to provide a reliable workflow process modification. This phase is in charge of providing a way to express transitions between activities as a set of Event Condition Action (ECA) rules (see Table 1). The event part provides a way to know when an activity (or more) has finished its execution. The condition part is useful to verify which workflow path is enabled, according to the boolean result. The action part determines the next activity that has to be carried out.

Table 1 Mapping from workflow pattern to ECA rules

Workflow pattern	*Variables*	*ECA rule*
Serial		When EndOf(A) Then Start.Activity(B)
AND-Join		When EndOf(A) AND EndOf(B) Then Start.Activity(C)
AND-Split		When EndOf(A) Then Start.Activity(B) AND Start.Activity(C)
OR-Join		When EndOf(A) OR EndOf(B) Then Start.Activity(C)
OR-Split	with a,b as booleans	When EndOf(A) Then if(a==true) Start.Activity(B) if(b==true) Start.Activity(C)
OR-Join/OR-Split	with a,b as booleans	When EndOf(A) OR EndOf(B) Then if(a==true) Start.Activity(C) if(b==true) Start.Activity(D)
Iteration	with a,b as booleans	When EndOf(A) Then if(a==true) Start.Activity(A) if(b==true) Start.Activity(B)

In order to represent a business process as a set of ECA rules, the business process definition file is defined in a AML-based format, and further translated into a data structure. After that, the patterns are identified by analysing every node in the generated structure. Finally, the business process is represented as a set of ECA rules.

Many algorithms and models have been proposed to provide adaptation when a business process modification is executed at runtime. In Smari et al. (2006) authors propose an approach to handle changes in dynamic workflow systems. Their approach performs modifications that must be made in the current workflow, and based on the results, a new workflow is generated.

We introduce a technological approach based on Goix et al. (2007), in order to implement business process adaptation at runtime. Mobile devices and Web-based applications are used by service providers to retrieve user's data, in order to mainly establish habits and preferences, but not limited to.

5 Use case

In this section, we illustrate our approach through an example related to an e-commerce transaction. This section is divided in two subsections: the first one shows the process execution without implementing our approach, and the second one illustrates our approach in action.

5.1 Static process execution

Figure 1(a) shows an e-commerce business process definition. Figure 1(b) illustrates the execution of the business process when: the order is correct, the credit is enough, the product is not InStock. Figure 1(c) illustrates the execution of the business process when: the order is correct, the credit is enough, the product is InStock.

The static execution of the e-commerce transaction is defined by the next set of activities:

1 The customer issues a purchase order request.

2 The workflow automatically receives the order, checks the items and judges whether it is correct. If not, the customer will be notified to make a modification.

3 When the order is confirmed, the credit check is carried out. If the credit is poor, the order would be denied. Otherwise, the system will accept the order request.

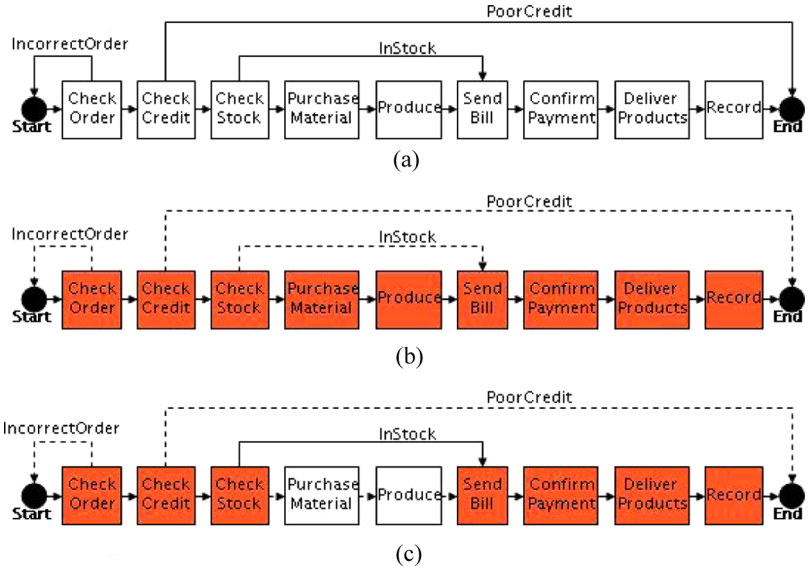
4 If the credit is enough, the system will check the stock in the plant. If there is not enough stock available, it will purchase the material and carry on production.

5 After that, the system sends the bill to the customer.

6 Before delivery, it waits for payment to be performed.

7 Finally, the process terminates after recording the whole operation.

Figure 1 (a) E-commerce business process definition, (b) and (c) e-commerce business process execution (see online version for colours)



5.2 *Our approach in action*

Once the business process has been defined, the first phase of our approach is executed (i.e., mapping transitions into a set of ECA rules). This phase identifies the patterns in the business process by analysing every node of the data structure generated. According to the sequence of activities, the mapping to ECA rules is executed (see Table 2).

Once the customer has submitted the order, an instance of the business process is created and the activity 'CheckOrder' is executed. Relevant current data can be used by the rule-based business process to enable a contextual decision to be made.

Let us suppose the rule-based system has established that the customer is: a VIP customer according to his profile and not reachable by e-mail at home but only via SMS. The changes applied by invoking the context-rule adaptation phase, are:

1 The rule-based system does not need to determine the credit for VIP clients. So the ECA rule that defines the transition 'CheckOrder-CheckCredit' is modified to allow straight communication between the activities 'CheckOrder' and 'CheckStock'.

2 The rule-based system dynamically adds to the business process instance: 'ConfirmPaymentBySMS', 'DeliverProducts' and 'UpdateCredit'. To allow this modification, the ECA rules that define the transitions 'SendBill-ConfirmPayment' and 'DeliverProducts-Record' are modified.

In order to execute the dynamic changes (e.g., task substitution, insertion, deletion) new ECA rules are added and other ones just modified (see Table 3).

Figure 2 illustrates how the business process instance is modified at run-time, according to the context information retrieved from the customer.

Table 2 Initial mapping from workflow pattern to ECA rules

Workflow pattern	*Variables*	*ECA rule*
R1	Boolean bOrder	When EndOf(CheckOrder) Then if(bOrder==false)Restart() else Start.Activity(CheckCredit)
R2	Boolean bCredit	When EndOf(CheckCredit) Then if(bCredit==true) Start.Activity(CheckStock) else End()
R3	Boolean bStock	When EndOf(CheckStock) Then if(bStock==true) Start.Activity(SendBill) else Start.Activity(PurchaseMaterial)
R4		When EndOf(PurchaseMaterial) Then Start.Activity(Produce)
R5		When EndOf(Produce) Then Start.Activity(SendBill)
R6		When EndOf(SendBill) Then Start.Activity(ConfirmPaymnt)
R7		When EndOf(ConfirmPayment) Then Start.Activity(DeliverProducts)
R8		When EndOf(DeliverProducts) Then Start.Activity(Record)
R9		When EndOf(Record) Then End()

Two dynamic modifications have been performed at runtime. They were carried out according to user-context information. Our approach provides a convenient way to apply dynamic changes in the business process instance; such changes come from the underlying rule-based system, which analyse context data to define new rules.

6 Approach validation

In this section, we validate our approach through both: execution performance and dynamicity. The time employed to perform specific activities is a very important characteristic. When the mentioned validation is performed, it is relevant considering characteristics related to the operation complexity in terms of execution time, and efficiency in terms of runtime business process modifications.

6.1 Business process execution performance

In this section, we illustrate our approach through an example characterised by a set of activities that perform recursive mathematical operations. It was necessary proposing a specific validation to verify whether the execution time decreased significantly or not. Execution time-based tests have been carried out to evaluate the performance of our approach. The business process definition is characterised by eight activities that perform recursive operations based on a typical Fibonacci algorithm (see Table 4).

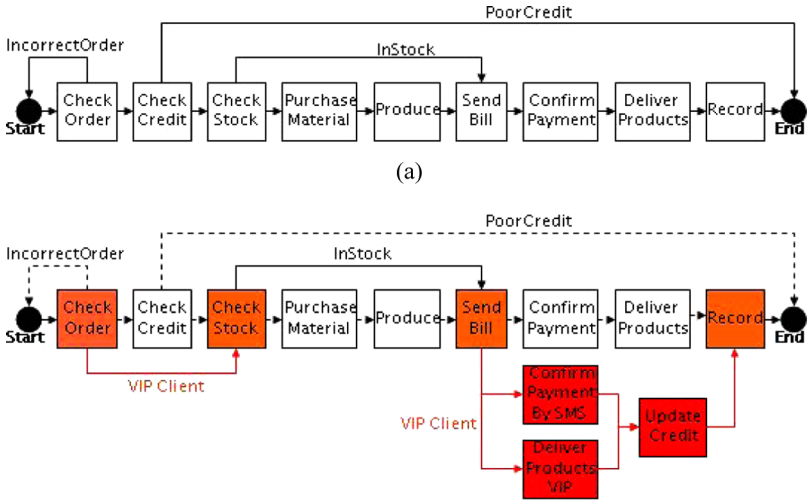
Table 3 ECA rules updated

Workflow pattern	*Variables*	*ECA rule*
R1	Boolean vipClient	When EndOf(CheckOrder) Then if(vipClient==true) Start.Activity(CheckStock) else Start.Activity(CheckCredit)
R2	Boolean bCredit	When EndOf(CheckCredit) Then if(bCredit==true) Start.Activity(CheckStock) elseEnd()
R3	Boolean bStock	When EndOf(CheckStock) Then if(bStock==true) Start.Activity(SendBill) else Start.Activity(PurchaseMaterial)
R4		When EndOf(PurchaseMaterial) Then Start.Activity(Produce)
R5		When EndOf(Produce) Then Start.Activity(SendBill)
R6	Boolean vipClient	When EndOf(SendBill) Then if(vipClient==true){ Start.Activity(ConfPaymentBySMS) AND Start.Activity(DeliverProductsVIP)} elseStart.Activity(ConfPayment)
R7		When EndOf(ConfPaymentBySMS) AND EndOf(DeliverProductsVIP) Then Start.Activity(UpdateCredit)
R8		When EndOf(UpdateCredit) Then Start.Activity(Record)
R9		When EndOf(ConfPayment) Then Start.Activity(DeliverProducts)
R10		When EndOf(DeliverProducts) Then Start.Activity(Record)
R11		When EndOf(Record) Then End()

Table 4 Activities

Activity	*Operation*
A	Fibonacci(10)
B	Fibonacci(15)
C	Fibonacci(20)
D	Fibonacci(25)
E	Fibonacci(30)
F	Fibonacci(35)
G	Fibonacci(40)
H	Fibonacci(45)

Figure 2 (a) Initial e-commerce business process definition and (b) context-based adaptation of business process instance (see online version for colours)



In order to represent the whole business process in terms of rules, we examine the system and extract the business logic that tended to be volatile. The rules extracted from the process definition are loaded and activity 'A' is performed (see Table 5).

Table 5 Mapping business process transitions from workflow pattern to ECA rules

Workflow pattern	*ECA rule*
R1	When EndOf(Activity(A)) Then Start.Activity(B)
R2	When EndOf(Activity(B)) Then Start.Activity(C)
R3	When EndOf(Activity(C)) Then Start.Activity(D)
R4	When EndOf(Activity(D)) Then Start.Activity(E)
R5	When EndOf(Activity(E)) Then Start.Activity(F)
R6	When EndOf(Activity(F)) Then Start.Activity(G)
R7	When EndOf(Activity(G)) Then Start.Activity(H)
R8	When EndOf(Activity(H)) Then End()

Each activity performs recursive operations, with different complexity levels, to simulate several situations where computing operations require more resources. In order to compare our approach with the mentioned workflow engine, we validate it by considering the time employed to complete the whole business process

execution. In order to obtain an acceptable data validation, it was necessary repeating ten times the same test. The execution time average, using our approach, is around 578,00 ms. The workflow engine employs around 15.687,00 ms to execute the business process proposed. It means 96.31% more time employed to execute the whole business process. Taking into account the results in Table 6, it is possible identifying the time average employed between activities.

Table 6 Execution time(ms) comparison

Transition (*start-start*)	*Rule engine* (*drools*)	*Workflow engine* (*bonita*)
A-B	16,00	0
B-C	31,00	15,0
C-D	31,00	0
D-E	47,00	16,0
E-F	282,00	140,0
F-G	31,00	1344,00
G-H	31,00	14172,00

Figure 3 illustrates the time employed to execute the defined business process activities by comparing our approach with the mentioned workflow engine.

6.2 Business process dynamicity

Based on Moody and Hillegersberg (2008) criteria, we have considered some characteristics in order to evaluate the flexibility according to flexible dynamic changes.

WS-flow languages such as BPEL are not flexible enough because they are still statically defining the WS types they use. BPEL focuses exclusively on the executable aspects of the process and does not contain elements to represent the graphical aspects of a process diagram. There is no standard graphical notation for WS-BPEL, in that way some vendors have invented their own notations.

On the other hand, approaches based on Petri Net elements capture quite different behaviour depending on the number of incoming and outgoing arcs. Since Petri nets is a language that uses only two different symbols, three relevant problems for modelling workflow processes have been found:

1 There is no specific support for patterns involving multiple instances and keeping track. Therefore, joining of instances is carried out by the designer.

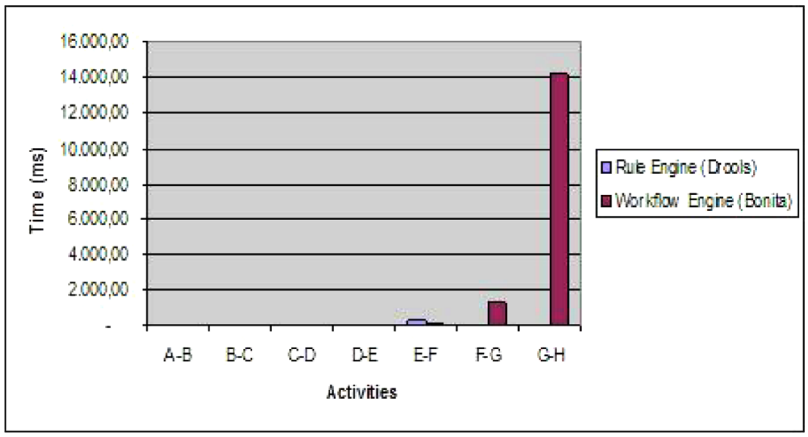
2 Advanced synchronisation patterns are difficult to model in terms of a high-level Petri net because the firing rule only supports two types of joins.

3 Removing tokens from various places without knowing where they reside.

YAWL is a workflow language that provides support for workflow patterns and dynamic adaptation of workflow models. Flexibility is provided through the concept of worklets, and supported by allowing a process designer to designate certain tasks to be substituted at runtime. A worklet handles one specific task in

a process activity. Any number of worklets can form the dynamic repertoire of an individual task.

Figure 3 Approaches comparison: execution time (see online version for colours)



Each task of a process is linked to a repertoire of actions and one of them is chosen at runtime to carry out the task. Dynamic substitution is based on data analysis and data inputs of the original workitem are mapped to the inputs of the worklet.

When the worklet has completed, its output data is mapped back to the original workitem, which is then checked back into the engine, allowing the original process to continue. This mapping operation does not provide enough flexibility to map the output to other activity in order to skip unnecessary business process operations. It means that dynamic components provided by YAWL, does not have the possibility to perform activities that are not associated with the specific repertoire of worklets.

The worklet selection enables flexibility only related to single activities by providing a process designer with the ability to substitute a workitem in a process at runtime with a dynamically selected worklet. The worklet is dynamically selected and invoked and may be created at any time, but it is not completely independent to provide dynamic changes in the business process structure.

The repertoire of worklets can be added to at any time but only provides dynamic ad-hoc process changes, without having the possibility to modify the original process specification. Variations in the business process specifications are very important to have a complete control of the way in which the business process should behave. For example, according to context data analysis by means of context-related rules, some specific business process would not need performing a block of activities that are not required into the business process anymore.

The worklet selection is a dynamic procedure into a static business process structure. This component is not flexible enough because does not provide the possibility to eliminate, avoid or skip blocks of specific activities at runtime. Instead the approach proposed in previous sections, supports dynamic modification of the business process structure by adding or cancelling context-rules that specify transitions between activities, assuming no dependencies between them.

7 Conclusions

Adapting a workflow process dynamically, according to context information retrieved from different kind of resources, has the purpose of providing flexibility, correctness and consistency. The key goal of rule-based systems is to provide relevant information and/or services based on current user context.

In this paper, we propose an approach that provides a reliable and more flexible way to handle dynamic business process changes based on context rules. The business process is represented in terms of rules, in order to provide more flexibility when a dynamic change is required.

The execution performance analysis has involved the time employed to executed a whole business process by using two different methodologies to represent the mentioned business process: ECA rules and workflow engine notation. Considering our approach, the time employed to pass from some specific transition to other one is zero. It means a high performance related to the time employed to connect the end and start point of two different activities performed in sequence. These results, reveal a high efficiency in terms of rule-based process execution.

On the other hand, the execution time average is around 578 ms. It means 96.31% less time compared to the result obtained by using a workflow engine. In particular, we analyse some principles based on computing performance to determine how business processes should behave according to low or high performance, and how context information can be used to infer performance policies. These aspects are important for improving the efficiency of the systems. A big advantage of our approach is that structural checking does not have to be performed since the structural modifications are based on runtime rules extraction or addition.

We also discuss differences of process modelling approaches. In particular, we analyse YAWL and BPEL, and discuss their strengths and weaknesses. These aspects are of importance for learning a modelling language, creating and understanding models. Such approaches have been compared according to flexible dynamic changes and graphic notation. Based on our findings, we propose a preliminary evaluation of the effectiveness of these modelling languages for empirical studies. There are still many relevant factors to improve the business process efficiency, mainly related to new available features (e.g., percentage of CPU usage, temperature and technology), in order to reduce the execution time.

As future work, we see many ways in which this work can be extended, we would like to explore the robustness of the results across diverse platforms, in order to provide a bigger range of applications studied. There are many possible extensions of the presented work, we intend to extend the dynamic changes by implementing a full business process management system. Another interesting future work is the application of such rule-based representation to achieve composition of two or more business processes.

References

Akhil, K. and Zhao, L.J. (2002) 'EROICA: A rule-based approach to organizational, policy management', in Meng, X., Su, J. and Wang, Y. (Eds.): *Workflow Systems, WAIM 2002, Lecture Notes in Computer Science*, Vol. 2419, pp.201–212.

Atrey, P.K., Hossain, M. and El-Saddik, A. (2008) 'Association-based dynamic computation of reputation in web services', *International Journal of Web and Grid Services*, Vol. 4, No. 2, pp.169–188.

Boszormenyi, L., Hellwagner, H., Kosch, H., Libsie, M. and Podlipnig, S. (2003) 'Metadata driven adaptation in the ADMITS project', *EURASIP Signal Processing: Image Communication Journal*, Vol. 18, No. 8, pp.749–766.

Business Rules (2000) *Defining Business Rules, What are They Really?*, http://www.businessrulesgroup.org

Eid, M.A., Alamri, A. and El-Saddik A. (2008) 'A reference model for dynamic web service composition systems', *International Journal of Web and Grid Services*, Vol. 4, No. 2, pp.149–168.

Ellis, C., Keddara, K. and Rozenberg, G. (1995) 'Dynamic change within workflow systems', *COCS '95: Proceedings of Conference on Organizational Computing Systems*, ACM Press, New York, NY, USA, pp.10–21.

Friedman-Hill, E. (2008) *Jess, the Rule Engine for the Java Platform*, Sandia Labs, http/www.jessrules.com

Goix, L., Valla, M., Cerami, L. and Falcarin, P. (2007) 'Situation inference for mobile users: a rule based approach', *Workshop on Managing Context Information and Semantics in Mobile Environments (MCISME)*, pp.299–303.

Kareliotis, C., Vassilakis, C., Rouvas, E. and Georgiadis, P. (2009) 'IQoS-aware exception resolution for BPEL processes: a middleware-based framework and performance evaluation', *International Journal of Web and Grid Services*, Vol. 5, No. 3, pp.284–320.

Khriss, I., Lévesque, E., Tremblay, G. and Jacques, A. (2008) 'Towards adaptability support in collaborative business process', *International MCETECH Conference on e-Technologies*, IEEE Computer Society, Montréal, Canada, pp.34–45.

Marquet, B., Gustave, C., Lefebvre, A., Nemchenko, S. and Chassande-Barrioz, S. (2002) 'Secured services in a multi-tier architecture', *World Telecommunications Congress, WTC 2002*, Paris, France.

Moody, D. and Hillegersberg, J. (2008) *Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams*, Springer-Verlag, Berlin, Heidelberg.

Nivala, A. and Sarjakoski, L. (2004) 'Preventing interruptions in mobile map reading process by personalization', *Proceedings of the 3rd Workshop on 'HCI in Mobile Guides'*, Glasgow, Scotland, pp.13–16.

Reichert, M. and Dadam, P. (1998) 'ADEPT flex-supporting dynamic changes of workflows without losing control', *JIIS (Journal of Intelligent Information Systems)*, Special issue on Workflow Management System, Vol. 10, No. 2, pp.93–129.

Rosenberg, F. and Dustdar, S. (2005) 'Business rules integration in BPEL – a service-oriented approach', *CEC 2005: The 7th International IEEE Conference on E-Commerce Technology*, Munich, Germany, pp.476–479.

Schonenberg, M., Mans, R., Russell, N., Mulyar, N. and Van der Aalst, W. (2008) 'Towards a Taxonomy of Process Flexibility', *CAiSE'08 Forum*, Montpellier, France, Vol. 344, pp.81–84.

Schou, S. (2008) 'Context-based service adaptation platform: improving the user experience towards mobile location services', *ICOIN 2008: International Conference on Information Networking*, pp.1–5.

Smari, W., Donepudi, S., Kim, S. and McQuay, W. (2006) 'Efficient handling of changes in dynamic workflow systems', *CTS'06: International Symposium on Collaborative Technologies and Systems*, pp.440–449.

Stantchev, V. and Schröpfer, C. (2009) 'Service-level enforcement in web-services-based systems', *International Journal of Web and Grid Services*, Vol. 5, No. 2, pp.130–154.

Sun, P. and Jiang, C. (2008) 'Analysis of workflow dynamic changes based on Petri net', *Information and Software Technology*, Vol. 51, pp.284–292.

Tan, W. and Fan, Y. (2006) 'Dynamic workflow model fragmentation for distributed execution', *Computers in Industry*, Vol. 58, pp.381–391.

Xia, Y. and Wei, J. (2008) 'Context-driven business process adaptation for ad hoc changes', *IEEE International Conference on e-Business Engineering*, pp.53–60