Dynamic Context-Aware Business Process: A Rule-Based Approach Supported by Pattern Identification

Jose F. Mejia Bernal
Department of Automation and Information
Politecnico di Torino
Turin, Italy
+390115647087
jose.mejiabernal@polito.it

Paolo Falcarin
Department of Automation and Information
Politecnico di Torino
Turin, Italy
+390115647022
paolo.falcarin@polito.it

Maurizio Morisio
Department of Automation and Information
Politecnico di Torino
Turin, Italy
+390115647033
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 structure in a set of rules, like ECA (Event Condition Action) rules, each of them representing a transition of the business process, i.e. an edge of the business process graph structure. As a consequence the business process engine can be realized by reusing and integrating an existing Rule Engine. We are proposing a way for representing Dynamic Business Process in terms of Rules based on patterns identification. With this approach it is easy to apply on a business process instance both user-based personalization rules and automatic rules inferred by an underlying context-aware system.

Keywords

Business process, business rules, context-awareness, rule engine

1. INTRODUCTION

A business process defines the context and the logical relationships between activities, and also specifies both the order of invocations (control flow) and the rules for data transfer (data flow). Such activities are configured in order to produce a specific output and associated with specific objectives.

A business rule is a statement that defines or constrains some aspect of the business [2]. Business rules are usually expressed either as constraints or in the form “if condition then action”. They provide the means to express, manage and update different kind of components contained into the whole business domain. In this way, such rules have to be expressed in terms of the defined activities and further integrated with the process specification.

The definition of a process and business rules requires often specific skills, usually related to people with a high level of expertise in design and programming. Another common problem is the lack of flexibility related to: business rules derivation based on dynamic changes in the business context.

On the other hand, we have seen that many approaches are not completely dynamic and effective when we need to automatically modify (add or delete an activity) the business process instance according to the changes in the business process context.

To address these issues, we propose an approach focused on providing a more dynamic and flexible adaptation of the business process. The main features of our approach include a suitable and flexible method for decomposing (based on patterns identification) and representing a business process structure in terms of rules; and a reliable method for executing dynamic business process adaptation, according to eventual modifications in the context information.

All this with the scope of increasing the flexibility and improving the adaptability of the system.

2. RELATED WORKS

In this section, we describe and discuss some commercial and academic solutions that provide support for adaptability.

In [5] a business-rule component provides support for rule sets ("if-then" rules) and decision tables. Selectors allow enhancing the behavior of business processes selectively based on business rules. A process can dynamically behave depending on business conditions.

There is a vast body of prior work on dynamic modification of business process at runtime. For example, [4] proposes concepts to manage the continuous changes in the processes. In another work [7], 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 [1] that extends the syntax of the ECA rules, but it does not provide an organizational rule modelling and enforcement framework for dynamic business processes.

Distributed workflow execution is characterized by separating one integrated workflow model into small partitions and allot them to different servers to be executed. [12] proposes a Petri net-based approach for dynamic fragmentation of a workflow model. Their approach divides the centralized process model while the process is executed. The fragments created can migrate to proper servers, where tasks are performed and new fragments are created.

Migration [11] is being considered one of the major issues of business process adaptability. Since migrating a process instance [6], from the original schema to a new one, involves a set of complex steps, we consider that such kind of solution could be very expensive in terms of memory, time and resources.

In [13], 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.

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 of this kind of adaptation is how to handle process changes, when they are needed only in a single instance.

These approaches differ from our approach in the next ways. Firstly, we propose a methodology for representing the business process in terms of rules. Since these rules do not depend on the business process definition, our approach provides more flexibility and facilitates the modification of the process instance, by replacing or eliminating the rules that define the transitions among activities. Secondly, they may unnecessarily adapt the process instance structure. In our approach, adaptation is only required when context information causes workflow changes. For instance, let's suppose that in a process instance, a new activity A3 is inserted between activities A1 and A2, because of new context information. This modification will be possible only after checking structural correctness between A1-A3 and A3-A2.

3. CONTEXT-AWARENESS AND RULES

One of the main issues in developing services and applications is the growing complexity in the interaction between the user and the many and more sophisticated functionalities and environments offered. To cope with this problem without sacrificing flexibility, context-aware systems aim at exploiting available knowledge about the user's situation in order to adapt their functionalities and provide an automated service personalization.

The representation of user's context data is usually multidimensional, as composed by many variables related to user's device and data, like: location, time, contacts, presence, agenda, type of used device, network status.

Different association-rules mining techniques are available to discover patterns of the form *context→activity*, expressing the fact that the user, in a particular context is likely to perform an activity. Such association rules can then be exploited in a wide range of personalization, recommendation and data analysis tasks.

A context-aware business process is then able to automatically adapt itself according to the user's context change. Such context-awareness can be implemented by defining a more complex business process, taking into account context-related rules in advance, and thus showing the same issues already faced when integrating business rules [3].

In order to flexible insert and withdraw context-related rules in the business process we propose to use a rule-based representation [8] to also represent business process, and thus a rule engine to execute the business process along with its related business rules and context-related rules.

The possibility given by a rule engine [9] of adding and removing rules at run-time in an easy and safety way leads to make dynamic business processes a more concrete option. Adaptation of a context-aware business process involves the ability to: perceive the status, attributes and changes of relevant elements in the environment; execute an action according to the current context; and select an action based on the possible future state of the system.

4. DYNAMIC BUSINESS PROCESS

Decomposing the initial business process structure in a set of rules is a process based on pattern identification. This approach consists of two 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.

4.1 Mapping Business Process to Rules

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 (jpdl/xml) is generated; and further parsed by a component defined as FileManager. The FileManager generates a data structure to represent the business process; each node of this structure contains the definition of every tag associated with the jpdl/xml file. After that, patterns are identified in the data structure by analyzing every node. Finally, the business process is represented as a set of ECA rules, according to the sequence in which the activities are suppose to be executed (see Figure 1).

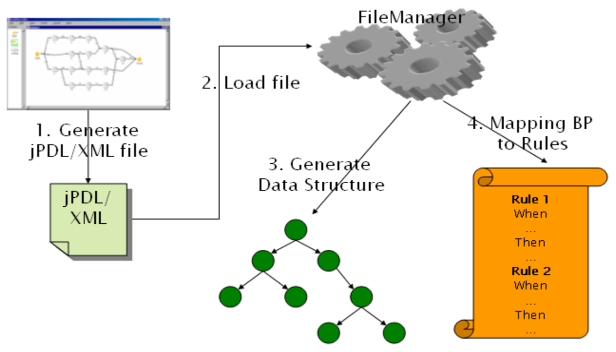
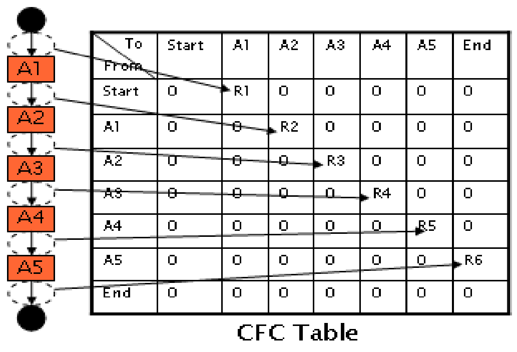


Figure 1. Mapping Business Process to ECA Rules

4.2 Context-Aware Business Process Adaptation

Many algorithms and models (specially based on petri nets) have been proposed to provide adaptation when a workflow change is executed at runtime. In order to represent the dependencies [10] between activities we have defined a Control Flow Checking (CFC) table to map the set of ECA rules that define transitions among activities (see Figure 2). This table provides a way to store any eventual modification of the business process instance.



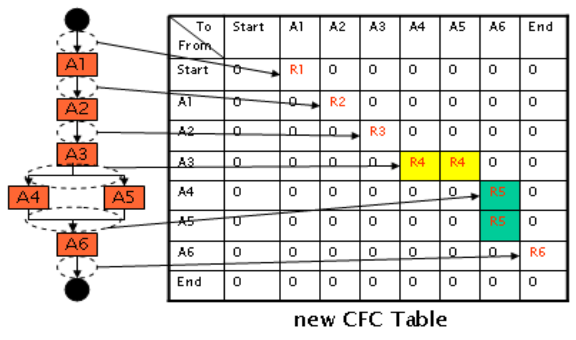
From \ To	Start	A1	A2	A3	A4	A5	End
Start	0	R1	0	0	0	0	0
A1	0	0	R2	0	0	0	0
A2	0	0	0	R3	0	0	0
A3	0	0	0	0	R4	0	0
A4	0	0	0	0	0	R5	0
A5	0	0	0	0	0	0	R6
End	0	0	0	0	0	0	0

Figure 2. Control Flow Checking Table

When some modification is executed, a new version of the CFC table is generated (see Figure 3). Adaptation is a process that involves the comparison of the new CFC with the old CFC table, in order to establish which transitions have to be modified.

Checking dynamic workflow changes is a procedure that: supports effective changes, reduces the costs associated, detects changes in active workflow instances and incorporates changes without shutting down the whole process.

This procedure generates the new CFC table; finds out modified rows by comparing the new CFC table with the old CFC table; checks possible structural inconsistencies in the new process by analyzing modified rows; and in case there is not any inconsistency, adds/modifies/deletes the rules involved in the changes.



From \ To	Start	A1	A2	A3	A4	A5	A6	End
Start	0	R1	0	0	0	0	0	0
A1	0	0	R2	0	0	0	0	0
A2	0	0	0	R3	0	0	0	0
A3	0	0	0	0	R4	R4	0	0
A4	0	0	0	0	0	0	R5	0
A5	0	0	0	0	0	0	R5	0
A6	0	0	0	0	0	0	0	R6
End	0	0	0	0	0	0	0	0

Figure 3. Control Flow Checking Table Updated

According to the information processed in every activity, the context-aware system is able to establish how the process instance could be dynamically modified. An example is propose in the next section to see in more detail how our approach works.

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 4(b) illustrates the execution of the business process when: the order is correct, the credit is enough, the product is not InStock. Figure 4(c) illustrates the execution of the business process when: the order is correct, the credit is enough, the product is InStock.

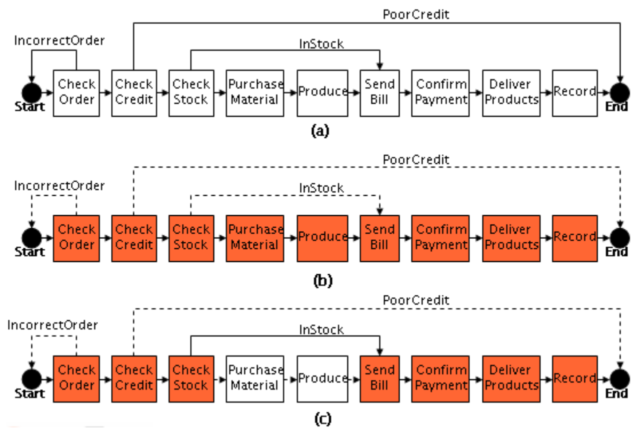


Figure 4. (a) E-Commerce business process definition, (b) and (c) E-Commerce business process execution

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

5.2 Our Approach in Action

Once the business process has been defined (Figure 5(a)), the first phase of our approach is executed (mapping transitions into a set of ECA rules). This phase identifies the patterns in the business process by analyzing every node of the data structure generated. According to the sequence of activities, the mapping to ECA rules is executed (see Table 2) and the initial Control Flow Checking table is created (see Table 3).

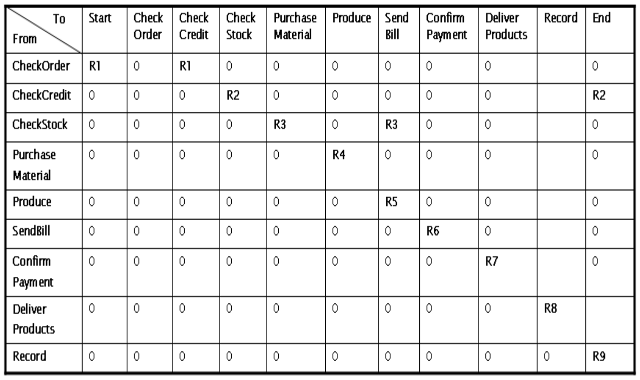
Table 2. Mapping from workflow pattern to ECA rules

Workflow pattern	Vars	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()

Once the customer has submitted the order, an instance of the business process is created and the activity "CheckOrder" is executed. Let's suppose the context-aware system has established that the customer is: a VIP customer according to his profile and not reachable by email at home but only via SMS. The changes applied by invoking the context-aware adaptation phase, are:

1. The context-aware 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 context-aware 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.

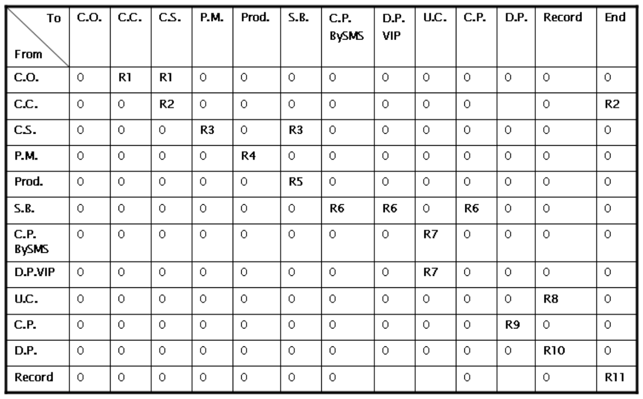
Table 3. Initial Control Flow Checking



To From	Start	Check Order	Check Credit	Check Stock	Purchase Material	Produce	Send Bill	Confirm Payment	Deliver Products	Record	End
CheckOrder	R1	0	R1	0	0	0	0	0	0		0
CheckCredit	0	0	0	R2	0	0	0	0	0		R2
CheckStock	0	0	0	0	R3	0	R3	0	0		0
Purchase Material	0	0	0	0	0	R4	0	0	0		0
Produce	0	0	0	0	0	0	R5	0	0		0
SendBill	0	0	0	0	0	0	0	R6	0		0
Confirm Payment	0	0	0	0	0	0	0	0	R7		0
Deliver Products	0	0	0	0	0	0	0	0	0	R8	
Record	0	0	0	0	0	0	0	0	0	0	R9

In order to modify the process instance, a new CFC table (see Table 4) is generated and compared with the old CFC table.

Table 4. Control Flow Checking Updated



To From	C.O.	C.C.	C.S.	P.M.	Prod.	S.B.	C.P. BySMS	D.P. VIP	U.C.	C.P.	D.P.	Record	End
C.O.	0	R1	R1	0	0	0	0	0	0	0	0	0	0
C.C.	0	0	R2	0	0	0	0	0	0	0		0	R2
C.S.	0	0	0	R3	0	R3	0	0	0	0	0	0	0
P.M.	0	0	0	0	R4	0	0	0	0	0	0	0	0
Prod.	0	0	0	0	0	R5	0	0	0	0	0	0	0
S.B.	0	0	0	0	0	0	R6	R6	0	R6	0	0	0
C.P. BySMS	0	0	0	0	0	0	0	0	R7	0	0	0	0
D.P.VIP	0	0	0	0	0	0	0	0	R7	0	0	0	0
U.C.	0	0	0	0	0	0	0	0	0	0	0	R8	0
C.P.	0	0	0	0	0	0	0	0	0	0	R9	0	0
D.P.	0	0	0	0	0	0	0	0	0	0	0	R10	0
Record	0	0	0	0	0	0	0			0		0	R11

Modified rows were found by comparing the new CFC table with the old CFC table. In order to execute the dynamic changes (task substitution, insertion, deletion, combination of these) new ECA rules are added and other ones just modified (see Table 5).

Table 5. ECA Rules Updated

Workflow pattern	Vars	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)} 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	Boolean vipClient	**When** EndOf(SendBill) **Then** if(vipClient==true){ Start.Activity(ConfirmPaymentBySMS) AND Start.Activity(DeliverProductsVIP)} else {Start.Activity(ConfirmPayment)}
R7		**When** EndOf(ConfirmPaymentBySMS) AND EndOf(DeliverProductsVIP) **Then** Start.Activity(UpdateCredit)
R8		**When** EndOf(UpdateCredit) **Then** Start.Activity(Record)
R9		**When** EndOf(ConfirmPayment) **Then** Start.Activity(DeliverProducts)
R10		**When** EndOf(DeliverProducts) **Then** Start.Activity(Record)
R11		**When** EndOf(Record) **Then** End()

Figure 5 illustrates how the business process instance is modified at run-time, according to the context information retrieved from the customer.

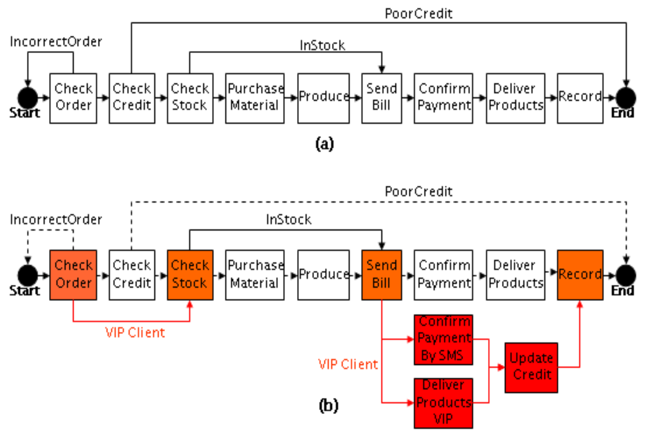


Figure 5. (a) Initial E-commerce Business Process Definition, (b) Context-based Adaptation of Business Process Instance

To sum things up, two dynamic modifications have been executed on the business process instance. They were carried out according to user-context information. Our approach provide a convenient way to apply to the business process instance, dynamic changes implicitly inferred by the context-aware system and supported by both: ECA rules mapping and business process adaptation.

6. CONCLUSIONS AND FUTURE WORK

Adapting a workflow process dynamically, according to context information retrieved from different kind of resources, involves critical phases that have the purpose of providing flexibility, correctness and consistency. The common idea in recent approaches is mainly related to provide new techniques that permits adaptation of process instances at runtime.

This paper provides a reliable and more flexible approach to handle dynamic changes (based on context data) in business process instances. The modification of a workflow process based on our approach consists of two phases: Firstly, the business process is represented in terms of rules, in order to provide more flexibility when a dynamic change is require. Secondly, the dynamic adaptation of the business process instance involves a reliable procedure based on the context information.

There are many possible extensions of the presented work, we intend to extend the dynamic changes by implementing a full business process management system by means of a rule engine, integrated with a context-aware subsystem. Another interesting future work is the application of such rule-based representation to achieve composition of two or more business processes.

7. REFERENCES

[1] Akhil K, Zhao LJ. EROICA: A rule-based approach to organizational, policy management. In: Meng X, Su J, Wang Y, editors. Workflow systems, WAIM 2002, Lecture notes in computer science, 2002, vol. 2419, 201–212.

[2] Business Rules Group, 2000. Defining Business Rules, What are they really?. Online at www.businessrulesgroup.org.

[3] Charfi, A., Mezini, M., Hybrid Web Service Composition: Business Processes Meet Business Rules. In *ICSOC'04, 2nd International Conference on Template Production*. New York City, NY, USA, 2004.

[4] Ellis, C., Keddara, K., Rozenberg, G., Dynamic change within workflow systems. In *COCS '95: Proceedings of conference on Organizational computing systems*. ACM Press, New York, NY, USA, 1995.

[5] IBM WebSphere Process Server, Features and Benefits, 2007.

[6] Khriss, I., Lévesque, E., Towards Adaptability Support in Collaborative Business Process, 2008.

[7] Reichert, M., Dadam, P., ADEPT flex -supporting dynamic changes of workflows without losing control. Journal of Intelligent Information Systems (JIIS). Special Issue on Workflow Management System, 1998, Vol.10, No. 2, 93-129.

[8] Rosenberg, F., Dustdar, S., Business Rules Integration in BPEL – A Service-Oriented Approach. In *CEC 2005: The 7th International IEEE Conference on E-Commerce Technology*, 2005, 476-479.

[9] Sandia, 2008. The Jess Rule Engine. On-line at http://herzberg.ca.sandia.gov/jess/

[10] Zhang, S., Wang, B., "The Research on Decision Approach of Data Dependence in Dynamic Workflow System". In *PDCAT'05: Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2005.

[11] Sun, P., Jiang, C., Analysis of workflow dynamic changes based on Petri net, 2008.

[12] Tan, W., Fan, Y., 2006, Dynamic workflow model fragmentation for distributed execution.

[13] Xia, Y., Wei, J., Context-driven Business Process Adaptation for ad hoc changes. In: *IEEE International Conference on e-Business Engineering*, 2008.