Dynamic Management for Business Processes Modeling & Execution in Workflows

Mati Golani matig@il.ibm.com

I.B.M Haifa Research Laboratory
ISRAEL

Information Systems Engineering Department
Faculty of Industrial Engineering and Management
Technion- Israel Institute of Technology

Abstract

Contemporary workflow-management systems cannot represent change or evolution of business processes. When a change is needed due to external reason, an offline procedure is invoked in order to create a new workflow engine template for the future instances in the workflow enactment module.

The standard interfaces do not deal with the business process metadata in a way that can actually change it as a reaction to inbound knowledge. There are many relevant cases, especially in the virtual enterprise arena, where the business process is not deterministic and is influenced by external parameters (such as the selection of virtual partners), so the knowledge of what should be done is available, however it is external to the system. There is a need to develop a modeling mechanism that enables to transfer process definitions in an automatic way, without the need for human interference. One way of confronting with these issues is the use of a rule-based engine to monitor business process execution. This engine will contain internal meta-rules that refer to metadata entities, i.e. rules that describe how to act on other rules (business process routing) when a change is detected, while executing all needed consistency checks.

1 Introduction

Workflow (WF) is a visual model of information flow and monitoring system that executes predefined actions at given predefined situations. The **sequence** of such actions (**activities**) defines a business process (BP). One or more agents can execute the business process. A significant reduction of cost as well as increase of efficiency can be achieved by managing these processes in an automatic or semi-automatic fashion, resulting in an improvement of product or service.

A reference model (figure 1) was developed on a generic basis by WFMC (workflow management coalition), in order to define various components of the workflow with emphasis on the functionality of each one, and the interfaces between them [wfmcD95]. The Workflow Enactment Service is a central module that acts as a hub to connect all other modules using APIs. This module is a software component that contains one WF engine or more that creates, manages and executes instances of WF. The enactment service module is responsible for the logical routing of activity execution, while the execution itself is done by other modules.

- Interface 1 defines the connection to the process-modeling environment, where all business processes are defined (build time).
- Interface 2 defines the connection to WF client applications, i.e. applications that provide an interaction with the user.
- Interface 3 defines similarly to interface 2, the connection to external applications, to be invoked by the WF system.
- Interface 4 defines the connection to other WF enactment services, in order to enable data exchange.
- Interface 5 defines the connection to the monitoring and administration module.

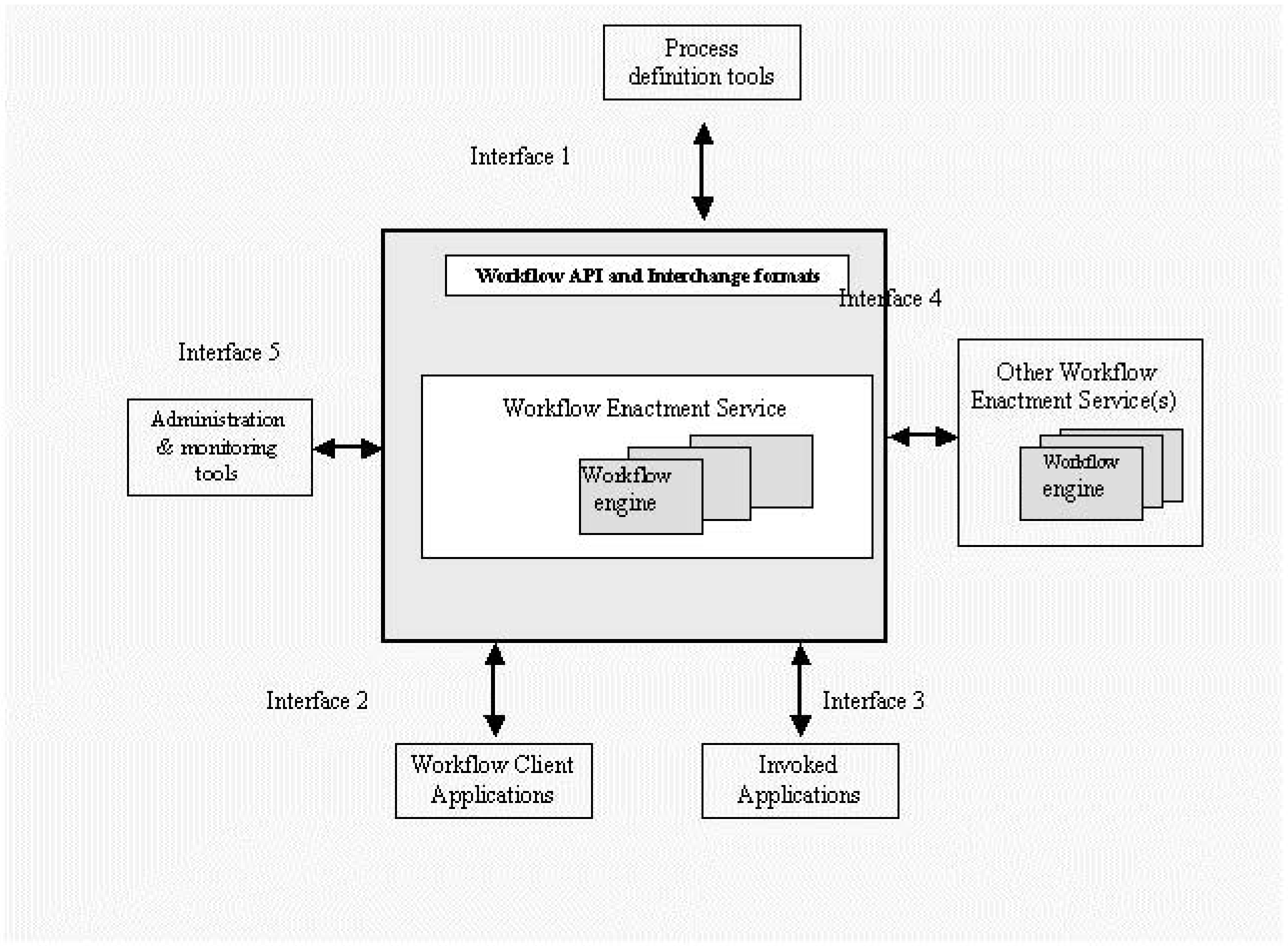
Process definition tools
Interface 1
Workflow API and Interchange formats
Interface 4
Interface 5
Administration & monitoring tools
Workflow Enactment Service
Workflow engine
Other Workflow Enactment Service(s)
Workflow engine
Interface 2
Interface 3
Workflow Client Applications
Invoked Applications

Figure 1: WF reference model design

Once the Business process model was defined (GUI, WF declarative language etc. – tool dependent) it is compiled and transferred by interface 1 to the enactment module.

1.1 Dynamic Updates

Observations show that in a dynamic environment organizations are required to constantly change their business processes, in order to survive the market competition and perform optimal usage of technological limitations. The change can be done on 3 levels:

Business process improvement: perform the same business process with higher efficiency. This usually involves organizational structure modification, and responsibility allocation.

Innovative modification: where the process deliverables remains the same, but the way of execution is redesigned.

Exception: The process modeling is based upon predefined entities that are processed through the BP execution (purchase orders, raw materials, invoices etc.). These entities have characteristics that may be updated and would effect the BP execution (in the case of exception). Furthermore, such exceptions can even cause the business process execution failure. For example, a material has a certain package weight that was updated to a higher weight, which in turn exceeds the weight limitation of the distributor for express deliveries, while the BP still expects this material to arrive on a special express delivery.

Standard commercial WF systems do not deal with model versioning, thus the current compiled version is the one to be executed. Research projects have dealt with this issue by defining types of update. Most of them refer to immediate/semi immediate types [D+98] [KCD99]. The most comprehensive one [SO98]partitioned the updates into these categories:

Flush	- all existing instances are permitted to continue executing according to the "old" version.
Abort	- all current instances are aborted.
Migrate	- all instances will perform future activities according to the updated version.
Adapt	- relates to identified exceptions and errors.
Build	- all instances are aborted, and start over again according to the updated version.

All the above mentioned (and more) described models and projects define manually the business process update. Classical WF system use the WF definition module in order to define such updates, and then compile it into a runtime version. More advanced systems (research prototypes) that combine active database technology as an integral part of the system, use the active module for deriving rules that reflect the business logic, and monitor its progress. These models can do a superb job until the time in which a change in the BP model is required. In this case there is a need to return to the build time module for reconstruction of the BP model and then runtime generation, that will derive the new rules to be monitored.

2 The Research question

The research question is twofold:
Construct a model that will enable dynamic automated modification on running WF systems; Resolve inconsistencies that may occur as a result of such modification.

2.1 *Motivation*

Virtual enterprises are formed in a competitive market as a result of the need to improve cost/performance ratio by cross-organizational cooperation. Cross-organizational operations require a dynamic and flexible mechanism that can handle the data flow between different partners [SN96][SN97]. Consider a virtual enterprise in which partner A delivers raw material k to partner B for further processing during an assembling process H, where the process H is dependent upon the nature of k. There may be a situation in which partner A, for some reason, has changed the characteristics of raw material k (exception, temporal change, permanent change). This should be reflected in process H (for example, there might be a need for machinery replacement). Once process H is active, a query should be sent to partner A, to receive such modifications. Alternatively, a message from A can be generated and sent to inform the responsible person at B. Notice that the update takes place in one enterprise, but defines the BP of another one.

In environments with a demand for fast response, an off line updating procedure decreases the efficiency of the organization and causes one of two depending on the update frequency:

- The system gives a good response time, but relies on possibly wrong data and knowledge.
- The system is reliable, but is occasionally in update status that prevents its fluent functionality.

A different attitude that may solve the problem is:

- The raw martial engineers (at partner A) update the local database regarding the new specifications.
- This update generates a rule with the “new” logic (materials, quantities, machinery, and routing).
- This rule is activated on partner A’s database, and is monitored since its definition time. Once a situation that relates to the relevant data is identified, the new data is applied, and the WF is respectively updated automatically.
- When a situation that defines the end of temporary modification is traced, the mechanism updates the WF system back to its initial state.

2.2 The Solution approach

2.2.1 Methodology

- Definition of a theoretical model for business process updates management and monitoring.
- Monitoring mechanisms definition for dynamic workflow processes. Several issues will be examined:
  Centralized/Distributed- although most WF system are categorized as centralized system that holds all the data, the option of a distributed monitoring mechanism should be considered, since the market is turning into a distributed environment work space (virtual enterprises etc.). In case the central option will be chosen, in order to get inbound data, it needs an interface with other external system mechanisms using the modification language (wrapped with XML).
  Combined data-metadata/fragmented- the meta data (for monitoring) can be an integral part of the database containing the “regular” data, or a an autonomous fragment
- Definition of BP structural modification language. This language will support the definition of the data/knowledge, which is required for BP modification. It will enable users/machine to represent the new BP process, including nodes (activities) and edges that were in the original BP (modified or erased).
- A verification process can run over the requested change, to ensure the modification authorization and correctness, using graph theory algorithms.
- Build a mechanism that uses the language, manages and monitors the processes (and activities) that are defined in the WF. The language will support composite event detection and WF oriented actions, definitions and modification (i.e. roles, activities, decision nodes, and routers). These

actions will be done automatically once the updated data is monitored, and the run time version will be updated.

2.2.2 The model components

Situation monitoring engine: there is a basic version at IBM Haifa Research Laboratory (HRL). It detects situations that are composed of simple events. It uses temporal, quantitative, and logical operators. It supports context driven filtration, and will be customized to support WF oriented events.
Database: a relational schema that holds the logical model of rules and situations.
Management & Monitoring Tool Module: this module is the main axis that connects and flows information among the other modules.
Process Definition Tool: A standard module in every WFMC standard supported WF tool, providing the option of assimilating it into the monitoring system. If needed, a standard compatible module that cooperates with the management & monitoring module will be built.

2.2.3 Design

The design schematics is shown in figure 2. This figure is based on the standard design (figure 1), and the darker "L" shaped area borders the related modules that are issued in this research. A functional core construction for event and activities detection, will support API 's launching.
Core XML wrapper for distributed connectivity for various applications.
Uses common interfaces according to WFMC. Interface 1 for Process definition tools, and interface 5 for Administration & monitoring tools. These two modules will be either taken from a WF vendor tool or there might be a need to create independent modules that use the standard interfaces (1 and 5).
The monitoring mechanism will be constructed of active database components, but the meta-rules structure is to be defined, considering two alternatives:

1. A model that is an extension of the classical ECA rule structure. This option is supported in most commercial databases, although its ability is quite poor when it comes to composite event definition. This option will reflect the business logic in rules that will spot events, and as a result will modify the WF structure.
2. A model that is based on the situation language *Amit* (IBM HRL) that was originally developed for middleware services monitoring [EA99]. Amit gets events as input and detects situations, which are composed of the events. Thanks to the large variety of operators, one can easily define a situation, which is quite impossible to define as a classical ECA rule. *Amit* language is currently not supported in commercial databases, but is a powerful tool for definition and detection of processes characterized with high complexity. Further more, *Amit* engine is platform independent.
3. A combination of both options (most probable solution), meaning: the use of active database for rule management and simple event detection. The situation manager will form an upper level, applied for definition and monitoring of complex situations.

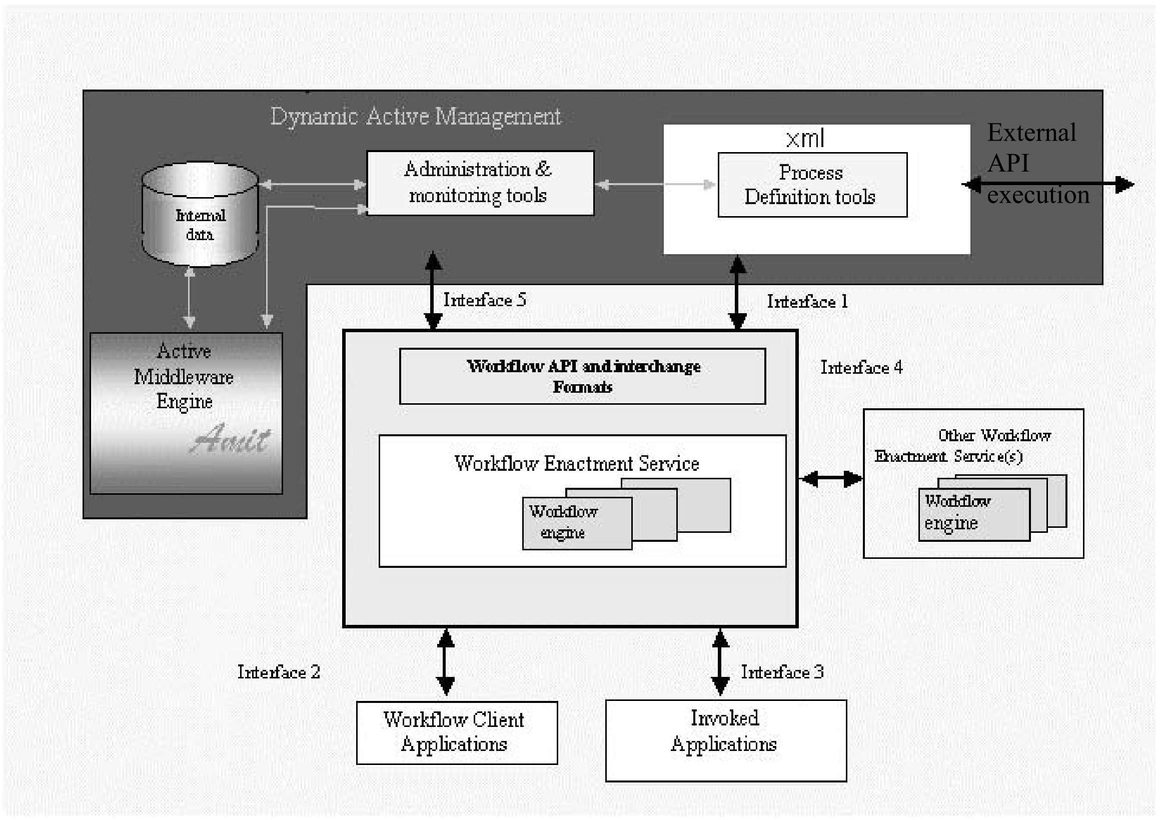

Figure 2: Dynamic Management Architecture Design

System data and fluent control data will be stored in relational schemes.
At this point some issues are raised.

- The perception of ownership/authorities/privileges regarding one changing others' business process definition.
- Inconsistency check to verify that the change has created a "connected" graph meaning there are no activities that are not connected to the path starting from the initial activity.
- Inconsistency check to verify that the final activity is always reached.

2.3 Case study

This case study shows the use of simple ECA rules to model and monitor BP execution through modifications. Figure 3 describes a 3 companies B2B network, where the retailer is the front end available to customers through the net, the orders are transferred to the manufacturer, and the distributor. Further data transfer that is not related to a new order is done directly between the manufacturer and the distributor. Such networks have the following characteristics:

- Business partners cooperate in some of their business processes (Amazon/FedEx/ Publishers etc.)
- The information system of each partner is not shared with the others.

- Knowledge that has been produced with in one partner's organization and affects the other partner processes is transferred to this partner.

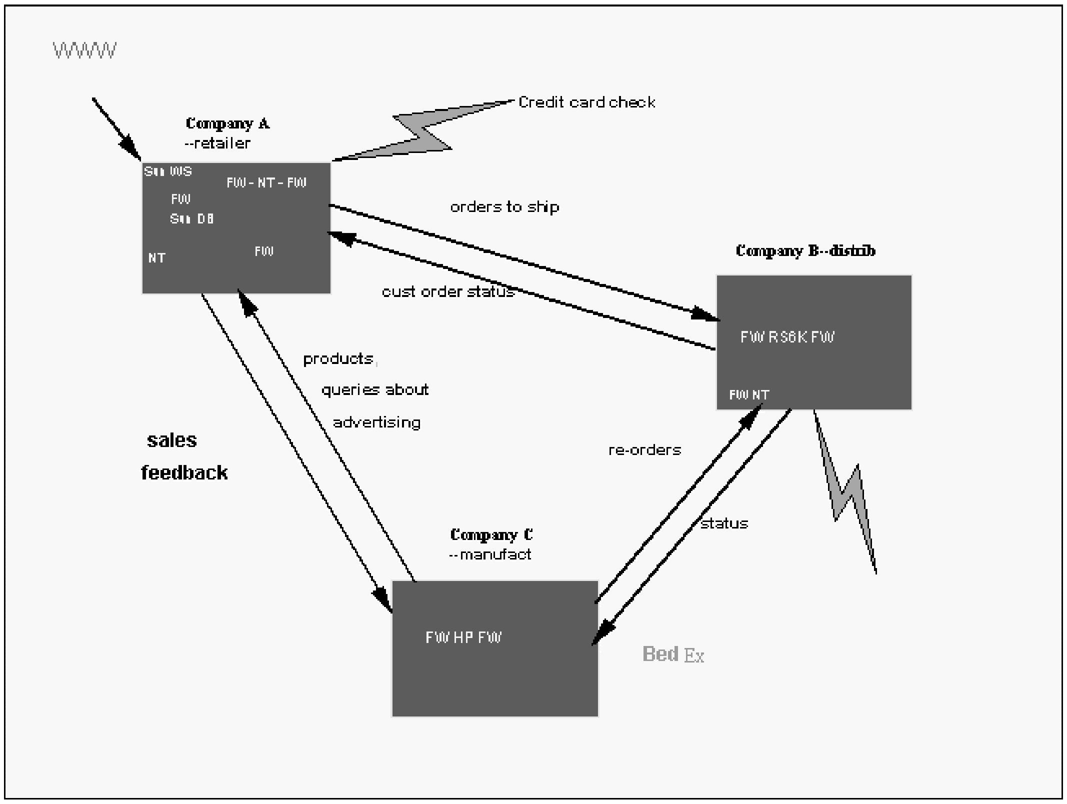

Figure 3: *an illustration of the interaction between business partners*

given the following scenario:

Tarzan, a Virtual book shop, is updated using common interfaces (EDI usually) that provide data which is relevant to current business processes activities, and is not relevant to undefined ad hoc activities.
BedEX – the distributor, informs Tarzan that for the next two weeks, all shipments to the Middle East are canceled (for new orders) and the shipping charges for the same destination for the following week are raised by 35%
This modification is visualized in figure 4. The update is performed as an off-line procedure… excluded from the monitoring system.

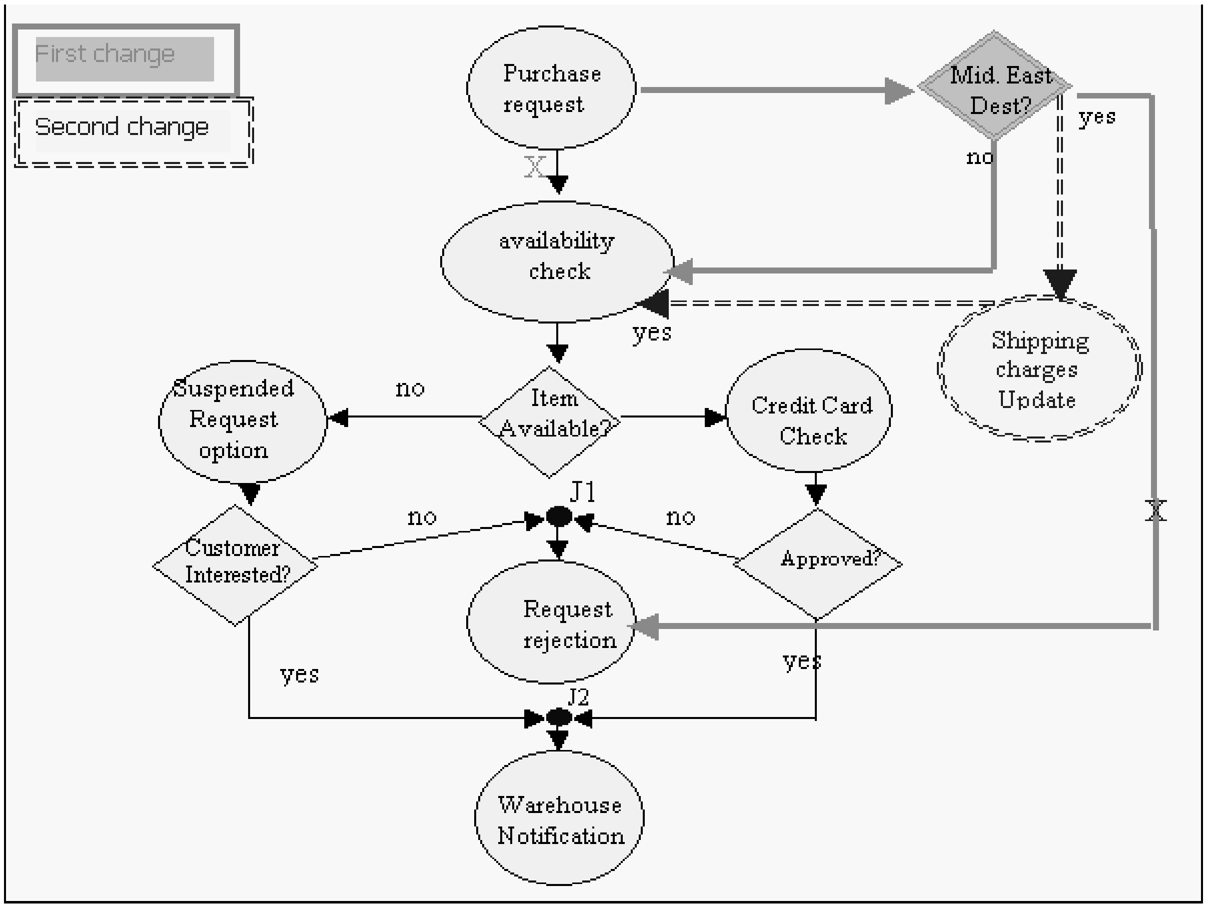

Figure 4: Order process flow in Tarzan

The database structure and content for modeling the business processes is as follows:

Table 1 defines the root task of each business process.

Table 1:Workflow (Wfname, Start task)

BookOrder	Purchase request

Table 2 lists the tasks related to each business process with assigned roles for each activity.

Table 2:Work task (Wfname, Wtname, role)

BookOrder	Purchase Request	Client
BookOrder	Availability Check	Auto
BookOrder	CreditCard Check	Auto
BookOrder	SuspendedRequestOption	Client
BookOrder	Request rejection	Auto
BookOrder	WarehouseNotification	Auto

Table 3 lists the routing nodes related to each business process, and defines their type.

Table 3:Routing task (<u>Wfname</u>, <u>Rtname</u>, type)

BookOrder	ItemAvailable?	Xor split
BookOrder	Approved?	Xor split
BookOrder	CustomerInterested?	Xor split
BookOrder	J1	Xor join
BookOrder	J2	Xor join

Table 4 lists the sequences segments of each BP. For cases of one activity follows another without splits or joins.

Table 4:Next (<u>Wfname</u>, <u>Task name</u>, Next task)

BookOrder	Purchase request	Availability check
BookOrder	Availability Check	Item Available?
BookOrder	Suspended RequestOption	Customer Interested?
BookOrder	CreditCardCheck	Approved?
BookOrder	Warehouse Notification	End
BookOrder	RequestRejection	End

Table 5 connects the routing nodes with the following tasks

Table 5:After fork (<u>Wfname</u>, <u>ForkTask</u>, <u>Next task</u>,Condition)

BookOrder	Item Available?	Credit Card Check	true
BookOrder	Item Available?	SuspendedRequestOption	false
BookOrder	CustomeInterested?	J2	true
BookOrder	CustomeInterested?	J1	false
BookOrder	Approved?	J2	true
BookOrder	Approved?	J1	false

This business logic is transferred into rules, which will monitor the system. The rules are ECA structured, meaning first an event is detected, then the condition is evaluated, and if it is true than the defined action is performed. Thus the business logic is maintained.

Initial BP Rules

Event	Condition	Action	Role
1 Purchase request(id)	True	Availability check(id)	Client
2 Availability check completed	Item.status=Available	CreditCardCheck	System
3 Availability check completed	Item.status!=Available	SuspendedRequestOption	System
4 SuspendedRequestOption	client.interested = true	WarehouseNotification (id)	Client
5 SuspendedRequestOption	client.interested =False	Request rejection(id)	Client
6 CreditCardCheck(id)	Result = OK	WarehouseNotification (id)	System
7 CreditCardCheck(id)	Result = Denied	RequestRejection (id)	System

The incoming input (knowledge) regarding the required change might be interpreted into meta data (knowledge monitoring) rules.
Suspend(1); - due to the 1st change (and the second change later on).

Updated rules table

Event	Condition	Action	Role
1 Purchase request(id)	True	Availability check(id)	Client
2 Availability check completed	Item.status=Available	CreditCardCheck	System
3 Availability check completed	Item.status!=Available	SuspendedRequestOption	System
4 SuspendedRequestOption	client.interested = true	WarehouseNotification (id)	Client
5 SuspendedRequestOption	client.interested =False	Request rejection(id)	Client
6 CreditCardCheck(id)	Result = OK	WarehouseNotification (id)	System
7 CreditCardCheck(id)	Result = Denied	RequestRejection (id)	System
8 PurchaseRequest	MidEastDest=True	RequestRejection (id)	Partner
9 PurchaseRequest	MidEastDest=False	AvailabilityCheck(id)	Partner
10 2 Weeks from now	True	Resume(1); delete(8,9,10);	System

A snapshot of the rules table after the 1st change

MetaData Rule
Suspended Rule

Updated rules table

Event	Condition	Action	Role
1 Purchase request(id)	True	Availability check(id)	Client
2 Availability check completed	Item.status=Available	CreditCardCheck	System
3 Availability check completed	Item.status!=Available	SuspendedRequestOption	System
4 SuspendedRequestOption	client.interested = true	WarehouseNotification (id)	Client
5 SuspendedRequestOption	client.interested =False	Request rejection(id)	Client
6 CreditCardCheck(id)	Result = OK	WarehouseNotification (id)	System
7 CreditCardCheck(id)	Result = Denied	RequestRejection (id)	System
8 PurchaseRequest	MidEastDest=True	ShipingChargesUpdate(id)	Partner
9 PurchaseRequest	MidEastDest=False	AvailabilityCheck(id)	Partner
10 ShippingChargesUpdate(id)	True	AvailabilityCheck(id)	Partner
11 3 Weeks from now	True	Resume(1); delete(8,9,10,11);	System

A snapshot of the rules table after the 2nd change

Thus, at any given time the rule table contains the logic of current desired behavior, while it may hold some more meta-data related to future changes. In a way it is similar to a snapshot table in a temporal database, but different by fact that the table here is dynamic and multiple versions are not maintained.

3 Research added value

All the current studies regarding dynamic process management deal with the change as an operation that is done manually by the user, and the discussion was on the effect on running WF instances, and future ones. Consequently, the issue in focus was that once the BP is updated, there are several ways the WF system may react, especially in cases where there are running instances of the ”old” BP model. Some studies that combine workflows with active databases, emphasized the conversion of BP logic into rules, in a static manner [DHL90],[K+95],[C+96], i.e., they referred to the given BP as a steady one, and modeled it with rules. These projects neglected the aspect of how a business process update is relevant to the area of active databases, although an active database may be a convenient infrastructure for that matter. Thus, using the given case study, all existing approaches would have modeled the BP in order to compile it to an updated run-time version that will react to the given changes.

The development of a mechanism that supports BP dynamic update performed on the runtime version is a significant extension to the use of active databases in the WF arena. It will manage business processes using active databases and will allow dynamic changes in business processes and automatic creation of runtime versions. This reduces dramatically user interaction and increases consistency and robustness of the system. In addition, a change, that it's origin in an event, and requires immediate response, is not applicable in classical workflow systems that demand the user's online interaction. Furthermore, event that encapsulates knowledge will allow update using this knowledge that is absolutely new to the system.

References

[C+96] F.Casati,et al.. Deriving Active Rules for Workflow Enactment. *Database and Expert Systems Applications. 7th International Conference, DEXA 96 Proceedings. 1996 XV pp. 94-115*

[D+98] A. Dogac, et al. Worflow Management Systems and Interoperability. *Springer-Verlag, 1998.*

[DHL90] U. Dayal, M. Hsu, R. Ladin . Orgenizing Long Running Activities with Triggers and Transactions. *Proc. ACM SIGMOD, 1990*

[EA99] O. Etzion, A. Adi. The Situation manager – an Application development tool for reactive applications, *IBM research Laboratory in Haifa, Isarel, Technical Report, 1999.*

[K+95] G.Kappel et al.. Workflow Management Based on Objects, Rules and Roles, *Data Engineering, vol. 18, n. 1 pp. 11-18, March 1995.*

[KCD99] P. Koksal, I. Cingil, A. Dogac. A Component-Based Workflow System with Dynamic Modifications, *Proc. of Next Generation Information Technologies and Systems (NGITS '99), Springer-Verlag, Lecture Notes in Computer Science, 1649, Israel, pp 238-255, July 1999.*

[SN96] L. Song, R. Nagi. An integrated Information Framework for Agile Manufacturing, *Industrial Engineering Research Conference proc. IIE Norcross, GA USA pp. 568-573, 1996*

[SN97] L. Song, R. Nagi. Design and Implementation of Virtual Information System for Agile Manufacturing, *IIE Transactions V 29 n 10 pp. 839-857, 1997*

[SO98] S. W. Sadiq, M.E. Orlowska. On Dynamic Modification of Workflows *Technical report, July 1998.*

[wfmcD95] The workflow reference model, *workflow management coalition, TC00-1003 issue 1.1 January 1995*