March 31, 2009

RT0850
Computer Science 21 pages

Research Report

Shared Management of Dynamic Business Process Extensions

Laurent Baduel, Hideki Tai, Takayuki Kushida

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Business Process Management Journal. , Vol. x, No. x, xxxx

Shared Management of Dynamic Business Process Extensions

Abstract

The global marketplace becomes more and more competitive, and business organizations need to team up and operate as a virtual enterprise to utilize the best of their resources for achieving their common business goals. As the business environment of an enterprise is highly dynamic, it is necessary to develop a workflow management technology that is capable of handling dynamic workflows across enterprise boundaries. This paper proposes a Workflow Extension Model (WEM) and a dynamic workflow management system of WEM for modeling and controlling the execution of multi-organizational business processes. WEM enables the explicit specification of dynamic properties associated with a business process model. It extends the underlying processes by adding connectors, conditions of application, extension process definition, and rules as its modeling constructs. Using WEM as the underlying model, the paper also describes the workflow engine which is extended by an extension service to trigger extensions during the execution of a workflow process to enforce business rules and policies and to adapt the process model at run-time.

Keywords:
Worflows and Process Extensions, Management, Dynamic Adaptations.

Introduction

Nowadays organizations compel workflow management in order to ease creation and execution of workflow so as to streamline business processes. They are often required to develop custom activities; for instance a multinational company must adapt its processes to comply with local laws and regulations, or a delivery service must change its way to ship materials depending on the weight, size and manipulation care. Adaptation and flexibility of those local processes are keys for business success. In this paper we will employ the term *business process* to identify a collection of related, structured activities or tasks that produce a specific service or product for a particular customer or customers. Business processes may describe IT, management, production, etc. operations. The term *workflow* indentifies a model to represent real work for further assessment. Workflow is a pattern of activity enabled by a systematic organization of resources, defined roles and mass, energy and information flows, into a work process that can be documented and learned. Workflows permit to implement business processes.

Most of existing workflow management solutions only handles static business processes [Apache ODE, 2007], [Hanson J., 2006], [MSWF, 2008]. Specific cases are then expressed as branches in a unique and global workflow process. This approach raises issues in terms of management and maintenance of large and complex error prone workflow processes, as well as extensibility and adaptation difficulty. Those companies would benefits a lot from the delegation of governance: global business processes should be defined at the coverage of company-wide while specific adaptations should be handled within a local organization. Such approach would avoid maintaining locally modified copies of the global process, which may lead to coherency losses between the multiple copies and the base process. We define an *adaptation* which is a modification of a business process making it able to handle new cases or providing new functionalities. Adaptation is a generic term as it does not specify which way it must be achieved.

We propose a workflow management method to extend existing processes, which we call as “base processes”, without changing them. It allows us to add multiple extensive processes conditionally to one base process. Those multiple extensions can be defined concurrently by different administrators. *Extensions* is the name we give to process parts designed to be plug onto regular business processes in order to make them adapted to new constraints or environments they should handle. There are three advantages for this approach. First the base processes are not copied nor modified to integrate adaptations. In that way losses of coherency between copies are avoided and creation of adaptations does not lead to a modification of the reference base process. Second, the multiple adaptations are independent from each others. The benefit is that one extension can be modified without concern about others. Third, the manageability of such system is its biggest value: by separating and sharing the administration of base processes and extensions we simplify the maintenance of such composed business processes.

Problem statement

Workflows adaptations may range from ad-hoc modifications of the process for a single customer to a complete restructuring for the workflow process to improve efficiency. Today's workflow management systems are often unsuitable to deal with workflow changes for several requirements. They typically support more or less idealized version of the preferred process. However, the real run-time process must be much variable than the process specified at design-time. For instance one may want to add or skip tasks in processes. An IT process whose role is to guide the installation of an operating system may be required to add some configuration steps like installing a specific language pack depending on the installation environment. Another process about the asset purchase may skip repeated approval task under specific conditions (e.g. limited cost and time emergency). The only way to handle changes is to integrate adaptation abilities within the system itself. If users are forced to bypass the workflow management system quite frequently, the system is more a liability than an asset. Therefore, we need to take up the challenge to find the innovative technique to add flexibility for workflow management without loosing the capability that is provided by today's systems.

There are two major types of adaptations: ad-hoc changes and evolutionary chanages. Ad-hoc changes are handled on a case-by-case basis. In order to provide customer specific solutions or to handle rare events, the process is adapted for a single case or a limited group of cases. Evolutionary changes are often the result of reengineering efforts. The process is changed to improve responsiveness to the customer or to improve the efficiency (do more with less). [Paul C. J. 2007] outlines that the trend is towards an increasingly dynamic situation where both ad-hoc and evolutionary changes are needed to improve customer service and reduce costs..

As depicted in the next section, for more than ten years different projects attempted to fill the need for adaptation of business processes. None of them succeed to become a reference, and be widely used in actual business IT systems.

We can point out two main reasons. The first and major one is the complexity of managing such systems. It is commonly tricky to guess which adaptation can be executed by which users. For instance within a process representing an insurance contract, hundreds of adaptations represent particular cases such as promotional campaigns, specific locations, contracts, agreements or time constraints. The overall resulting process is huge and hardly manageable. It is difficult to guess which adaptations might be triggered during a designated process instance. Moreover the creation of new adaptations may override, invalidate, or shortcut existing adaptations. The administration of such system becomes incredibly messy and is not acceptable in its current status for business. One of the main focuses of this paper is to provide a solution to dynamically adapt processes with an automated assistance to identify and sort adaptations. Management of adaptations is shared between several administrators, each group of them responsible for a set of adaptations it is skilled to deal with. It becomes easier to manage a great number of adaptations linked to processes.

The second reason is the total absence of validation mechanisms to ensure the correctness of the adapted process in most of exiting systems. During the edition of a new adaption the Workflow Extension Mechanism should ensure that important properties of the basic process are maintained after the process has been modified to enact the adaption. For instance, mandatory tasks should not be skipped or tasks that must be executed once and only once should not be allowed to be re-executed because of the adaptation. Such functionality to verify the correctness of a process at user level is offered by our system but is out of the scope of this paper that is centered on processes and adaptations management.

Related work

Workflow management is now widely accepted as the technology to support various business processes. Nowadays many commercial solutions for workflow systems are available. However, one may regret that a large majority of those workflow systems are restricted to centralized and internal use within the boundary of small organizations. They do not offer mechanisms to deal with runtime adaptation nor with shared administration of process definitions.
The Workflow Process Definition Language, WPDL for short [WfMC, 1995] [WfMC, 1999], has become the reference standard for workflow. WPDL has been originally developed by the Workflow Management Coalition (WfMC) in the first half of nineties. It offers textual grammar for the specification of process definitions and comes with a meta-model providing a set of modeling constructs for defining business processes. Typically a business process is represented by a set of inter-related activities connected by different kinds of connection lines. The activities represent tasks performed by human or computer that are related to a context (workflow data, environment, and operators). The connection lines that link those activities control the flow within a workflow process. Unfortunately, WPDL has been designed to model business processes only in the scope of a unique organization whose needs are uniform for all its parts. WPDL fails to provide a system allowing the creation of inter-organization or inter-division business processes. Along with evolution of IT industries and complex business interactions the need for the possibility to dynamically adapt processes to particular cases raised.

Several projects attempted to address this issue by introducing some dynamic capabilities to workflows. Their solutions appeared and can be categorized as follow:

- The *evolutionary changes* have a structural nature. From a certain moment in time, the process changes for all new cases to arrive at the system. Those changes may result from the definition of a new business strategy, the modification of a law, etc. It mainly consists in the modification of running process. Such modification of process is a global and does not allow specific adaptation.
- The *inheritance of workflows* defines a base process containing the necessary tasks of a process and allowing additional tasks. The base process identifies the sequences that can be extended. One can regret that it becomes impossible to extend a piece of the process that has not been originally intended to be extended.
- The *dynamic inter-organizational workflow management* provides complex extensions by modifying the base process itself at runtime depending on rules pre-defined by the user. Unfortunately, this approach does not permit a shared and concurrent administration of extensions. Moreover the extensions may be in conflict with each others.

Later several research studies were made to solve this issue. The approach chosen by some project was to define events and rules used to define and enact the control flow in a business process model. The most representative example among this solution is the EvE project [Geppert and Tombros, 1998]. The EvE project introduced a distributed Event-Condition-Action (ECA) rule-based enactment architecture. The use of events and rules to handle exceptions and failures during workflow execution falls into another category. In this category, events are produced outside a workflow execution, by either system environment or customers. Corresponding rules will be triggered when these events occur. An example is the WIDE project [Ceri et al., 1997]. The WIDE project uses a distributed architecture for workflow management based on a database management system. It is enhanced with rule evaluation capabilities in order to allow the definition of ECA rules to support exception handling and implement asynchronous behaviors during workflow execution. One may regret that this approach is very centralized and even so does not provide management mechanism in addition to the adaptation mechanism.

More recently the project AgentWork [Müller, 2002] is based again on the concepts of event and rules, but slightly different. AgentWork deals with dynamic adaptation appearing when the workflow instance encounters unexpected failures. The main difference between the approach of AgentWork and the ECA rules presented above comes from the fact that the rules and events are directly part of the workflow model. In order to allow adaptation at runtime one describes at edition time the adaptation to perform. The paper describes the given moments at which adaptations start and the synchronous or asynchronous nature of the events. During execution of workflow the system emits the events as described and activates the rules to trigger adaptation. A second major difference with ECA model is that the approach promoted by AgentWork is not restricted within a single organization as it events can be defined and managed in a distributed environment [Lee, 2000]. [Meng and al. 2006] names the AgentWork's approach ETR for Event-Trigger-Rule. Even if AgentWork deals with distributed management, it is unfortunate that no mechanism is introduced to share the management activities. They remain strongly centralized even if implemented on a distributed environment.

Several projects attempted to solve the dynamic adaptation problem using the Object-Oriented paradigm: [Zur Muehlen M. and Becker J., 1999], [Basten T. 1997], [Basten T. 2002], [Manolescu D. A., 2001a], [Manolescu D. A., 2001b], and [Sadiq, W. et al. 2006]. Thanks to dynamic bindings this approach could provide mechanisms to call a task instead of a similar one (an ancestor or interface task). The benefit of this approach was more to produce enterprise-oriented implementations of workflow engines for object-oriented languages than the actual flexibility it may have allow to adapt processes. The rigid architecture defined by inheritance limited the ability to freely modify a process. Moreover dynamic bindings may be quite unpredictable thus making static edition of processes and extensions a difficult task.

Lately a trend was to consider that a dynamic workflow management system should to be able to dynamically modify a workflow definition in order to adapt to dynamic business conditions and exceptional situations. [Reichert and Dadam 1998] presents a formal foundation for supporting dynamic structural changes of running workflow instances. [Muller and Rahm, 1999] describes a rule-based approach for the detection of semantic exceptions and for dynamic workflow modifications, with a focus on medical workflow scenarios. The work in the TAM project [Zhou et al., 1998] presents a dynamic restructuring of distributed transactional activities. These works mainly focus on the structural changes of process models. Finally, DynaFlow [Meng and al. 2006] supports both structural (e.g. drop an activity or bypass some activities) and semantic (e.g. replace an activity or modify a transitional condition) changes to an inter-organizational business process model. The 'Code Generation' approach, which is used to develop the workflow engine in DynaFlow, makes it easy to support these changes. The adaptability a workflow instance is enhanced with when these changes to the process model are performed dynamically by the business rules that are triggered by synchronous events posted by the running business process. By modifying the process itself at runtime DynaFlow permits a great flexibility however no management mechanism ensures validity or isolates extensions among them.

Despite some projects tried to address the problem of correctness or validity of a composed process [van der Aalst, W.M.P. , 1999], [van der Aalst, W.M.P. , 2003] [Kin, et al., 2004] none of the solutions presented above provide an integrated mechanism to seamlessly define, extend, adapt, and change a business process in a decentralized manner that hierarchically assigns specific modifications to specific environments and data. The projects presented here propose mechanisms to extend a process but do not offer a complete end-user oriented approach to manage such functionality. They miss both the concept of extension that can be easily created, attached, and remove from a process, and the technique to manage them within large organizations. One of challenages for the workflow management system was to offer to customers a simple, elegant, and flexible way to easily manage multiple adaptations efficiently implemented. Those will be done with the proposed management shared process extensions in the paper.

Extension

Proposal

Workflows systems are usually data-centered, i.e., every piece of work is related and executed for a specific data object. Examples of data object are an asset of a company, a person profile, a tax declaration, an order or a request for information.

Workflow Management Coalition (WfMC) uses the term "process instance" to denote the dynamic version of process definitions attached to an object and which need to be handled by the workflow management system (e.g. workflow engine). A task, also referred to as "activity" by the WfMC, is an atomic piece of work. Tasks are not specific for a single instance but are of course related to the object. In principle, a task can be executed for any process instance. In this paper, we use term of workflow as a way to implement a business process. Similarly the term extension denotes the implementation of an adaptation.

The proposed approach consists in the addition of a new component included within the workflow engine it self that describe how an adaptation must behave, under which constraints, and when it should stop. This new component comes with procedures to enhance management of adaptations.

Description of this solution
Our technology provides a mechanism to share the administration of process extensions. It allows multiple administrators (process extensions' editors) to deal with isolated and/or intersected (and included) extensions. We define that two extensions are *isolated* if they can not happen at the same time and at the same point during the execution of a process (i.e. they can not enter in conflict). When editing one extension its isolated peers should not be displayed to the administrators. On the other hand, *intersected* and *included* extensions may happen at the same time and at the same point during the execution of a process (i.e. they can enter in conflict). It is necessary to identify the extensions in intersection and let the administrator solve possible problems (by ordering extension for instance).
We propose a new technique to identify isolated and intersected extensions for processes. The goal is to simplify the edition of process extensions. When editing an extension on a process the administrator will be shown the entire set of extensions associated to the process (the entire contents of the *Extensions Table*). It may result in a very complex and large picture of the process. The editing method in this paper will select the necessary set of extensions to display regarding to a specific extension in order to ease the administrator's work. As extensions are only executed in accordance with their *Condition of Application* this takes an important role in our selection.

This mechanism takes place in 4 steps.

The first step consists in the definition of the *CoA* of a new extension. An administrator creates a new extension and starts by defining its *CoA*

In the second step the variables trees is built with values used in the *CoA*s of the existing extensions. For each variable we build a tree following the rules: a child node is a value included in it parent node and branches are disjoint values. In Figure 0 the black boxes, roots of the trees, represent the case "any". The nodes in the tree are conditions that refine the condition of their ancestors.

The third step is about the selection of displayed extension. Based on the *Conditions of Application* of the existing extensions ($CoAs_{ext}$) and of the new extension (CoA_{new}) the isolated extensions are hidden thus showing only extensions whose *CoA* may intersect with the new extension. The rules to decide if an extension is shown or hidden use the variables trees as follow:

- If CoA_{ext} and CoA_{new} belong to different trees, or
  If CoA_{new} is ancestor of CoA_{ext} (CoA_{ext} and CoA_{new} belong to a same tree),
  it is an **intersection** → show the existing extension

- If CoA_{ext} is ancestor of CoA_{new} (CoA_{ext} and CoA_{new} belong to a same tree),
  it is an **inclusion** → show the existing extension
- If CoA_{ext} and CoA_{new} belong to different branches of a same tree, it is an **exclusion (isolation)** → hide the existing extension

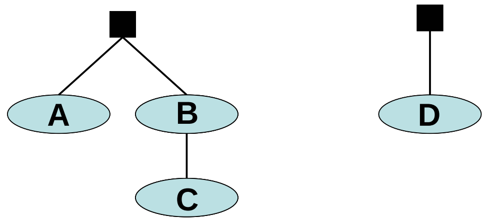


Figure 0 - The relationship in trees

Figure 0 represents two trees, the first one composed by three nodes A, B, and C, and a second containing a unique node D. The relationship among those nodes is as follows: D intersects A, B, and C as it belongs to a different tree. Symmetrically A, B, and C intersect D. B includes C as B is an ancestor of C. A is in exclusion with B and C: they belong to different branches of the same tree.

Finally during the fourth step the administrator completes the definition of the new extension. The administrator finishes defining the new extension (*entry-point, branching connectors, process, order, and exit-point*). The administrator uses the *order index* to solve possible ambiguity in execution order among extensions whose entry point is similar. *The order index* is a real number comprise between 0 (excluded) and 1 (excluded).
The administrator is also free to narrow the *CoA* of the new extension; in that case the step 3 will be re-executed in order to hide other extensions that enter in isolation to the narrowed *CoA*.

The *Extensions Table*, even if distributively managed, is a centralized set of information. This provides several benefits such as global validation or audit of the extensions. A super-administrator may perform automatic checks on the *Extensions Table* to verify that no locally defined extension is illegal regarding the base process and to verify the compliance with standards or rules that the company may follow.

Discussion of benefits

In summary, the Workflow Extension Model allows to provide adaptations that are evaluated at runtime. The extensions are created after a base process with which they are associated. The base process defines the general objectives and rules to achieve a complex task while extensions refine the base process in order to adapt the general case to specific, local, or particular cases. By opposition with many projects presented earlier in this article our approach of process adaption allows new paths in a business process and not only the replacement of designated tasks. Flexibility of such approach is of course greater.

The first advantage is to avoid any modification of the base process (at edition time and execution time). In that way processes remain a strong reference to understand the goal they achieve as they give the general outline of what will happen. Base processes are not copied. Multiple copies of a base process may ease the adaptation mechanism but present a risk for some copies to introduce differences within the base process itself rather than in the extensions. The possible loss of coherency between copies of base process is a great danger for management and correctness of business process.

The second advantage is the isolation of multiple and independent extensions. Each extension is independent from the others (except in the particular case of hierarchical or nested extensions). It means that each extension is self-sufficient and does not require modification on other extensions. This permits to freely modify one extension with limited care about the others. The mechanism we described above deal with possible interaction between extensions. One user who may follow several extensions on one process must be aware of them; the mechanism presented here helps him to filter which extension he/she is subject to execute depending on his/her environment of execution.

Thus the final and major advantage is the manageability it offers to operators of business processes. The separation in base processes and extensions allows keeping the base processes simple as they do not include anymore the entire set of adaptations. It is then much easier to add new extensions: the system will automatically select which already existing extensions to show to the operator creating a new one. The base process is shown with the subset of all its extensions that could be triggered in a common execution environment to the new extension attempted to be created. The separation in base processes and extensions also allows reducing the number of updates of process definitions. As the extensions deal with adaptations all the updates regarding adaptations are performed on the extensions thus maintaining the base process unchanged. It is valuable as extensions are locally managed and their modifications only impact their users. In a standard and globally integrated management of adaptations all users of a process suffers from all modifications of any of its embedded adaptations. This is a critical problem when a process is widely used and contains lots of adaptations. The proposed approach can avoid this problem.

Workflow engine, design & implementation

Principles and implementation

As presented in [Bergmann S., 2008] Workflow engines are generally implemented as state-machines, i.e. a model of behavior composed of a finite number of states, transitions between those states, and actions. In most of existing implementations we identify the main components of workflows engine as a *Process Table* that stores entire set of process definitions, a *Process Definition* Table that describes processes (transitions & tasks), a *Task Definition* Table that associates actual operations to process tasks, and a *Process Instance Table* that provides a view of running processes.

A workflow engine periodically selects a process from the Process Instance Table then looks up in the Process Definition Table to discover the next task to perform. Tasks are operated by human (enter data, approve action, etc.) or automated by computer (call to a web service, query on database, etc.). Then this procedure is repeated until the process instance reaches an end point in the process definition or raises some sorts of unhandled errors.

We have created a new table that we included in the core tables presented above. This new table describes the way an extension is attached to a process. It does not describe the extension itself; we relay on the standard process definition table to store the definition of extensions as well as the definitions of base process.

Entry point	*Condition of Ap.*	*Branching*	*Process*	*Order*	*Exit point*
Task 1	`OS_Type=Windows`	**unconditional**	**$E_{(1)}$**	**0.5**	Task 1
Task 2	`user.domain=Japan`	**parallel**	**$E_{(2)}$**	**0.5**	End
Task 2	`user.domain=USA`	**parallel**	**$E_{(3)}$**	**0.5**	End

Table 1 – The Extensions Table of the process shown in Figure 3

As shown in the example of Table 1, the Extension Table is basically composed of the following data fields:

(1) the *entry-point* from where in the base process the extension process must be started,

(2) the *Condition of Application* (*CoA*) that triggers the evaluation of the extension process. At edit time the *CoA* is used to detect which other extension must be shown or hidden to the administrator.

(3) the *type of connectors* to the base process that defines the branching semantic of the extension (conditional, unconditional,

parallel, etc.)

(4) the *extension process itself* describing the particular adaptation to some concern,

(5) the *order index* used to order several extensions that may happen at the same point,

(6) the *exit-point* where in the base process must be resumed after the extension process termination,

We also extend the Process Instances Table with a new field, the *Exit-points Stack*, keeping references to the exit-points when executing an extension process. The scope of our contribution to enact a workflow extension model is shown by the shadowed box in Figure 1.

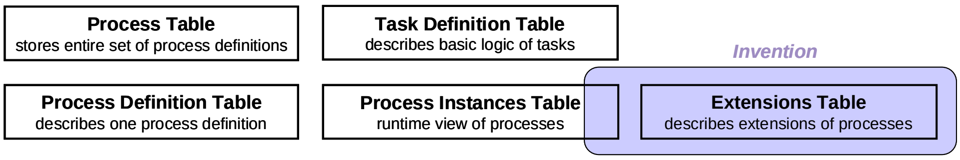


Figure 1 - The existing and the new

As shown in see Figure 2 the way a workflow engine operates becomes slightly different: after selecting a process in the Process Instances Table, the workflow engine picks up an extension process and identifies the next task to perform. If extension processes are found to be applied at this point (*entry-point*) they are sorted regarding their *order index* and the first extension whose *Condition of Application* is satisfied is executed in place of the regular task. The *exit-point* is registered in the Process Instance Table. This table handles the exit-points in a stack object in order to permit recursive extension. When the execution of the extension process ends, the base process is resumed at the exit point popped up from the stack. Bold steps (shadowed boxes) are parts of our contribution.

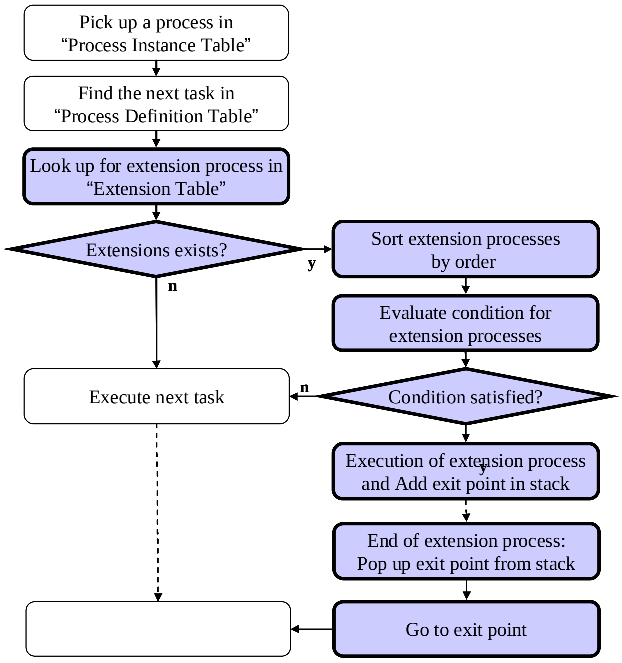


Figure 2 - The workflow engine flowchart

Example scenario

In the use case briefly introduced below (described in Table 1), the condition to evaluate the extension processes is based on some environment values such as `user.domain` and `OS_Type`. Figure 3 presents the base process associated with its extensions defined in the *Extensions Table* of Table 1.

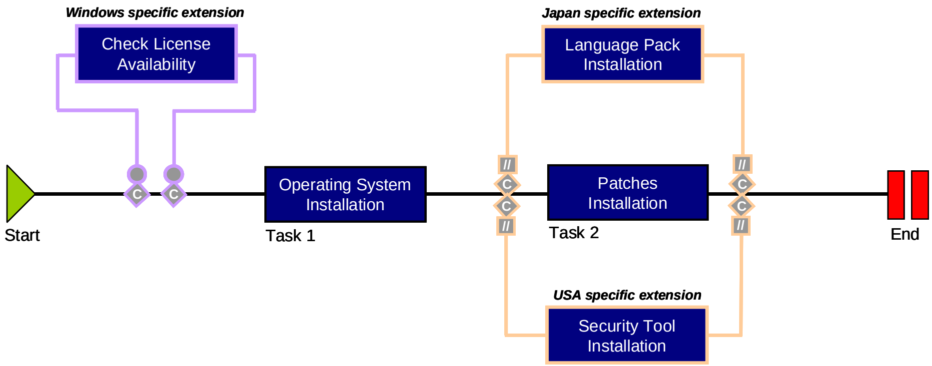


Figure 3 – An OS installation process with various specific extensions

The first extension ($E_{(1)}$) is unconditionally performed by all users whose domain is Japan. Two other extensions ($E_{(2)}$ and $E_{(3)}$) consist in the parallel execution of a new task along with the execution of a base process task.

We can consider to add new extensions on the process depicted in Figure 1, and will go through the 4 steps mechanism.

First administrators define the *Condition of Application* of the new extensions they intend to create. We define three administrators who independently want to add one new extension to the base process of Figure 3. Each of them defines a *CoA* as described in Figure 4. The first administrator, on the left, wants to define an extension related to users whose domain is Tokyo. The second administrator, in the center, is introducing an extension for users in America installing a Linux operating system. Finally the third administrator's will is to create an extension that will concern all users world-wide.

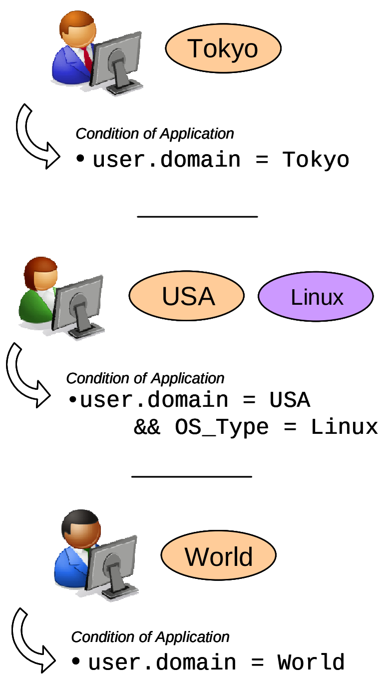


Figure 4 – Administrators define CoAs

Secondly, variables trees with values used in the *CoA*s of the existing extensions are built. Figure 5 shows possible variables trees for the process of Figure 1. The left tree can organize the different values associated with the `user.domain` variable within the extensions attached on the base process. The domains "Tokyo" and "Kyoto" are included in the domain "Japan". Similarly, "Austin" is included in "USA", "Japan" and "USA" are included in "World", and "World" is placed under the general root case "ANY". To build the trees we rely on the data definitions. We assume that variable type are defined along with some *comparable* interface or *comparator* object able to deal at least with *included* and *excluded* primitives.

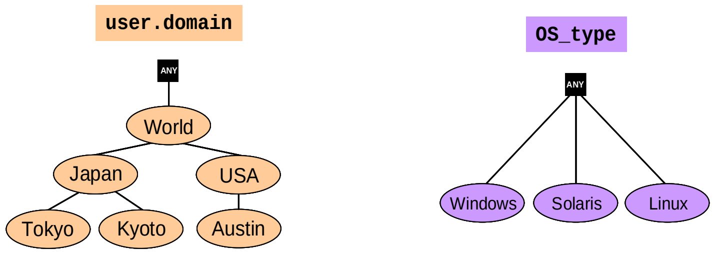


Figure 5 – Variables trees

The third step triggers the selection mechanism to identify which extensions must be shown or hidden to the three administrators. The rules to select the extension to show and hide are evaluated between the *CoA* of a new extension and the *CoA*s of existing extensions.
Finally as presented in Figure 6 each administrator has a different view of the base process and the existing extensions.

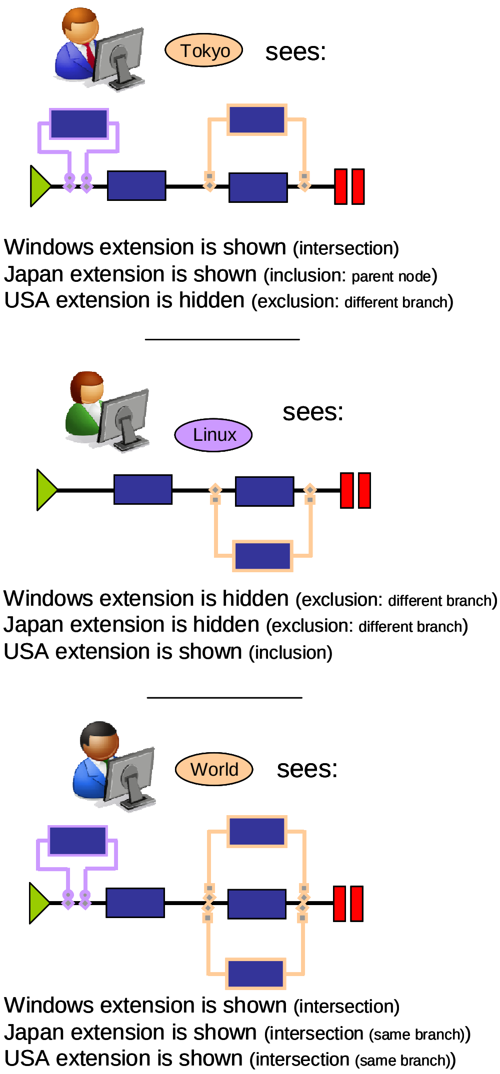
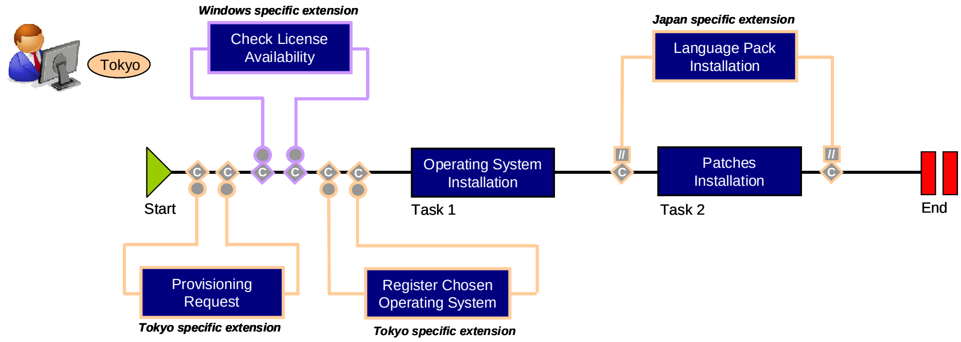


Figure 6 – Administrators' view of the base process and its existing extension regarding to the CoA they provide for a new extension.

In the fourth and last step administrators fully define the new extensions. Let's continue this scenario with the case of the most left administrator who defines a Tokyo specific extension. This administrator wants to provide two new extensions before the Task1 (OS installation) for "provisioning the request" and "register the OS". An ambiguity about the order to perform those two extensions and the existing one "Check license" appears. The administrators must properly set the *order indexes* to order those extensions. Figure 7 presents the final view of the process after the administrator complete to define the two new extensions.



	Entry point	Condition of Ap.	Branching	Process	Order	Exit point
	Task 1	`OS_Type=Windows`	unconditional	$E_{(1)}$	0.5	Task 1
	Task 2	`user.domain=Japan`	parallel	$E_{(2)}$	0.5	End
	Task 2	`user.domain=USA`	parallel	$E_{(3)}$	0.5	End
NEW	Task 1	`user.domain=Tokyo`	unconditional	$E_{(4)}$	0.3	Task 1
NEW	Task 1	`user.domain=Tokyo`	unconditional	$E_{(5)}$	0.6	Task 1

Figure 7 – The process view of the administrator after addition of two extensions and the Extensions Table.

Conclusion

The major benefit of our approach is an increased manageability of the business processes. Related works hardly deal with the administration of dynamic processes and only focus on the extension execution rather than considering the entire life cycle of processes and process extensions.

Summary of advantages

In order to give an evidence of the eased management introduced by our workflow extension model we will consider the case of an international organization whose offices are spread in many countries. The head office of this organization wants all its foreign offices to observe the rules and process it defines. On the other hand, those offices will ask for adaptations of the processes to comply with their local laws, customs, or business rules. In a workflow model that does not permit the creation of extensions the general administrator (or administrative group) receives unceasing requests for integrating local adaptations to the general processes. It commonly results in several issues: 1) the administrator may have a limited knowledge or understanding of all local constrains and so produce errors in the adaptation he provides; 2) in case of sudden peak of requests the central administrator role represents a bottleneck that delays the creation of adapted processes; 3) the global process resulting from all adaptations is very complex and error-prone; 4) moreover any new modification of an existing adaptation that regards only one local office has to be propagated to all local offices even if they are not concerned by this adaptation.

Evaluation

To make it more concrete, we show the example to track how many modifications and updates are necessary with and without our workflow extension model. Let's consider a company composed of 10 divisions (head office, foreign offices, financial department, IT department, etc.). The head office produces business mainlines under the form of base processes that all other divisions would follow. Each of those divisions is free to add its own extensions on top of the base processes. Once the set of processes and extensions has been created, we are interested in observing their life cycles.

Without a workflow extension model, the processes are common to all the company's divisions and completely integrate the entire set of adaptations. Adaptations usually take the form of conditional branching in the processes. The current result is that any modification required by a division, even for its exclusive purposes, is shared with the entire company in a process definition becoming more and more complex has the company size is growing. Moreover confidentiality may be an issue: for instance the IT division may not be aware of internal procedures of the financial division. In this configuration if a division, let's say the IT division, introduces numerous extensions and modifies them often the entire set of users, from all divisions, will have to suffer from the numerous migrations to the ever new versions of the globalized processes.

With the workflow extension model in this paper, the impact of modifications and updates are limited to the division that has produced them. Let's imagine the distribution of employees is equal within all the division. In the scenario presented above only 10% of the employees (the IT division out of 10) suffer from the migration of the processes to its new version. 90% of process migrations are avoided. Moreover only the persons who are concerned by a specific adaptation are concerned by the updates of this adaptation. It is easier for an IT guy to understand what is changing in the process he uses that it is for another guy who never uses this particular adaptation in a globalized process. The last advantage in moving adaptations out of a unique globalized process is to stabilize the base process. Frequent updates regarding adaptation and previously required on top of a globalized process are now performed on the extensions. It increases the life time of each version of the base processes, thus introducing a greater global stability in the set of business practices within the entire company.

In terms of software performances the addition of our extension model had no sensible impact on the load of the application. CPU usage does not vary as our implementation relays on existing mechanism and consists mainly in re-routing the flow rather than in the introduction of additional mechanisms. The memory usage remains constant because no dynamic value is stored in memory; the architecture model keeps all information in database. The data storage that maintains process definitions increases in an insensible way. Only the definitions of extension are added. The definitions of nodes and objects that consume much larger space are not changed. Finally the logic itself only introduces a lookup to find existence of extensions and if several extensions exists in the same point a sort. Those operations are negligible in the whole process: they represent less than 0.1% of the operations required for the flow to move from one task to another.

To conclude, by opposition to existing solution our innovative approach is much more dynamic and adaptable as extensions can be plugged or removed without modification or copies of base process. This is permits by the introduction of a new table within the core of workflow engine along with a mechanism to help shared and distributed management. The governance of final workflows is shared between different entities thus dividing the complexity of base workflow. The edition of new extensions is eased by hiding existing extensions in isolation. Finally, the *Extension Table* centralizes the information about all extensions thus allowing simplified inspections and audits.

References

[Bogia and Kaplan, 1995] Bogia, D. P., and Kaplan, S. M., 'Flexibility and control for dynamic workflows in the WORLDS environment' Proceedings of conference on Organizational computing systems

[WfMC, 1995] WfMC: The Workflow Reference Model, WFMC-TC-1003, January 1995.

[Basten T. 1997] Basten, T., 'Life-cycle inheritance: A Petri-net-based approach' Application and Theory of Petri Nets 1997, volume 1248 of Lecture Notes in Computer Science, p62—81, 1997

[Ceri et al., 1997] Ceri, S., Grefen, P. and Sanchez, G. 'WIDE – a distributed architecture for workflow management', Proceedings of the Seventh International Workshop on Research Issues in Data Engineering, Birmingham, UK, April 1997.

[Geppert, A. and Tombros, D., 1998] Geppert, A. and Tombros, D.'Event-based distributed workflow execution with EVE', IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing, The Lake District, England, September 1998.

[Reichert and Dadam 1998] Reichert, M. and Dadam, P. 'Adept_flex-supporting dynamic changes of workflows without losing control', Journal of Intelligent Information Systems, Special Issue on Workflow and Process Management, Vol. 10, No. 2, pp.93–129, 1998.

[Zhou, T. et al., 1998] Zhou, T., Pu, C. and Liu, L. 'Dynamic restructuring of transactional workflow activities: a practical implementation method', Proceedings of the Seventh International Conference on Information and Knowledge Management, Washington D.C., November 1998.

[Muller and Rahm, 1999] Muller, R. and Raham, E. 'Rule-based dynamic modification of workflows in a medical domain' Proceedings of BTW99, p429 – 448, Freiburg, Germany, March 1999.

[van der Aalst, W.M.P. , 1999] van der Aalst, W.M.P. 'Generic workflow models: how to handle dynamic change and capture management information?' Cooperative Information Systems, p115—122, 1999.

[Zur Muehlen M. and Becker J., 1999] Zur Muehlen, M., and Becker, J., 'Workflow Management and Object-Orientation - A Matter of Perspectives or Why Perspectives Matter', OOPSLA Workshop on Object-Oriented Workflow Management, 1999

[WfMC, 1999] WfMC 'Interface1: process definition interchange V 1.1 final (WfMC-TC-1016-P)', October, Available at: http://www.wfmc.org. 1999

[Lee, 2000] Lee, M. 'Event and rule services for achieving a web-based knowledge network', PhD Dissertation, Department of Computer and Information Science and Engineering, University of Florida, Available at: http://www.cise.ufl.edu/tech-reports/tech-reports/tr00-abstracts.shtml, TR-002. 2000.

[Manolescu D. A., 2001a] Manolescu, D. A., An extensible Workflow Architecture with Object and Patterns, TOOLSEE 2001.

[Manolescu D. A., 2001b] Manolescu D. A., Micro-workfow a workflow architecture supporting compositional object-oriented software development, PhD Thesis, University of Illinois at Urbana-Champaign, 2001.

[Basten T. 2002] Basten, T., 'Inheritance of Workflows: An approach to tackling problems related to change', Theoretical Computer Science, vol. 270, 2002

[Müller, 2002] Müller, R. 'Event-oriented dynamic adaptation of workflows: model, architecture and implementation', PhD Dissertation, University of Leipzig, 2002.

[van der Aalst, W.M.P. , 2003] van der Aalst, W.M.P., 'Inheritance of Interorganizational Workflows: How to agree to disagree without loosing control?', Information Technology and Management, Vol.4, Issue 4, p345—389, October 2003.

[Kin, et al., 2004] Kim. J., Sparagem, M., and Gil, Y., 'An inteligent Assistamt for Interactive Workflow Composition', Proceedings of the International Conference on Intelligent User Interface, Portugal, 2004.

[Hanson J., 2006] Hanson J. 'Manage your business processes with JBoss jBPM', JavaWorld.com, May 2006

[Meng and al. 2006] Meng, J., Su, S.Y.W., Lam, H., Helal A., Xian, J., Liu, X., and Yang, S. 'DynaFlow: a dynamic inter-organisational workflow management system' Int. J. Business Process Integration and Management, Vol. 1, No. 2, p101—115, 2006

[Sadiq, W. et al. 2006] Sadiq, W.; Sadiq, S.; Schulz, K. 'Model Driven Distribution of Collaborative Business Processes', Proceedings of the International Conference on Services Computing, September 2006.

[Apache ODE, 2007] Apache ODE 'User Guide' http://ode.apache.org/user-guide.html, 2007

[Paul C. J. 2007], Paul C. J. 'The process of building a process manager: architecture and design patterns' IBM Systems Journal, volume 46, issue 3, p479—495, July 2007.

[Pinheiro W. A., et al. 2007] Pinheiro W. A., Vivacqua, A.S., Barros, R. Mattos, A.S., Cianni, N.M, Monteiro, P.C. Jr., Martino, R.N. Marques, V., Xexéo, G., Souza, J.M., 'Dynamic Workflow Management for P2P Environments Using Agents', Proceedings of the 7th international conference on Computational Science, 2007.

[Bergmann S., 2008] Bergmann S. 'Design and Implementation of a Workflow Engine', PhD Thesis, Rheinische Friedrich-Wilhelms-Universität, Germany, 2008.

[MSWF, 2008] Microsoft Library, 'Microsoft Workflow Foundation Overview' MSDN Library, .NET Framework 3.5, 2008