Int. Journal of Business Process Integration and Management

Management and Execution of Business Process Adaptations

Abstract

The global marketplace is becoming more and more competitive, and business organizations need to team up and operate as virtual enterprises to best utilize their resources in achieving their shared business goals. Since 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 using WEM to model and control 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 definitions, and rules as 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:
Workflows and Process Extensions, Management, Dynamic Adaptations.

Introduction

Nowadays organizations are seeking workflow management to facilitate the creation and execution of workflows while streamlining business processes. They are often required to develop customized activities. For example, a multinational company must adapt its processes to comply with local laws and regulations, or a delivery service must change how materials are shipped depending on the weight, size, and handling constraints. The adaptability and flexibility of those localized 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, or other operations. The term *workflow* indentifies a model to represent real work for further assessment. A workflow is a pattern of activity enabled by a systematic organization of resources, defined roles and material, energy, and information flows, that together form a work process that can be documented and manipulated. Workflows permit implementing business processes.

Most existing workflow management solutions only handle static business processes [Apache ODE, 2007], [Hanson J., 2006], [MSWF, 2008]. Specific cases are expressed as branches in a unique and global workflow process. This approach raises issues in terms of management and maintenance for large, complex, and error-prone workflow processes, as well as for extensibility and adaptation difficulty. These companies would benefit greatly from the delegation of governance. Global business processes should be defined company-wide, while specific adaptations should be handled within local organizations. This approach would avoid maintaining locally modified copies of the global processes, thus avoiding a loss of coherency between the multiple copies and the canonical process. We define an *adaptation* as a modification of a business process to handle new cases or to provide new functionalities. Adaptation is a generic term, and it does not attempt to specify any methods for the modifications.

We propose a workflow management method to extend existing processes, which we call "base processes", without changing them. It allows us to conditionally add multiple extended processes 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 appended onto regular business processes to adapt them to new constraints or environments. There are three advantages to this approach. First the base processes are not copied nor modified to integrate the adaptations. This avoids losses of coherency between copies and the creations of the adaptations do not cause modifications of the reference base process. Second, the multiple adaptations are independent from each other. Each extension can be modified without concern about the others. Third, the manageability of this system is extremely valuable: by separating and sharing the administration of base processes and extensions we simplify the maintenance of these combined business processes. Our solution has been implemented as a research prototype built on a commercial product.

Problem statement

Workflows adaptations may range from ad hoc modifications of a process for a single customer to a complete restructuring of the workflow process to improve efficiency. Today's workflow management systems are often inappropriate for dealing with workflow changes for competing 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 example, employees 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 need some extra configuration steps such as installing a specific language pack depending on the installation environment. Another process for the purchase of assets may skip repeated approval tasks under specific conditions (e.g. limited cost and emergency-caused constraints on time). The only way to handle such changes is to integrate adaptation capabilities within the system itself. If users are forced to bypass the workflow management system too frequently, the system is more of a liability than an asset. Therefore, we need to find innovative techniques to add flexibility for workflow management without losing the capabilities that are provided by today's systems.

There are two major types of adaptations: ad hoc changes and evolutionary changes. Ad hoc changes are handled on a case-by-case basis. To provide customer-specific solutions or to handle rare events, the process is adapted for a single case or a limited group of cases as needed. Evolutionary changes are often the results 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] argues 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 have attempted to fill the needs for adaptations of business processes. None of them have become a standard reference, and none are widely used in actual business IT systems.
There are two main reasons. The first and major one is the complexity of managing such systems. It is generally difficult to guess which adaptations can be executed by which users. For example, within a process representing the creation of an insurance contract, hundreds of adaptations represent particular cases such as promotional campaigns, specific locations, contracts, agreements, or time constraints. The resulting overall process is huge and scarcely 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 bypass existing adaptations. The administration of such a system becomes incredibly complicated and is not acceptable or reliable enough for business use. One of the main focuses of this paper is to provide a solution to dynamically adapt processes with automated assistance to identify and handle the adaptations. Management of the adaptations can be shared among several administrators, with each of them responsible for a set of adaptations that administrator is trained to deal with. This makes it easier to manage a large number of adaptations linked to processes.

The second reason is the total absence of validation mechanisms to ensure the correctness of the adapted processes in most existing systems. During the addition

of a new adaptation, the Workflow Extension Mechanism should ensure that the important properties of the basic process are maintained even after the process has been modified for the adaptation. For instance, mandatory tasks should not be skipped or tasks that must be executed once and only once should not become repeatable executed tasks because of the adaptation. Our system includes functions to verify the correctness of a process at the user level, but they are out of the scope of this paper, which is centered on the management of processes and adaptations.

Our methodology was to study the weaknesses of existing solutions and to identify the ways in which they fail to fulfill the requirements of actual organizations. We considered business process scenarios (many of them used in our own international company) and investigated the reasons why none of the existing workflow systems were able to address all of the problems making it difficult to dynamically adapt, encapsulate, separate, and hierarchically organize the different processes. It soon became clear that even if the related work allows for adapting business processes to particular needs, the management of those adaptations is quite limited, far below what is expected by the managers of a business. We decided to concentrate on building a model for process adaptations in which the management of processes and extensions is hierarchically organized and shared.

Our method to solve the problems we faced involves three main steps. First a kernel process must be identified. This process should be the smallest set of shared tasks and be complete and executable. In a second step extensions are created and added to the kernel process. Those extensions are processing components. They are incomplete by themselves and need to be attached to a complete process to be executed. Finally the last step consists of organizing the extensions for the various conditions and events that trigger them. This step constitutes the major part of our contribution. Proper management of the extensions simplifies the creation of numerous extensions while preserving the kernel process and reducing the overall number of process updates.

Related work

Workflow management is now widely accepted as a technology to support various business processes. Nowadays many commercial solutions for workflow systems are available. However, the large majority of those workflow systems are restricted to centralized and internal use within the boundary of a small organization. They do not offer mechanisms to deal with runtime adaptation nor with shared administration of process definitions.
The Workflow Process Definition Language, WPDL [WfMC, 1995] [WfMC, 1999], has become the reference standard for workflow. WPDL was originally developed by the Workflow Management Coalition (WfMC) in the early nineties. It offers a text-based 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 as a set of interrelated activities with different kinds of connection lines among them. The activities represent tasks performed by humans or computers that are related to each context (as defined by the workflow data, environment, and operators). The connection lines that link related activities control the flow within a workflow process. Unfortunately, WPDL is designed to model business processes only within the scope of a specific organization with uniform needs that span the entire organization. WPDL fails to provide a system allowing for the creation of inter-organization or inter-division business processes. The evolution of the IT industry and increasingly complex business interactions require handling dynamically adapted processes for special cases.

Several projects attempted to address this problem by introducing various dynamic capabilities for workflows. Here is a categorization of these solutions:

- *Evolutionary changes* have a structural nature. From a certain moment in time, a process changes for all of the new cases that enter the system. Such changes may result from the definition of a new business strategy, the modification of a law, etc. They mainly involve the modifications of a running process. Such a process modification is a global change and does not allow a constrained adaptation.
- The *inheritance of workflows* defines a base process containing the necessary tasks of a process and allowing for additional tasks. The base process identifies the sequences that can be extended. Unfortunately it might be impossible to extend parts of the process that were not carefully designed to allow for extension.
- *Dynamic inter-organizational workflow management* provides for complex extensions by modifying the base process at runtime depending on rules defined in advance by the users. Unfortunately, this approach does not permit shared or concurrent administration of the extensions. Also, the extensions may be in conflict with each other.

Later several research studies targeted these problems. The approach chosen by some projects was to define events and rules used to define and support the control flow used in a business process model. The most representative example among these solutions was the EvE project [Geppert and Tombros, 1998]. The EvE project introduced a distributed Event-Condition-Action (ECA) rule-based execution architecture. The use of events and rules to handle exceptions and failures during workflow execution falls into another category. In that category, events are produced outside a workflow's execution, either by the system environment or by 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 to allow the definition of ECA rules to support exception handling and to implement asynchronous behaviors during workflow execution. Unfortunately, this approach is very centralized but even so does not provide management mechanisms in addition to the adaptation mechanism.

-

More recently the project AgentWork [Müller, 2002] was based on the same concepts of event and rules, but with a slightly different approach. AgentWork deals with dynamic adaptations appearing when a workflow instance encounters unexpected failures. The main difference between the approach of AgentWork and the ECA rules comes is because the rules and events are directly part of the workflow model. To allow adaptations at runtime each adaptation must be fully described when it is added. The paper describes the permitted times when adaptations can start and the synchronous or asynchronous characteristics of the events. During the execution of a workflow the system emits the events as described and activates the rules to trigger the adaptation. A second major difference from the ECA model is that the approach used by AgentWork is not restricted to a single organization, since events can be defined and managed in a distributed environment [Lee, 2000]. [Meng and al. 2006] called the AgentWork approach ETR for Event-Trigger-Rule. Even though AgentWork deals with distributed management, it lacks any mechanism to share the management activities. They remain strongly centralized even in a distributed environment.

Several projects attempted to solve the dynamic adaptation problems 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]). By using dynamic bindings this approach can provide mechanisms to call one task rather than a related task (an ancestor or interface task). The advantage of this approach is more in producing enterprise-oriented implementations of workflow engines for object-oriented languages than in the actual flexibility for adapting the processes. The interest in this approach was revived by the Web services approach. The degree of mechanization in BPM is limited, which slows the necessary dynamic evolution of business processes. [Cai J. et al, 2005] and [Hepp, M. et al., 2005] argued that Web services frameworks are a natural fit for creating more flexible systems. A rigid architecture defined by inheritance limits the ability to freely modify a process. Also, dynamic bindings may be quite unpredictable, making the static addition of processes and extensions a difficult task.

Recent research has focused on how a dynamic workflow management system can dynamically modify workflow definitions to adapt to dynamic business conditions and exceptional situations. [Reichert and Dadam 1998] presented a formal foundation for supporting dynamic structural changes of running workflow instances. [Muller and Rahm, 1999] describe 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 transactions. These projects focused mainly on the structural changes of the process models. Most recently, DynaFlow [Meng and al. 2006] supports both structural (e.g. dropping an activity or bypassing some activities) and semantic (e.g. replacing an activity or modifying

a transitional condition) changes to an inter-organizational business process model. The 'Code Generation' approach, which was used to develop the workflow engine in DynaFlow, makes it easy to support such changes. The adaptability of a workflow instance is enhanced when the changes to the process model are handled dynamically by business rules that are triggered by synchronous events posted by a running business process. By modifying the process itself at runtime DynaFlow permits great flexibility, but there is no management mechanism to ensure validity or to isolate the extensions .

Even though some projects tried to address the problems of correctness or validity of a composite process [van der Aalst, W.M.P., 1999], [van der Aalst, W.M.P., 2003] [Kin, et al., 2004], [Goedertier S. et al. 2008] none of the solutions provided an integrated mechanism to seamlessly define, extend, adapt, and change a business process in a decentralized manner while hierarchically assigning specific modifications to specific environments and data. [Adamides E. and Karacapilidis N., 2006] and [Wang X., 2009] introduced a knowledge-centered framework to deal with a collaborative business. Unfortunately this approach is limited by and depends upon the availability and the technical abilities of a technical facilitator. The projects presented there propose mechanisms to extend a process but do not offer a complete end-user-oriented approach to manage such functionality. They lack both the concept of extensions that can be easily created, attached, and removed from a process, and techniques to manage them within large organizations. One of the challenges for our workflow management system was to efficiently implement a simple, elegant, and flexible way to easily manage multiple adaptations. The mechanism for hiding or showing extensions we introduce in our model is a valuable feature for simplifying process management while enabling smart collaboration in process addition. In addition, other advantageous side effects are increased flexibility, dynamism, and stability of the core processes.

Extension

Proposal

Workflows systems are usually data-centered, so that each piece of work is related to and executed for a specific data object. Examples of data objects are an asset of a company, a personal profile, a tax declaration, a purchase order, or a request for information.

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

-

The proposed approach involves the addition of a new component within the workflow engine itself. This component describes how each adaptation must behave, under which constraints, and when it should stop. This new component comes with procedures to enhance the management of adaptations.

Description of this solution

Our technology provides a mechanism to share the administration of process extensions. It allows multiple administrators (process extension editors) to deal with isolated or intersecting (and included) extensions. We say that two extensions are *isolated* if they cannot occur at the same time and at the same point during the execution of a process (which means they can not enter into conflict). When editing one extension its isolated peers should not be displayed to the administrators. In contrast, *intersecting* and *included* extensions may occur at the same time and at the same point during the execution of a process (which means they can enter into conflict). It is necessary to identify the extensions in an intersection and let the administrator solve any possible problems (perhaps by sequencing the extension).
We propose a new technique to identify isolated and intersecting extensions to processes. The goal is to simplify the addition of process extensions. When editing an extension to a process the administrator will be shown the entire set of extensions associated with that process (the entire content of the *Extensions Table*). This may result in a very complex and large view of the process. The editing method in this paper selects the necessary set of extensions to display that are related to a specific extension, thus simplifying the administrator's work. Since extensions are only executed in accordance with their *Conditions of Application* this is an important aspect of our selection mechanism.

This mechanism involves 4 steps:

The first step is the definition of the *CoA* for the new extension. An administrator creates the new extension and starts by defining its *CoA*

In the second step variables trees are built with the values used in the *CoA*s of the existing extensions. For each variable we build a tree following these rules: a child node is a value included in the parent node and the branches are disjoint values. In Figure 1 the black boxes, the roots of the trees, represent the case "any". The nodes in the tree are conditions that refine the conditions of their ancestors.

The third step involves the selection of extensions to display. Based on the *Conditions of Application* of the existing extensions ($CoAs_{ext}$) and of the new extension (CoA_{new}) the isolated extensions are hidden leaving only those extensions for which the *CoA* may intersect with the new extension. The rules to decide if an extension is shown or hidden use the variables trees:

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

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

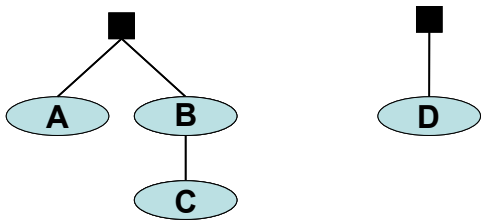


Figure 1 - The relationships in trees

Figure 1 represents two trees, the first one composed of three nodes A, B, and C, and a second containing a unique node D. Here are the relationships among these nodes: D intersects A, B, and C, since it belongs to a different tree. Symmetrically A, B, and C intersect D. B includes C since B is an ancestor of C. A is excluded from B and C since 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, processes, order, and exit-point*). The administrator uses the *order index* to solve possible ambiguity in execution order among the extensions whose entry points are similar. *The order index* is a real number from between 0 (excluded) to 1 (excluded).
The administrator is also free to narrow the *CoA* for the new extension. In that case Step 3 will be executed in order to hide the other extensions that become isolated because of the narrowed *CoA*.

The *Extensions Table*, even if managed in a distributed way, is a centralized set of information. This provides several benefits, such as global validation or auditing of the extensions. A super-administrator may perform automatic checks on the *Extensions Table* to verify that no locally defined extensions are conflicting with the base process and to verify compliance with the company's standards and rules.

Discussion of advantages

In summary, the Workflow Extension Model allows to provide adaptations that are evaluated at runtime. The extensions are created from a canonical 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 custom cases. In contrast to the many projects reviewed in this article, our approach to process adaptation allows new paths in a business process and is not limited to the replacement of designated tasks. The flexibility of our approach is also much higher.

-

The first advantage is that of avoiding any modifications of the base process (at addition time or execution time). In this way the original processes remain as a clear reference for understanding the goals that are being achieved, since they provide a general outline of what will happen. The base processes are not copied. Multiple copies of a base process may simplify the adaptation mechanism but create risks that some copies may introduce differences among the copies of the base process itself rather than among the extensions. The possible loss of coherency between copies of the base process is a great danger hindering the management and verification of business processes.

The second advantage is the isolation of multiple and independent extensions. Each extension is independent of the others (except in the special cases of hierarchical or nested extensions). This means that each extension is self-sufficient and does not require modifications to other extensions. This permits freely modifying one extension with limited concern about the others. The mechanism we described deal with possible interactions between extensions. A user who may be tracking several extensions for one process must be aware of them, and the mechanism presented here helps filter which extensions can be executed depending on the execution environment.

Thus the final and major advantage is the manageability our approach offers to the owners of business processes. The separation between the base processes and the extensions makes it possible to keep the base processes simple, since they no longer include the entire set of adaptations. This makes it much easier to add new extensions, since the system will automatically select the already existing extensions that must be shown to the operator who is creating a new one. The base process is shown with the subset of all its extensions that could be triggered in a shared execution environment in which the new extension is being created. The separation between the base processes and the extensions also allows reducing the number of updates of the process definitions. Since the extensions deal with the adaptations, all of the updates affecting the adaptations are done on the extensions, thus leaving the base process unchanged. This is valuable so that the extensions can be locally managed and their modifications will only impact their own users. With standard and globally integrated management of all adaptations all of the users of any process may be affected by all of the modifications to any of the 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, using a model of behavior consisting of a finite number of states, transitions between those states, and actions. In most of the

existing implementations we can identify the main components of the workflow engine as a *Process Table* that stores the entire set of process definitions, a *Process Definition* Table that describes the processes (transitions and tasks), a *Task Definition* Table that associates the actual operations to process tasks, and a *Process Instance Table* that provides a view of the running processes.

A workflow engine periodically selects a process from the Process Instance Table then uses the Process Definition Table to discover the next task to perform. Tasks are done by humans (entering data, approvals, etc.) or automated by computers (calling a Web service, querying a database, etc.). This procedure is repeated until the process instance reaches an endpoint in the process definition or encounters an unhandled error. The commercial product we based our prototype on supports two standards, BPMN for the modeling and BPEL for the execution. Actually this commercial product uses only a subset of BPMN, but our extension can be extended to the entire BPMN standard.

We have created a new table that we include with the core tables. This new table describes the way an extension is attached to a process. It does not describe the extension itself, and we relay on the standard process definition table to store the definitions 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 – An Extensions Table for the process shown in Figure 4

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

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

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

(3) the *type of connectors* to the base process that defines the branching semantics of the extension (conditional, unconditional, parallel, etc.)

(4) the *extension process itself*, describing the particular adaptation to address certain problems,

(5) the *order index* used to sequence several extensions that may be executed at the same time,

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

We also extend the Process Instances Table with a new field, the *Exit-points Stack*, which tracks the references to the exit-points when executing an extension process. The scope of our contribution for a workflow extension model is

shown by the shadowed box in Figure 2.

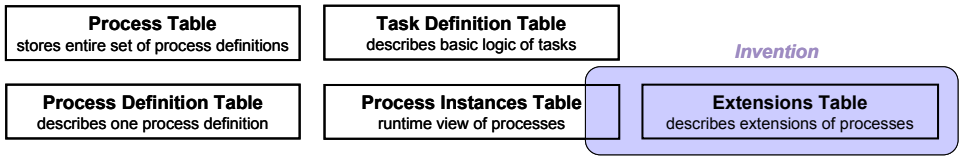


Figure 2 - The existing and the new

As shown in see Figure 3 the way a workflow engine operates is slightly changed: after selecting a process in the Process Instances Table, the workflow engine selects an extension process and identifies the next task to perform. If appropriate extension processes are found at this point (the *entry-point*), then they are sorted according to their *order index* and the first extension for which the *Conditions of Application* are satisfied is executed instead of the original task. The *exit-point* is registered in the Process Instance Table. This table handles the exit-points as a stack object to permit recursive extensions. When the execution of the extension process ends, the base process is resumed at the exit point popped from the stack. The bold steps (shaded boxes) are our contributions.

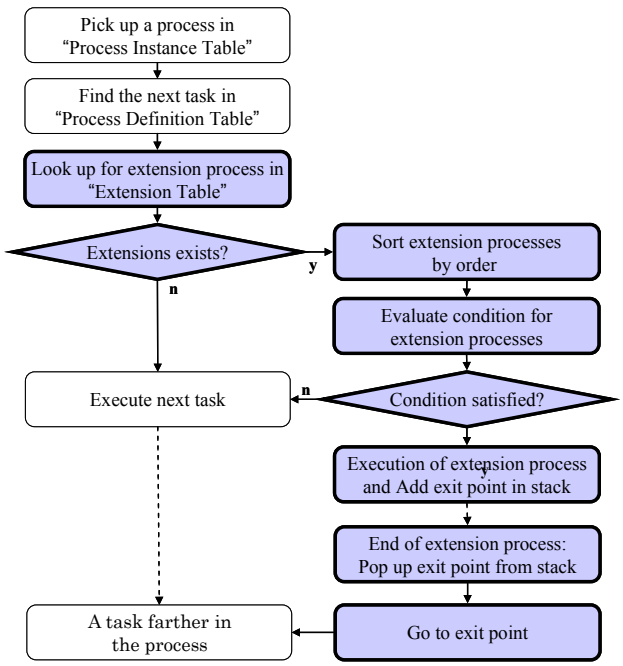


Figure 3 - The workflow engine flowchart

Example scenario

In our use case example (described in Table 1), the conditions to evaluate the extension processes are based on certain environment values such as `user.domain` and `OS_Type`. Figure 4 presents the base process associated with extensions defined

in the *Extensions Table* of Table 1.

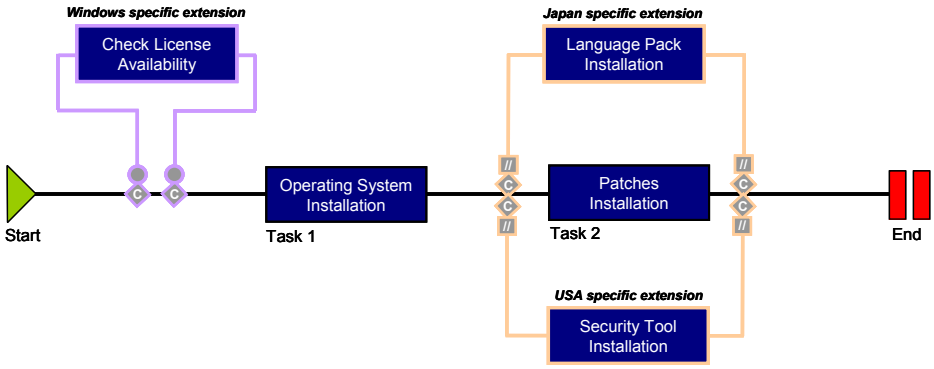


Figure 4 – An OS installation process with various specific extensions

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

We now consider adding new extensions to the process depicted in Figure 2, and will go through the 4-step mechanism.

First, the administrators define the *Conditions of Application* of the new extensions they intend to create. Our example involves three administrators who independently want to add one new extension to the base process of Figure 4. Each of them defines a *CoA* as described in Figure 5. The first administrator wants to define an extension related to users in the domain Tokyo. The second administrator, Admin B, is introducing an extension for users in the U.S. who have installed a Linux operating system. Finally Admin C's goal is to create an extension that will affect all users worldwide.

-

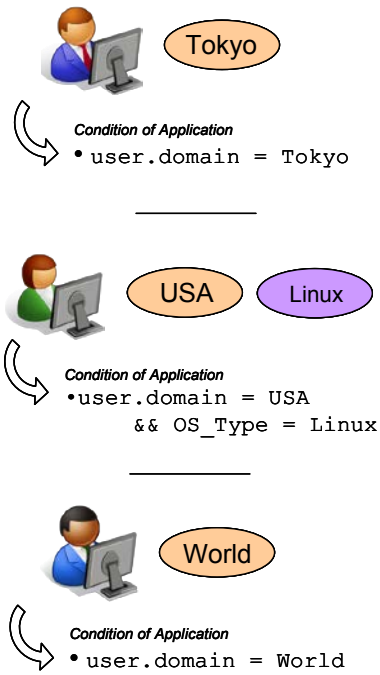


*Figure 5 – Administrators define CoA*s

Second, variables trees with the values used in the *CoA*s of the existing extensions are built. Figure 6 shows the possible variable trees for the process of Figure 2. The left tree contains the different values associated with the `user.domain` variable within the extensions attached to 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 types are defined along with some *comparable* interface or *comparator* object able to deal at least with *included* and *excluded* primitives.

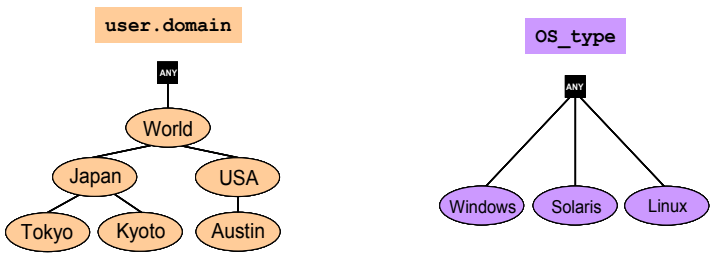


Figure 6 – Variables trees

The third step triggers the selection mechanism to identify which extensions must be shown to or hidden from the three administrators. The rules to select the extensions to show and hide are evaluated by using the *CoA* of a new extension

and the *CoA*s of the existing extensions.
Finally as shown in Figure 7 each administrator has a different view of the base process and the available extensions.

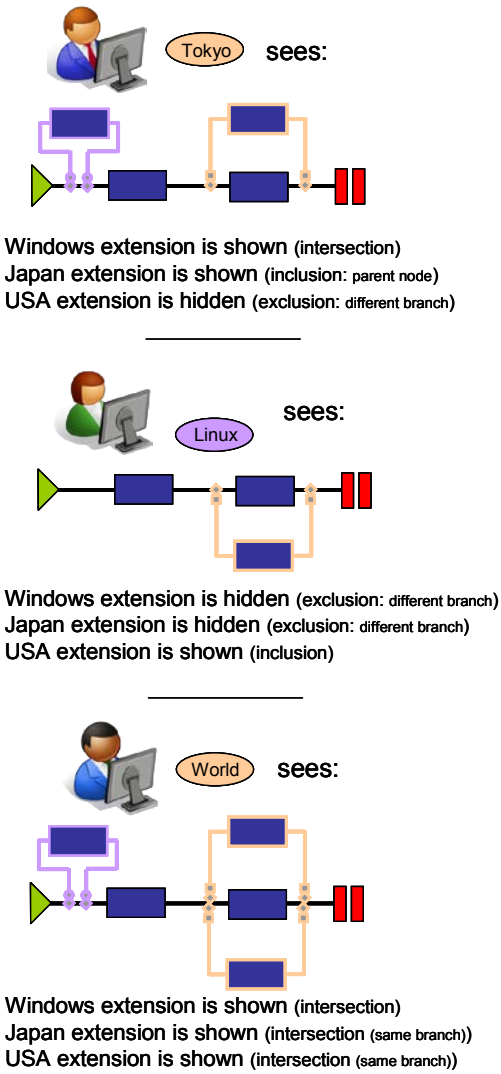
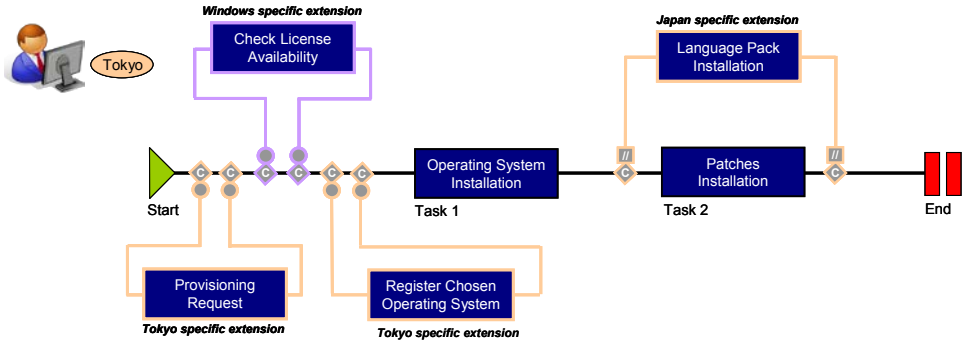


Figure 7 – Administrators' views of the base process and available extensions according to the CoA they specified for each new extension.

In the fourth and final step the administrators fully define their new extensions.
Let's continue this scenario with Admin A who defines a Tokyo-specific extension. This administrator wants to provide two new extensions before Task1 (OS installation) for "provisioning the request" and "register the OS". There is an ambiguity about the order in which to perform these two new extensions and the existing "Check license" extension. The administrators must properly set the *order indexes* to sequence those extensions. Figure 8 presents the final view of the

process after the administrator finishes defining 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 8 – The process view of Admin A after the addition of two extensions and the Extensions Table.

Conclusion

The major benefit of our approach is increased manageability of the business processes. The prior related work scarcely deals with the administration of dynamic processes and only focuses on the execution of extensions rather than considering the entire lifecycle of processes and process extensions.

Summary of advantages

In order to give evidence of the simplified management approach of our workflow extension model, we will consider the case of an international organization whose offices are scattered among many countries. The head office of this organization wants all of its foreign offices to conform with the centrally defined the rules and process. At the same time, many of 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) will continually receive requests for local adaptations that must also be integrated with the general processes. This generally results in several problems: (1) the central administrator may have limited knowledge and understanding of all of the local constraints and so introduce errors in the adaptations; (2) when there is a sudden peak of requests, the central administrator becomes a bottleneck delaying the creation of adapted processes; (3) the global process resulting from all of the adaptations is very complex and error-prone; and (4) each new modification of an existing adaptation that affects only one local office

will still be propagated to all of the local offices even when they are not affected by that adaptation.

Evaluation

To make it more concrete, we show an example of tracking how many modifications and updates are required 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 guidelines in the form of base processes that all of the other divisions must follow. Each of the 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, all of the processes are shared among all of the company's divisions and completely integrated with the entire set of adaptations. Adaptations usually take the form of conditional branches in the processes. The result is that any modification required by any division, even for its exclusive purposes, is shared with the entire company in a process definition that becomes more and more complex as the company grows. In addition, confidentiality may become an issue. For example, the IT division may not be aware of the internal procedures of the financial division. In this configuration, if one division, such as the IT division, introduces numerous extensions and modifies them often, then the entire set of users, from all divisions, will be affected, possibly negatively, by the numerous migrations to the new versions of the globalized processes.

With the workflow extension model in this paper, the impact of each modification or update is appropriately limited to the division that has produced it. If the employees are equally divided among all of the divisions, then only 10% of the employees (the IT division out of the 10) need be affected by the migration of a new process for that division. The other 90% of process changes can also be ignored by the IT division. In addition, the effects can even be limited to only those people who are actually affected by a specific adaptation. The IT-specific changes to processes need not affect other employees who have no need of those adaptations. The final advantage in moving the adaptations out of a special globalized process is that this approach also stabilizes the base processes. The frequent updates affecting adaptations that were previously required on top of a globalized process are now performed only on the extensions. This increases the lifetime of each version of the base processes, resulting in greater global stability in the set of business practices used within the entire company.

The addition of our extension model has no detectible impact on the performance of our test application. The CPU usage does not change since our implementation relies on existing mechanisms and consists mainly of rerouting flows rather than introducing additional mechanisms. The memory usage remains constant because

no dynamic values are stored in memory. Our architectural model keeps all of the information in the database. The additional data storage to hold the process definitions is quite small, and can normally be ignored, since it involves only the definitions of the extensions. The definitions of the nodes and objects that consume most of the storage are unchanged. Finally the logic itself only adds a search for extensions and a sorting operation if several extensions apply at the same time. Those operations are negligible in the overall process, representing less than 0.1% of the operations required for the flow from one task to another.

Sometimes there are problems when different divisions may produce similar extensions for a specific workflow. This will result in duplicated work in producing those extensions and in some extra storage for essentially duplicated information, but this problem is outside the scope of this paper. However [Nakamura et al. 2009] investigated analytic tools that can identify such shared extensions and create a catalogue of possible extensions to attach to a base process based on the best practices. There are also performance issues related to the choice of an inefficient extension when a more efficient version exists, but this is also beyond the scope of our prototype. [Helquist et al, 2009] give some hints about evaluating workflow compositions. Their model incorporates measurements and metrics related to global geographic and team issues, so that process designers can predict more accurately the most successful deployment options. Such meta-data metrics should be included in the extension catalogue.

In conclusion, in contrast to existing solutions our innovative approach is much more dynamic and adaptable since the extensions can be added or removed without modifying or copying the base process. This is permitted by the introduction of a new table within the core of workflow engine along with a mechanism to help share and distribute the management information. The governance of the final workflows is shared between the various participants, thus reducing the complexity of the base workflow. The addition of new extensions is eased by isolating and hiding the existing extensions. Finally, the *Extension Table* centralizes the information about all of the extensions and thus simplifies the 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, pp. 62–81, 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 Rahm, 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, pp. 115–122, 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', Ph.D. 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-workflow a workflow architecture supporting compositional object-oriented software development, Ph.D. 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', Ph.D. 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, pp. 345—389, October 2003.
[Kim, et al., 2004] Kim. J., Sparagem, M., and Gil, Y., 'An Intelligent Assistant for Interactive Workflow Composition', Proceedings of the International Conference on Intelligent User Interface, Portugal, 2004.

-

[Cai J. et al., 2005] Cai J., Lu S. C-Y., Grobler F., Case M., and Jing N., 'Modeling and managing collaborative processes over the internet', Business Process Management Journal vol.11, issue 3, p255–274, 2005

[Hepp, M. et al., 2005] Hepp, M. Leymann, F. Domingue, J. Wahler, A., and Fensel, D., 'Semantic business process management: a vision towards using semantic Web services for business process management', IEEE International Conference on e-Business Engineering, Beijing, China, 2005.

[Adamides E. and Karacapilidis N., 2006] Adamides E. D. and Karacapilidis N., 'A knowledge-centred framework for collaborative business process modelling', Business Process Management Journal, Vol. 12, Issue 5, pp. 557–575, 2006

[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, pp. 101–115, 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, pp. 479–495, 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', Ph.D. Thesis, Rheinische Friedrich-Wilhelms-Universität, Germany, 2008.

[Goedertier S. et al. 2008] Goedertier S., Haesen R., Vanthienen J., 'Rule-based business process modeling and enactment', Int. Journal of Business Process Integration and Management Vol. 3, No.3 pp. 194 – 207, 2008

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

[Helquist J. et al. 2009] Helquist, J. H. Cox J. J., Walker A., 'Exploring diverse process and team alternatives through virtual process simulation', Business Process Management Journal Vol. 15, Issue 5, pp. 633–652, 2009

[Nakamura M. et al. 2009] Nakamura M., Kushida T., Bhamidipaty A., and Chetlur M. 'A Multi-layered Architecture for Process Variation Management', International Workshop on Services Composition (SC2009 workshop), Bangalore, India, 2009

[Wang X., 2009] Wang X., 'Specifying the business collaboration framework in the Contract Expression Language', Int. Journal of Business Process Integration and Management Vol. 4, No.3 pp. 200 – 208, 2009