Using self-reconfigurable workplaces to automate the maintenance of evolving business applications

Conference Paper · October 2005

DOI: 10.1109/ICSM.2005.100 · Source: IEEE Xplore

2 authors, including:

Ying Zou
Queen's University
154 PUBLICATIONS **1,935** CITATIONS

SEE PROFILE

Using Self-Reconfigurable Workplaces to Automate the Maintenance of Evolving Business Applications

Qi Zhang and Ying Zou
Department of Electrical and Computer Engineering
Queen's University, Kingston, Ontario
{qi.zhang, ying.zou}@ece.queensu.ca

Abstract

In this ever changing business environment, business processes are constantly being customized to reflect the up-to-date organizational structure and business objectives. Technology updates and innovation also affect the way business is carried out. A workplace application provides an interactive electronic working environment that integrates software applications to assist users in performing their daily work more efficiently. Managing and maintaining workplace applications within an organization is a challenging job, since it often involves labor intensive manual reconfiguration to adapt the workplace to the changes in business processes. In this paper, we propose a dynamic reconfigurable workplace framework that supports the changing nature of the business domain. This framework updates the workplace at run time, minimizes the interruption to users' work, and simplifies the evolution of a business application. The effectiveness of the framework is studied by examining changes to several business processes and the ability of the framework to update the corresponding workplaces.

1. Introduction

Business processes encapsulate the knowledge of operations and services provided by an organization. Typically, a workflow definition represents a business process as a sequence of steps. It describes essential tasks, business roles, and resources required by a business process. For example, a *Book Purchasing* business process may consist of a number of tasks, such as *Request Book*, *Select Payment Method, Approve Book Order and Ship Book*. Multiple roles, such as *Customer*, *Sales Manager and Shipping Clerk*, participate in this business process. A Sales Manager approves an order, and creates an invoice. A Shipping Clerk prints a shipping slip and contact the shipping company for pickup. A workplace application provides an electronic working environment that integrates information and services to support employees in performing their daily work. A workplace application is built on top of a Workflow Management System (WfMS)[1]. A WfMS has a workflow engine that analyzes workflow definitions (corresponding to business processes) and assigns tasks to workplaces. Each role in an organization (*e.g.,* Sales Manager or Shipping Clerk) has a workplace application. Tasks are distributed to an appropriate workplace based on the workflow definitions. For example, when a book is selected by a customer, the *Approve Book Order* task is displayed the workplace for a Sales Manager. Once the order is approved, the workplace for the Shipping Clerk is updated with a *Print Shipping Slip* task. An application is assigned to perform each task.

In this ever changing business environment, business processes are constantly customized to meet the requirements of an organization. Business processes are frequently modified to achieve business goals. Furthermore, technology updates and innovation also affect the way business is carried out. In today's reality, a change, such as adding a new task to an existing business process, or replacing a manual business step with automation, is a labor intensive and requires complicated manual re-configuration or re-development of the software application that implements the business process. Especially, there are no explicit bindings between the tasks specified in workflow definitions, and the software applications that execute these tasks. This leads to difficulties in identifying the appropriate locations in software applications that need to be modified or re-configured in order to reflect changes in business processes. The inability to gracefully adapt to changes has driven research interests in developing autonomous systems that can be dynamically configured to accommodate evolving business processes.

However, most of researchers focus on developing techniques that allow a workflow engine to automatically distribute tasks specified in the new/modified workflow definitions. From the workplace perspective, Uploading new applications that perform the new tasks or removing applications that conducted deleted tasks involves shutting down the whole workplace system in order to perform system updates. This system interruption often causes lost work and forces the users to repeat workflow tasks. For example, a Customer has to make the same order twice if the system was updated before a book purchase process was done. To improve work efficiency and reduce financial loss, it is important to ensure that workflow definition changes are automatically reflected to workplace applications, so that role players (i.e., users) carry out new/modified tasks without affecting tasks accomplished or in progress under the old workflow definition. In this paper, we propose a run-time reconfiguration framework that identifies modifications in workflow definitions, and automatically populates the modifications to the appropriate workplaces by associating the new/modified tasks with new applications. The proposed self-reconfigurable framework can reduce the downtime of workplace applications and the costs of maintaining them.

This paper is organized as follows: Section 2 gives an overview of WfMSs and workplace applications. In Section 3, we present a motivating example. Section 4 describes the proposed framework to achieve the self-reconfigurability in a workplace application. Section 5 introduces our prototype self-reconfigurable workplace system and the case studies we conducted. Section 6 surveys related work. Finally, Section 7 concludes the paper.

2 Workflow Management Systems and Workplace Applications

2.1 Workflow Management Systems

The Workflow Management Coalition (WfMC) defines a reference model for workflow management systems (WfMS) [19]. A WfMS typically consists of a workflow modeling tool, a workflow engine and a workflow client application. A workflow modeling tool allows business users to model workflow definitions. A WfMS provides a workflow definition interface to import and store workflow definitions. A workflow engine interprets a workflow definition, and distributes tasks to workplace applications (i.e., workflow client applications) at run-time. Tasks are executed by applications that access enterprise database, and services.

2.2 Workplace Applications

Workplace applications ease the day-to-day changes to business processes. They provide a central location that facilitates the distribution of change information [3] and upgrading of business applications that implement business processes. Workplaces leverage Web portal technology, and integrate with WfMSs, enterprise databases, and other enterprise applications, such as knowledge management and decision support systems. The Web portal platform serves as a standard middleware to provide interoperability with underlying applications and the infrastructure. Applications implement the functionality provided in a workplace. Typical applications include email, calendar and specific task related applications. Infrastructure provides common functions to a workplace application, such as administration, access control, installation and un-installation of applications.

As an example of a workplace application, a screenshot of a prototype workplace is illustrated in Figure 1. The workplace consists of several major components, such as Work Items, Process List, and Process Progress Status. The Work Items component lists a collection of Tasks that are assigned by a workflow engine and are to be completed by the user. The Process List component gathers a set of business processes which the user participates in. Business processes are organized based on the role of the user in a business process. Multiple roles may participate in a business process to fulfill tasks through collaboration mechanisms provided by workplace platforms. A user can be assigned multiple roles. As illustrated in the Process List component, the user is assigned to two roles, including Product Manager and a Manager. Once a user clicks on an assigned task the corresponding application that implements the task is displayed in the center area of the user interface of the workplace. In this example, an application is invoked to accomplish the task of *Create Catalog*. The application enables the user to create a new category in a catalog hierarchy or select an existing category. The *Process Progress Status* component indicates the progress of user's work in a business process.

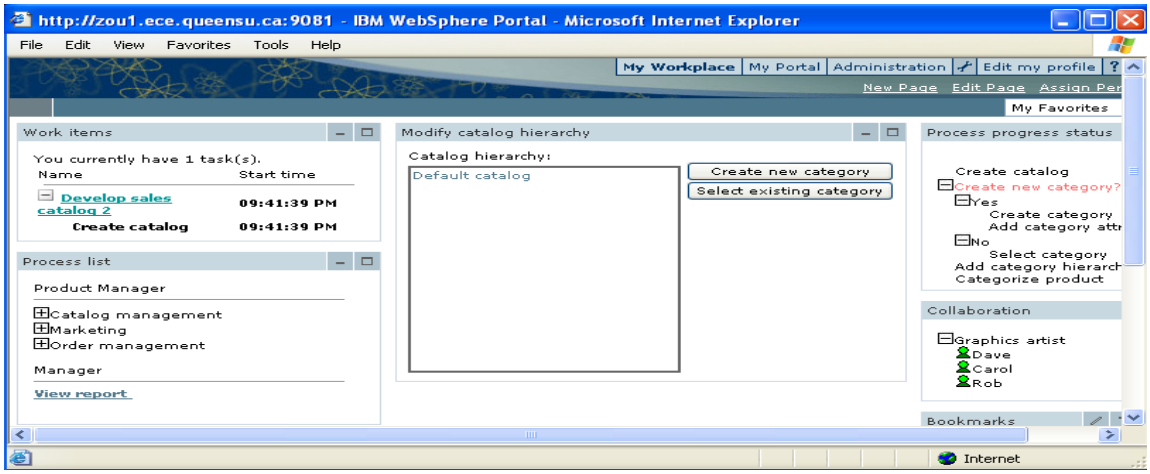

Figure 1: Example of a workplace

2.3 Bindings between Tasks and Applications

To ease the localization of affected applications that implement an evolving workflow definition, we define a set of bindings between tasks and applications. A task to application binding can be specified when the workplace is created for the user, and is updated when a workflow definition is modified. Figure 2 illustrates a workflow definition comprised of four tasks. *Task 1 is bound to Application A, and executed by Application A*. Similarly, *Task 2* is bound to *Application B* in *user 1*'s workplace, whereas *Task 3* and *Task 4* are bound to *application C* in *user 2*'s workplace.

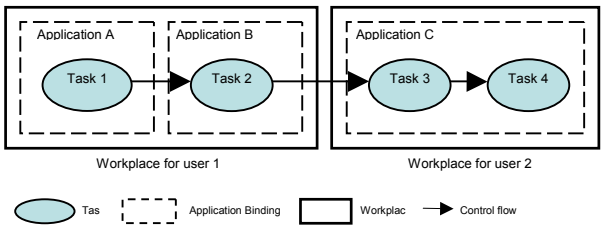

Figure 2: Bindings between applications and tasks in a workflow definition

At run-time, a workflow engine initiates a workflow instance from a workflow definition. A workflow instance consists of several task objects. Each task object corresponds to a task specified in a workflow definition. When a task object needs to be executed, the workflow engine sends a request to the workplace. Consequently, the workplace fetches the binding information between the task and its associated application. Furthermore, the workflow engine assigns the task to the user's workplace. The user accomplishes the task by executing the associated application in the workplace.

3 A Purchase Order Example

We present a purchase order example to illustrate how modifications in workflow definitions can be reflected in the workplace. As depicted in Figure 3, a *Purchase Order* workflow definition handles customer's purchase request and generates an invoice upon approval by four consecutive tasks. The *Order Request* task and *Select Payment Method* task are performed by the role of Customer by filling out a purchase application on a *Purchase Order Website*. Consequently, the role of Sales Manager verifies the order and approves it in the *Purchase approval* task, and creates the invoices in the *Create invoice* task. Moreover, the workflow definition specifies data flowing between tasks, including *Product info* data, *Customer order* data, and *Invoice* data. In the workplace, the application, named *Order Management Application,* is associated with, and used to perform the *Purchase Approval* and the *Create Invoice* tasks, respectively.

Suppose the company decides that the Sales manager needs to verify the credit information of the customer before a purchase can be approved. This causes a new task, called *Check Customer Credit,* to be inserted before the *Purchase Approval* task in the *Purchase Order* workflow definition. Nevertheless, a data, called *Purchase Record,* is added to the workflow definition, which contains both the order information and customer account information. The modified workflow definition is shown in Figure 4. In this case, a new application called *Credit Checking Application* bound to *Check Customer Credit* task needs to be installed.

The new workflow definition is imported into the MfMS. A new workflow instance is spawn under this new workflow definition. To reflect the changes in the

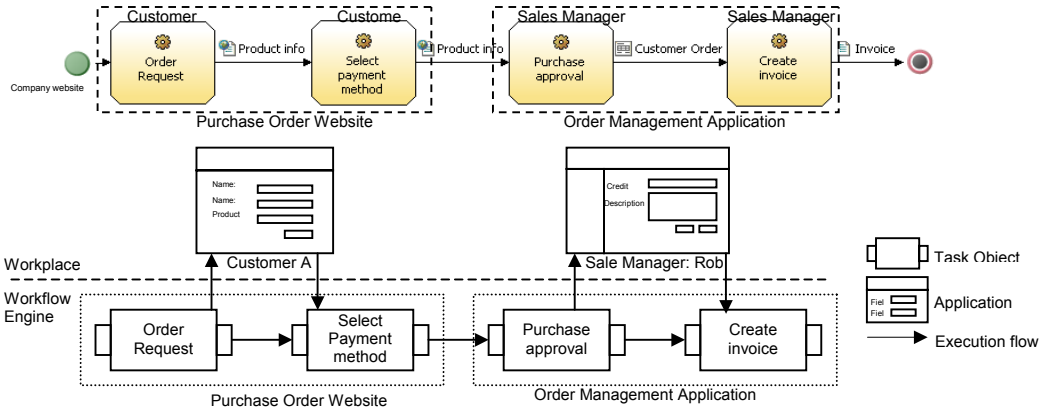

Figure 3: Workplaces for purchase order workflow definition

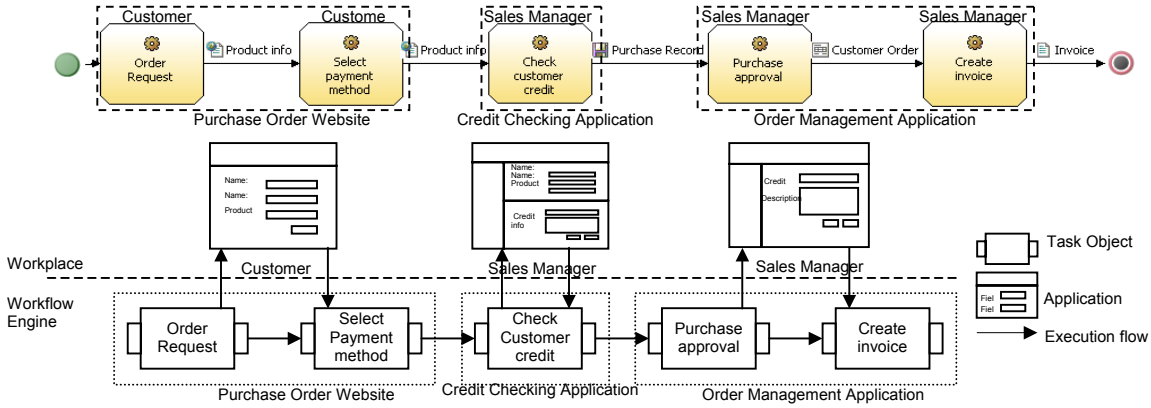

Figure 4: Workplaces for modified purchase order workflow definition

Purchase Order workflow definition, the *Credit Checking* application needs to be installed into the workplace. We could simply shut down the old workplace applications (i.e., the Customer's workplace and the Sale Manager's workplace) and upload the applications associated with the new workflow instance into the workplace. However this may not be feasible in many circumstances, such as business processes with high availability and cross-organizational processes. For example, a Sales Manager is engaged in other business activities, and collaborates with other roles. The system interruption of his/her workplace may cause the loss of the work requests from other roles, and cause the suspension of other workflow instances that depend on the work accomplished by this role. Furthermore, shutting down the system may cause a decrease in business performance and hence increase maintenance cost. As a result, run-time reconfiguration of the workplace is necessary to address these issues.

Our research addresses the following issues which arise in the reconfiguration of a workplace:

- Bindings between tasks and the applications that execute the tasks may change, such as software upgrades. We treat different versions of an application as independent applications.
- A role may be assigned to other tasks. A list of new applications is added into each role player' workplace.
- As a workflow definition evolves, such as task addition, deletion, and modification, user's activities in the workplace are affected. For example, a new task is added to a current executing workflow instance, the user must conduct the additional task.

4. A Framework for Dynamically Reconfigurable Workplace Applications

To empower business workplaces to be responsive to the evolving environment, our research leverages workflow definitions to build a self-reconfigurable workplace for employees in an organization to manage their day-to-day work. We strive to derive a self-reconfigurable framework that can automatically detect changes made to workflow definitions, and adapt changes to the workplaces and their corresponding applications.

To achieve this objective, we establish the associations between the tasks and the applications that implement these tasks. Therefore, changes to workflow definitions can be easily tracked down to the affected roles, and the corresponding workplaces can be updated using our proposed self-reconfigurable framework with no need for manual intervention. In the following subsections, we discuss the proposed self-reconfigurable framework in more details.

4.1 Workflow Change Detection and Workplace Configuration Generation

In this section, we elaborate on our approach for detecting changes in workflow definitions, and generating an updated workplace configuration that incorporating the changes in workflow definitions. A workplace configuration includes the bindings between tasks and their executable applications, interface definitions of the applications, and the workplace layout. Different users have their own workplace. The content and layout of each workplace is determined by each user's individual configuration.

In order to capture changes in workflow definitions, we develop a generic workflow meta-model that can abstract the essential characteristics of workflow definitions and represent the workflow definitions in a unified format. This meta-model also allows us to explicitly describe possible changes to a workflow definition. Typically, a workflow is defined in terms of nodes and connectors. The connectors include control flow connectors and data flow connectors. These connectors guide the control and data flow of the process. Essentially, the generic structure of a workflow can be viewed as a 6-tuple, defined as follows. We refer this 6-tuple as a workflow meta-model.

W = <Node, CFC*, DFC*, Application*, Role*, Data*>*
Node = (Name, Role, Application, InData, OutData)
CFC = (From, To, Condition)
DFC = (From, To, Data)*
InData = (Data)*
OutData = (Data)*

- * denotes that the occurrence of an element in the tuple is zero or more.
- *Node* represents an activity in a workflow definition, such as tasks (e.g., Check Credit Card) and decisions (e.g., approve a purchase "if" the credit card is valid).
- *CFC* is the control flow connectors that link two *Nodes*.
- *DFC* is the data flow connectors that direct *Data* between *Nodes*.
- *Application* represents an executable program that fulfills one or more tasks. Some typical applications are an email client, a document manager, and interfaces to other enterprise applications.
- *Role* represents roles involved in the business process, such as Sales Manager, Customer, and Payment Manager in Figure 3.

Differences between workflow definitions of two versions can be identified by comparing the two sets of tuples. Based on the changes obtained, we can automatically produce a workflow configuration file and then execute a script for generating a new workplace configuration updates. Figure 5 sketches the process of generating new workplace configurations.

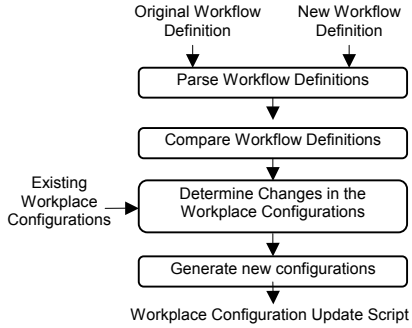

Figure 5: Generating new configuration updates for a workplace

As a result of changes in a workflow definition, new bindings between tasks and applications as well as new data interfaces may be introduced. Certain applications need to be removed from the workplace in the case that their related tasks are removed in the new workflow definition; some applications may be added in the event that new tasks use them in the workflow definition. As the example mentioned in Figure 3 and 4, we can determine which applications need to be replaced by comparing their binding information. For our example, the result of the comparison is as follows:

For the purchase order workflow definition, illustrated in Figure 3,

Binding (Purchase Form) = (Customer, {Order Request, Select Payment Method})
Binding (Order Management Application) = (Sales Manager, {Purchase Approval, Create Invoice})
OutData (Purchase Form) = {Product info}
InData (Order Management Application) = {Product Info}
OutData (Order Management Application) = {Invoice}

For the modified purchase order workflow definition, illustrated in Figure 4,

Binding (Purchase Form) = (Customer, {Order Request, Select Payment Method})
Binding (Credit Checking Application) = (Sales Manager, {Check customer credit})
Binding (Order Management Application) = (Sales Manager, {Purchase Approval, Create Invoice})
OutData (Purchase Form) = {Product info}
InData (Credit Checking Application) = {Product info}
OutData (Credit Checking Application) = {Purchase Record}
InData (Order Management Application) = {Purchase Record}
OutData (Order Management Application) = {Invoice}

By comparing these two processes, we find that the binding for the *Credit Checking Application* is added in Sales Manager's workplace. The interface for *Order Management Application* is changed. As a result, the workplace needs to install new *Account Management Application* and *Order Management Application*.

4.2 Workplace Dynamic Reconfiguration

In the proposed reconfiguration process, we aim to maintain the work consistency when a workplace is switched from one workflow definition to an updated one while minimizing the disturbance to the users' currently working within a workplace application. We first examine whether a currently running workflow instance is valid to continue execution under the new workflow definition. We identify a set of possible states that an application goes through in its life cycle. By combining the consistency of workflow instances and the states of their associated applications, we derive a dynamic reconfiguration process to carry out the workplace reconfiguration at run time.

4.2.1 Workflow Instance Consistency

Changes in a workflow definition may cause the currently progressing workflow instances to become inconsistent with the new workflow definition. As a result, each workflow instance must be verified for consistency under the new workflow definition. A variety of workflow instance validation approaches have been proposed in the workflow literature [15, 30, 31]. In general, a workflow engine provides several settings that allow a workflow engine to be reconfigurable to the new workflow definition. The reconfigurability settings of a workflow engine can be summarized in the following four cases.

Case 1: In the case that an existing workflow instance is specified to use the old workflow definition by a workflow engine, such a workflow instance can continue its execution without being affected.

In the case that a workflow definition is modified while the workflow instance from the old workflow definition is executing, the following two situations can occur in a workflow engine:

Case 2: The workflow instance can adapt to the new definition without being affected. This case is possible if the execution sequence of the workflow instance is valid under the new definition. For the workflow definition example described in Figure 3, the customer is permitted to pay by cheque, which is an additional payment method introduced in a new workflow definition. This change to the workflow definition does not affect the exiting payment methods. Therefore, the workflow instance under the old workflow definition is valid to execute under this new workflow definition.

Case 3: The workflow instance can adapt to the new definition. However, the application to execute the affected task is required to terminate. Consequently, the workflow engine rolls back the workflow instance to a previous state. We define this affected task as an inconsistency spot. For example, in the new workflow definition of Figure 4, the customer is only allowed to pay by cheque, and other payment methods are not permitted. The application enacting *Select Payment* task has to be terminated, and the workflow instance is rolled back to the *Order Request* task by a workflow engine. The support for the recovery of a workflow instance is required in a workflow engine for this situation.

Case 4: The workflow engine is set to terminate the workflow instance continues to execute under the old workflow definition.

When an inconsistency occurs in a currently progressing workflow instance, the workplace retrieves the reconfigurability settings of a workflow engine and decides whether to immediately switch to the new workflow definition or wait until the old workflow instances complete.

4.2.2 Life Cycle of an Application in Workplace

In general, the life cycle of an application that implements one or more tasks in a workflow definition contains five states: *Inactive*, *Suspended*, *Active*, *Completed and Terminated*, as illustrated in Figure 6.

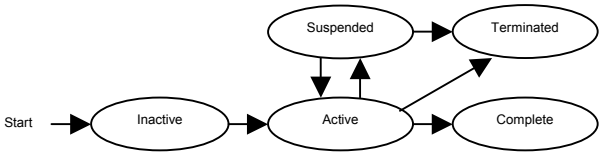

Figure 6: State transitions of an application in the workplace

Inactive refers to the state in which an application has not been yet activated (i.e. the starting condition has not been met yet). For example, a user has not started a task that is implemented by an application. *Active* represents a state in which the starting condition of an application has been met; the user is currently using it in the workplace. *Completed* denotes a state in which a task enacted by an application is completed, and user confirmed the

completion of the task. *Suspended* state means a task has not been completed, the application bound to the task is temporary suspended, and the user will continue to work on it. *Terminated* indicates the execution in the application has failed

From the perspective of a user's workplace, a task in a workflow instance is distributed to a user's workplace following the execution order specified in a workflow definition. The task is queued for service in a waiting list of the user's workplace. For the example illustrated in Figure 1, the *Create Catalog* task is listed in Work Items component, and waits for the user to trigger the corresponding application to complete this task. In this context, the application that enacts the task is either in *Active* or *Suspended* state. An active application indicates that the user is currently working on the task. A suspended application is either initiated by the current user or requested by other roles in the same workflow instance; but the user is not working on this task. Essentially, a user can actively engage in at most one active application that executes a task at any point.

4.2.3 Workplace Reconfiguration Criteria

In our reconfiguration approach, we identify a reconfiguration point at which a set of applications that are associated with tasks specified in new workflow definitions can be uploaded into the workplace. The aim of this reconfiguration approach is to minimize the inconsistencies to the users' work. For the example in Section 3, our reconfiguration process avoids the case that the sales manager has to verify the same purchase order twice, because of the changes to the workflow definition. We are concerned about active and suspected applications, since these applications could cause inconsistency to the state of the workplace after reconfiguration. An inactive application can be reconfigured at any point.

We utilize a path analysis of all currently running workflow instances for the same workflow definition and compare the relative position of the active/suspended applications to the inconsistency spots in the workflow definition. Based on this relative position, we define the reconfiguration criteria by distinguishing the following three scenarios.

- If the current active/suspended application is in the upstream of the inconsistency spot, the reconfiguration process updates the configuration for the new workflow definition. In this scenario, the users' current work can carry on in the new workflow instance. For the example in Figure 3, when a customer is at the point of the *Select Payment Method* task, the workplace can be safely reconfigured to adapt the changes if the inconsistency spots of the new workflow definition are located in the downstream of the *Payment Method* task.
- If the current active/suspended application is in the downstream of the inconsistency spot, the reconfiguration process updates the configuration for the new workflow definition. The user's current work can continue through the unchanged tasks in the new workflow definition. For example in Figure 3, when a sales manger is at the point of the *Create Invoice* task, the workplace can be safely configured to add the new binding information between *Create Credit Check* application and *Check Customer Credit* task. Because the binding between the *Create Invoice* task and *Order Management* application remains the same in the new workflow instance. The changes to the workflow definition will take effect the next time the workflow definition is initiated.
- If the current active/suspended application is in the inconsistency spot, we can take two approaches. As described in Case 3 in Section 4.2.1 in which the workflow engine decides to adapt the current workflow instance to the new definition by rolling back, the current application can be terminated, and the reconfiguration process updates the configuration for the new workflow definition. For the example in Section 3, if a sales manager is working on the *Purchase Approval* task, when the reconfiguration occurs. The *Purchase Approval* task is terminated. After the workplace reconfiguration, the sales manager will perform the *Check Customer Credit* task. However, the reconfiguration process can also follow Case 1 discussed in Section 4.2.1, which allows the old workflow instance to complete without changes.

Multiple workflow instances can exist in a user's workplace as the user is assigned tasks from different workflow definitions. The reconfiguration process is applied to workflow instances in which changes are introduced in the corresponding workflow definition. Other workflow instances under unchanged workflow definitions are kept intact. Once the reconfiguration point is determined, the reconfiguration process automatically loads the workplace configuration updates.

4.3 Proposed Framework for Self-Reconfigurable Workplace Applications

We propose a self-reconfigurable framework to allow a workplace application to automatically accommodate continuous and constant changes in business processes. The framework utilizes Web portal as a platform and builds on top of WFMS. A sample screenshot of a

prototype workplace is developed using this framework, as illustrated in Figure 1. The major components of the framework are depicted in Figure 7. The *Workflow* component acts as an interface to a workflow engine in the WFMS; it accepts tasks distributed by the workflow engine, and displays user interfaces of the application that executes the task to an end user. The *Work Items* component (corresponding to the *Work Items* in Figure 1) retains a list of tasks to be completed by a user. The *Process Progress Status* component (corresponding to the *Process Progress Status* component in Figure 1) is used to display the execution status of each workflow instance. The *Configuration* component stores the workplace configuration information in a database. The *Administration* component maintains user profiles and establishes the associations between roles and users.

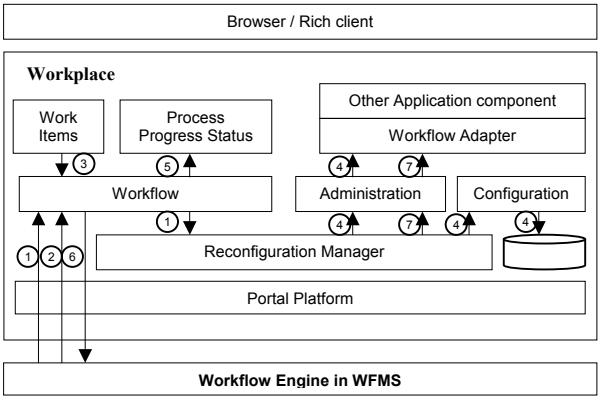

Figure 7: A Self-reconfigurable workplace framework

Each application in the workplace has an associated *Workflow Adapter* component which defines data and event interfaces to the workflow management system. The data interface specifies the data format used by the applications to exchange workflow relevant data information. The event interface is used to send and receive events from the workflow management system. In a sense, a *Workflow Adapter* component is a wrapper for the application, which handles the underlying communication issues.

The *Reconfiguration Manager* component implements the reconfiguration process discussed in Section 4.2. It discovers the changes made to workflow definitions and automatically modifies workplace configuration and the task binding information stored in the *Configuration* component. This is done through communication with the workflow engine and dynamically applying updates at run time. This dynamic binding checking improves the flexibility of the workplace application since the binding can be dynamically changed at run time.

4.4 Dynamic Reconfiguration Procedure

As a summary, our dynamic reconfiguration process is shown in Figure 7. It consists of 7 steps:

(1) Upon receiving a new definition from the back-end workflow engine, the *Workflow* component forwards the new definition to the *Reconfiguration Manager* component, which generates a new workplace configuration, as mentioned in Section 4.1.
(2) The *Workflow* component retrieves the workflow reconfigurability settings of the workflow engine, as mentioned in Section 4.2.1.
(3) The *Workflow* component verifies the workplace reconfiguration criteria for each workflow instance and determines if *active* and *suspended* applications are affected, as discussed in Section 4.2.3.
(4) The *Reconfiguration Manager* component performs the reconfiguration in the workplace. New applications are installed, and the bindings between the tasks and applications are updated by the *Configuration* Component at run time. The workplace also gives a notice to the user, and explains the outcomes to the tasks in the workplace.
(5) The *Workflow* component updates the status of the affected workflow instances.
(6) The *Workflow* component notifies the workflow engine that reconfiguration is complete. After this point, both the workplace and the workflow engine can resume their normal operations.
(7) In the case where an application is no longer required, the workplace removes the obsolete applications when there are no workflow instances which depend on the applications.

Now we come back to the example described in Section 3. Suppose there is a customer creating a new purchase order and is at the point of the *Select Payment Method* task when the workflow definition is changed in the WFMS. The workflow engine is set to adapt the currently progressing workflow instances to the new definition. The *Workflow* component verifies the workplace reconfiguration criteria and determines that the inconsistency spot (i.e., *Check Customer Credit* task) is in the downstream of the current active task (i.e., *Select Payment Method* task). The binding between the *Select Payment Method* task and the application, *the Purchase Order Website*, remains the same in the new workflow definition. In this context, the reconfiguration process that adapts the workplace to the new workflow definition will not affect the customer's interaction with the *Purchase Order Website*. Therefore, the new configuration of the workplace is generated by the *Reconfiguration Manager* component. The *Reconfiguration Manager* then performs

the dynamic reconfiguration. Once complete, the *Workflow* component notifies the workflow engine about the completion of the reconfiguration, and workplace then resumes its normal operation.

5. Case Studies

We developed a proof-of-concept prototype portal workplace using IBM Websphere Portal v5.0 [24]. The screenshot is illustrated in Figure 1. The portal workplace has 26 workflow definitions for a typical commerce application. The workplace prototype runs on an Intel Pentium 4 desktop computer at 2.66GHz with 1Gb of RAM. We listed two example workflow definitions in Table 1.We studied the feasibility of the proposed reconfiguration approach and examined the performance of reconfiguration processes, which is crucial to the user perception of the working environment.

Table 1: Delay introduced by reconfiguration

	Workflow Definition 1		Workflow Definition 2	
	Test Case 1	Test Case 2	Test Case 1	Test Case2
Number of workflow instances running	1	3	1	2
Number of running workflow instances affected	1	2	1	1
Reconfiguration Script Generation Time (ms)	4031	4094	4125	4344
Workplace Reconfiguration Time (ms)	53220	69392	568996	535963
Instance Reconfiguration Time (ms)	<1	<1	<1	<1
Work Items update Time (ms)	<1	<1	<1	<1
Experienced Deletion Time by Workplace User (ms)	<1	<1	<1	<1
Actual Deletion Time (ms)	6062	5422	72955	88830
Total Delay time (ms)	57251	73486	573121	540307

In the case studies, we aim to determine the following performance characteristics in our system: (1) The time needed for each step in the reconfiguration process (2) The effect of processing multiple workflow instances at runtime. In the first example (*Workflow Definition 1* column in Table 1), we tested a workflow definition that involves two tasks that are bound to one application. We conducted two test cases for this workflow definition. In the first case, there was one workflow instance running and it was affected by the reconfiguration process. In the second case, there were three running workflow instances, and two of them were affected by the reconfiguration process. In the second example of workflow definition (*Workflow Definition 2* column in Table 1), tasks are bound to five applications. The two test cases were conducted on the second workflow definition. In the first case, one workflow instance was affected by the reconfiguration process. In the second test case, two workflow instances are initiated and one workflow instance of them was affected by the reconfiguration process.

We found that the reconfiguration process takes the most amount of time (up to 568996ms or 7.5 minutes) during our experiments, as shown in the row of *Workplace Reconfiguration Time* in Table 1. Deletion operations also take time to complete, as listed in the row of *Actual Deletion Time* in Table 1. However, the actual deletion is performed when the workplace is idle. Therefore the experienced deletion time is less than 1 second, as depicted in the row of *Experienced Deletion Time* in Table 1. In addition, we observed that the instance reconfiguration time is negligible, as indicated in the row of *Instance Reconfiguration Time* in Table 1. Hence the total delay time in each case is the sum of the time measurement for each step except the actual deletion time.

Uploading new configurations in a Websphere Portal Server is an expensive operation, since it involves communicating with a back-end database. However, the reconfiguration process is conducted in the background, and only has impact on the user who is working on the affected workflow instances. However, the user can continue to work on other unaffected workflow instance during the reconfiguration process. We expect a performance increase by replacing the current desktop PC with a powerful enterprise level server. Adapting a running workflow instance to the new workflow definition is not time consuming, since we observed that handling multiple workflow instances does not increase the *Instance Reconfiguration Time*. The applications we used in our experiment are simple and do not access any back-end services. Furthermore, the workflow definitions studied do not involve large amount of data, hence the workflow instance reconfiguration time is short. However, we expect longer delays for handling workflow instances for applications involving large amount of database access. Nevertheless, we expect the delays not to be noticeable by the users of the workplace.

In summary, the results of case studies show that the reconfiguration process can be automatically performed without interrupting users' work. This reduces the cost of system interruption, and eases the maintenance process for a frequently evolving business application.

6. Related Work

Research issues relating reconfigurable workflow engines have been studied in the past decade. In [25, 26] a comprehensive study on flexibilities in workflow engine is given: In [5, 6, 7, 8, 9, 10], the concept of workflow change primitives and workflow consistency criteria are proposed. In [30, 31], the flexibilities are implemented in the workflow management system. However, these researchers only address the issues of maintaining workflow instance consistencies with respect to workflow definitions. None have examined the workflow execution environment, such as a workplace, to be dynamically updated with changes from the workflow definitions.

In [32, 35], the authors identified the issues to separating workflow definitions domain from organizational domain. They present a prototype workplace, which contains a task workspace and a work list interface. However, the problem of dynamically propagating changes from workflow definitions to workplace is not examined.

The AFLOWS workflow management system [29] utilizes agent technologies for task collaboration. The binding between workflow tasks and applications are computed dynamically at runtime. The GENESIS system [21] is a workflow management system that supports cooperative distributed environments. The exception handling process was performed either automatically, or through manual intervention. However, the behavior of the user interface of the system during exception process was not mentioned in the paper. Our approach aims to minimize the impacts on user's experience during the dynamic workplace reconfiguration.

The web service oriented workflow execution and collaboration is the emerging area of research in this field [27, 33]. Their major focus is on the underlying service execution collaboration, not at the user-interaction level.

7. Conclusion

In this paper we presented a framework for adapting workplace applications to meet the changing nature of business domains. We proposed a workflow meta-model that captures the changes in workflow definitions, such as workflow node addition and deletion, or connector addition and deletion. We defined a reconfiguration process that allows a workplace to automatically adapt to these changes at run-time. A prototype is developed to highlight the feasibility of our approach. The case studies show that the reconfiguration process can be automatically performed without interrupting users' work.

The future work of this research includes examining the performance and management issues, and refining the current approach to incorporate more usability and reliability features.

References

[1] Edward A. Stohr, J. Leon Zhao, "Workflow Automation: Overview and Research Issues" Information Systems Frontiers, September 2001, Volume 3 Issue 3

[2] http://www.ibm.com/software/integration/wbimodeler

[3] Matthias Vering, Grant Norris, Peter Barth, James R. Hurley, Brenda Mackay, David J. Duray, Matthias Vering, The E-Business Workplace, John Wiley & Sons, Inc, June 2001

[4] Corporate Portals Empowered with XML and Web Services, Butterworth-Heinemann Newton, MA, USA, 2002

[5] Casati, S. Ceri, B. Pernici, and G. Pozzi. "Workflow Evolution" In Proceedings of ER '96, pages 438-455

[6] M. Weske, "Formal Foundations and Conceptual Design of Dynamic Adaptations in a Workflow Management System" Proceedings of the 34th Annual Hawaii International Conference on System Sciences, January 2001

[7] Formal Verification of Workflow Schemas, C. Karamanolis, D. Giannakopoulou, J. Magee, S. M. Wheater, Submitted for publication to the 12th Conference on Advanced Information Systems Engineering (CAiSE 2000), October 1999

[8] Classen, I., Weber, H. and Yanbo Han "Towards Evolutionary and Adaptive Workflow Systems" In Proceedings of First International Enterprise Distributed Object Computing Workshop(EDOC '97), 24-26 Oct. 1997

[9] Shazia Sadiq "Workflows in Dynamic Environments – Can they be managed?" Proceedings of the Second International Symposium on Cooperative Database Systems for Advanced Applications (CODAS99), Woollongong, Australia. March 27-28, 1999.

[10] S. Rinderle, M. Reichert, P. Dadam, "Flexible Support of Team Processes By Adaptive Workflow Systems" Distributed and Parallel Databases, Kluwer Journal, 16, pp 91-116, 2004

[11] Joao Paulo A Almeida, Maarten Wegdam, Marten van Sinderen, Lambert Nieuwenhuis, "Transparent Dynamic Reconfiguration for CORBA" In Proceedings of the 3rd International Symposium on Distributed Objects and Applications, pages 197--207. IEEE Computer Society, September 2001

[12] Abdelmadjid Ketfi, Noureddine Belkhatir, "Open Framework for Dynamic Reconfiguration of Component-based Software", Proceedings of the 2004 International Conference on Software Engineering Research and Practice (SERP'04), Monte Carlo Resort, Las Vegas, Nevada, USA, June 21 - 24, 2004

[13] Keith Whisnant, Zbigniew T. Kalbarczyk, Ravishankar K. Iyer "A System Model for Dynamically Reconfigurable Software" IBM Systems Journal, Volume 42, Number 1, 2003

[14] J. Dowling and V. Cahill, "Dynamic Software Evolution and the K-Component Model" Workshop on Software Evolution, OOPSLA. 2001

[15] De Palma N., Bellissard L., Riveill M., "Dynamic Reconfiguration of Agent-based Applications" ,European Research Seminar on Advances in Distributed systems (ERSADS'99), Madeira, Portugal, April 1999

[16] http://www-128.ibm.com/developerworks/library/ws-bpel

[17] http://www.wfmc.org/standards/XPDL.htm

[18] http://www.wfmc.org/standards/docs/if2v20.pdf

[19] "Workflow Reference Model", The Workflow Management Coalition Specification, http://www.wfmc.org/ standards/docs/tc003v11.pdf

[20] H. Liu, M. Parashar and S. Hariri, "A Component-based Programming Framework for Autonomic Applications," Proceedings of the 1st IEEE International Conference on Autonomic Computing (ICAC-04), IEEE Computer Society Press, New York, NY, USA, pp. 278 - 279, May 2004

[21] L. Aversano, A. De Lucia, M. Gaeta, P. Ritrovato, S. Stefanucci, M.L. Villani, "Managing Coordination and Cooperation in Distributed

Software Processes: the GENESIS Environment", Software Process: Improvement and Practice, vol. 9, no. 4, 2004, pp. 239-263.
[22]http://www.lotus.com/products/product5.nsf/wdocs/workplacehome
[23] M. Hammer and J. Champy, J., 1993. Reengineering the Corporation: A Manifesto for Business Revolution. HarperCollins, New York.
[24] http://www.ibm.com/software/genservers/portal/
[25] Heinl, P., Horn, S., Jablonski, S., Neeb, J., Stein, K. and Teschke, M., "A Comprehensive Approach to Flexibility in Workflow Management Systems", Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration, 1999, San Francisco, California, USA, February 22-25, 1999, ACM 1999, pp79-88
[26] Jørgensen, H. D., "Interaction as a framework for flexible workflow modeling", Proceedings of International ACM SIGGROUP Conference on Supporting Group Work, Boulder CO, September. 30 – Oct 3, 2001
[27] Khalid Belhajjame, Genoveva Vargas-Solar, Christine Collet: "Defining and Coordinating Open-Services Using Workflows" ODBASE 2003: 110-128
[28] GRASSO, A., MEUNIER, J.-L., PAGANI, D., and PARESCHI, R., "Distributed coordination and workflow on the World Wide Web", Computer Supported Cooperative Work: The Journal of Collaborative Computing, 1997, pp 175-200.
[29] K. Belhajjame, G. Vargas-Solar, and C. Collet. "A flexible workflow model for process-oriented applications", Proceedings of the 2nd International conference on Web Information Systems Engineering, WISE'2001, Kyoto-Japan, December 2001.
[30] J.J. Halliday, S.K. Shrivastava, and S.M. Wheater. "Flexible Workflow Management in the OPENflow System", In Proceedings of the 4th International Enterprise Distributed Object omputing Conference(EDOC 2001), 4-7 September 2001, Seattle, Washington, , pages 82–92.
[31] Santosh K. Shrivastava, Stuart M. Wheater, "A transactional workflow based distributed application composition and execution environment" ACM SIGOPS European Workshop 1998, 74-81
[32] Halliday, J.J., Shrivastava, S.K., and Wheater, S.M. "Implementing support for work activity coordination within a distributed workflow system" In Proceedings of the Third International Conference on Enterprise Distributed Object Computing (EDOC '99), (Sept. 1999), 116–123.
[33] Chiu, D.K.W., Cheung, S-C., Karlapalem, K., Li, Q., Till, S.: "Workflow View Driven Cross-Organizational Interoperability in a Web-Service Environment" In Proc. of Web Services, E-Business, and the Semantic Web, CAiSE 2002 International Workshop, WES 2002, Canada, 2002.
[34] L. Aversano, G. Canfora, A. De Lucia, S. Stefanucci, "Automating the Management of Software Maintenance Workflows in a Large Software Enterprise: A Case Study", Journal of Software Maintenance and Evolution: Research and Practice, vol. 14, no. 4, 2002, pp. 229-255.
[35] A. Sheth, "From contemporary workflow process automation to adaptive and dynamic work activity coordination and collaboration", Proceedings of the 8th International Workshop on Database and Expert Systems Applications, p.24, September 01-02, 1997