Improving the Usability of e-Commerce Applications Using Business Processes

Ying Zou, *Member*, *IEEE Computer Society*, Qi Zhang, and Xulin Zhao, *Student Member*, *IEEE*

Abstract—e-commerce applications automate many daily business activities. Users interact with e-commerce applications through menu-driven User Interface (UI) components such as toolbars, dialogs, and windows. However, the tremendous number of functionalities may overwhelm the users. Users struggle to locate the appropriate UI components to accomplish the tasks required by business processes. In this paper, we enhance e-commerce applications by improving their usability using the knowledge embedded in business process definitions. Our improved application provides contextual information to fulfill each business task. The improved application guides users through the various tasks in a step-by-step fashion. Through a controlled experiment, we demonstrate that our improved application offers a better usability experience for novice users by giving them more guidance and reducing the time needed to locate the next UI component in a complex UI.

Index Terms—Graphical user interface, user interface reengineering, business process, process definition, usability.

1 Introduction

A business process is a sequence of tasks carried out to achieve the business objectives of an organization. For example, a book purchasing process may consist of several tasks such as selecting a book from the catalog, paying with a credit card, and printing a receipt. A process definition specifies the business tasks (for example, selecting a book), roles (for example, customers and sales representatives), and data (for example, a book order request) involved in a business process.

E-commerce applications support and execute business processes for various business domains such as call centers and online retail stores. To fulfill the growing business requirements, e-commerce applications have gradually evolved to provide sophisticated functional features through their graphical user interfaces (GUIs). Users perform daily business activities by interacting with e-commerce applications. However, locating the appropriate UI components, such as toolbars, dialogs, and windows, is often not obvious to the users. Consequently, users, especially novice users, struggle to determine the most appropriate UI components to accomplish a particular task. Users require continuous guidance when using e-commerce applications because the UIs are frequently updated to reflect the continuous evolution of the underlying business processes. In addition, the UIs of e-commerce applications are designed from the perspective of the IT personnel rather than from the perspective of the business users [62]. The design of navigation within UIs may not follow the users' natural work rhythm. The UIs of current e-commerce applications have many problems. Two particular problems are determining the steps to follow in a complex UI and inconsistencies between a business process and the UI's implementation. These problems increase operational costs and decrease user productivity. A more systematic approach for designing e-commerce applications is needed to allow users to carry out their work more efficiently.

In this paper, we propose an approach to improve the usability of e-commerce applications. The approach leverages information stored in process definitions (for example, task processing sequences and role information) to improve the UI of an existing e-commerce application. Our improved application guides users who are interacting with an e-commerce application by prompting the next UI component. Our approach dynamically displays the UI components that are relevant to the tasks and hides the irrelevant UI components. This paper extends our earlier work presented at the 22nd IEEE International Conference on Software Maintenance (ICSM 2006) [70]. In this paper, we study the benefits of our approach through a controlled usability experiment.

Section 2 of this paper describes sample scenarios, which demonstrate the limitation of the existing UI of an e-commerce application. The scenarios also highlight the benefits of our proposed approach. Section 3 introduces the information recorded in a business process definition. We focus primarily on the information relevant to our approach. Section 4 presents our approach. We describe the techniques that are used for establishing links between the business tasks and the UI components and for generating navigation sequences for the UI. We also present the architecture of our business-process-driven UI. Our proposed approach restructures existing e-commerce applications to fit within this architecture. This architecture provides runtime support and tracks the interaction of

- *Y. Zou and X. Zhao are with the Department of Electrical and Computer Engineering, Queen's University, Walter Light Hall, 19 Union Street, Kingston, ON, K7L 3N6, Canada. E-mail: ying.zou@queensu.ca, 4xz5@qlink.queensu.ca.*
- *Q. Zhang is with the Devid R. Cheriton School of Computer Science, University of Waterloo, 200 University Ave. West, Waterloo, ON, N2L 3G1, Canada. E-mail: q8zhang@cs.uwaterloo.ca.*

Manuscript received 19 Dec. 2006; revised 2 May 2007; accepted 12 June 2007; published online 25 June 2007.
Recommended for acceptance by D. Binkley.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0285-1206.
Digital Object Identifier no. 10.1109/TSE.2007.70708.

0098-5589/07/$25.00 © 2007 IEEE Published by the IEEE Computer Society

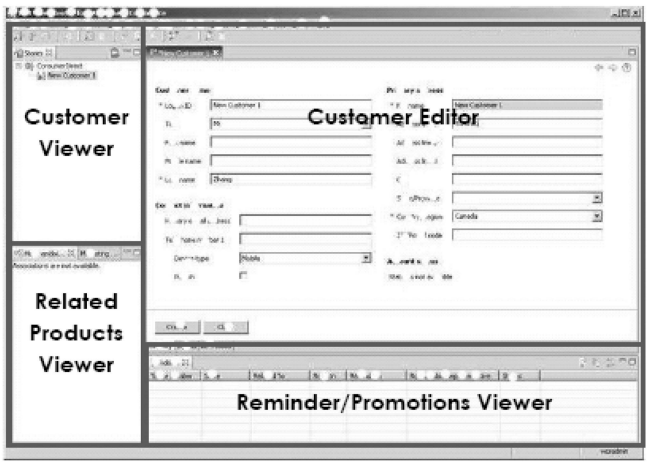

Fig. 1. Annotated UI of a call center application. (Figs. 1, 3, and 5 are blurred to avoid disclosing sensitive information.)

users with UI components. Section 5 evaluates the accuracy of our automatic restructuring. Section 6 presents a controlled experiment that is used for evaluating the usability of the improved UI generated by our approach. Section 7 contrasts our approach with related work in business process analysis, business process reengineering, business process automation, UI reengineering, and Web-application reengineering. Section 8 concludes this paper and explores future work.

2 Sample Scenarios

In this section, we illustrate several usability problems for e-commerce applications through an example e-commerce application. Generally, usability is a software quality attribute that quantifies the ease with which a user can use a tool in a specified context [9], [29], [30]. Usability consists of five attributes:

1. *learnability*, which measures the ease of learning the functionality of an application,
2. *low error rate*, which measures the number of mistakes that users make while using the application,
3. *memorability*, which measures the ease of remembering the functionality of the application,
4. *efficiency*, which measures the ease of use and the level of productivity that the users of the application can attain, and
5. *user satisfaction*, which measures the enjoyment of the users who are using the application.

Fig. 1 shows an annotated screen shot of a call center application. Customer Service Representatives (CSRs) use the application for creating and managing customer orders when receiving phone calls from customers. The application is used for creating and managing customer orders over the phone. The screen shot is annotated to show the following UI components:

1. a *customer viewer* displays the stores and the current customers registered in each store,
2. a *customer editor* is used for editing the records for a customer,

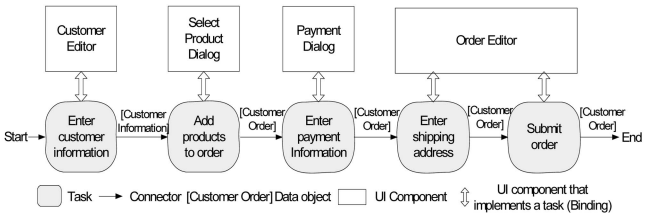

Fig. 2. Bindings between business process tasks and UI components in a purchase order process.

3. a *related-products viewer* offers similar or related products that a customer may be interested in purchasing,
4. a *reminder viewer* displays any reminders for a CSR, and
5. a *promotions viewer* presents promotions for a product.

Due to limited screen space, the reminder viewer often covers the promotions viewer (as shown in Fig. 1).

Such a call center application supports approximately 30 business processes. A UI component, such as a menu, an editor, or a viewer, is used for fulfilling one or more business tasks. Fig. 2 shows the definition of a typical "Purchase Order" business process for this call center application. To fulfill this process, a CSR must navigate through different UI components. A CSR first searches for the menu that is able to initiate the customer editor. A CSR, especially a novice one, needs to be familiar with the functionality of many UI components (for example, buttons in the menu and the toolbar) and the applicability of each UI component to the task. After entering the customer information in the customer editor, the CSR needs to select the correct UI component to activate the "Select Product" dialog to add products. At this point, no help is available for a CSR who must locate the UI components needed to complete each task within a business process. In short, a CSR must memorize the functionality of every UI component so that he can perform every task in a business process. During a phone order from a customer, a CSR usually opens numerous windows and dialogs to handle multiple tasks within different processes. A CSR has to recall the correspondence between each active task and the UI components. Recalling this correspondence would likely result in the loss of CSR efficiency and an increased error rate.

The related-products viewer and promotions viewer display data only when a product is added to an order in the "Add Product to Order" task. However, when no data is available for display, these viewers remain open, as shown in Fig. 1. These unused viewers, if not closed by the user, reduce the already limited screen space. A CSR may be unaware of the availability of data in these viewers, especially when the CSR places the promotions viewer and the reminder viewer underneath other UI components to save screen space. The current UI fails to capture the work context of a CSR and the UI does not adjust to help users work more effectively.

In our research, we intend to support the CSR in the above scenarios. We use information from the business process definitions to provide UI navigational and con-

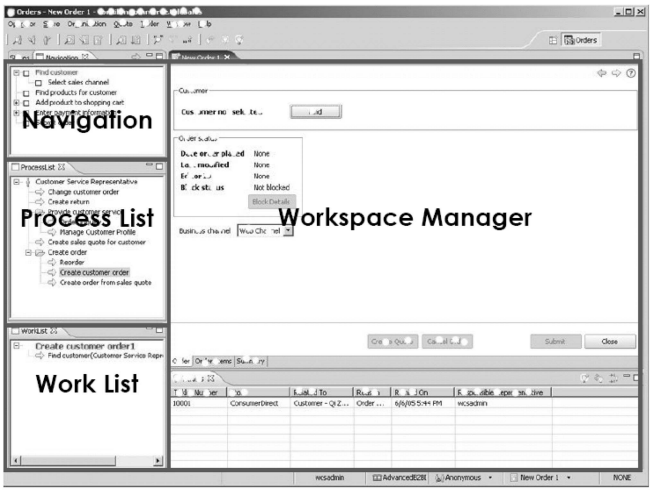

Fig. 3. Annotated screenshot of the improved UI for the call center application.

textual support for the CSR. Fig. 3 shows an annotated screen shot of the improved UI for the call center application. The improved UI offers the following features:

- support for evolving business processes,
- support for context awareness, and
- support for automatic navigational transitions.

2.1 Support for Evolving Business Processes

To ensure that the implementation of the UI and the underlying business processes are consistent, we generate the appropriate content of the UI based on the definition of the corresponding business process. The framework for generating UIs is presented in [71] and [74]. We automatically generate the following components that are essential in accomplishing the tasks within business processes:

1. **Workspace manager.** This provides a screen space to load the UI for back-end business applications. The workspace manager assists a user in accomplishing all business tasks related to the roles (for example, CSR and Manager) assumed by the user. The workspace content, shown in Fig. 3, is taken directly from the UI of the existing call center application. The bindings between UI components and business tasks are specified in a configuration file.
2. **Process list.** This displays a set of processes in which a role assigned to the current user needs to perform the first task. If the first task in the process is not performed by the role assigned to the user, this business process is not shown in the process list. Once a user selects a process name from the process list, a process instance is spawned. A reference to the process instance is placed in the work list.
3. **Work list.** This displays a set of task instances which require the interaction of the current user. These task instances are grouped by the process instances to which they belong. The task instances are displayed in the work list based on the progress of the various process instances in the system. A task instance in the work list can be initiated by the current user or can be forwarded from other process instances that are initiated by other users through their own UI. If a process instance is at a task that does not require the intervention of the current user, then the task instance is not listed in the work list of that user. At any time, a user may select a task instance from the work list to work on. Once a task instance is selected, the UI components corresponding to this task instance are launched in the workspace. A user can switch between tasks in different process instances by using the work list.
4. **Navigation.** This displays a simplified view of the structure of a process definition. The navigation component indicates the progress of a process instance and highlights the task that is currently in progress.

2.2 Support for Context Awareness

Context awareness is the ability of a UI to sense and analyze the context by using various sources. The UI then takes different actions based on the current context. From the perspective of an e-commerce application, the availability of UI components is determined by using different contexts, such as the involved business processes, the resources available to execute a task, and user information. To improve the use of the limited screen space, UI components react dynamically based on the underlying business processes. The improved UI allows users to focus on the current tasks through relevant menus. Context awareness allows the UI to provide proactive assistance, which increases the users' efficiency [57]. For example, when data is available for display in the promotions viewer, the UI automatically opens the viewer. When the user switches from one process instance to another, the improved UI automatically displays the relevant information for the latter process instance and hides the information relevant to the former process instance.

2.3 Support for Automatic Navigational Transitions

To avoid the need to memorize the functionality of UI components, we provide users access to the appropriate UI components. We use the binding information between tasks and UI components to divide a business process into segments. Each segment may contain a sequence of tasks that are performed by the same UI component. If a task is mapped to more than one UI component, we provide step-by-step task instructions in the UI to improve the learnability of the UI [57]. For example, in Fig. 2, the "Enter Shipping Address" and "Submit Order" tasks are in the same segment since they use the same UI component (that is, the order editor). In this case, once a CSR completes the "Enter Shipping Address" task in the first page of the order editor, the workspace manager automatically switches over to the second page. The automatic transition allows the user to continue with the order submission without wasting time searching for the subsequent UI component.

3 Business Processes

The Workflow Management Coalition (WfMC) provides standard definitions for business processes [68]. There are three types of annotations that are used for describing business processes:

1. **Activities (that is, tasks**). These define how the work is actually done. An activity can have arbitrary granularity. Tasks can be automatically executed by applications or manually performed by humans. An activity can represent a trivial task, such as "Selecting Payment Methods," or a more complex task, such as "Validating a Credit Card." Users are assigned to roles such as CSR or supervisor. Each task is assigned to a particular role. Users acting as different roles must collaborate to fulfill a business process. For the example business process shown in Fig. 2, all of the tasks are assigned to the CSR role. An "Authorize Special Offers" task may be added after the "Enter Customer Information" task. The new task would be assigned to a supervisor role. The supervisor must approve the special offers before the CSR can communicate them to the customer.
2. **Control flows.** These define routing constraints for executing activities, including a sequence of activities, OR-relations, parallelisms, iterations, preconditions, and postconditions. An OR-relation describes a single thread of control which selects an execution path among two or more alternatives. Parallelism allows two or more threads of control to proceed autonomously and independently until all of the threads of control are merged once they are completed. Iterations are used for repeating the execution of one or more tasks. Preconditions and postconditions represent entry and exit criteria for a particular task.
3. **Data flows.** These specify the input, output, or both of an activity. Data flow also indicates dependencies between two related tasks. For example, when purchasing flight tickets, the ticket is the data, which flows from the "Search for Flights" task to the "Make Payment" task. The output of the "Search for Flights" task (that is, ticket availability) is the input of the "Make Payment" task. An example of data flows, shown in Fig. 2, is the "[Customer Information]," which is an output of the "Enter Customer Information" task and an input of the "Add Products to Order" task.

4 Our Approach for Improving UI

In a typical usage scenario of the improved UI, as discussed in Section 2, a user starts a process instance by selecting a process name from the process list shown in Fig. 3. The UI components for the first task in the selected process instance are launched in the workspace area. Once a task is completed, the corresponding UI components are closed and the UI component for the subsequent task is launched. As the user progresses through the process, the improved UI updates the contents of the work list and the navigation components. Creating such a UI requires a static-analysis phase and a dynamic execution environment.

In the static-analysis phase, we analyze the process definitions and the source code of the e-commerce application to determine the following information:

1. **The different roles and their tasks.** We analyze the definitions of all business processes in an application to recognize the various roles in these processes. A user can be assigned to multiple roles. Once we recognize the various roles assigned to a user, we can create a personalized UI for each user and ensure that each user interacts only with the relevant UI components. For example, the UI for the supervisor of a CSR would list the supervisor's tasks. The supervisor's UI would be updated when a CSR enters the customer data in the CSR's UI. The supervisor may then authorize special promotions for that particular customer. After the authorization, the UI for the CSR would show the updated offers. The CSR could then communicate these offers to the customer.
2. **The bindings between tasks and the UI components which implement these tasks.** We analyze the source code of the UI components to identify the bindings between tasks and UI components. Using our static-analysis results, we can insert events into the source code of the UI components to allow us to track, during runtime, the progress of each process instance.

Fig. 4 illustrates the high-level architecture of the dynamic execution environment. Using the information gathered from the static-analysis phase, we can create and update the content of the UI components (for example, process list) as a user progresses through a process instance. An event engine provides navigation transitions and context awareness for the UI. The event engine launches applications by issuing function calls to the UI components which implement a task. The event engine listens to the task events triggered by UI components to determine the executing task in a process instance. The event engine communicates the status of in-progress process instances through update events to the process list, the work list, and the navigation component. Examples of requests are starting a new process instance and switching between process instances.

In the following sections we elaborate on the following:

1. the steps required to recover roles and map roles to tasks,
2. the techniques used for binding the tasks to the UI components which implement these tasks, and
3. the operations of our dynamic execution environment.

4.1 Recovering Roles and Their Mappings to Tasks

We analyze all process definitions and extract information related to each defined role. The information includes tasks conducted by a role, control flows (which regulate the execution of the tasks), and data flows between the tasks. A process may contain subprocesses which describe activities

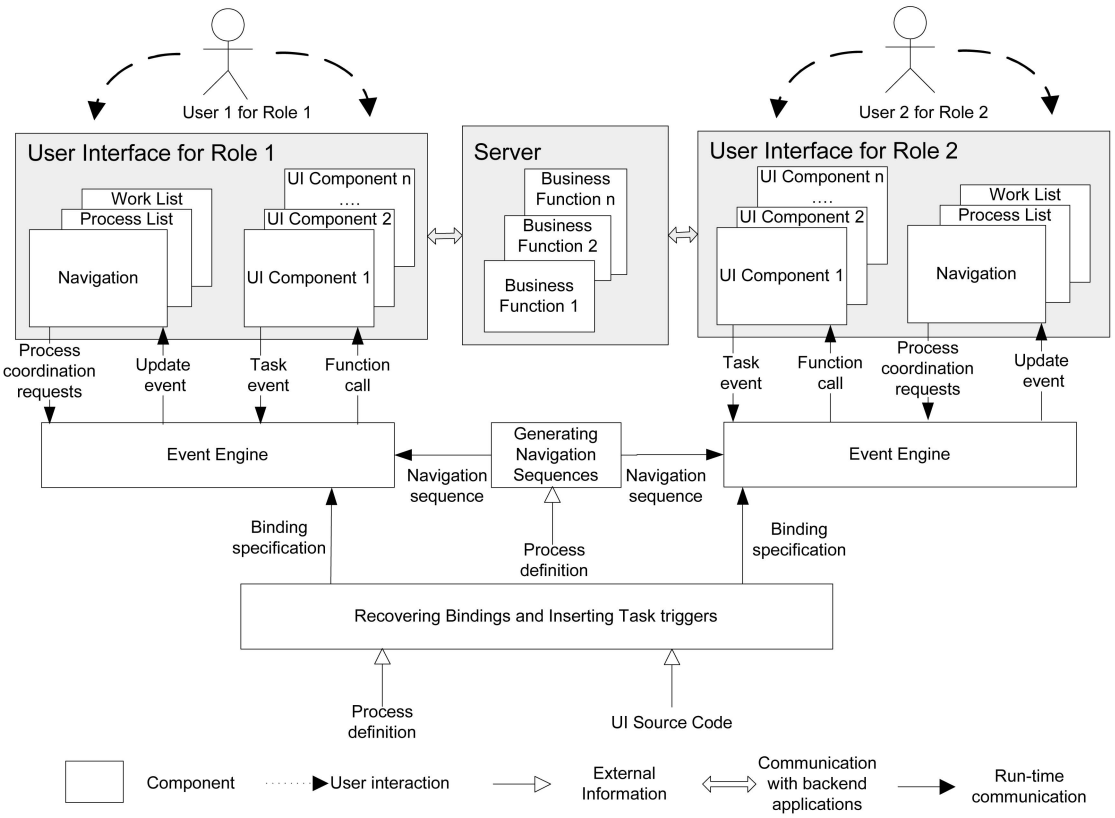

Fig. 4. Architecture for our business process driven user interface.

or tasks to be reused in other business processes. To accurately describe the tasks pertinent to a role, we further expand the subprocesses and identify role information in the subprocesses. Some tasks, the so-called human tasks (for example, "Specifying Credit Card Information"), require human interaction and thus are associated with UI components for intercepting input from the users. Other tasks, the so-called automatic tasks (for example, "Validating Credit Card"), are invoked automatically without human interaction and therefore are not related to any UI components. The navigation component displays human tasks and their execution orders relevant to a specific role. To improve the applicability of our approach to different business process specification standards (for example, XML Process Definition Language (XPDL) [69] and Business Process Execution Language (BPEL) [7]), we store the role information in a common format that is independent of a particular business process specification standard.

4.2 Recovering Bindings between Tasks and UI Components

The top-level UI components, such as windows and dialogs, hierarchically contain a set of low-level UI components such as buttons, labels, and text fields. Windows and dialogs are identified by their titles. In the case of Microsoft Office Word, the dialog for opening a file is titled "Open." Menus and other UI components have a fixed set of properties such as size, label, and background color. For example, a cancel button is labeled with "Cancel" as its screen name. A UI accepts input from a user, generates system events which trigger back-end functions, and displays the results to the user. The system events are predefined events associated with a set of hierarchical UI components such as windows, dialogs, and menus.

A process definition captures high-level abstractions of an application's execution flows. A task represents the lowest level of detail in a process definition. The functionality of a task is implemented using one or more UI components. The granularity of UI components matches well with the high level abstraction of a task in a process definition. Therefore, we capture the correspondences between the existing UI components and the tasks specified in process definitions. Once we identify the correspondences, we insert task event statements in the identified code blocks. The inserted statements trigger task events during the execution of a UI component. These triggered events are used by our dynamic execution environment to provide context awareness and navigational support for the executing business processes. The detailed steps are described in the following.

4.2.1 Recovering Task-UI Bindings

A task in a business process is described in terms of its name, input data, output data, and internal attributes (for example, human task or automatic task). Since the structure of the source code for the UI is highly dependent on the UI platform, we use the structure of a business process and the literal string description of tasks to locate the code that implements a particular task. We match similar literal words used in the task names and the task descriptions with the source code for the UI components. By examining the design and implementation of the UI of an e-commerce application, we identify the following generic matching rules:

Fig. 5. Example mappings between tasks to UI components.

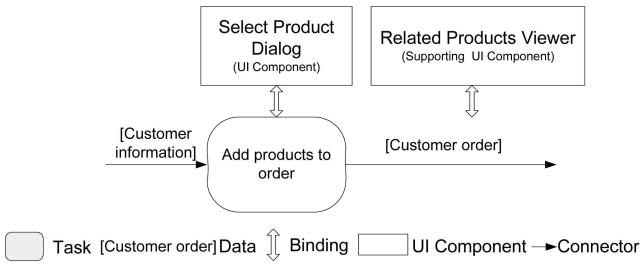

Fig. 6. Supporting UI component bound to data.

1. **Match the task name with the names of UI components.** This case occurs when one or more UI components, such as windows and dialogs, are dedicated to performing a task. For example, a "Find Order" task is implemented in the "Find Order" dialog UI code. This dialog is used for finding orders. Therefore, the "Find Order" task is bound to the "Find Order" dialog.
2. **Match the task name with the name of a widget in a UI component.** To improve the efficiency of a UI component, developers group a few tasks in one window or dialog to optimize the UI's delivery of information. Therefore, a task can be performed by one or more UI components. For example, as depicted in Fig. 5, a user can enter payment information via a composite UI component within a window. The composite UI component is composed of several UI components, including a group widget (that is, the boundary box with the title of "Payment"), a table, and three buttons (that is, "Add," "Edit," and "Delete"). The hierarchical structure of the UI indicates that the group widget encapsulates the table widget and the three button widgets. Moreover, the title of the boundary box (that is, "Payment") is similar to the task name (that is, "Enter Payment Information") in the "Purchase Order" process. In this case, we leverage the hierarchical structure and grouping relations to match a task name with a group of UI components.
3. **Match the task name with the data items in a UI component.** Some complex UI components, such as Tree and Table, require additional data items to describe the content of a UI component. For instance, as shown in Fig. 5, the table consists of several columns with names such as shipping description, shipping carrier, and estimated shipping. In this case, we further examine the names of the data items specified in a UI component and match a task name with the internal information associated with a UI component.
4. **Match the name of a task with the names of function calls to back-end functions of UI components.** For instance, the "Create Order" task may be bound to an "Order Editor" window if the "Order Editor" window invokes the function CreateOrder() as a back-end function.
5. **Match the input and output data of a task with the data required or produced by a UI component.** The UI of an e-commerce application often provides supporting UI components such as image viewers and product promotion reminders, which are integrated into the UI. These supporting UI components do not directly correspond to tasks in a process definition. Typically, such supporting UI components are useful in specific contexts. For example, when a customer is purchasing a product, a promotion viewer can prompt discount information for the CSR once the product is selected. At other times, the content of the promotion viewer is unavailable. To ensure efficient use of screen space, we display the supporting UI components based on the availability of content. We analyze the data dependencies between the UI components that display data requested by a task or produced by a task and the UI components that perform that task. In particular, the input and output data of a task serve as a starting point to locate the supporting UI components. We bind the supporting UI components with the input and output data of the task. As illustrated in Fig. 6, when data is generated from one UI component and passed into the supporting UI component, we launch the supporting UI component.

We define a binding specification which captures the UI components that implement a task. We use the "Purchase Order" process shown in Fig. 2 as an example to describe the format of our binding specification. As shown in Fig. 7, a ProcessBinding element specifies the name of a business process. For each task in a business process, a TaskBinding element specifies the associated UI components, which are known as UIComponent tags. The Name of the UIComponent refers to the name of the source code class which implements the UI component. The template of such a configuration is automatically generated from our static analysis of the source code and the process definitions.

4.2.2 Inserting Task Event Triggers

Once we identify the bindings between the tasks and UI components, we determine the location in the source code, where a task event is triggered when a UI component is executed. A task event is used for indicating the current active task in a process instance. Every task is linked with a unique event which is generated directly from the unique

```
<Configuration>
<ProcessBinding ProcessName="Purchase Order">
 <TaskBinding  TaskName="Enter customer information"
          Optional="false">
    <UIComponent  Name="UI.Editor.CustomerEditor" />
 </TaskBinding>
 <TaskBinding TaskName="Add products to order"
          Optional="false">
    <UIComponent Name=" UI.Dialog.SelectProductDialog"  />
 </TaskBinding>
 <TaskBinding TaskName="Enter payment information"
           Optional="false">
   <UIComponent Name=" UI.Dialog.PaymentDialog" />
 </TaskBinding>
 <TaskBinding TaskName="Enter shipping address"
            Optional="false">
  <UIComponent Name="UI.Editor.OrderEditor" />
 </TaskBinding>
 <TaskBinding TaskName="Submit order" Optional="false">
    <UIComponent Name="UI.Editor.OrderEditor" />
  </TaskBinding>
 </ProcessBinding>
</Configuration>
```

Fig. 7. An example binding specification for the purchase order process.

task name, which is specified in the process definition. Each event has three types: "start," "end," and "cancel." A start task event is triggered when a UI component is initiated to execute a task. Similarly, when a UI component is closed, an end task event is triggered. The cancel task event is triggered when a UI component is cancelled. We have adopted three heuristics to identify the location to insert task event triggers:

1. The start task event trigger is inserted where a user performs a menu action such as pressing a button, selecting a menu item, or choosing an option in a combo box in a UI component. The position to place a start task event trigger depends on the granularity of the UI component implementing the task. We consider two cases. In the case where a task is implemented by an entire UI component, the start task event trigger is located at the point where the UI component is about to be displayed. For example, the "Find Customer" task starts when a "Find Customer" dialog pops up. In the case where a task is carried out by a widget, the start task event is located at the point where the widget is clicked. For instance, the "Submit Order" task is launched when a "Submit" button is clicked in an "Order Editor" window.
2. The end and cancel task event triggers are usually located where the output data of a task have been derived. A task may produce output data when a task is completed. In this case, we examine the location of the output data from a task and place either the end or cancel task event trigger based on the availability of the data. For example, if a "Find Customer" task is performed in a "Find Customer" dialog, then the task ends when customer information is retrieved in the "Find Customer" dialog. Otherwise, the "Find Customer" task is cancelled if no customer information is returned.
3. We do not insert triggers for *optional* tasks which do not require user interaction. Developers may choose to provide default values or settings for UI components such as combo boxes and text boxes. In this context, a user only needs to verify if the default value is appropriate and may not need to perform any actions in the UI. We call this type of task *optional* tasks. The remaining tasks, which require the attention of a user, are called mandatory tasks. Typically, optional tasks can be detected by determining the default values of widgets when a UI component is initialized. At runtime, the event engine determines the completion of an optional task once the engine receives the trigger for a start task event of the task subsequent to the optional task.

```
public class TaskEvent {
  // Event name
  public String Name;
  // Event type, such as Start, End, etc.
  public String Type;
  // Business process related data for each task
  TaskContext context;
}
```

Fig. 8. Task event definition.

```
// In source code of UI component, at trigger point
TaskEvent event = new TaskEvent(
    "Enter customer information", "Start", null);
IEventHandler.getInstance().handleTaskEvent(event);
```

Fig. 9. Triggering a task event in the source code of a UI component.

Once the locations for inserting triggers for task events are identified, a wizard tool helps developers by automatically inserting statements that trigger task events. A task event has three attributes, including name, type, and context, as shown in Fig. 8. The TaskName is the name of a task which triggers the event. The Type determines the type of the task event trigger. The TaskContext holds a set of data related to a process instance which can be passed to the receivers of the event. Examples of data are the parameters to the event engine and the process instance identifier. Fig. 9 gives an example to demonstrate how a task event is sent to the event engine. A trigger for the start task event of the "Enter Customer Information" task is sent to an IEventHandler, which is a reference variable to the event engine.

4.3 Our Dynamic Execution Environment

To improve the usability of the UI, our dynamic execution environment guides users through a sequence of UI components to accomplish tasks within a business process. The dynamic execution environment also provides context awareness assistance to support users working simultaneously on multiple process instances.

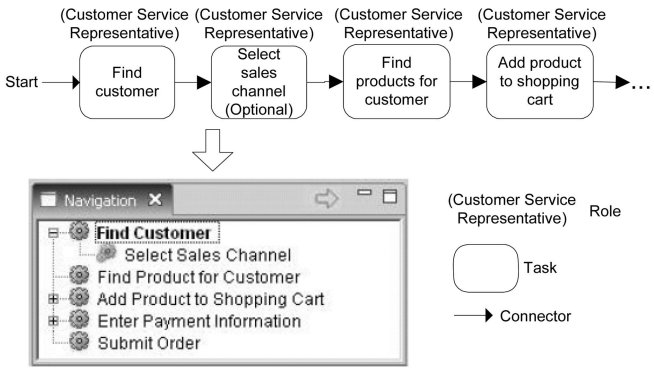

Fig. 10. Handling optional tasks in the navigation component.

4.3.1 Showing Navigation Sequences for Business Processes

To illustrate the tasks that a user must perform in a business process, we display the structure of the business process in the navigation component. However, a business process normally contains detailed information related to tasks. A user will be overwhelmed by the large number of tasks and will be distracted by the multiple execution paths specified in a process definition. To reduce the complexity of the navigation sequence, we show a simplified structure of the business process. The simplified structure of a business process only shows mandatory tasks as an initial navigation sequence and displays an optional task when its related mandatory task is completed. For the example business process shown in Fig. 10, the "Select Sales Channel" task is an *optional* task since it is performed in a combo box with a default value specified. Therefore, we display "Select Sales Channel" as a sublevel of the "Find Customer" task with a different icon. When the "Find Customer" task is complete, the CSR can either perform the "Select Sales Channel" task to select a nondefault value or proceed to the subsequent task (that is, the "Find Products for Customer" task). This technique for displaying optional tasks allows users to focus on mandatory tasks while giving them the chance to conduct optional tasks as needed.

To reduce the complexity of displaying possible navigation sequences, we include only one navigation sequence at a time. For OR-relations in a business process, the navigation component dynamically displays the possible navigation sequence based on the user's previous selections. Parallel tasks are displayed by the navigation component as sequential tasks. Business process parallelism simply implies independence between business operation paths. Tasks in different parallel paths can be performed in any order. Different graphic icons are used for distinguishing sequential tasks, alternative tasks, and parallel tasks. The simplification techniques reduce the complexity of displaying the navigation sequence and provide a clearer view of the overall progress of a business instance.

4.3.2 Providing Context Awareness Guidance

A context is composed of a process instance, an active task, the input and output data of the task, and the related UI components. As a user progresses through the UI, contextual information is gradually collected from the task events received from the UI components, the navigation sequence, and the binding specification. The navigation sequence indicates the active tasks. The task events indicate the status of the current active task. By examining the binding specification, the event engine can dynamically determine which UI components and supporting UI components should be displayed to allow the user to accomplish the current business task. Once a UI component is no longer used for subsequent tasks, the UI component closes automatically.

However, on some occasions, when a task ends, a user may want to verify the result displayed in a UI component before proceeding to the next UI component. Automatically opening a UI component may interfere with the user's perception of the UI. We have implemented a "next" button to give users control over the pace of their navigation. Once the user presses the "next" button, the current UI components are closed and the UI components that implement the subsequent task in a business process are opened. By default, the "next" button is disabled. The "next" button is enabled when a task in the current UI components generates data that must be verified by the user. In the cases where the current completed task leads to multiple tasks performed in parallel, the "next" button is enabled and users can bring up the UI component that implements one of the parallel tasks. Since the navigation component always highlights the current tasks to be performed, the user can easily navigate to the other concurrent tasks by selecting them from the navigation component when the concurrent tasks need to be fulfilled by the same user.

5 Improving the UI of a Call Center Application Using Business Processes

To verify the applicability of our approach, we restructured the UI of an e-commerce call center application. Our approach enhanced the existing UI of the application by providing contextual and navigational support. In this section, we present the implementation details associated with applying our approach on the UI.

5.1 The Call Center Application

The call center application is implemented on the Java-based Eclipse Rich Client Platform (RCP) [17]. The application is an early version of a tool being developed for commercial release. This application provides full functionality for a CSR to interact with customers and sell products over the phone. However, the application's UI usability is not optimal, especially to novice users. Generally, users of call center applications have limited computer experience. Call centers usually experience high turnover rates among CSRs. Therefore, many CSRs can be considered as novice users. Because most CSRs are novice users, the usability of the UI is a critical issue due to the high cost of training the large number of novice users.

5.2 Acquiring and Analyzing Process Definitions

For the studied application, the process definitions that describe the activities in a call center are modeled independently of the software development. The process

TABLE 1
Sizes of the Major Tools

Tools	Java (LOC)
Process Definition Parser	1,658
Role Model Generator	540
UI Model Generator	1,212
Navigation Sequence Generator	132
Code Generator	1,020

definitions of the call center application are provided by a business solution architect and are modeled by the IBM WebSphere Business Modeler (WBM) [64] in the Eclipse environment. The process definitions are stored using a schema defined by IBM and represented by XMI and Eclipse Metamodel Framework (EMF). We developed a parser for analyzing process definitions stored in the schema of XMI and EMF to extract information required by our approach. To handle other process definition languages such as XPDL or BPEL, a specialized parser for each language needs to be developed.

For other e-commerce applications, the process definitions are not always documented. Therefore, we need to first recover the process definitions from the source code by using the techniques described in [24], [72], [73]. The names of tasks in a business process are directly derived from source code artifacts (for example, widget names and function calls). To ease the analysis of the process definitions for generating the dynamic execution environment, we export the recovered process definitions into the IBM WBM [24].

5.3 Generating the Dynamic Execution Environment

We develop several tools to automatically generate the execution environment used by the improved UI:

1. A process definition parser parses process definitions modeled using the IBM WBM.
2. A role model generator extracts the tasks relevant only to a specific role (for example, a CSR or a Supervisor) from process definitions.
3. A UI model generator uses the information in role models for the presentation and layout in the improved UI.
4. A navigation sequence generator produces the navigation sequence, which describes the possible processing orders for each process definition.
5. A code generator generates source code for the event engine and the UI components in the improved UI, including the process list, the work list, and the navigation component.

The sizes of the major tools used for generating the dynamic execution environment are illustrated in Table 1.

5.4 Recovering Bindings between Tasks and UI Components

We bind the tasks specified in the process definitions with the original UI components of the call center application. We have automatically identified bindings by using the heuristics discussed in Section 4.2.1. All five heuristics are applied to UI components and task names. We observe that the names of most tasks and UI components are two words long. We also note that the names of tasks and UI components tend to use common words and follow a similar naming convention (that is, a verb followed by a noun). The chances of correctly matching similar tasks and UI components are relatively high. Since the call center application is still under active development, not all tasks specified in the process definitions are implemented in the software. Some tasks are automatic tasks that are invoked automatically without user involvement. We manually remove the not-implemented and automatic tasks from the process definitions to generate a precise navigation sequence.

TABLE 2
Accuracy of Heuristics for Binding Recovery

Process ID	# of Tasks in a Process	# of Human Tasks	# of Matched Human Tasks	Matching Accuracy
1	11	11	9	82%
2	12	12	10	83%
3	13	13	11	85%
4	15	14	12	86%
5	16	15	13	87%
6	49	41	36	88%
7	14	14	13	93%
8	25	24	23	96%
9	4	4	4	100%
10	8	2	2	100%
11	3	3	3	100%
12	8	7	7	100%
13	2	2	2	100%
14	3	2	2	100%
15	5	5	5	100%
16	6	6	6	100%
17	5	1	1	100%
18	8	8	8	100%

To evaluate the applicability of our heuristics, we develop a tool that matches UI components with human tasks in the definitions of 18 processes. We then manually evaluate our matching accuracy in consultation with the business solution architect and the developers of the application. The accuracy of the matching refers to the percentage of correct matches. The accuracy of the matching for each business process is listed in Table 2. The matching accuracy is lower in the first eight business processes. We mismatch the bindings for the "Handle Item Availability"

TABLE 3
Breakdown of Users

Group Name	Experience	# of Users
Expert	Experienced users who have used the original UIs for over one year	2
Novice-Tutorial	Novice users who take a 15 minute tutorial	5
Novice-NoTutorial	First time users who did not receive any tutorial or guidance	5

and "Enter Order Price Adjustments" tasks. Both tasks are used in the first eight business processes, where the matching accuracy is less than 100 percent. The total number of nonduplicate human tasks is 91 and the number of accurate matching is 89. If we measure the matching accuracy over the total number of nonduplicate human tasks, then the overall matching accuracy is 97.8 percent.

Using our heuristics, we cannot match the "Handle Item Availability" task. We manually examine the source code for the UI of the call center application and find that a dialog exists for users to handle available items in an inventory. However, no UI component is named by using any of the words among "Handle Item Availability." For the "Enter Order Price Adjustments" task, our tool matches it to the "Order Summary" dialog. However, the task is implemented by another UI component, called the "Order Items" dialog. The accuracy of our matching heuristic is relatively high and it is useful for semiautomated matching when the correctness is manually verified.

5.5 Integrating with Existing UI Components

The improved UI uses the same Eclipse RCP style as the original UI of the call center application, as shown in Fig. 3. To integrate the dynamic execution environment with existing UI components, we manually insert task event triggers in the source code of the existing UI components once the location of the source code is automatically identified. These UI components are bound to tasks and no additional modifications to the existing code are needed. We add approximately 91 task events in total to the code of the call center application. These task events are handled by the event engine.

6 Usability Study

We conduct a controlled experiment to study the usability of our improved UI for the call center application. The study compares the usability of the original UI against the usability of the improved UI. In this section, we describe the design of the experiment and our hypotheses. We also discuss our results and the threats to their validity. We follow the presentation of the experiment suggested in [67].

6.1 Design of the Experiment

6.1.1 Study Subjects

Nielsen [43] suggests that the best usability study for gathering qualitative measures should involve three to five users. We recruited 12 users with different levels of experience for our usability study. Table 3 gives an overview of the study subjects. All of the subjects use computers on a regular basis and are familiar with graphical UIs since they use such interfaces to perform daily activities such as checking e-mails, browsing the Internet, and shopping online. We wanted to compare the usability of the two UIs by users with different experience levels, so we divided the subjects into three groups: an expert group (Expert), a novice group that receives a tutorial (Novice-Tutorial), and a novice user group that does not receive a tutorial (Novice-NoTutorial).

The expert user group (Expert) consisted of two users who have used the original UI for over 1 year. The expert group was also involved in our research project and understood the features of the improved UI.

The novice user group consisted of 10 users who were graduate students in computer engineering and computer science. The novice user group had no prior experience of using either of the UIs. We randomly selected five users (Novice-Tutorial) in the novice group and gave them a 15 min tutorial on both the original and the improved UIs. The remaining five novice users did not receive a tutorial (Novice-NoTutorial).

In the tutorial, we stepped through both versions of the UI and demonstrated how we fulfilled all of the scenarios used in our study to the Novice-Tutorial group. At each step, we explained the information required for the user to perform a task and demonstrated the results expected after the fulfillment of each task. The tutorial allowed us to give the five novice users more guidance in using the UIs.

During the experiments, an observer sat beside the users and recorded the major problems that they encountered and the length of time required for performing the various scenarios.

TABLE 4
Summary of Tasks Performed in Each Scenario

Scenario #	Used Business Processes	# of Tasks
1	Manage Profile	5
2	Create Order	15
3	Manage Profile, Create Order	20

6.1.2 Scenarios Used in the Study

We asked the users to carry out three different scenarios by using the two versions of the UI. Users had to perform one or two business processes in each scenario. A summary of the scenarios is listed in Table 4. We used the "Manage Profile" and "Create Order" business processes in the scenarios. The "Manage Profile" is a simple process which contains five tasks. The "Create Order" is a more complex scenario which contains 15 tasks. A detailed description of these two processes is specified as follows:

- The **"Manage Profile"** business process describes the tasks for managing a customer's profile. To keep a customer's profile up to date, a CSR often conducts this business process while performing other business processes related to the customer. Specifically, a CSR first finds a customer profile and views the past

contact history of the customer. Depending on the customer's status, a CSR can update a customer's profile. In the end, a CSR may attach comments to the profile.

- The **"Create Order"** business process specifies tasks for a CSR to create an order for a customer. First, a CSR obtains the customer's data during a phone conversation and searches for the customer's record. If the customer has a record in the system, the CSR will follow a sequence of tasks to place an order for the customer. Otherwise, the CSR creates a new record for the customer and places the order for the customer. Placing the order involves finding products, adding products to the shopping cart, and selecting a shipping address.

These two business processes were combined to create three different scenarios:

Scenario 1. The user performs a simple business process, namely, the "Manage Profile" process. To fulfill this business process in the original UI, a user needs to switch between the menus and the UI components a few times to locate the UI components for the different tasks in the process. We want to evaluate whether the user prefers the strict processing order in the improved UI. The strict order guides users in performing a business process while limiting them to specific alternative paths.

Scenario 2. The user performs a more complex business process, namely, the "Create Order" process, which contains more tasks and control flow structures. Using this business process, we want to evaluate the effect of dynamically hiding, viewing, opening, and closing UI components, and the usage of the "next" button for guiding different users through tasks in a complex business process.

Scenario 3. The user simultaneously performs two different instances of the "Manage Profile" and "Create Order" processes. For the improved UI, supporting viewers (that is, a noneditable window) such as promotions viewers are only visible when the user is working on an instance of the "Create Order" business process. The promotions viewer is hidden when the user is working on an instance of the "Manage Profile" process. Similarly, the customer's profile information viewer is only displayed when the user is working on an instance of the "Manage Profile" process. Therefore, the correct UI components are displayed according to the context of the running process instances. In this scenario, we want to evaluate if our improved UI can help users manage multiple process instances simultaneously.

6.1.3 Evaluation Criteria for Usability

As discussed in Section 2, usability is measured in terms of the following attributes: error rate, memorability, efficiency, user satisfaction, and learnability. To quantitatively compare the two UIs, we used a set of usability metrics to assess each of the aforementioned usability attributes. We also asked users to fill out a questionnaire regarding their satisfaction to qualitatively measure the usability of the improved UI.

Error rate measures the rate of cancellation and system rejection. We record the number of times that a user cancels a UI component such as a window or a dialog when trying to locate the UI component needed to perform a task in a business process. We also track the number of times that the application rejects a user's selection of a UI component. Each rejection is indicated by a warning message. For example, if a user submits an order without providing the payment information, then the system would reject the selection since it is out of context. We measure the number of cancellations and system rejections for each user after a scenario is completed. To compare different scenarios, we normalize the error rate over the total number of tasks performed in a scenario. A UI with a low error rate is desirable.

Memorability can be measured by using retention over time [51]. In practice, help and feedback prompts are used for improving the memorability of a UI. We adapt two metrics to evaluate memorability: the percentage of tasks that a user requests for help while fulfilling the business processes (Help%) and a Mean Recognition Score (MRS) [34], [63]. The Help% metric is derived by measuring the number of times that an application displays a context-specific help messages that is triggered by the user requesting help. The number is divided by the total number of tasks in a scenario. A UI with high memorability has a low value of Help%.

The MRS determines the accuracy with which an application can automatically prompt users with the correct dialogs or windows while users are performing consecutive tasks. MRS is produced by counting the number of correctly prompted dialogs or windows. We normalize the MRS by dividing it by the total number of tasks in a scenario. A high MRS indicates a UI that has high memorability since users need to memorize fewer UI features.

Efficiency can be measured directly by using the time needed for a user to complete a business process (that is, elapsed time) or indirectly by measuring the performance of a user. Kamm et al. [34] propose that high learnability, low memorability, and low error rate have a positive impact on the performance. For example, requesting help has a negative impact on the performance. We measure the performance of the user by using (1), which is proposed by Kamm et al. [34]:

$$\text{Perf.} = 0.25 * \text{MRS} + 0.33 * \text{Success Rate} - 0.33 * \text{Help}\%, \tag{1}$$

where MRS is the Mean Recognition Score, Success Rate refers to the rate of successfully completed tasks, and Help% measures the percentage of tasks that a user requests help while performing a scenario.

User satisfaction is a subjective measure of the comfort and acceptability of use by users. To provide data on user satisfaction, users in our study completed the following short survey. We indicate the rationale for asking each question in brackets. The users were not shown the rationale for the questions. The survey contains the following questions:

- Were the UI components needed to launch a business process easy to locate? (process list performance)
- Did the system provide the information that you wanted to see? (context awareness performance)

TABLE 5
Results of Usability Evaluation for Scenario 1

Usability Attribute	Metrics	Expert		Novice-Tutorial		Novice-NoTutorial	
		Original UI	Improved UI	Original UI	Improved UI	Original UI	Improved UI
Error Rate	Error Rate	0	0	0	0	0	0
Memorability	Help%	0	0	0	0	0.4	0
	MRS	0.2	0.8	0.2	0.8	0.2	0.8
Efficiency	Elapsed Time (sec)	70	76	153	122	247	127
	Success Rate	1	1	1	1	1	1
	Performance	1.05	1.20	1.05	1.20	0.92	1.20
User Satisfaction	Subjective Satisfaction	23	28	21	72	16	26
Learnability	Learnability	1	1	0.46	0.62	0.28	0.60

TABLE 6
Results of Usability Evaluation for Scenario 2

Usability Attribute	Metrics	Expert		Novice-Tutorial		Novice-NoTutorial	
		Original UI	Improved UI	Original UI	Improved UI	Original UI	Improved UI
Error Rate	Error Rate	0	0	0.07	0	0.13	0
Memorability	Help%	0	0	0.13	0.07	0.27	0
	MRS	0.20	0.87	0.20	0.87	0.20	0.87
Efficiency	Elapsed Time (sec)	224	245	480	360	825	380
	Success Rate	1	1	1	1	1	1
	Performance	1.05	1.22	1.01	1.19	0.96	1.22
User Satisfaction	Subjective Satisfaction	24	29	21	27	17	27
Learnability	Learnability	1	1	0.47	0.68	0.27	0.64

- In the interaction, was it easy to locate the next UI component to perform the task that you wanted? (ease of navigation)
- During the interaction, did you know what you could do at each point of the interaction? (user's knowledge of the UI)
- Did the system work as you expected? (expected system behavior)
- Would you use the system regularly in the future? (future use)

The survey has multiple-choice Likert scale responses ranging over values such as almost never, rarely, sometimes, often, and almost always. To produce a numerical score, each survey response is mapped into the range of 1 to 5, with five representing the greatest degree of satisfaction. All of the responses are summed up to produce a User Satisfaction measure which ranges from 6 to 30 for each scenario.

Learnability measures the time needed to learn a UI [51]. We are interested in understanding the ease of learning our business process driven UI in comparison to the original UI. To measure the learnability of a UI, we compare the time needed for an expert to perform the same scenario relative to the time needed for a novice user to perform a scenario. For example, if an expert takes, on average, 50 sec to perform a scenario, while a novice user takes 100 sec to perform the same scenario, then the learnability of the UI is 0.5. A UI with high learnability is desirable.

6.2 Hypotheses and Analysis of Results

Tables 5, 6, and 7 are the descriptive statistics for the three scenarios of the three groups of users (that is, Expert, Novice-Tutorial, and Novice-NoTutorial). For each user group, the table shows the means of the various usability metrics for that particular user group. Simply comparing the means of the various metrics is not sufficient since the differences may be due to the natural variability of the data and may not be statistically significant. Instead, we statistically test the results of our study. In particular, we formulate the following hypotheses for our study:

1. **Usability and type of UI.** The usability of the improved UI is higher in comparison to the usability of the original UI for novice users. However, the usability of the improved UI is lower for expert users.
2. **Usability and tutorial.** A tutorial does not increase the usability of the improved UI for novice users. However, a tutorial increases the usability of the original UI for novice users.

We test our hypotheses by using statistical tests which assume a level of significance of 5 percent (that is, $\alpha = 0.05$). We perform all of our statistical analyses on the data for novice users. Due to the small number of expert users (only two users), we cannot statistically test our results for expert users. However, for reference, we report the metrics for the two expert users in Tables 5, 6, and 7. The metrics for both

TABLE 7
Results of Usability Evaluation for Scenario 3

Usability Attribute	Metrics	Expert		Novice-Tutorial		Novice-NoTutorial	
		Original UI	Improved UI	Original UI	Improved UI	Original UI	Improved UI
Error Rate	Error Rate	0	0	0.05	0	0.10	0
Memorability	Help%	0	0	0	0	0.50	0
	MRS	0.20	0.85	0.20	0.85	0.20	0.85
Efficiency	Elapsed Time (sec)	266	291	721	500	1200	520
	Success Rate	1	1	1	1	1	1
	Performance	1.05	1.21	1.05	1.21	0.89	1.21
User Satisfaction	Subjective Satisfaction	22	20	20	28	16	28
Learnability	Learnability	1	1	0.37	0.58	0.22	0.56

expert users are consistent, with little variability between both users. For each hypothesis for the novice users, we compare the means of the usability metrics discussed in Section 6.1.3.

For our statistical analysis, we conduct parametric and nonparametric tests. We use a t-test [39] for the parametric test and a paired Wilcoxon signed-rank test [39] for the nonparametric test. We study the results of both types of tests to determine whether there are any differences between the results reported by both types of tests. In particular, for nonsignificant differences reported by the parametric tests, we check if the differences are significant according to the nonparametric Wilcoxon test. The Wilcoxon test helps ensure that nonsignificant results are not simply due to the departure of the data from the assumptions of the t-test. For the results presented in the following, the results of both types of tests are consistent and we only show the P values for the t-tests.

6.2.1 Usability and Type of UI

For the first hypothesis, we compare the mean of the usability metrics for novice users by using the original UI $\mu(\text{Metric}_{\text{Original}})$ and the mean of the usability metrics for novice users by using the improved UI $\mu(\text{Metric}_{\text{Improved}})$. We believe that the modified UI would increase the usability of the call center application, in particular for novice users. For example, we expect that the learnability metric would increase. Therefore, we expect that the difference in means for the learnability metric would improve. We can then formulate the following test hypothesis:

$$\text{H}_0 : \mu(\text{Learnability}_{\text{Original}} - \text{Learnability}_{\text{Improved}}) = 0,$$
$$\text{H}_\text{A} : \mu(\text{Learnability}_{\text{Original}} - \text{Learnability}_{\text{Improved}}) \neq 0.$$

$\mu(\text{Learnability}_{\text{Original}} - \text{Learnability}_{\text{Improved}})$ is the population mean of the difference between the learnability metric for each UI. If the null hypothesis H_0 holds (that is, the derived P value $> \alpha = 0.05$), then the difference in mean is not significant and is likely due to the variability of the data. If H_0 is rejected with a high probability (that is, the derived P value ≤ 0.05), then we are confident about the performance improvements of the improved UI.

We perform similar hypotheses testing on all of the usability metrics. Table 8 shows the results of the t-test for the hypotheses for the usability metrics. The table shows the difference in means and the percentage of improvement. To present all of the metrics in a consistent matter, we show the absolute value of the difference since, for some metrics (for example, Help%), a decrease in metric value represents an improvement, whereas, for other metrics (for example, Performance), an increase in value represents an improvement. All statistically significant cells are colored gray. The results shown in Table 8 indicate that the differences in the metrics between both UIs are significantly different from zero. An analysis of the metrics reveals that all metrics have improved in the improved UI. We can conclude that the usability of the improved UI is statistically better than the usability of the original UI.

Examining Tables 5, 6, and 7, we notice that the expert users take more time to complete all three scenarios for the improved UI. The original UI is designed around a data flow. For example, a UI component such as a window or a dialog is designed for each data object such as a customer, an order, or a quote. Users can enter all of the information related to that data object in a UI component. Inside each UI component, users can use buttons or tabs to switch between

TABLE 8
Statistical Analysis for Usability versus Type of UI

Usability Attribute	Metric	$\mu(\lvert\text{Metric}_{\text{Original}} - \text{Metric}_{\text{Improved}}\rvert)$	$P(H_0 \text{ holds})$
Error Rate	Error Rate	0.0583 (+100%)	0.018
Memorability	Help%	0.205 (+95%)	0.001
	MRS	0.6388 (+319%)	0.013
Efficiency	Elapsed Time	18.819 (+43%)	0.04
	Performance	0.227 (+23%)	0.017
User Satisfaction	Subjective Satisfaction	8.7 (+47%)	<0.001
Learnability	Learnability	0.27 (+78%)	0.047

different pages in the UI component. An expert user who is familiar with the UI can perform business processes with high efficiency. When two business processes have common tasks, an expert user can fulfill a common task once in the corresponding UI component. The result of the task can be shared with other business processes. However, in the improved UI, data from common tasks in different business processes cannot be shared. Therefore, common tasks have to be performed repeatedly in each business process. For example, if two separate business processes contain an "Enter User Shipping Information" task, then, in the improved UI, the expert user must perform this task twice instead of performing both processes in parallel and sharing the task data. Due to the inability of the expert user to share task data between processes, the elapsed time for expert users using the improved UI is longer than the time for the original UI. In short, our experimental results show that the improved UI may slow down expert users since users have to follow a strict order to conduct their tasks. The strict order is not desirable for expert users who may have acquired different workarounds and speedups. Unfortunately, we cannot statistically test this finding due to the limited number of expert users. However, we confirm our findings with the expert users who pointed to their frustration in having to reenter data and in having to follow the strict order. The expert users expressed their frustration as well in the subjective survey by giving the improved UI a lower satisfaction rate in scenario 3 (20 for improved UI versus 22 for original UI). Both expert users indicated that they do not prefer using the improved UI in the future.

For the improved UI, we have received positive feedback from the novice users. The novice users appreciate the strict order in the improved UI. Users note that they like that the navigation sequence shows only mandatory tasks and that it dynamically adapts to a user's selection. The navigation sequence reduces the complexity of presenting the progress of a process instance and gives users an overview of the remaining steps. The navigation sequence also records the progress and the status of the process instances and is valuable in helping users resume their work, even if they took a break during the evaluation. The context awareness is considered to be desirable, especially for supporting UI components that are mixed with other UI components or which require users to manually initiate them. Novice users do not need to search for UI components themselves. Instead, they simply click on the "next" button and the UI dynamically opens and closes the appropriate UI components. Moreover, the "next" button replaces multiple mouse clicks with one simple click when searching for the subsequent UI components. The "next" button reduced the time required by users to locate UI components and improved their work efficiency.

As pointed out in [44], usability attributes such as ease of learning, maximum efficiency of use, low error rate, and subjective satisfaction conflict with one another. It is challenging to balance the expectations of novice users and expert users in one UI design. A compromise is usually reached in the design of a UI to meet the expectation of the majority of the users of an application (that is, novice or expert). An application that has a user base with high turnover is best serviced with a UI designed for novice users. Conversely, an application that is used frequently by a stable user base would require a UI designed for expert users to improve the efficiency.

We believe that supporting novice users has precedence since a large number of users of e-commerce applications tend to be novice users [42]. As business processes evolve over time, the underlying implementation such as the UI of an e-commerce application must change. Many expert users are likely not to have expert knowledge of all business processes offered by large complex applications. Moreover, an expert may not be an expert after the UI is updated to accommodate evolving business processes. In our approach, the content of the UI is automatically regenerated from the updated process definitions. Using our approach, the changes to a business process are immediately reflected in the UI. The strict order of the improved UI ensures that expert users and novice users do not miss new changes in old processes. Moreover, the improved UI can make optimal use of screen space by automatically opening and closing the supporting UI components when no data is available. In short, our business process driven UI is beneficial since it reduces the probability of errors to novice and expert users (especially if they are overworked).

6.2.2 Usability and Tutorial

To familiarize users with the UI, half of the novice users were given a tutorial about both UIs. We would like to examine the benefits of conducting such a tutorial for the original UI and for the improved UI. Due to the high cost associated with training users, we hope that our improved UI can reduce the need for tutorials.

We study the benefits of the tutorial on the usability attributes by comparing the usability metrics for novice users with tutorial and the metrics for novice users without tutorial. We first compare the metrics for the novice users using the original UI to determine if the tutorial was helpful for the original UI. We then compare the metrics for the improved UI to determine whether the tutorial is helpful for the improved UI. For example, for the learnability metric, we formulate the following test hypotheses to study the benefits of the tutorial on the original UI for novice users:

$$\mathrm{H}_0: \mu(\mathrm{Learnability}_{\mathrm{OriginalUI\_Tutorial}} - \mathrm{Learnability}_{\mathrm{OriginalUI\_NoTutorial}}) = 0,$$

$$\mathrm{H}_\mathrm{A}: \mu(\mathrm{Learnability}_{\mathrm{OriginalUI\_Tutorial}} - \mathrm{Learnability}_{\mathrm{OriginalUI\_NoTutorial}}) \neq 0.$$

$\mu(\mathrm{Learnability}_{\mathrm{OriginalUI\_Tutorial}} - \mathrm{Learnability}_{\mathrm{OriginalUI\_NoTutorial}})$ is the population mean of the difference between the learnability metric for novice users with and without tutorial for the original UI. If the null hypothesis H_0 holds (that is, the derived $\mathrm{P\ value} > \alpha = 0.05$), then the difference in mean is not significant and is likely due to the variability of the data. If H_0 is rejected with a high probability (that is, the derived $\mathrm{P\ value} \leq 0.05$), then we are confident that the tutorial is of little value. This analysis is done for the original UI and the improved UI. Table 9 is the summary of both analyses. Cells in gray are statistically significant, whereas the other cells are not. The table indicates that the tutorial is valuable for the original UI since the tutorial improves the usability for novice users for all usability

TABLE 9
Statistical Analysis of the Effect of Tutorial

Usability Attribute	Metric	$\mu(Metric_{Tutorial} - Metric_{NoTutorial})$	$\mu(Metric_{Tutorial} - Metric_{NoTutorial})$
		Original UI	Improved UI
Error Rate	Error Rate	-0.038	0
Memorability	Help%	-0.34	0.02
	MRS	0	0
Efficiency	Elapsed Time	-21.91	-1.11
	Performance	0.113	-0.0007
User Satisfaction	Subjective Satisfaction	4.33	0.33
Learnability	Learnability	0.123	0.0201

metrics except for Error Rate and MRS metrics. However, the tutorial does not have any statistically significant effect on the usability of the improved UI for novice users and is likely not needed.

6.3 Threats to Validity

We now discuss the different types of threats that may affect the validity of the results of our experiment.

External validity tackles the issues related to the generalization of the result. Our study asks users to conduct three scenarios on a single e-commerce application. The scenarios are selected in consultation with the developers of the application. The developers feel that these scenarios are the most frequently used scenarios. The e-commerce application is a large and complex application that is currently deployed in an industrial setting. Nevertheless, we should, in the future, study the benefits of our approach by using other applications and other scenarios to determine if our results hold for other applications and domains.

In our experiment, a limited number of expert users participated in our usability study since it is difficult to find users who have used the systems for more than 1 year other than the original developers. In practice, the users of the studied application are novice users with limited computer experience. The novice user groups in our usability study were graduate student volunteers. These volunteers likely have more experience in using computers than a regular CSR. Nevertheless, we believe that the additional experience would make them more comfortable in using and navigating through both versions of the UI. Hence, the usability improvement demonstrated in our experiment is more likely to be a lower bound rather than an upper bound.

Internal validity is a concern with the issues related to the design of our experiment. In particular, we need to account for the order in which users use the UI and the order in which they perform the scenarios. In our experiment, all three user groups complete the scenarios in the same order and always start with the improved UI. To avoid any bias, the novice users are not informed whether

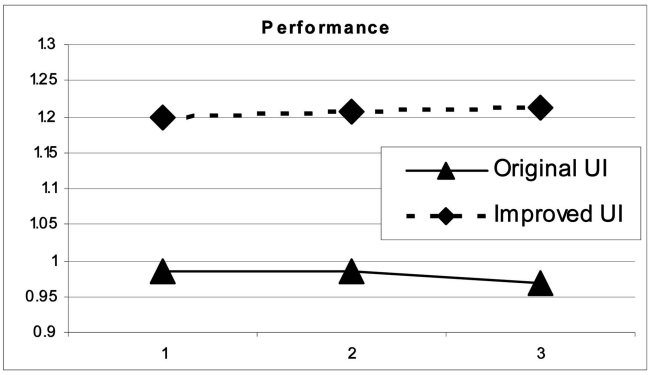

Fig. 11. Comparing the mean values of the performance metric across the three scenarios.

the UI that they are using is the original or the improved UI. We feel that this setup should not reflect positively on our improved UI. This setup will also not reflect negatively on the original UI. The guidance provided by the improved UI will likely help users in performing the scenarios using the original UI. The users' experience in using the improved UI should make the users more familiar with the components of the original UI.

Users performed the scenarios in the order of their complexity. We feel that the order of scenarios from the easiest (that is, scenario 1 was done first) to the most complex is more realistic since users in an industrial setting are likely to be mentored through easy scenarios before being given more complex scenarios. Scenarios 1 and 2 have no common UI components. However, scenario 3 contains the UI components from scenarios 1 and 2. The two earlier scenarios help users in performing scenario 3.

To quantify the learning effect, we compared the means of the various usability metrics across the three scenarios for all novice users for both original and improved UIs. Fig. 11 shows the means for the performance metrics for all novice users on both UIs. Fig. 11 shows that there is a slight improvement in performance going from scenarios 1 to 2, but the performance drops for scenario 3 in the original UI. However, the improvements are trivial. To determine if there is a learning effect, we perform a two-way ANOVA [39], $\alpha = 0.05$, which uses a scenario (1, 2, or 3) and a UI type (improved UI or original UI) as independent variables and the usability metrics as dependent variables. The ANOVA results show that the differences in means of all of the usability metrics are statistically significant for the UI type, but the differences are not significant for the scenarios. The differences are not significant for the interaction between scenarios and UI types. These results indicate that the learning effect is not statistically significant. We performed one-way Kruskal-Wallis [39] nonparametric tests by using scenarios as independent variables and the usability metrics as dependent variables. The results of the Kruskal-Wallis tests confirmed that the differences of means for the usability metrics across scenarios are not significant. Although our statistical tests show that there is no visible learning effect between scenarios, we expect, in practice, that there would be a learning effect. We do not think that the learning effect will affect our results since

novice users are commonly trained on simple scenarios before being moved to more complex ones.

Users performed the three scenarios by using the improved UI before performing the scenarios again by using the original UI. To prevent user fatigue from affecting the results of our study, users were given a 10 min break between the experiments on the different UIs.

Construct validity is a concern as to the meaningfulness of the measurement. In our study, we measured the usability of a UI by using a variety of metrics from the literature. However, usability is a subjective issue in many cases and we account for this through the use of a subjective survey rather than the use of usability metrics.

7 Related Products, Standards, and Research

7.1 Business Process Modeling and Automation

Workflow Management Systems (WfMS) [66], such as IBM's WebSphere MQ Workflow [65] and Oracle's Workflow [11], are often used for automating the execution of business processes. The WfMC defines a reference model for WfMS. Typically, a WfMS consists of business process modeling tools, business process analysis tools, workflow engines, and applications. A business process modeling tool allows business users to model business process definitions. To optimize a business process and detect design faults (for example, bottlenecks), analysis tools are used for simulating, diagnosing, and verifying process definitions [66]. Techniques for analyzing business processes include action analysis, process mapping, coordination analysis, and social grammar analysis [3].

To automate the execution of business processes, a workflow engine interprets a process definition, enacts tasks at runtime, and distributes tasks to roles. The business process automation deals with generating the implementation from the definitions of business processes and with managing the execution of the generated implementation [19].

Despite the many features of WfMSs, they have a number of limitations, such as the lack of interoperability among WfMSs, the lack of support for correctness and reliability, and weak tool support for analyzing, testing, and debugging the implementation of business processes. A WfMS is heavyweight in nature and is mostly used for integrating complex cross-organizational e-commerce applications. Frameworks such as OOHDM [21], UWA [35], WebML [1], and OO-HMETHOD [8] offer a lightweight solution for simple Web-based e-commerce systems [35]. Such frameworks and WfMS engines offer tools for creating applications from process definitions. However, little focus is put on ensuring that the created applications are easily used by novice users. Our work captures valuable information stored in the business process definitions and uses the information to create applications that are easier to use. Our work can be integrated into such frameworks and WfMSs to improve the usability of created business applications.

Business process reengineering is the rethinking and the redesign of business processes to improve the key performance measures such as cost, quality of services, and speed [22]. A typical reengineering scenario is caused by changes in five interrelated factors, including operational processes, the emergence of technology, human resource structure, communication, and organizational structure [25], [32]. However, technology only enables changes in business processes, whereas other factors (for example, human resource structure and communication) need to be coordinated to ensure a successful business process reengineering project [2], [20], [27]. Researchers propose a variety of techniques to ensure the alignment of business processes and software systems. Ganti and Brayman [18] present general guidelines on migrating legacy systems into a distributed environment. The organizational structure and business operations are examined to reengineer business processes. Our work focuses on automatically restructuring the existing UI of an application instead of reengineering the business processes. Once a business process is reengineered, then our approach can automatically restructure the UI of the reengineered process. By automating the restructuring process, we can ensure that business processes and the applications that implement them can evolve together while staying in sync.

7.2 UI Design and Reengineering

Much of the research on UI reengineering focuses on migrating UIs, either from text-based platforms to GUI-based platforms [40], [41], [56], [59] or from one GUI platform to another [53], [54]. Brambilla et al. [5] point out that reengineering existing UIs improves the performance and user satisfaction, shortens the training, and reduces error rates. Typically, a UI reengineering process starts by reverse engineering the UI components by using static [37] and dynamic analyses [53], [54]. We use a heuristic-based approach which is similar to [37] to identify bindings between business tasks and UI components such as windows and dialogs. We insert triggers for task events in the existing code to coordinate the open and close actions of the UI components with the progress of a process instance.

To improve the usability of UIs, Sliski et al. [52] utilize the bindings between activities and UI components to develop an environment to integrate software toolsets. However, developers must manually specify the bindings and processes that fit the behaviors of the toolsets. In contrast, our proposed approach automatically generates navigation sequences directly from process definitions. In addition, we are able to handle multiple process instances (that is, user scenarios) and provide context awareness assistance.

Studies show that novice users and experienced users exhibit different behaviors when using a UI [38], [61]. Novice users prefer to follow instructions or navigations in a step-by-step fashion. Experienced users expedite their work by using direct access options such as shortcuts. As indicated by Nielsen [42], Web designs should focus on novice users. Ease of use is one of the most important quality indicators of a Web site. Guidelines [26], [48] and design patterns [23], [50], [55], [58], [60] are used for capturing design knowledge and for guiding designers in the design of UIs. Moreover, mistakes in UI designs are collected as antipatterns (also called shame halls) such as the Interface Hall of Shame Web site [28] and the Web

Bloopers Web site [33]. Both sites collect a large number of antipatterns in UI designs, explain the limitations of the designs, and propose typical workarounds. Our work focuses on improving the usability of the e-commerce applications for the novice users. The antipatterns and design guidelines could be integrated into our work and used for improving the usability of the UI generated using our approach.

Role-based access control models are used for limiting users from accessing functional features that are not designed for them. Examples of research on designing access control policies are XACML [45], Ponder [10], and authorization engines such as the Flexible Authorization Framework (FAF) [31]. A role-based access control model is particularly important when different roles are using the same UI. Functional features (for example, buttons and menus) are enabled or disabled according to the current role and its access rights. Integrating access control features into the UI increases the complexity of the UI. De Lucia et al. design the access policies of roles by means of a visual-language-based tool that provides a metaphor-oriented layer above the RBAC model [15]. In our work, we simplify the UI by presenting to the users the functional features that are relevant to their current role(s) according to the process definitions.

7.3 Web Applications Reengineering

Currently, most e-commerce applications adopt a Model-View-Controller (MVC) architecture [6]. In this architecture, a model manages the data and behaviors of the application and invokes back-end functions. Views describe the content and layout of the UI and present the computation results to users. The views can be implemented by various UI technologies such as Web portal technologies, Eclipse-Style RCP, and hypertexts. Controllers are used for gathering data from the users. This architecture separates UIs (that is, views) from business logics (that is, model).

Various researchers have conducted studies on reverse-engineering techniques and tools for analyzing legacy applications and adapting the UI of existing systems to the MVC architecture. Bodhuin et al. [4] have presented a wrapping strategy to adapt a Cobol legacy system to the MVC architecture. A toolkit is developed to extract all the information related to views from the Cobol source code. The legacy UI is replaced with a Web UI using Java Server Pages. The wrappers are automatically generated for the communication between the new Web UI (that is, views) and the business logics (that is, models) implemented in the original Cobol system. Ping and Kontogiannis [46] propose a refactoring approach that analyzes the page flows in an existing hyperlinked Web site and restructures the page flows toward the MVC architecture.

Ricca and Tonella [49] propose a migration approach to transform static Web sites into dynamic ones by using a page flow analysis, graph traversal algorithms, pattern matching, and clustering techniques. Antoniol et al. [1] propose a methodology for recovering the design of a static Web site by abstracting the navigational structure of a Web site. The recovered design is represented using the relationship management data model and the ER+ model proposed with the relationship management methodology. To understand the behavior of Web applications, the proposed approaches [12], [13], [14] analyze the static and dynamic contents of Web applications to comprehend the dynamic interactions among the components in Web applications. UML diagrams are extracted from existing Web applications to facilitate the understanding of the functionality of Web applications. Clone detection techniques are proposed in [16] to identify cloned Web pages with similar content and replace the clones with dynamic pages. Most of the above-mentioned approaches focus on enhancing the maintainability of Web applications to ease the maintenance and evolution of these applications. The primary stakeholders in these approaches are the software developers and maintainers. However, in our work, we focus on enhancing the usability of these applications. Our primary stakeholder is the user of these applications. In particular, we would like to ensure that novice users are able to use these applications without requiring extensive training. Our approach can be applied to the UIs of Web and non-Web-based applications.

8 Conclusion and Future Work

In this paper, we present an approach to restructure the UI of an existing e-commerce application to improve its usability. The improved UI provides navigational guidance to help users accomplish business activities. The improved UI provides context awareness assistance by displaying UI components relevant to the context of a user's current activities. We conducted a usability study, which showed that the improved UI can improve the user experience for novice users by offering more guidance and by reducing the time needed for novice users to perform business scenarios. The improved UI requires less training. Since the UIs of e-commerce applications are continuously updated to accommodate the evolving business processes, novice users will require little assistance and training to use the latest version of the application.

Our approach has some limitations. First, our UI, guided by business processes, often specifies a single navigation sequence to carry out business activities. However, developers tend to offer many alternatives to perform tasks. For example, a CSR can add multiple products by using the "Add Product" editor. The corresponding "Add Product" task, as specified in the process definition, is nonrepeatable. Our usability study shows that expert users are adversely being forced to follow a strict order when conducting their work. In the original UI, an expert user is able to complete a business process by using the most efficient navigation sequence. The most efficient sequence involves the least searching for the next UI components and the minimal number of buttons clicks. On the other hand, our study shows that the improved UI improves the productivity of novice users and increases the accuracy for both novice and expert users, especially when business processes evolve. In the future, we plan to incorporate usability design criteria to optimize the navigation sequence with fewer clicks and fewer UI components. In this case, a novice user can be guided to click fewer buttons and experience fewer navigational transitions, just like an expert user.

Another limitation of our approach is that we treat each process instance of a business process independently of other processes. In practice, users may want to use data such as "User Information" as input to other process instances. Using our approach, the "User Information" must be entered separately for each process instance. The business processes cannot be linked to one another. In the future, we plan to add functionality to permit users to populate data fields in an active process instance from data captured in previously executed business processes. This technique would reduce the amount of data reentry that users have to perform and would improve the usability of our improved UI (especially for expert users).

Regarding our current implementation, we use a semi-automated approach to identify bindings and insert triggers for task events. The automated process works reasonably well in identifying bindings since task names and UI components tend to follow a similar naming convention. The triggers are manually inserted into the source code. In the future, we plan to automate this process.

In our current usability study, we were unable to recruit more than two expert users. We cannot statistically validate the usability of the improved UI for expert users. However, our usability study is indented to demonstrate that our approach can improve the usability of the UI for novice users. In the future, we plan to apply our approach to other e-commerce applications, to receive feedback from actual novice users and more expert users, and to address the shortcomings of our improved UI.

ACKNOWLEDGMENTS

This research is sponsored by IBM Canada, Centers for Advanced Studies (CAS), and the National Sciences and Engineering Research Council of Canada (NSERC). The authors would like to thank Jen Hawkins, Bhadri Madapusi, Ross McKegney, and Tack Tong of the IBM Canada Toronto Laboratory for their valuable feedback on this work. They would like to thank the anonymous reviewers and Harry Sneed for their valuable feedback on the manuscript.

REFERENCES

[1] G. Antoniol, G. Canfora, G. Casazza, and A. De Lucia, "Web Site Reengineering Using RMM," *Proc. Second Int'l Workshop Web Site Evolution,* pp. 9-16, 2000.

[2] L. Aversano, G. Canfora, A. De Lucia, and P. Gallucci, "Business Process Reengineering and Workflow Automation: A Technology Transfer Experience," *J. Systems and Software,* vol. 63, no. 1, pp. 29-44, 2002.

[3] S. Biazzo, "Approaches to Business Process Analysis: A Review," *Business Process Management J.,* vol. 6, no. 2, pp. 99-112, 2000.

[4] T. Bodhuin, E. Guardabascio, and M. Tortorella, "Migrating COBOL Systems to the Web by Using the MVC Design Patterns," *Proc. 10th Working Conf. Reverse Eng.,* pp. 329-338, 2003.

[5] M. Brambilla, S. Ceri, S. Comai, P. Fraternali, and I. Manolescu, "Specification and Design of Workflow-Driven Hypertexts," *J. Web Eng.,* vol. 1, no. 2, pp. 163-182, Apr. 2003.

[6] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stahl, *A System of Patterns.* Wiley, 1998.

[7] Business Process Execution Language for Web Services, http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf, Apr. 2007.

[8] C. Cachero, J. Gómez, and O. Pastor, "Object-Oriented Conceptual Modeling of Web Application Interfaces: The OO-HMethod Presentation Abstract Model," *Lecture Notes in Comuter Science,* vol. 1875, pp. 206-215, Springer-Verlag, 2000.

[9] M. Costabile, "Usability in the Software Life Cycle," *Handbook of Software Eng. and Knowledge Eng.,* vol. 1, World Scientific, 2001.

[10] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The Ponder Policy Specification Language," *Lecture Notes in Computer Science,* vol. 1995, pp. 18-28, Springer-Verlag, 2001.

[11] "Feature Overview Oracle 9i Application Server: Oracle Workflow," http://www.oracle.com/technology/products/integration/workflow/workflow_fov.html, Dec. 2006.

[12] G.A. Di Lucca, M. Di Penta, G. Antoniol, and G. Casazza, "An Approach for Reverse Engineering of Web-Based Applications," *Proc. Eighth Working Conf. Reverse Eng.,* pp. 231-240, 2001.

[13] G.A. Di Lucca, A.R. Fasolino, U. De Carlini, F. Pace, and P. Tramontana, "WARE: A Tool for the Reverse Engineering of Web Applications," *Proc. Sixth European Conf. Software Maintenance and Reengineering,* pp. 241-250, 2002.

[14] G. Di Lucca, A. Fasolino, U. De Carlini, and P. Tramontana, "Abstracting Business Level UML Diagrams from Web Applications," *Proc. Fifth IEEE Int'l Workshop Web Site Evolution,* pp. 12-19, 2003.

[15] A. De Lucia, M. Giordano, G. Polese, G. Scanniello, and G. Tortora, "Role Based Reengineering of Web Applications," *Proc. Seventh Int'l Symp. Web Site Evolution,* pp. 103-110, 2005.

[16] A. De Lucia, R. Francese, G. Scanniello, and G. Tortora, "Reengineering Web Applications Based on Cloned Pattern Analysis," *Proc. 12th Int'l Workshop Program Comprehension,* pp. 132-141, 2004.

[17] Eclipse Rich Client Platform, http://wiki.eclipse.org/index.php/Rich_Client_Platform, 2007.

[18] N. Ganti and W. Brayman, *Transition of Legacy Systems to a Distributed Architecture.* John Wiley & Sons, 1995.

[19] D. Georgakopoulos, M. Hornick, and A. Sheth, "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure," *Distributed and Parallel Databases,* vol. 3, pp. 119-153, 1995.

[20] D. Grant, "A Wider View of Business Process Reengineering," *Comm. ACM,* vol. 45, no. 2, pp. 85-90, 2002.

[21] N. Guell, D. Schwabe, and P. Vilain, "Modeling Interactions and Navigation in Web Applications," *Proc. World Wide Web and Conceptual Modeling Conf.,* pp. 115-127, 2000.

[22] M. Hammer and J. Champy, *Reengineering the Corporation.* Harper Collins, 1993.

[23] S. Henninger, "A Methodology and Tools for Applying Context-Specific Usability Guidelines to Interface Design," *Interacting with Computers,* vol. 12, no. 3, pp. 225-243, 1999.

[24] M. Hung and Y. Zou, "Recovering Workflows from Multi-Tiered E-Commerce Systems," *Proc. 15th Int'l Conf. Program Comprehension,* pp. 198-207, 2007.

[25] K. Hunt, G. Hansen, E. Madigan, and R. Phelps, "Simulation Success Stories: Business Process Reengineering," *Proc. Winter Simulation Conf. '97,* pp. 1275-1279, 1997.

[26] IBM Web Design Guidelines, http://www-03.ibm.com/easy/page/572, 2007.

[27] J. Iden, "Business Process Reengineering: Examining Some Major Roadblocks to Increased Self-Control for the Employee," *Proc. Conf. Organizational Computing Systems,* pp. 75-82, 1995.

[28] Interface Hall of Shame, http://homepage.mac.com/bradster/iarchitect/shame.htm, 2007.

[29] *ISO 9126 Standard,* http://www.issco.unige.ch/ewg95/node13.html, Apr. 2007.

[30] *ISO 9241: Ergonomics Requirements for Office Work with Visual Display Terminal (VDT),* Parts 1-17, 1997.

[31] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian, "Flexible Support for Multiple Access Control Policies," *ACM Trans. Database Systems,* vol. 26, no. 2, pp. 214-260, 2001.

[32] M. Jansen-Vullers, M. Netjes, and H.A. Reijiers, "Business Process Redesign for Effective E-Commerce," *Proc. Sixth Int'l Conf. Electronic Commerce,* pp. 291-382, 2004.

[33] J. Johnson, *Web Bloopers: 60 Common Web Design Mistakes and How to Avoid Them.* Morgan Kaufmann, 2003.

[34] C. Kamm, D. Litman, and M. Walker, "From Novice to Expert: The Effect of Tutorials on User Expertise with Spoken Dialogue Systems," *Proc. Fifth Int'l Conf. Spoken Language Processing,* pp. 1211-1214, 1998.

[35] G. Kappel, B. Proll, W. Retschitzegger, W. Schwinger, and T. Hofer, "Modeling Ubiquitous Web Applications—A Comparison of Approaches," *Proc. Third Int'l Conf. Information Integration and Web-Based Applications and Services,* 2001.

[36] N. Koch, A. Kraus, C. Cachero, and S. Melia, "Modeling Web Business Processes with OO-H and UWE," *Proc. Third Int'l Workshop Web-Oriented Software Technology,* July 2003.
[37] E. Merlo, J. Girard, K. Kontogiannis, P. Panangaden, and R. De Mori, "Reverse Engineering of User Interfaces," *Proc. First Working Conf. Reverse Eng.,* pp. 171-179, 1993.
[38] R. Molich and J. Nielsen, "Improving a Human Computer Dialogue," *Comm. ACM,* vol. 33, no. 3, pp. 338-348, 1990.
[39] D. Montgomery and G. Runger, *Applied Statistics and Probability for Engineers,* third ed. John Wiley & Sons, 2003.
[40] M. Moore and L. Moshkina, "Migrating Legacy User Interfaces to the Internet: Shifting Dialogue Initiative," *Proc. Seventh Working Conf. Reverse Eng.,* pp. 52-58, 2000.
[41] M. Moore, S. Rugaber, and P. Seaver, "Knowledge Based User Interface Migration," *Proc. Int'l Conf. Software Maintenance,* pp. 72-79, 1994.
[42] J. Nielsen, "Novice vs. Expert Users," *Jakob Nielsen's Alertbox,* 2000.
[43] J. Nielsen, "Success Rate: The Simplest Usability Metrics," *Jakob Nielsen's Alertbox,* Feb. 2001.
[44] M. Padilla, "Strike a Balance: User's Expertise on Interface Design," IBM white paper, http://www-128.ibm.com/developerworks/web/library/wa-ui/#2, 2003.
[45] *Core Specification: eXtensible Access Control Markup Language (XACML) Version 2.0,* OASIS, May 2007.
[46] Y. Ping and K. Kontogiannis, "Refactoring Web Sites to the Controller-Centric Architecture," *Proc. Eighth European Conf. Software Maintenance Reeng.,* pp. 204-213, 2004.
[47] C. Plaisant, A. Rose, and B. Sheideman, "Low Effort, High Payoff User Interface Reengineering," *IEEE Software,* vol. 14, no. 4, pp. 66-72, July 1997.
[48] *Research-Based Web Design and Usability Guidelines,* http://www.usability.gov/pdfs/guidelines.html, 2007.
[49] F. Ricca and P. Tonella, "Understanding and Restructuring Web Sites with ReWeb," *IEEE Multimedia,* vol. 8, no. 2, pp. 40-51, Apr.-June 2001.
[50] J. Scholtz and S. Laskowski, "Developing Usability Tools and Techniques for Designing and Testing Web Sites," *Proc. Fourth Conf. Human Factors and the Web,* 1998.
[51] B. Shneiderman, *Designing the User Interface.* Addison-Wesley, 1998.
[52] T. Sliski, M. Billmers, L. Clarke, and L. Osterweil, "An Architecture for Flexible, Evolvable Process-Driven User Guidance Environments," *ACM SIGSOFT Software Eng. Notes,* vol. 26, no. 5, pp. 33-43, 2001.
[53] E. Stroulia, M. El-Ramly, P. Iglinski, and P. Sorenson, "User Interface Reverse Engineering in Support of Interface Migration to the Web," *Automated Software Eng. J.,* vol. 10, no. 3, pp. 271-301, 2003.
[54] E. Stroulia, M. El-Ramly, and P. Sorenson, "From Legacy to Web through Interaction Modeling," *Proc. 18th Int'l Conf. Software Maintenance,* pp. 320-329, 2002.
[55] J. Tidwell, *Designing Interfaces.* O'Reilly, 2005.
[56] K. Tucker and K. Stirewalt, "Model Based User Interface Reengineering," *Proc. Sixth Working Conf. Reverse Eng.,* pp. 56-65, 1999.
[57] User Interface Architecture, IBM Corp., http://www-03.ibm.com/easy/page/1392/$File/IBM_UIA.pdf, 2007.
[58] J. Vanderdonckt, "Development Milestones towards a Tool for Working with Guidelines," *Interacting with Computers,* vol. 12, no. 2, pp. 81-118, 1999.
[59] A. Vanniamparampil, B. Shneiderman, C. Plaisant, and A. Rose, *User Interface Reengineering: A Diagnostic Approach,* Technical Report CS-TR-767, Dept. of Computer Science, Univ. of Maryland, 1995.
[60] M. van Welie Web Design Patterns, http://www.welie.com/patterns/, 2007.
[61] J. Vassileva, "A Task-Centered Approach for User Modeling in a Hypermedia Office Documentation System," *User Modeling and User Adapted Interaction,* vol. 6, no. 2-3, pp. 185-223, 1996.
[62] M. Vering, G. Norris, P. Barth, J.R. Hurley, B. Mackay, and D.J. Duray, *The E-Business Workplace.* John Wiley & Sons, June 2001.
[63] M. Walker, J. Boland, and C. Kamm, "The Utility of Elapsed Time as a Usability Metric for Spoken Dialogue Systems," *Proc. IEEE Workshop Automatic Speech Recognition and Understanding,* pp. 1167-1170, 1999.
[64] WebSphere Business Modeler, http://www-306.ibm.com/software/integration/wbimodeler/, Dec. 2006.
[65] WebSphere MQ Workflow, http://www-306.ibm.com/software/integration/wmqwf/flowcoal.html, Dec. 2006.
[66] M. Weske, W. van der Aalst, and H. Verbeek, "Advances in Business Process Management," *Data and Knowledge Eng.,* vol. 50, pp. 1-8, 2004.
[67] C. Wohlin, P. Runeson, M. Host, M.C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering—An Introduction.* Kluwer, 2000.
[68] *Workflow Management Coalition Terminology & Glossary,* http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf, Dec. 2006.
[69] *Workflow Process Definition Interface—XML Process Definition Language,* http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf, Apr. 2007.
[70] Q. Zhang, R. Chen, and Y. Zou, "Reengineering User Interfaces of E-Commerce Applications Using Business Processes," *Proc. 22nd IEEE Int'l Conf. Software Maintenance,* pp. 428-437, 2006.
[71] Q. Zhang, Y. Zou, T. Tong, R. MacKegney, and J. Hawkins, "Automated Workplace Design and Reconfiguration for Evolving Business Processes," *Proc. IBM Centre for Advanced Studies Conf.,* pp. 320-333, 2005.
[72] Y. Zou, T. Lau, K. Kontogiannis, T. Tong, and R. McKegney, "Model-Driven Business Process Recovery," *Proc. 11th Working Conf. Reverse Eng.,* pp. 224-233, 2004.
[73] Y. Zou and M. Hung, "An Approach for Extracting Workflows from E-Commerce Applications," *Proc. 14th Int'l Conf. Program Comprehension,* pp. 127-136, 2006.
[74] Y. Zou and Q. Zhang, "A Framework for Automatic Generation of Evolvable E-Commerce Workplaces Using Business Processes," *Proc. 28th Int'l Conf. Software Eng.,* pp. 799-802, 2006.

Ying Zou received the BEng degree from Beijing Polytechnic University, the MEng degree from the Chinese Academy of Space Technology, and the PhD degree in 2003 from the University of Waterloo, Canad. She is currently an assistant professor in the Department of Electrical and Computer Engineering at Queen's University, Canada. She is a visiting faculty fellow of the Center for Advanced Studies (CAS), IBM Canada. Her research interests include software engineering, software reengineering, software reverse engineering, model-driven software development, and business process management. She is a member of the IEEE Computer Society.

Qi Zhang received the BASc degree from the University of Ottawa, Canada, in 2003 and the MSc degree from Queen's University, Canada, in 2006. He is currently working toward the PhD degree at the University of Waterloo, Canada. He received and IBM CAS fellowsip in 2005 and was a research assistant at the IBM Toronto Laboratory, Markham, Ontario, in the summers of 2004, 2005, and 2006. His research interests include software maintenance, software reengineering, and user-interface design for e-commerce systems.

Xulin Zhao received the BEng degree from Shandong University, China, in 2001 and the MEng degree from Beihang University, China, in 2004. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering at Queen's University, Canada. His research interests are model-driven software development, quality of models, and usability evaluation. He is a student member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.