Customizing Business Processes in Web Applications

Gustavo Rossi*, Hans Albrecht Schmid**, and Fernando Lyardet***

*LIFIA-Universidad Nacional de La Plata, Argentina
gustavo@sol.info.unlp.edu.ar
**University of Applied Sciences, Konstanz, Germany.
schmidha@fh-konstanz.de
*** Technical University of Darmstadt
fernando@tk.informatik.tu-darmstadt.de

Abstract In this paper we discuss several issues related to the introduction of business processes in the life cycle of Web based E-commerce applications. We first argue that business processes have been so far neglected by modeling and design methodologies treating them as by-products of conceptual and navigational design artifacts, and as a consequence introducing different design and usability problems in the final products. We introduce a novel approach in which processes and activities are treated as first class citizens during application modeling and design. In the core of the paper we analyze the problem of customizing business processes to different user profiles or individuals. We show that using our approach we obtain modular and evolvable solutions.

1. Introduction

With the introduction of new technological advances brought by the World Wide Web, the Internet is being used as a platform for the implementation of complex business applications. Even the most simple e-commerce application includes some kind of embedded business process that must be correctly executed to guarantee the success of the application. The growing trend on automating business tasks by making the underlying applications interoperable using the Internet, brings new problems to software designers. However while there has been a considerable amount of work related with the specification and implementation of business processes in the context of conventional workflow-like applications [7], or with the use of the Internet platform for supporting complex interactions between distributed processes (e.g. using Web Services) [1], the interplays between processes and the usual navigational paradigm of Web applications remains barely unexplored. Moreover, mature Web design methods like OOHDM [9] or WebML [2] either neglect processes or just treat them as by-products of the navigational specification thus introducing design and usability problems.
In [8], we introduced a novel approach for designing business processes in Web Applications by extending the OOHDM design framework with processes and activities both in the conceptual and navigational models. This approach is built on sound software engineering principles: it treats business processes and activities as

first class citizens in the development life cycle; by further decoupling activities from the processes in which they are executed, and by clearly separating process control flow from the navigational behavior of the overall application, our approach improves activities reuse and helps to get rid of inconsistent states and incorrect execution behaviors. This kind of behaviors arises as the result of the interplay between navigation (the usual way of exploring the Web) and the execution of the business process. Objectifying processes and clearly indicating the effect of navigation in the state of process objects is one of the key strategies of our approach. In this paper we elaborate our previous ideas and discuss the problem of customizing business processes.

The structure of the paper is as follows: we first summarize our approach for introducing business processes in OOHDM emphasizing the interplays among processes and navigation. We next present our approach to customize processes to different user profiles or individuals and then present some concluding remarks and further work.

2. Modeling and Designing Business Processes in Web Applications

Introducing business processes in software applications is a difficult task, and it is widely recognized that the different components of a business process should be treated as first-class objects. For example, in [7] the authors introduce components for processes, synchronization features, histories, etc., in the context of a general architecture for workflow applications. We have put our focus in the integration of typical concepts of business process execution with the usual navigational semantics of the WWW. For achieving this objective, instead of defining a new approach from scratch, we decided to enrich OOHDM, a state-of-the-art design method for "conventional" Web applications design, with process features. The result has been appealing as we could use the standard mechanisms for extending the OOHDM framework (namely adding new meta-classes and refining existing behaviors).

OOHDM (as other Web design methods like [2]) focuses on three different design concerns: conceptual or application modeling, navigation design and interface design (we ignore interface issues in this paper). During conceptual modeling, domain classes and application functionality are described using UML [3] primitives.

We partition the conceptual model in two types of classes (described using UML stereotypes): entities and processes. While entities model usual business objects, processes represent set of activities that must be performed to achieve a goal. A process is a composite [4] of activities, which encapsulate their own state (active, suspended, etc); control flow is further decoupled from activities and represented in the corresponding process.

In Figure 1 we show the OOHDM conceptual schema of a simple CD store. At the bottom of Figure 1, we show entities like CD, or ShoppingCart, and at the top of Figure 1, process and activity classes. The conceptual model shows that a business process like CheckOut is typically composed of several activities like the Login Act, ConfirmItems Act, ShippingAddress Act, DeliveryOptions Act, ConfirmAll Act, and PaymentOptionsAct. This is represented by an aggregation relationship in the conceptual schema. Control flow between activities can be represented using UML activity diagrams (we omit them in this paper for the sake of conciseness). The main consequences of treating activities as first class objects are that we are able to reuse the same activity class in different business processes, and that we can implement different flow of activities for the same process, for example to customize it to different users (as shown in Section 3).

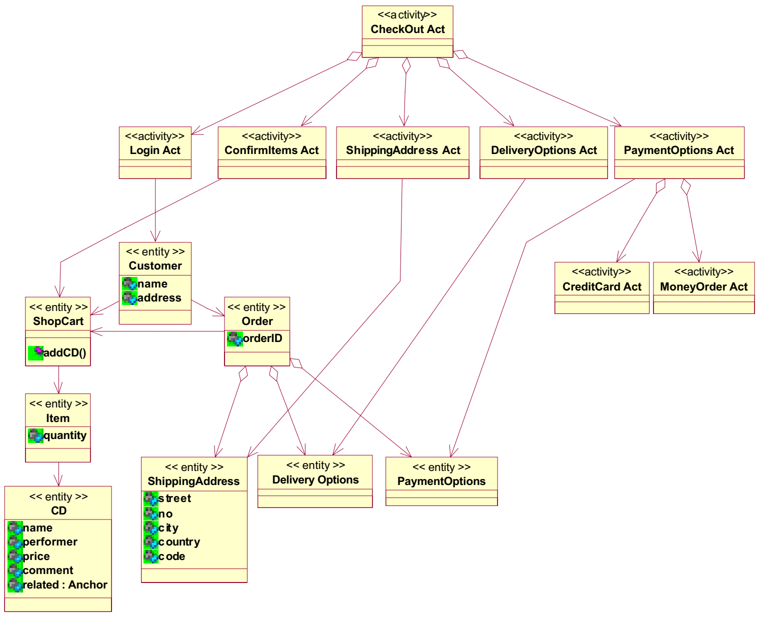

<<activity>> CheckOut Act
<<activity>> Login Act
<<activity>> ConfirmItems Act
<<activity>> ShippingAddress Act
<<activity>> DeliveryOptions Act
<<activity>> PaymentOptions Act
<<activity>> CreditCard Act
<<activity>> MoneyOrder Act
<< entity >> Customer
name
address
<< entity >> ShopCart
addCD()
<< entity >> Order
orderID
<< entity >> Item
quantity
<< entity >> CD
name
performer
price
comment
related : Anchor
<< entity >> ShippingAddress
street
no
city
country
code
<< entity >> Delivery Options
<< entity >> PaymentOptions

Figure 1: Conceptual Schema of CD store with processes

In OOHDM, the navigational model describes the objects that the user will perceive (i.e. the nodes), and the navigation topology (links and indexes); node classes represent a view of conceptual classes, while links represent the hypermedia counterpart of application relationships. OOHDM structures the navigational space into navigational contexts. A context usually represents a set of nodes in a particular task the user is performing. For example a customer can search and access CDs of a particular group, CDs in a musical gender, or CDs in a period of time, etc. Each time he accesses a particular CD of one of these sets, we may provide him with context-related features such as going to the next or previous CD in this set. We can also allow or prevent the user from performing certain actions according to the actual context.

We have slightly extended the OOHDM meta-model by defining activity nodes (the process counterpart of nodes) as shown in Figure 2.

In the navigational model, activity nodes, like LoginAct Node and ShippingAddressAct Node (Figure 2 right) play the same role as navigational nodes. They describe, in an abstract way, the visible attributes, anchors and operations with which the user will interact during process execution. The interface of an activity node (for example a Web page) will contain buttons like “cancel” or “next” that control the input processing of an activity and the control flow to a subsequent activity. Activity nodes like LoginAct Node are shown in the context of the corresponding process node to which they belong (a composite in OOHDM), like the CheckOutAct Node. This is indicated by drawing the activity nodes within the box of the process in which they are actually executed. Incoming links (like “resume” from CD node) do not point to activities but to the process node; when a process node must be activated, it gives control to the current activity (e.g. the one that has been suspended or that must be initiated).

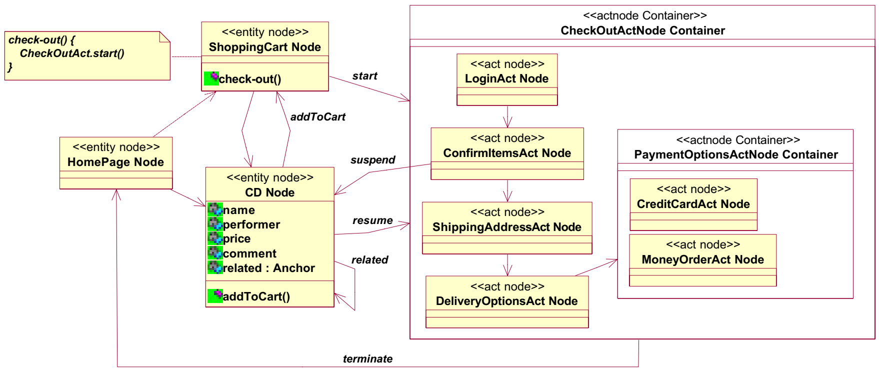


Figure 2:Navigational Schema of CD store with activity nodes

We have added some new types of links to the OOHDM meta-model, namely “suspend”, “abort” and “terminate” links in order to deal with those links whose source node is an activity node. Suspension links complement the usual link navigational semantics by triggering a message to suspend the activity that corresponds to the source node. In our example, when the user navigates from the ConfirmItems activity node to the CD node (following the “suspend” link), the process is suspended (and will be later resumed in the corresponding state). Abortion and termination links are similar to the suspend link.

The use of activity nodes and associated links help to overcome the usability problems occurring when implementing business processes in the Web by emulating them as navigation sequences. In our approach, when an activity is left, it is suspended, aborted or terminated and the corresponding process is aware of this change of state. This awareness is achieved as the outgoing links trigger the change of state in the corresponding activity/process.

When the process is resumed it can then be started in the corresponding activity because, as previously mentioned, when a process node is started/resumed it gives control to the corresponding child activity; at the same time, as activities are modeled as first class objects, they can store their state and be re-initiated safely.

A further problem arises when leaving a process using a navigation link. For example in the checkout process, we can navigate to a CD page from the ConfirmItems activity as shown in Figure 2. Should the user be allowed to add the CD to the shopping cart again, while the checkout process is suspended, and if so what would be the semantics of this action? We may not want to allow this operation since the checkout process may already have created the order with the items currently in the shopping cart. Therefore, an item added during navigation would not be taken into account when the checkout process is resumed, thus confusing the customer. More generally, when a business process is suspended, a user should not perform operations that modify the state of resources being used by the process.

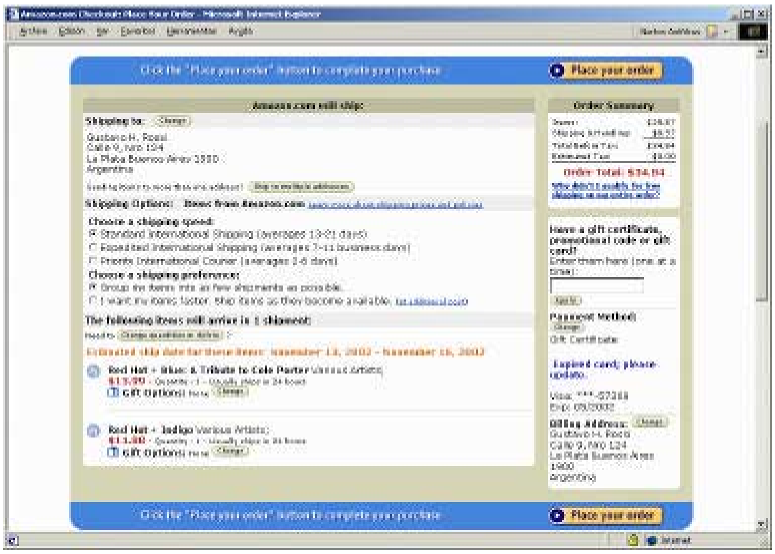
A good solution is to remove the action: “add to cart” from a CD node, when we access the CD after leaving the checkout process. We can achieve this objective easily by combining processes with OOHDM navigational contexts. In our approach, every process defines a navigational context: this means that when a user suspends a process, navigation occurs in the navigational context of this process. The navigational context of a process specifies, in the same way as a usual navigational context, which restrictions or additions apply to a node when it is accessed in the context of this process. In this way, we can make a “fine-tuning” of the features of nodes when accessed in the context of a business process or even from a particular activity in the process.

3. Customizing Processes

Customization has become hype in areas such as electronic commerce; we can find hundreds of applications that claim to be fully customizable to different user profiles or individuals. There are many different customization patterns: for example we may personalize the links allowing different users to explore different pages such as in Amazon recommendations; we can adapt the contents and/or the structure of a page to let different users access individualized contents, as in my.yahoo.com, etc.

Regarding business processes, we may also have many alternatives: for example in the checkout process of www.amazon.com, a new customer has to follow the previously mentioned step by step procedure (one page for each activity), while a registered customer just confirms all data in one page as shown in Figure 3; a customer can also sign for what is called “one-click” check-out in which the process is “automatic”, etc.

A more elaborated customization policy may provide special offers for holders of a particular credit card; in this case we may need to “expand” the normal process control flow to add a new activity to let the user select from a number of offers. Notice that many of these customized behaviors may require modifying the application code, e.g. the process classes; thus, the way in which we design processes and activities is critical to achieve modular and painless software evolution. For the sake of conciseness we will only focus customization to the user profile, i.e. personalization, ignoring other customization criteria such as date, time, actual network connection, etc. However, the principles exposed here are easily applied to the most general customization case.


Amazon.com Checkout: Place Your Order - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Click the "Place your order" button to complete your purchase
Place your order
Change
Ship to multiple addresses
Change quantities or delete
Apply
Internet

Amazon.com will ship:

Shipping to:

Gustavo H. Rossi
Calle 9, nro 124
La Plata Buenos Aires 1900
Argentina

Sending items to more than one address?

Shipping Options: Items from Amazon.com [illegible]

Choose a shipping speed:
- [✓ Standard International Shipping (averages 13-21 days)]
- Expedited International Shipping (averages 7-11 business days)
- Priority International Courier (averages 2-6 days)

Choose a shipping preference:
- [✓ Group my items into as few shipments as possible.]
- I want my items faster. Ship items as they become available. [illegible]

The following items will arrive in 1 shipment:

Estimated ship date for these items: November 13, 2002 - November 16, 2002

Red Hot + Blue: A Tribute to Cole Porter Various Artists;
$15.99 - Quantity: 1 - Usually ships in 24 hours
Gift Options: None

Red Hot + Indigo Various Artists;
$11.88 - Quantity: 1 - Usually ships in 24 hours
Gift Options: None

Order Summary

Items:	$26.87
Shipping & Handling:	$8.57
Total Before Tax:	$34.84
Estimated Tax:	$0.00

Order Total: $34.84

Why didn't I qualify for free shipping on my entire order?

Have a gift certificate, promotional code or gift card?
Enter them here (one at a time):

Payment Method:
Gift Certificate

Expired card; please update.

Visa: ***-57218
Exp: 09/2002

Billing Address:
Gustavo H. Rossi
Calle 9, nro 124
La Plata Buenos Aires
1900
Argentina

Figure 3: One page checkout in Amazon for registered users

We have claimed elsewhere [10] that customization should be addressed using a design more than an implementation view. This means that once we understand *what* we want to personalize, we have to express this personalization feature using the corresponding design primitives, before deciding *how* it will be implemented. Treating processes and activities as first class objects, and modeling them in the context of the OOHDM framework allow us to apply most of the design rules defined in [10] for achieving seamless process customization.

The simpler example of process customization consists in providing different navigation/interface functionality for the same process or activity to customize it to different user roles. For example when an Amazon employee performs the checkout process he may be provided with different navigational options that a regular user can not see; in OOHDM this is achieved by defining different navigational schema, providing different linking topologies, one for each user role as shown in the simplified diagram of Figure 6. Both navigational views in Figure 4 share the same conceptual model (e.g. the one in Figure 1). However, while performing checkout the customer can navigate to the CDs in the cart while the employee can also access information about benefits related with the products he is buying. Notice that this "pure" navigational customization is transparent for process and activity classes.

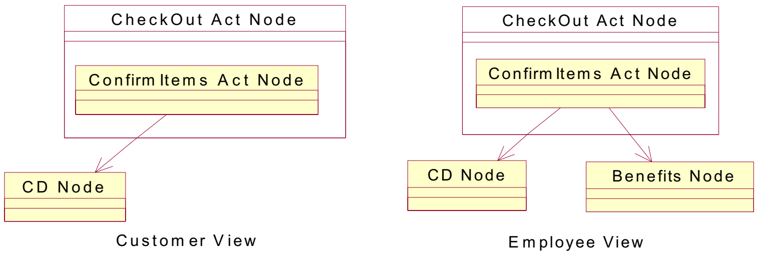


Figure 4: Customizing processes using OOHDM views

We may want to provide a simplified checkout interface for registered users like the one in Figure 3. Again, the OOHDM viewing mechanism allows specifying the ExpressCheckOut activity node containing all the information provided by the former simpler activities as shown in Figure 5.

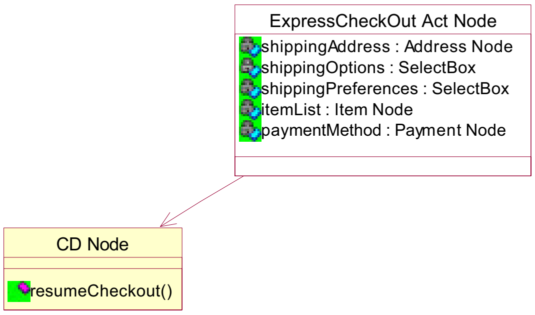

ExpressCheckOut Act Node
shippingAddress : Address Node
shippingOptions : SelectBox
shippingPreferences : SelectBox
itemList : Item Node
paymentMethod : Payment Node
CD Node
resumeCheckout()

Figure 5: Modeling an express checkout activity node

This activity provides an interface in the spirit of OOHDM composite nodes. Each component node, e.g. item List or shipping address, replaces the corresponding activity in Figure 2. Notice that each of these attributes belongs to a type (e.g. Item Node), which is itself a full-fledge node class. There is no need to define an “internal” control flow; however, if this activity is left following an anchor, the semantics of suspension and resumption remain as discussed before, as well as the notion of navigational process context.
However there is a subtlety in this example regarding the process state and the control flow: when the checkout process is started it has to check whether it is dealing with a registered user to define the corresponding activity view.
An elegant solution to this problem consists in delegating the decision about which activity must me started, to a user profile object; the profile object then returns the corresponding (activity) view as shown in Figure 6. The process object will then be ready to receive the “done” message to finish processing, for example defining its current state as being in the ConfirmAll activity

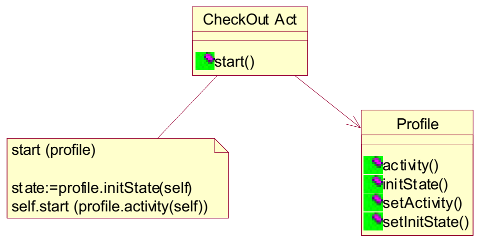


Figure 6: Customizing different views of the same process

In Figure 6 the start method in CheckOut asks the profile to return the initial state and then starts the corresponding activity, also provided by the profile object. Notice that class Profile also contains methods for defining states and activities; in this way we avoid having to sub-class Profile for different user profiles.
Finally, we might want to completely customize the control flow between activities according to the user profile. For example certain customers that paid a special fee when registering have a express delivery option, and thus they do not have to choose one of the options; or a set of cheaper products may be offered to customers paying with a Visa card and thus, a new activity has to be introduced. If we want to achieve complete process customization according to the user profile, the best solution is to decouple the control flow from the process object and delegate it to the corresponding profile object as shown in Figure 7.
In the micro-arquitecture of Figure 7, the behavior for deciding which is the "next" activity is delegated to the profile object as well as the information on the current state.

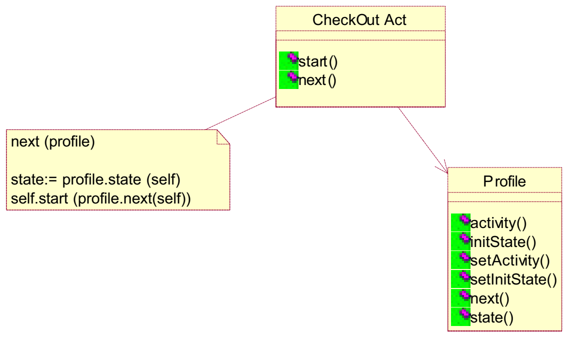


Figure 7: Decoupling process control flow for achieving customization

A generic architecture for achieving more complex customization policies such as those related with network connection, interface appliances, etc., is described in [5]. It further decouples those policies from the application code (in particular from the process objects) in order to separate the different concerns involved in the problem: the customization rules, the process (and domain) objects and the profile. Our approach can be easily used in the context of this architecture just by further separating the rules that guide processes to a separate component.

4. Concluding Remarks

In this paper we have elaborated our approach for introducing business processes in Web applications to show how to customize business processes. We have shown that treating processes as first class citizens allows us to model different customization strategies: for example we can adapt a business process either to an individual or to different user profiles in a seamless way, i.e. without having to deal with messy code. For the sake of space we have not discussed other possible customization examples. For example it is not difficult to adapt the process control flow to the context in which the process is being executed by combining the idea of navigational context with the requirements posed by customization.

We have used the proposed design method successfully for a number of Web applications, both in student projects and in cooperation with software houses in real world projects. Some of these applications are a customer relation management system for small and medium sized shops and companies, which embodies different business processes, a cooperative travel agency where users can share traveling opportunities, and several Web shops. We are currently working on defining a software arquitecture to implement business processes in Web applications by extending the Model-View Controller paradigm [6] with a new layer: the process layer. In this way we can overcome the well-known disadvantages of the MVC when used in applications with complex logic. In the same direction we are studying how to improve our notation by including some advanced UML features such as stereotypes and constraints. We are finally studying how to extend the OOHDM meta-model to include features related with workflow applications in which different users can collaborative participate in the execution of the same business process.

References

1.Business Process Specification Schema, www.ebxml.org

2. Ceri, S., Fraternali, P: Web Modeling Language (WebML): a modeling language for designing Web sites. Proceedings of the 9th. International World Wide Web Conference, Elsevier 2000, pp 137-157.

3. Conallen, J., Building Web Applications with UML. Addison Wesley 2000.

4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Elements of reusable object-oriented software. Addison Wesley 1995.

5. Kappel G., Retschitzegger W.. Modeling Ubiquitous Web Applications: The WUML approach. International Workshop on Data Semantics in Web Information Systems (DASWIS-2001), Springer Verlag LNCS, 2001.

6. Krasner, G. Pope, S.: A cookbook for Using Model-View-Controller interface paradigm in Smalltalk 80. Journal of Object Oriented Programming, August/September 1988, 26-49

7. Manolescu, D. and Johnson, R. A micro-workflow component for federated workflow. OOPSLA Workshop on O-O Workflow, available at http://micro-workflow.com
8. Schmid H. A., Rossi, G.: "Modelling and Designing Business Processes in Web Applications" in Proceedings of EC-Web 2002, the International Workshop on E-Commerce and the WWW, Springer Verlag LNCS, 2002.
9. Schwabe, D., Rossi, G.: "An object-oriented approach to web-based application design". Theory and Practice of Object Systems (TAPOS), Special Issue on the Internet, v. 4#4, pp.207-225, October, 1998.
10. Schwabe, D, Rossi, G, Guimaraes, R: Cohesive Design of Personalized Web Applications. IEEE Internet Computing, pp 34-43, March 2002