Chapter I

Design Recovery of Web Application Transactions

Scott Tilley, Florida Institute of Technology, USA

Damiano Distante, University of Lecce, Italy

Shihong Huang, Florida Atlantic University, USA

Abstract

Modern Web sites provide applications that are increasingly built to support the execution of business processes. In such a transaction-oriented Web site, the user executes a series of activities in order to carry out a specific task (e.g., purchase an airplane ticket). The manner in which the activities can be executed is a consequence of the transaction design. Unfortunately, many Web sites are constructed without proper attention to transaction design. The result is a system with unpredictable workflow and a lower quality user experience. This chapter presents an example of the recovery of the "as-is" design model of a Web application transaction. The recovery procedure is prescriptive, suitable for implementation by a human subject-matter expert, possibly aided by reverse engineering technology. The recovered design is modeled using extensions to the transaction design portion of the UML-based Ubiquitous Web Applications (UWA) framework. Recovery facilitates future evolution of the Web site by making the transaction design explicit, which in turn enables engineers to make informed decisions about possible changes to the application. Design recovery of a commercial airline's Web site is used to illustrate the process.

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Introduction

As with other kinds of software systems, Web sites undergo maintenance and evolve over time in response to changing circumstances. For complex Web sites supporting the execution of business processes, evolution can be particularly challenging. The hidden nature of the transaction model in the overall design of most Web sites further exacerbates the situation.

Business processes are realized as transactions that are triggered as the user executes a series of activities in order to carry out a specific task (e.g., purchase an airplane ticket). The manner in which the activities can be executed is a consequence of the transaction design. Therefore, the quality of the transaction design can have a direct influence on the quality of the user experience.

Unfortunately, many Web sites are constructed without proper attention to transaction design. It is quite common to incorrectly treat a transaction as a sequence of navigational steps through pages of the Web application (Rossi, Schmid, & Lyardet, 2003; Schmid & Rossi, 2004). The result is a system without an explicit transaction design, which leads to unpredictable workflow, maintenance difficulties, and a potentially frustrating session for the user.

This chapter presents an example of the recovery of the "as-is" design model of a Web application transaction. The recovery procedure is prescriptive, suitable for implementation by a human subject-matter expert, possibly aided by reverse engineering technology (Tilley, 2000; Müller et al., 2003). The recovered design is modeled using extensions to the transaction design portion of the Ubiquitous Web Applications (UWA) framework (UWA, 2001f). Recovery facilitates future evolution of the Web site by making the transaction design explicit, which in turn enables engineers to make informed decisions about possible changes to the application. Design recovery of a commercial airline's Web site is used to illustrate the process.

The next section outlines UWAT+, which is a refinement of the UWA transaction design model. The section "The Design Recovery Procedure" describes the design recovery procedure, including a formalization of the transactions, the creation of the Execution Model, and the construction of the Organization Model. The section "An Illustrative Example" demonstrates the use of the procedure on a representative Web site from the travel industry. Finally, "Summary" goes over the main points of the chapter and outlines possible avenues for future work.

UWAT+

The Web provides a distributed information system infrastructure as the base platform for application deployment. Indeed, one of the reasons for the success of e-commerce business today is the transactional behavior that the Web offers. However, for many Web sites that are already in use and in need of maintenance, this widely used behavior is often too complex, consisting of several ill-defined sub-transactions which can hinder

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

systematic evolution. The transaction design for such applications needs to be more explicit, flexible, and take users' goals into account.

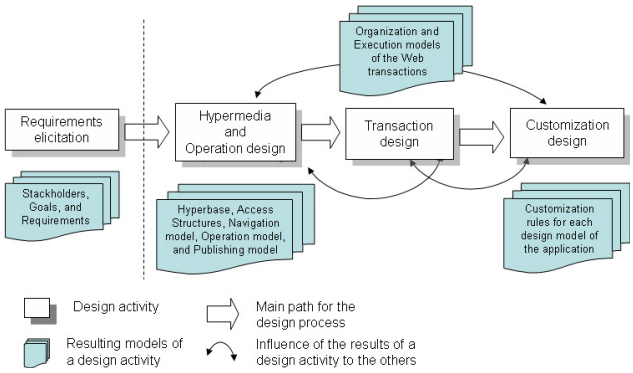
The UWA framework provides a complete design methodology for ubiquitous Web applications that are multi-channel, multi-user, and generally context-aware. As illustrated in Figure 1, the UWA design framework organizes the process of designing a Web application into four main activities (UWA, 2001a): (1) requirements elicitation (UWA, 2001b); (2) hypermedia design and operation design (UWA, 2001c); (3) transaction design (UWA, 2001d); and (4) customization design (UWA, 2001e). Each design activity results in a unique design model, which can iteratively affect the creation of other designs elsewhere in the process.

The UWA framework represents an excellent platform on which to build the conceptual modeling portion of the design recovery procedure. This section outlines a refined and extended version of the UWA framework, called UWAT+, which focuses specifically on extensions to the transaction design model. In the UWA vernacular, "transactions" represent the way business processes are addressed and implemented in Web-based applications. The extensions to the UWA transaction model include simplifications and extensions related to the definition of Activity and enhancements to several aspects of the Organization and Execution models, which are (according to the UWA) the main models on which the design of a Web transaction is based. Extensive details of UWAT+ are provided in Distante (2004); this section provides an overview of the salient features used for design recovery.

Changes to the Definition of Activity

Activities taken into account by the Organization and Execution model of a transaction implementing a business process should only be those that are meaningful for the user of the Web-based application; system-related activities and data-centered operations can be de-emphasized. This implies that in UWAT+, the OperationSet of an activity is no longer considered, mainly because it is primarily related to data-level details and to the implementation of a transaction, whereas user-centered design recovery is more concerned with conceptual models.

Figure 1. An overview of the UWA application design process



Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

An Activity's PropertySet is redefined to be more user-oriented, through the introduction of a new property (Suspendability), and the tuning of the semantics associated with the previously existing properties. The extended PropertySet set is now Atomicity, Consistency, Isolation, Durability, and Suspendability (ACIDS).

Changes to the Organization Model

The Organization model describes a transaction from a static point of view, modeling the hierarchical organization in terms of Activities and sub-Activities in which the Activity can be conceptually decomposed. It also describes the relations among these activities and the PropertySet of each of them. The Organization model is a particular type of Unified Modeling Language (UML) class diagram (Booch, Rumbaugh, & Jacobson, 1998), in which activities are arranged to form a tree; the main activity is represented by the root of the tree and corresponds to the entire transaction, while Activities and sub-activities are intermediate nodes and its leaves.

In UWAT+, significant changes have been made to the Organization model by dividing the possible relations between an activity A_1 and its sub-activities $A_{1.1}$ .... $A_{1.n}$ into two categories: the Hierarchical Relations and the Semantic Relations. As shown in Figure 2 and Figure 3, the two categories are defined as follows:

Hierarchical Relations: The set of "part-of" relations from the Organization model. It is composed of relations such as *Requires*, *RequiresOne*, and *Optional*.

Semantic Relations: The set of relationships that are not a "part-of" type. Relations among sub-activities of different activities are normally part of this kind of relation. The list semantic relations currently consists of the *Visible*, *Compensates*, and *Can Use*.

The changes to the Organization model provide a better modeling instrument with which design recovery can be accomplished. In particular, the distinction between hierarchical and semantic relations permit the designer to reason about transactions in a manner that is not possible with the unadorned UWA model. This in turn can lead to improvements in support for the business processes realized by the Web application.

Changes to the Execution Model

The Execution model of a transaction defines the possible execution flow among its Activities and sub-Activities. It is a customized version of the UML Activity Diagram (Bellows, 2000), usually adopted by the software engineering community to describe behavioral aspects of a system. In the execution model, the sequence of activities is described by UML Finite State Machines, Activities and sub-Activities are represented by states (ovals), and execution flow between them is represented by state transition (arcs).

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Figure 2. An example of the Organization model highlighting the hierarchical relationships

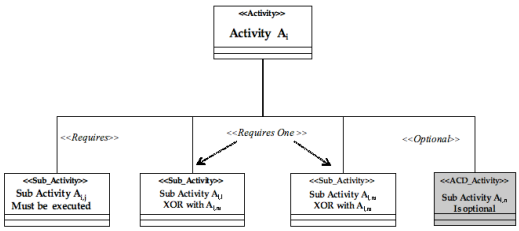
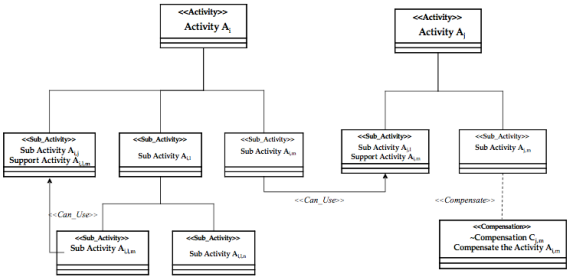


Figure 3. An example of the Organization model highlighting the semantic relationships



The original Execution model includes both user- and system-design directions for the developer team. Since our focus is more on the former than the latter, several changes have been introduced into UWAT+.

Commit and Rollback Pseudo-State: These two pseudo-states that exist in the original UWA execution model have been removed. Positive conclusion of an Activity is directly derived by the execution flow in the model, while the failure or the voluntary abort of it is modeled by the unique pseudo-state of "Process Aborted" in an Execution model.

Transition Between Activities: Each possible user-permissible transition between activities must be explicitly represented in the model with a transition line between them. The actions that trigger the transition should be specified on the transition line with a transition label. Compensation activities (activities which rewind the results of others) needed to allow a transition between two activities are implicit and controlled by the system. No transition of the Execution model can be associated with the action of the user selecting the "Back" navigation button in the browser, which should be disabled in order to avoid client-side to server-side data inconsistencies.

Transition Labels: A classification of the possible labels that can be associated with the transition lines of an Execution model has been introduced, with a simple labeling mechanism being used to indicate the category of the transition:

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

- A: Action invoked by the user;
- C: Condition(s) required for Activity execution;
- R: Result of activity execution;
- S: State associated with system due to Activity.

Failure Causes and Actions Table: A list of causes of Activity failure and possible actions the user or the system can take is maintained. The list also explains why an Activity fails and how the user or the system can react.

Adoption of Swimlanes: It is suggested that swimlane diagrams (OMG, 2003) be adopted when it is useful to describe how two or more user-types of the application collaborate in the execution and completion of a transaction.

The changes to the Execution model provide better visibility into the dynamic execution paths the user will experience while completing a specific transaction. By making such paths explicit, improvements in the transaction design can be more easily accomplished. However, for such paths to be modeled properly for existing Web sites, they must first be recovered.

The Design Recovery Procedure

Given an existing Web site, the goal is to populate an instance of the UWAT+ model described in the previous section with data from the site's content and structure. The resultant model can then be used to guide restructuring decisions based on objective information concerning the quality attributes of the business process' implementation by the Web-based application. The model can be recreated using a three-step prescriptive design recovery procedure: (1) formalization of the transactions; (2) creation of the Execution model; and (3) construction of the Organization model for each of the identified transactions.

A human subject-matter expert can accomplish this design recovery procedure without any tool support. However, as described in the subsection "Future Work" at the end of this chapter, the use of automated reverse engineering technology (Chikofsky & Cross, 1990) may improve the efficacy of the process. Extensive details of the design recovery procedure using reverse engineering are provided in (Distante, Parveen, & Tilley, 2004); this section provides an overview of these three steps.

Formalization of the Transactions

In the first step of the process, the user-types of the application and their main goals/tasks are formalized. Only goals/tasks that can be defined as "operative" are considered and a transaction is associated with each of them. Overlapping tasks of two or more user-types suggests UML swimlanes in the corresponding transaction's Execution model. At the end of this step the list of transactions implemented by the application is obtained.

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Creation of the Execution Model

For each of the transactions found in the first step, the Execution model is created by first performing a high-level analysis of the transaction in order to gain a basic understanding of its component Activities and Execution Flow. The transaction is then characterized as "simple" (linear) or "composite" (with two or more alternative execution paths). If the transaction is composite, then it should be further decomposed into sub-transactions until only simple transactions remain. Each simple transaction can be investigated separately. To each transaction (simple and composite) an Activity of the Execution model (and later of the Organization model) is associated.

A first draft of the Execution model is created for each simple transaction identified by executing it in a straightforward manner. Failure events are not yet taken into account in the model. The draft Execution model is then refined with deeper analysis of the transaction. All the operations available to the user during the execution of the transaction are invoked. Erroneous or incomplete data are provided in order to model failure states and possible actions the user can undertake. In this analysis phase, new secondary execution flows of the transaction can be found, and the reverse modeling technique could be invoked recursively as needed.

Finally, the table that describes the possible failure causes and the corresponding user actions or system invocations is investigated for each of the sub-activities that have been found.

Construction of the Organization Model

Once the Execution model has been obtained for a transaction, the Organization model can be constructed, which will model the transaction from a static point of view. The Execution model is used to determine the set of Activities and sub-Activities of a transaction. In the case of a simple transaction, the set is determined by all the Activities and sub-Activities encountered in the single flow of execution available to the user. In the case of a composite transaction, the set is composed of the union of the Activities and sub-Activities of the single transaction that have been found for the composite transaction.

The tree structure of the Organization model is constructed by aggregating sub-Activities that are conceptually part of an Activity ancestor. Singleton Activities that are related to the transaction are modeled with a Can-Use Semantic Relation. Each arc in the tree represents either a hierarchical or semantic relation. To define Hierarchical Relations, the analyst can refer to the Execution Flow defined by the Execution model and conditions and execution rules defined in it. However, defining the semantic relations still requires direct inspection of the application.

For each Activity and sub-activity, it is necessary to define the value for the ACIDS PropertySet. The analyst is required to refer to the definition given for each of the properties in the UWA documentation and discover the value to be assigned to each of them through direct inspection using the Web-based application.

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

An Illustrative Example

To illustrate the potential benefits of design recovery, this section of the chapter focuses on the use of the procedure described in the previous section. This technique is used to recover the as-is transaction design model using the formalism outlined in the section "UWAT+" of a real-world Web-based application. The application selected is the flight reservation system of Alitalia airlines (Alitalia, 2004). The Italian Alitalia Web site (*www.alitalia.it*) was chosen because it is representative of a commonly used e-commerce application, and one that appears to offer significant room for improvement from a user's perspective. The analysis refers to a period of observation from November to December 2003. It should be emphasized that it was the Italian version of the Alitalia Web site that was analyzed; the versions for other locales, such as the USA, have been found to be quite different.

The specific transaction from the Web site used to illustrate the design recovery procedure is called "*Round-Trip Flight Reservation.*" The next two subsections describe the Organization and Execution models representing this transaction recovered from the Alitalia Web site. The subsection "Discussion" narrates some of the perceived shortcomings of the recovered transaction design that become apparent using these two models.

The recovery was realized using a manual reverse engineering process. There is no inherent reason why this process could not be made more efficient through the use of appropriate tool support. For example, one possible useful tool would be a UML editor with UWAT+ profiles. However, such tools do not as yet exist.

The Recovered Organization Model

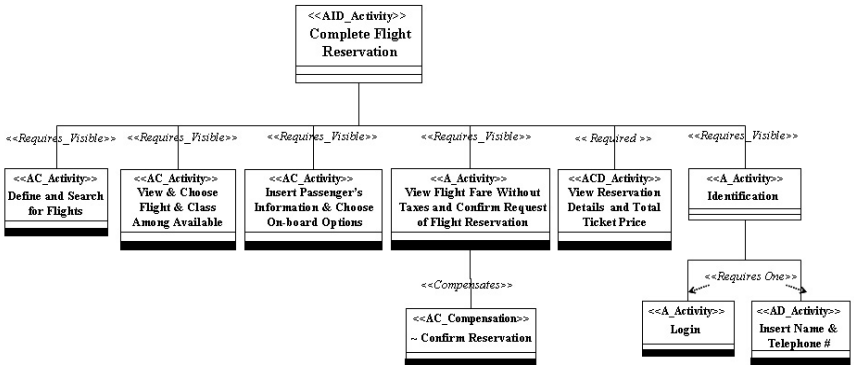
The recovered "as-is" Organization model of the "*Complete Flight Reservation*" process is shown in Figure 4. The Organization model has a tree structure, with the Activity corresponding to its root representing the main process. The model includes the Hierarchical and Semantic Relations existing among the Activities, and for each Activity, the PropertySet it verifies.

The Activity names used in the model are purposely lengthy, so that they indicate some of the characteristics worthy of attention later in the recovery process. The user-type simulated in the analysis is "*Nonregistered User.*" Unless otherwise stated, this is the user-type implied in the following discussion.

A registered user can choose to execute one of the following three activities to reserve a flight using the Alitalia Web site: *Fast Flight Reservation*, *Complete Flight Reservation*, and *Managing Reserved Flights*. These activities are in fact connected to the main activity of the diagram (the reservation process) with a *Requires One* relation. This last activity is available only to the user type *Registered User*. The *Payment* activity is an optional activity that the users could execute if they wish to purchase the corresponding ticket online. The hierarchical relation of *Optional* that links it to its ancestor indicates this.

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Figure 4. The "as-is" Organization model of the "Complete Flight Reservation" Activityin the Alitalia.it Web site



The model in Figure 4 details the *Complete Flight Reservation* Activity; the other Activities (which are represented by a filled version of the UML Class Stereotype), are omitted for lack of space. The PropertySet of the *Complete Flight Reservation* activity is set to *AID* (*Atomic*, *Isolated*, and *Durable*). It was observed that the user could experience inconsistency among data visualized, so the Activity is not *Consistent*. Moreover, the Activity is not *Suspendable* because it cannot be suspended in any time; instead, it must be completed during one usage session of the application.

The diagram also shows the sub-activities into which the *Complete Flight Reservation* Activity can be conceptually decomposed. These are *Define and Search for Flights*, *Choose Flight & Class Among available*, *Insert Passenger's Information & Choose On-board Options*, *View Flight Fare Without Taxes and Confirm Request of Flight Reservation*, *View Reservation Details and Total Ticket Price*, and *Identification*. All of these are activities required for the main *Complete Flight Reservation* Activity to be completed.

The activities that correspond to the leaves of the tree are elementary ones the user normally executes in a single episode. The model shows that the user can accomplish the *Identification* Activity by either logging into the system (*Login* activity, available only for registered users) or providing a name and a telephone number *(Insert Name and Telephone #* activity). These two activities are in fact related to their ancestor with a *Requires One* hierarchical relation.

Most of the activities in the recovered Organization model were found to be *Visible* since the changes they affect on data are visible by other activities during a session. The *~Confirm Reservation* activity is a *Compensates* activity which in essence rewards the effects of a successfully completed reservation when the user decides to delete it.

The Recovered Execution Model

The recovered "as-is" Execution model of the *Complete Flight Reservation* process is shown in Figure 5. The model details the activities of *Complete Flight Reservation* and

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Figure 5. The "as-is" Execution model of the "Complete Flight Reservation" Activityin the Alitalia.it Web site

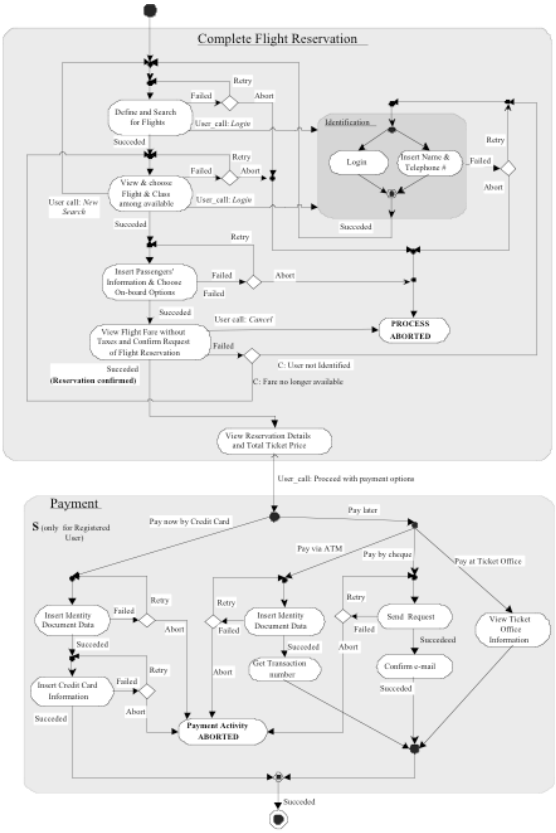


Figure 6. The activity of "Define and Search for Flights" in the Alitalia.it Web site

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Payment of the Organization model described in the "UWAT+" section. In the following discussion, the most linear flow of execution the user can experience while reserving a seat using the Alitalia Web site is used. (Note that while the Execution model also depicts the sub-Activities that compose the *Payment* Activity and describes the set of payment options the user can choose from, this activity is quite simply structured and is therefore not the focus of the design recovery process.)

The process of *Round-Trip Flight Reservation* requires five steps to be completed. These steps are illustrated in sequence by referring to the Execution model in Figure 5 and the screenshots of the Web page of the application supporting each of the five activities.

Step 1. The user starts the flight reservation process by defining the request of "*Round-Trip Flight Reservation*" and starting the flight search (*Define and Search for Flights*). Figure 6 shows a screenshot of the Alitalia Web page for this Activity. For this Activity to be successfully executed, the user must indicate the number of passengers, the preferred class, the departure and destination airports, and the departure and return dates. Default values are provided for most of the required input parameters; utilities that might have been helpful, such as a calendar, are not available to the user.

Step 2. If the flight search is successful, a list of possible flights (with different routes between departure and destination, and different times) is proposed for the itinerary (with an indication of the traveling class) specified in the previous step. The user is then required to choose the preferred path from the departure airport to the destination, and class of travel (*View & Choose Flight & Class Among Available*). Even if the preferred traveling class was specified in Step 1 of the process, the system still shows flights belonging to other classes. Figure 7 shows the screenshot of the Web page that enables the user to execute this Activity.

Step 3. To proceed ahead toward the completion of the reservation process, the user is required to provide all the information about the traveling passengers and choose for them the On-Board options such as the preferred meal and eventual needs for assistance

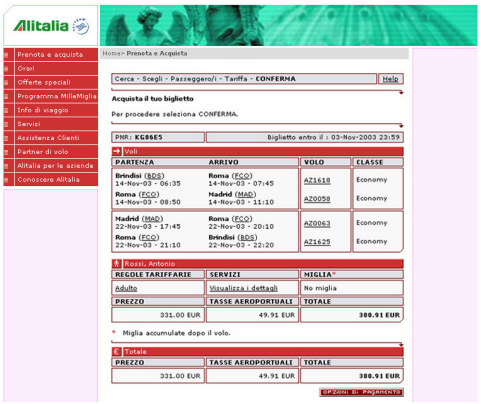
Figure 7. The activity of "Choose Path & Class Among Available" in the Alitalia.it Web site

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Figure 8. The "Insert Passenger Information & Choose On-board Options" Activity

Figure 9. The "View Fare Without Taxes and Confirm Request of Flight Reservation" Activity

Figure 10. The "View Reservation Details and Total Ticket Price with Taxes" Activity



services (*Insert Passenger's Information & Choose On-board Options*). Figure 8 shows a screenshot of the Web page in charge of this activity.

Step 4. After the previous activities have been successfully completed in sequence, the flight details and fare (without taxes) are shown. The user is asked to confirm the choices in order to effectively request the flight reservation and the system to commit it (*View Flight Fare Without Taxes and Confirm Request of Flight Reservation*). This is shown in Figure 9.

Step 5. If the flight reservation request succeeded, then the activity of *Complete Flight Reservation*, represented in Figure 5 by a large rounded rectangle as background, can

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

be considered successfully completed. As shown in Figure 10, at this point the user is provided with the flight reservation code, the date they must purchase the ticket, details of the reserved flight, and the total price (taxes included) (*View Reservation Details and Total Ticket Price*). A necessary condition for the reservation confirmation to be successful is that the user has previously been identified to the system by executing the *Identification* Activity. This is accomplished by either logging into the system (in the case of a registered user), or by specifying first and last name and providing a telephone number (in the case of an unregistered user).

When Step 5 is completed, the user can exit the process or proceed with the *Payment* for purchasing the ticket corresponding to the reservation just confirmed by the system. However, only the link that starts the *Payment* activity is made explicit by the application; neither "Exit" nor "End Process" button is provided to the user to indicate that the reservation process has been completed and can be exited. The model shows that only the user call "*Proceed with Payment Options*" is provided to the user. This may be considered a navigation problem, but if known at design time, it could have been avoided.

In the case of a registered user, the reservation is stored in Personal Travel Book and from here the user can pay the ticket online at a later time. In the case of an unregistered user, if the process is exited without contextually executing the *Payment* activity (e.g., simply closing the browser or navigating outside the pages related with the process), the user has no way to restart the Activity and to buy the ticket in a later session. The previous consideration tells us that the *Payment* activity is *Suspendable* only for registered users.

As mentioned at the beginning of this section, Steps 1 to 5 depict the normal, linear execution flow for the process of *Complete Flight Reservation*. However, depending on the user's choices and the system's responses, several execution paths can be followed. For example, the execution flow can take a different path from the one described if one of the involved activities fails.

Failure causes and corresponding actions that can be undertaken by the user or the system in response to them are described by *Failure Tables*. Two of these are summarized in Table 1 and Table 2. The first table refers to the activity of *Define and Search for Flights*, while the second refers to the possible failure causes identified for the activity of *View Flight Fare Without Taxes and Confirm Request of Flight Reservation*.

Discussion

A number of observations can be made regarding possible areas of improvement for the Alitalia transaction design, based on the as-is recovered Organization and Execution models and taking into account other information extracted from the application during its direct analysis. Five areas that can be considered the most important are: the suspendability of the entire process of *Complete Flight Reservation*, the *Identification* Activity, the *Insert Passenger's Information & Choose On-board Options* activity, visualizing total cost, and restarting the process. Each of these shortcomings (SC) is discussed in turn.

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

SC1: Suspendability of the Entire Process

The first consideration concerns the *Suspendability* of the entire "*Complete Flight Reservation*" transaction. As shown by the Execution model, none of the activities involved in the process are *Suspendable*. For example, the user cannot store a flight they found interesting to continue the reservation process in a following session.

In addition, the *Payment* activity, as noted in Step 5 of the previous section, is *Suspendable* for registered users only. An unregistered user has no chance to purchase the ticket in a following usage session of the application. One could argue that this is an implementation issue. One could also argue that this is a matter of security policy. In either case, the designer could document the tradeoffs regarding design decisions such as *Suspendability* if the rationale were made explicit in the model.

SC2: The Identification Activity

The second important consideration is related to the *Identification* Activity. As shown by the Organization model in Figure 4, the *Identification* activity is required for the successful completion of the "*Complete Flight Reservation*" process. In this case, the shortcomings lie in the way the user is forced to access and execute this activity and what its execution causes.The user can start the *Identification* Activity with the user call *Login* (following a link provided by the application) only when executing the *Define and Search for Flights* or the *Choose Flight & Class Among Available* activities. Moreover, as described by the transition line of the Execution model in Figure 5, executing the *Identification* activity causes a loss of the state of the transaction (implementation of the process) and forces the user to restart the process.This also happens when the user reaches the confirmation step (Step 5) of the reservation process and, because the user did not do it before, the activity fails in requesting the user to identify themselves.

SC3: The Insert Passenger's Information & Choose On-board Options

Another aspect worthy of attention is the *Insert Passenger's Information & Choose On-board Options* Activity. As shown by the Execution model in Figure 5 and described in the previous section (Step 3), this Activity is required in order to carry out the reservation process. The reverse modeling process exposed the fact that this activity is executed before the user knows the fare of the selected flight (Step 4), and each time the user conducts a new search or selection of flights. It rapidly becomes very frustrating to the user to have to repeatedly input the same information. Iteratively searching for flights by changing dates and itineraries looking for the best fare available is a very common activity in any airline flight reservation application. Rather than acting as it does now, the Alitalia system should instead request the Passenger's Information only one time, and then display the fare of the chosen flight during the entire session of usage. The fact that the system loses the information entered by the user when executing this activity is also modeled by the lack of the *Durability* property in the Organization model in Figure 4.

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Table 1. Failure table for Activity "Define and Search for Flights"

Cause of Failure	Application Behavior	Possible Actions	
		Action	Triggered by
A mandatory field is not inserted.	The application informs the user to fill the field.	Retry Abort	User User
The user specifies the same date for the departure and return flights in an international itinerary.	The application informs the user that return date trip cannot be the same of the departure.	Retry Abort	User User
The user asks for a flight with the same departure and destination.	The system informs the user of the error.	Retry Abort	User User

Table 2. Failure table for Activity "View Flight Fare Without Taxes and Confirm Request of Flight Reservation"

Cause of Failure	Application Behavior	Possible Actions	
		Action	Triggered by
The user does not choose any suggested flight.	The application informs the user that he must choose one of the suggested flights.	Retry Abort	User User
The system cannot find a flight in the space of ±3 days since the specified day.	The system informs the user that any flight cannot be found and invites the user to retry.	Retry Abort	User User
The flight is not available for a number of reasons	The application redirects the user to complete search page.	Abort	System

SC4: Visualizing Total Cost

The fourth issue regards the visualization of the total cost, including taxes, of the chosen flight. Users can easily find themselves guessing between available flights shown by the system in Step 2, but needing to reach Step 4 to see the flight fare. Moreover, the price shown in this Step, as the names of the Activity suggests, does not include taxes, and only after confirming the reservation request will the user finally know the total ticket price.

SC5: Restarting the Process

Last but not least is the consideration about how much easier it is for the user to start over with another search for flights looking for a better fare. Since this is a common goal for most of the users, the application should offer the opportunity to start a new search

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

at nearly every point of the reservation process. Instead, as the Execution model in Figure 5 shows, the application explicitly provides a way to start a new search only at Step 2, which is prior to the user knowing the price of a chosen flight. The user can attempt to get around this limitation by using the "Back" button in the browser, but this invites problems related to session expiration.

Summary

This chapter presented an example of design recovery of Web application transactions. The design recovery procedure relies on UWAT+, which is a conceptual model that is based on extensions to the transaction design portion of the UWA framework. The prescriptive recovery procedure is composed of three steps, which can be accomplished by a human subject-matter expert. A commercial airline's flight reservation system was used to illustrate the procedure.To use the procedure, there is a one-time learning curve required for the engineer to become familiar with the UWA design framework and the UWAT+ refinements. However, once this is done, the procedure is relatively easy to understand and systematic to apply. The designer is provided with a sequence of clear steps to be carried out and a set of well-defined concepts to refer to, represented by means of the well-known notation of UML diagrams. Applying the reverse modeling technique allows the analyst to draw from the application and effectively represent with the models a lot of information perceived by the user and worthy of attention from their point of view.The design recovery procedure provides the analyst/designer with a tool (broadly intended and not specifically a software tool) that is able to represent most of the aspects related to the user execution of a process to carry out their goals. UWAT+ relies on two models, the Organization model and Execution models, that, taken together, are suitable for describing at a conceptual level the design of Web application transactions. These are strong bases of discussion and comparison of ideas and strategies that the Web application realizes.

Future Work

One area of future work we foresee is to develop tool support for the design recovery procedure. Such tool support would greatly improve the likelihood of adoption by easing the reverse modeling task. It would also make the analysis phase faster and more thorough than the current manual approach. Supporting tools could range from commercial UML diagramming editors, provided with UWAT+ Organization and Execution model profiles (plug-in), to semi-automatic tailored tools able to analyze and model the Activities of an identified transaction.Another area of future work is to use the recovered design as a guide to reengineering the Web site's transactions. The following three steps outline a possible reengineering technique for Web application transactions:

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

1. Perform the transaction design recovery of the Web site using the procedure described in this chapter;
2. Analyze the recovered "as-is" UWAT+ transaction design model and evaluate it according to quality attributes such as usability and fulfillment of business requirements; and
3. Develop a new version of the transaction via restructuring of the as-is model, resulting in a candidate "to-be" model that better meets the user's expectations and improves the user's experiences using the Web site.

Evidence-based techniques such as empirical studies could be used to verify that the resultant Web site is "better" in some quantifiable way than the original. The difficulty is in quantifying "better," both for the designer and the developer. For the designer, one measure might be shorter time-to-market for a complex Web application, while still retaining and even improving functionality and lower subsequent maintenance costs. For the user, the measure is likely to remain usability—something that is notoriously difficult to measure, but ultimately the most important attribute of all for any application.

Acknowledgements

Tauhida Parveen contributed to the development of an early draft of this chapter.

Endnotes

- Contact author Scott Tilley for correspondence.

References

Alitalia (2003). Available online at *www.alitalia.it*

Bellows, J. (2000). Activity diagrams and operation architecture. *CBD-HQ White paper*. Available online at *www.cbd-hq.com*

Booch, G., Rumbaugh, J., & Jacobson, I. (1998). *The unified modeling language user guide*, (Rational Corporation Software). Reading, MA: Addison-Wesley.

Chikofsy, E. & Cross, J. (1990). Reverse engineering and design recovery: A taxonomy. *IEEE Software*, *7*(1), 13-17.

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Distante, D. (2004). *Reengineering legacy applications and web transactions: An extended version of the UWA transaction design model.* Unpublished Doctoral Dissertation, University of Lecce, Italy.

Distante, D., Parveen, T. & Tilley, S. (2004, June 24-26). Towards a technique for reverse engineering web transactions from a user's perspective. In *Proceedings of the 12th International Workshop on Program Comprehension*, IWPC 2004, Bari, Italy, (pp. 142-150). Los Alamitos, CA: IEEE Computer Society Press.

Müller, H., Jahnke, J., Smith, D., Storey, M.-A., Tilley, S., & Wong, K. (2003). Reverse engineering: A roadmap. In A. Finkelstein (Ed.), *The future of software engineering* (pp. 47-60). New York: ACM Press.

Object Management Group (OMG) (2003). *Unified language modeling specification, version 1.5.* Available online at *www.omg.org.*

Rossi, G., Schmid, H. & Lyardet, F. (2003). Engineering business processes in web applications: modeling and navigation issues. *Proceedings of the 3rd International Workshop on Web Oriented Software Technology (IWWOST 2003),* Oviedo, Spain.

Schmid, H. & Rossi, G. (2004). Modeling and designing processes in e-commerce applications. *IEEE Internet Computing,* January/February.

Tilley, S. (2000). The canonical activities of reverse engineering. *Annals of software engineering, 9,* (pp. 249-271). Dordrecht, The Netherlands: Baltzer Scientific / Kluwer Academic.

UWA (Ubiquitous Web Applications) Project (2001a). *Deliverable D3: Requirements investigation for Bank121 pilot application.* Available online at *www.uwaproject.org*

UWA (Ubiquitous Web Applications) Project (2001b). *Deliverable D6: Requirements elicitation: model, notation and tool architecture.* Available online at *www.uwaproject.org*

UWA (Ubiquitous Web Applications) Project (2001c) *Deliverable D7: hypermedia and operation design: Model and tool architecture.* Available online at *www.uwaproject.org*

UWA (Ubiquitous Web Applications) Project (2001d). *Deliverable D8: Transaction design.* Available online at *www.uwaproject.org*

UWA (Ubiquitous Web Applications) Project (2001e). *Deliverable D9: Customization design model, notation and tool architecture.* Available online at *www.uwaproject.org*

UWA (Ubiquitous Web Applications) Project (2001f). *The UWA approach to modeling ubiquitous Web application.* Available online at *www.uwaproject.org*

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Chapter II

Using a Graph Transformation System to Improve the Quality Characteristics of UML-RT Specifications

Lars Grunske, University of Potsdam, Germany

Abstract

This chapter presents the concept of graph-based architecture evolution and how this concept can be applied to improve the quality characteristics of a software system. For this purpose, the UML-RT used as an architectural specification language is mapped to a hypergraph-based data structure. Thus, transformation operators can be specified as hypergraph transformation rules and applied automatically.

Introduction

Over the past few years, software intensive technical or embedded systems have increasingly been implemented in software components (Douglas 1999; Gomaa 2000; Liggesmeyer 2000). These software components have to fulfill requirements relating to quality characteristics or nonfunctional properties (NFPs), such as safety, availability, reliability, and temporal correctness. If a system does not fulfill these requirements, the

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.