Web Site Evolution via Transaction Reengineering

Scott Tilley
Dept. of Computer Sciences
Florida Institute of Technology
stilley@cs.fit.edu

Damiano Distante
Dept. of Innovation Engineering
University of Lecce, Italy
damiano.distante@unile.it

Shihong Huang
Dept. of Computer Science
Florida Atlantic University
shihong@cse.fau.edu

Abstract

In 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, partially influenced by the constraints implied by the business model underlying the Web application. Unfortunately, many Web sites are constructed with the transaction design hidden in the overall system implementation. The result is a system with unpredictable workflow, which can make evolution difficult. This paper presents a technique for Web site evolution via transaction reengineering. The reengineering process consists of the recovery of the "as-is" design model of a Web application transaction, an analysis of the result to determine desirable restructuring options, and a redesign of the transaction model based on this analysis. The reengineering process relies on formalism that is a user-centered extension of the Transaction Design Model of the Ubiquitous Web Applications (UWA) framework. The goal of the reengineering process is to emerge with a transaction design that better reflects the user experience and also facilitates disciplined evolution of the Web-based application. An example from the travel industry is used to illustrate the process.

Keywords: conceptual modeling, design recovery, reengineering, transactions, Web site evolution

1. Introduction

The Web provides a distributed information system infrastructure that can be used 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. Business processes are realized by means of transactions, which in this context can be interpreted as high-level workflows corresponding to user tasks (e.g., purchasing an airplane ticket). The manner in which the activities can be executed is a consequence of the transaction design and the constraints imposed on it by the underlying business process model(s).

As with other kinds of software systems, transaction-oriented Web sites are required to evolve over time in response to changing circumstances. For example, the addition of new flight options to an existing route, or the introduction of a new payment method. Unfortunately, many Web sites are constructed without proper attention to transaction design. Web transactions need to be designed (in the software engineering sense) because of their complexity. Treating transaction design as a by-product of the development process results in applications prone to erroneous behaviors and poor usability. For example, it is quite common to incorrectly treat a transaction as a sequence of navigational steps through pages of the Web application [7][8]. 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 paper presents a technique for Web site evolution via transaction reengineering. The prescriptive reengineering process consists of three steps: (1) the recovery of the "as-is" design model of a Web application transaction; (2) an analysis of the result to determine desirable restructuring options; and (3) a redesign of the transaction model, introducing the restructuring options from the recovered "as-is" model, resulting in the "to-be" design model. The reengineering process relies on formalism that is a user-centered extension of the Transaction Design Model of the Ubiquitous Web Applications (UWA) framework [17]. The recovery step makes the transaction design explicit, which in turn enables engineers to make informed decisions about possible changes to the application. The result is a transaction design that better reflects the user experience and also facilitates disciplined evolution of the Web-based application.

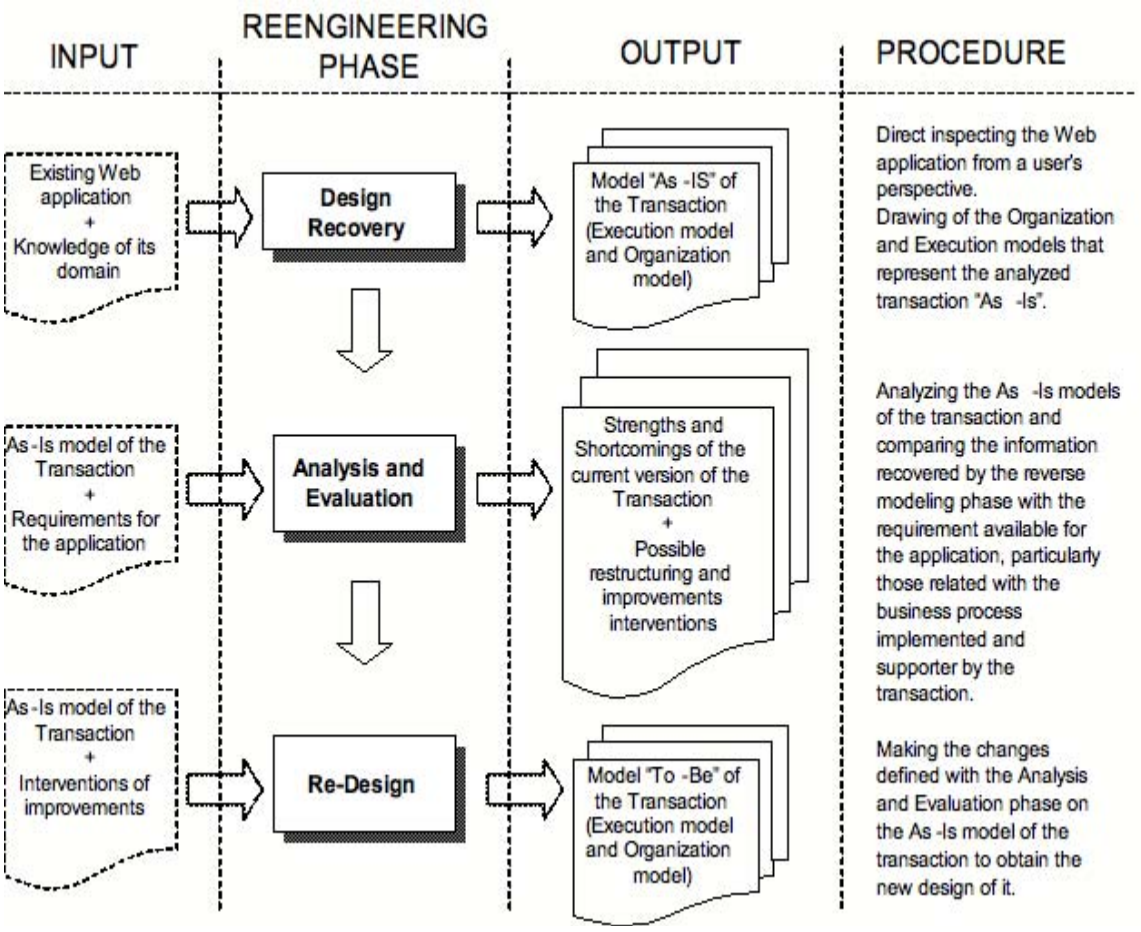

Figure 1: An overview of the transaction reengineering process

The next section of the paper outlines the transaction reengineering process. Section 3 highlights salient aspects of the design recovery procedure of a commercial airline's Web site. Proposed restructurings of the recovered transaction design models are detailed in Section 4. Finally, Section 5 summarizes the paper and outlines possible avenues for future work in the area of Web application transaction reengineering.

2. The Transaction Reengineering Process

Reengineering is one mechanism for achieving disciplined software evolution. The reengineering process for Web-based business-oriented applications is similar to that of traditional distributed information systems. However, the singular features of the Web, such as its heterogeneous nature, its implicit design model (particularly for transactions), and its inherent complexity (and often correspondingly low quality) suggest that the reengineering process be based on an established decision-guiding framework. For transaction reengineering in particular, the UWA framework provides a suitable starting point.

This section outlines selected aspects of the transaction reengineering process, the entirety of which is depicted in Figure 1. The formalism underlying the process is a revised version of the UWA Transaction Design Model [4], which is the portion of the UWA framework that focuses specifically on the design of Web application transactions. The revisions to the original version of the model have been introduced in order to focus more on the perspective of the end-user of the Web application.

The design recovery procedure that relies on this model is also summarized in this section. Extensive details of this portion of the reengineering process are provided in [5]. A detailed illustration of the results of using the design recovery process is provided in [11].

2.1 The Transaction Design Model

The UWA design framework provides a complete design methodology for ubiquitous Web applications that are multi-channel, multi-user, and context-aware. The UWA design framework organizes the process of designing a Web application into four main activities [12]: (1) requirements elicitation [13]; (2) hypermedia and operation design [14]; (3) transaction design [15]; and (4) customization design [16].

Using the UWA methodology, the transaction design process produces two conceptual models: the Organization Model and the Execution Model. The Organization Model describes a transaction from a static point of view. It uses a particular UML class diagram [6] in which the Activities involved in the transaction are represented by class stereotypes, which are arranged to form a tree. The Activity represented by the root of the tree corresponds to the entire transaction; component activities and sub-Activities are intermediate nodes and leaves of the tree that represent sub-transactions and elementary activities, respectively.

The Execution Model of a transaction defines the possible execution flow among its component activities and sub-activities. It is a customized version of the UML Activity Diagram [9]. The sequence of activities is described by UML Finite State Machines, in which activities and sub-activities are represented by states (ovals), and execution flow between them is represented by state transition (arcs).

The formalism used in the reengineering process is an extended and refined version of this UWA Transaction Design Model. Changes to the original model include simplifications and extensions related to the definition of Activity (which a Web transaction is composed of), and to several aspects of the Organization and Execution Models.

Changes to the Definition of Activity

One of the revisions to the UWA Transaction Design Model concerns the definition of Activity and related aspects. Activities are defined as units of work the user has to execute in order to fulfill a goal. Thus, only user Activities are considered and modeled; system-related Activities are neglected in this context. The OperationSet in the original model that is associated with an Activity is no longer considered, since it is primarily related to data-level details and transaction implementation issues.

In addition, 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

Significant changes have been made to the UWA's original 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. The distinction between hierarchical and semantic relations permits the designer to reason about transactions in a manner not possible with the unadorned UWA model.

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 consist of the *Visible*, *Compensates*, and *Can Use*.

Changes to the Execution Model

Several changes have been introduced into the UWA's Execution Model to focus attention on design issues that have direct user impact. For example, the Commit and Rollback pseudo states have been removed; the inclusion of an Activity is now 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.

In the revised version of the Execution Model, 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 of A (Activity invoked by the user), C (condition(s) required for activity execution), R (result of Activity execution), or S (state associated with system due to Activity). A list of causes of Activity failure and possible actions the user or the system can take is maintained, to explain why an activity fails and how the user or the system can react

Compensation Activities (Activities that rewind the results of others) needed to allow a transition between two activities are implicit and controlled by the system. It is suggested that UML swimlanes diagrams [10] be adopted when it's useful to describe how two or more user types of the application collaborate in the execution and completion of a transaction. No transition of the

Execution Model can be associated to 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.

The above changes to the Execution Model provide better visibility into the dynamic execution paths that the user will experience while completing a specific transaction. By making such paths explicit, improvements in the transaction design can be more easily accomplished.

2.2 Design Recovery

Given an existing Web site, the goal is to populate an instance of the revised UWA model as described above with data obtained by direct inspection and analysis of the site's content and structure as experienced by a real user. The resultant model can then be used to guide reengineering 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 formalized transaction. A human subject-matter expert can finish this procedure without any tool support. However, the use of automated reverse engineering technology [3] may improve the efficacy of the process.

Formalization of the Transactions

In the first step of the design recovery procedure, the user-types of the application and their main goals/tasks are formalized. A transaction is associated with each operative goal/task. At the end of this step, the list of transactions implemented by the application is obtained.

Creation of the Execution Model

For each of the transactions found in the first step of the design recovery procedure, the Execution Model is created by first performing a high-level analysis of the transaction in order to gain a basic understanding of its Activities and execution flow. The transaction is then characterized as "simple" (linear), or "composite" (with two or more alternative execution paths).

A first draft of the Execution Model is created for each simple transaction identified by executing it in a straightforward manner. 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 failures states and possible actions the user can undertake. This information is used to refine the Execution Model. 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 created for a transaction, the Organization Model can be constructed, which will model the transaction from a static point of view. In the case of a simple transaction, the Activity set is determined by all the Activities and sub-Activities encountered in the single flow of execution allowed 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 has been found for the composite transaction.

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 and discover the value to be assigned to each of them through direct inspection using the Web-based application.

2.3 Analysis and Evaluation

The result of the design recovery procedure is a model of the Web application transactions using the revised version of the UWA Transaction Design Model. The next step of the reengineering process is to perform a user-oriented analysis and evaluation of the recovered "as-is" design. This analysis can be based on characteristics of the design such as its functional effectiveness (how well it meets current business needs) and technical efficiency (a reflection of the quality of the design using software engineering measures).

This step of the reengineering process aims to define a set of possible restructuring options for the current design and implementation of the considered transactions addressing the shortcomings highlighted by the analysis. These options are then considered as input to the redesign step.

2.4 Redesign

The final step of the reengineering process is the redesign of the "as-is" transaction model, resulting in the "to-be" version of the transaction design. Introducing the changes defined during the analysis phase into the recovered "as-is" design model produces the new design.

The recovered models are thus used as basis for creating the new design.

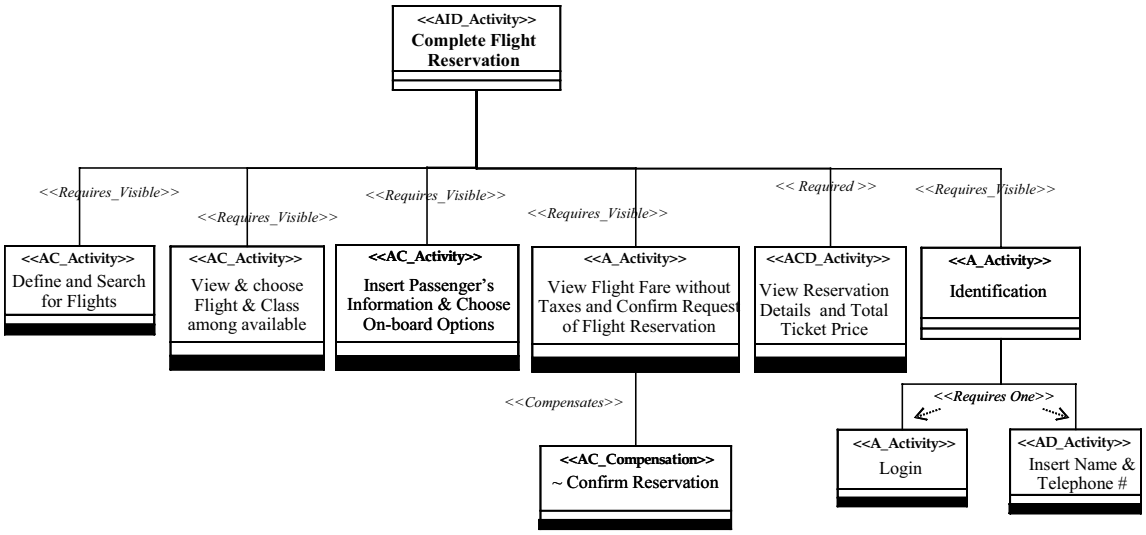

Figure 2: The "as-is" Organization Model of the "Complete Flight Reservation" Web transaction

3. The Recovered Transaction Design

The previous section outlined the transaction reengineering process. The process was applied to the *Complete Flight Reservation* transaction of the Alitalia.it Web site [1]. The *Complete Flight Reservation* is one of the two procedures offered by the Alitalia.it Web site for booking a flight. The second way, called *Fast Flight Reservation*, is a quicker way that assumes default choices for optional parts of the flight reservation. Complete details of the application of the design recovery process and the recovered transaction design for this Web site are provided in [11]. This section summarizes the shortcomings (SC) of the current transaction design based on the recovered model.

The recovered "as is" Organization and Execution Models of the *Complete Flight Reservation* Web transaction from the Alitalia.it airlines Web site are shown in Figure 2 and Figure 3, respectively. Together, the two models represent the current conceptual design of the Web transaction as perceived by the user.

As described by the Organization Model in Figure 2, in order to book a round-trip flight by means of the *Complete Flight Reservation* procedure, the user is required to execute six elementary Activities: identify themselves to the system (*Identification*), search for a flight (*Define and Search for Flights*), choose a flight among available (*View & choose Flight & Class among available*), specify passenger information and on board services (*Insert Passengers' Information & Choose On-board Options)*, confirm the chosen flight *(View Flight Fare without Taxes and Confirm Request of Flight Reservation*), and view the reservation details with total price (*View Reservation Details and Total Ticket Price)*. All six Activities are required as shown by the *Requires_Visible* type hierarchical relation that connects them to the parent activity *Complete Flight Reservation,* which represents the entire Web transaction. The Organization Model also illustrates that none of the six activities is Suspendable, thus the reservation procedure must be completed in one session, or it must be aborted.

The Execution Model of Figure 3 describes the execution rules that regulate the execution of the set of Activities involved in the reservation transaction. In particular, the model reveals several shortcomings in the usability of the Web transaction of reservation. These shortcomings (SC) are summarized below.

SC1: Executing the Identification Activity

The model exhibits that the execution of the *Identification* Activity (logging into the system or providing few personal information) causes the reservation transaction to be restarted, no matter what its state is. This is not a desirable situation.

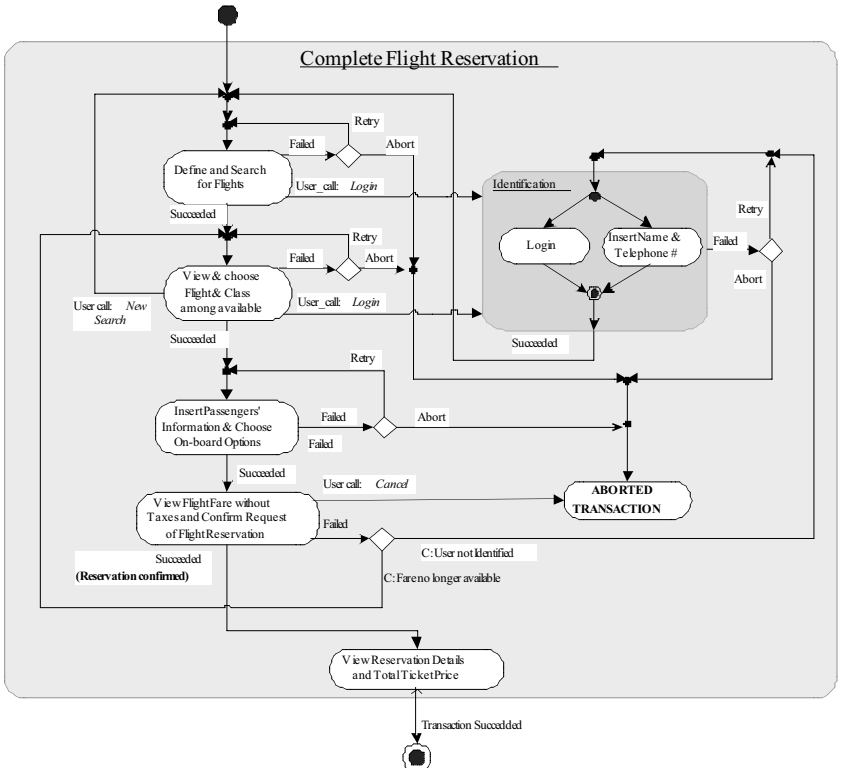

Figure 3: The "as-is" Execution Model of the "Complete Flight Reservation" Web transaction

SC2: Specifying Passenger Information and Options

The position of the *Insert Passengers' Information & Choose On-board Options* Activity in the execution flow of the entire transaction is a concern. This Activity must be executed every time the user tries to look for a better flight by repeating the *Define and Search for Flights* Activity, before knowing the fare associated with a flight by means of the *View Flight Fare without Taxes and Confirm Request of Flight Reservation* Activity.

SC3: Visualizing the Total Cost of a Flight

In the current design of the *Complete Flight Reservation* Web transaction, the visualization of the total price for the selected flight is incomplete. As shown by Execution Model, the user has to execute four Activities before they can know the total price of the selected flight. Moreover, this information is needed each time the user searches for new travel solutions by changing some of the search parameters.

SC4: Restarting the Reservation Web Transaction

The recovered model illustrates the difficulty of restarting the reservation process by repeating the search for flight. The application explicitly provides a way to start a new search (i.e., to re-execute the *Define and Search for Flights* Activity) only when executing the *View & choose Flight & Class among available* and not later. In particular, there is no other way to complete this activity, other than using the "Back" button in the browser. Relegating navigation management to the "Back" button of the browser can have dangerous implications, in terms of the incoherence between the states of the Web transaction as stored by the system (server side) and the states as perceived by the user (client side) [2].

4. The Restructured Transaction Model

The previous section identified several shortcomings in the recovered transaction design. These shortcomings suggest possible restructuring options to lead to redesigned transaction model for the Web application.

The restructured Organization and Execution Models for the *Complete Flight Reservation* Web transaction are shown in Figure 4 and Figure 5, respectively. Taken together, they describe a possible new "to-be" design for the *Complete Flight Reservation* transaction. The "to-be" versions of the Web transaction design models (the Organization Model and the

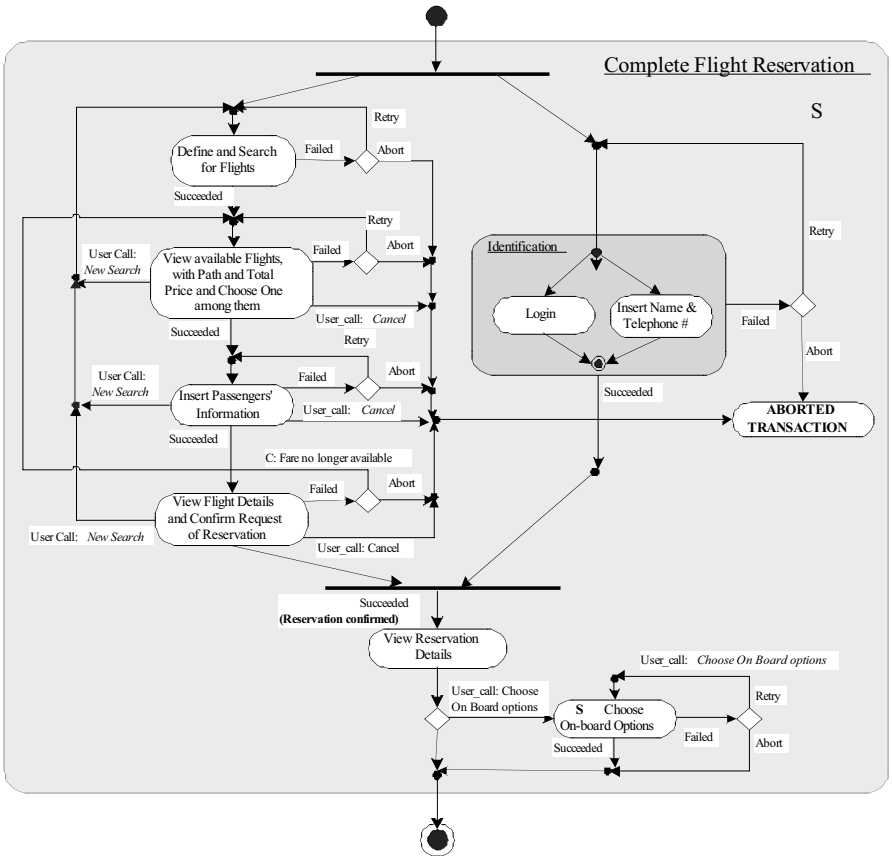

Figure 4: The "to-be" Execution Model of the "Complete Flight Reservation" Web transaction

Execution Model) derive from the "as-is" version recovered during the reverse modeling phase as shown in Figure 2 and Figure 3. The restructurings introduced in the "to-be" models address the shortcomings noticed in the recovered transaction design.

4.1 The Restructured Organization Model

The restructured Organization Model is shown in Figure 4. The first change is to replace the *Choose Flight & Class Among Available* Activity with a new Activity named *View available Flights, with Path, Total Ticket Price, and Choose One among them*. This new Activity is intended to enable the user to have the information about the available flights fitting their needs all at once. This change means that the total price of each flight will be shown immediately in the list of available flights resulting from a search (the price shown will be the total price including taxes). This change also considerably lessens the user's "trial and error" practice of looking for best fare flights. This change is intended to overcome shortcomings SC3 and SC4 discussed above.

To completely address shortcoming SC3, a slight change is needed to the implementation of the *View Flight Fare without Taxes and Confirm Request of Flight Reservation* Activity. In consideration of the previous change, the new version of this Activity will show all the details of the ongoing flight reservation, including the total ticket price with taxes, before the user's confirmation of the reservation request. A new Activity called *View Flight Details and Confirm Request of Reservation* takes its place.

The Activity *Insert Passengers' Information & Choose On-board Options* is split into two new Activities: *Insert Passengers' Information* and *Choose On-board Options.* To address the problem reported in shortcoming SC2, the former of these new Activities should be *Required* and *Durable,* while the latter can be *Optional* and *Suspendable*.

4.2 The Restructured Execution Model

The restructured version of the Execution Model shown in Figure 5 reports most of the changes

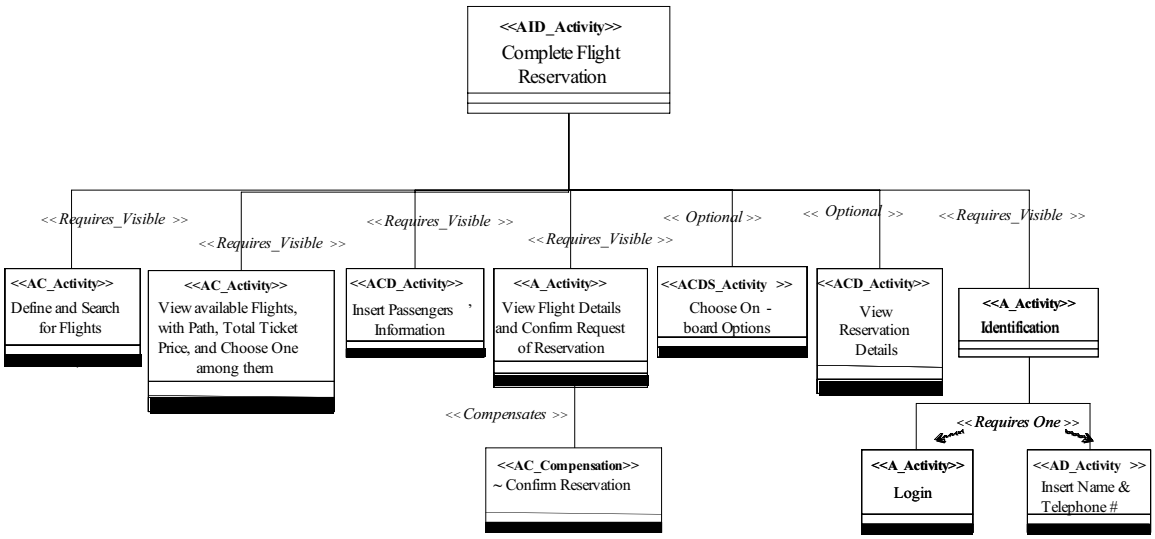

Figure 5: The "to-be" Organization Model of the "Complete Flight Reservation" Web transaction

introduced in the Organization Model as well as other major changes dealing with the execution flow of the process. The most important of these changes and the motivations on which they are based are highlighted in the discussion below.

As described by the model, in the proposed new version of the *Complete Flight Reservation* Web transaction the *Identification* Activity will be executable at any time and concurrent to any Activity of the *Complete Flight Reservation* Web transaction. This is represented in the model enclosing the *Identification* Activity and all the other Activities required to accomplish the flight reservation between synchronization bars. The synchronization bars also indicate that executing the *Identification* Activity will not cause the loss of state of the transaction, forcing the user to start it over again. The *Identification* can even be delayed and accomplished later in the Activity by the user just before a reservation request is confirmed. This change is intended to address the shortcoming SC1.

Thanks to the changes in the Activities composing the restructured Web transaction, the user will now be able to search and learn details about available flights, including their total cost, in a single step, just by executing the *Define and Search for Flights* Activity. As shown by the execution flow depicted in the proposed restructured version of the Execution Model, the user is required to provide the passengers' information only when they have decided on a particular flight that fits their request and satisfies their budget.

As shown by the available transition lines, the Activity of *Define and Search for Flights* can now be repeatedly executed from any point of the *"Complete Flight Reservation*" transaction until the reservation is confirmed. The application is supposed to explicitly provide the user with the option to invoke the User_call *New search*. This solution addresses the "trial and error" scenario described above.

The Activity of *Choose On-board Options* is now separated from the Activity of *Insert Passengers' Information,* delayed in the transaction after the completion of the reservation and having the *Suspendable* property. Its execution is optional; it will be executed only if the user chooses it via the action User_call *Choose on board options*.

4.3 Discussion

We are aware that some of the criticisms brought against the recovered transaction design of the Alitalia.it reservation transaction discussed in this section and identified as shortcomings could derive from conscious choices made by the application's designers. We do not claim that the proposed redesigned transaction models for the "to-be" version of the application represent the best version of the transaction we analyzed. The restructuring interventions we proposed, in fact, are intended to offer only a possible solution to the weakness that we, as users, observed in the structure and the execution flow of the transaction we examined. We stress that defining what is "better" from the user point

of view is beyond the objectives of this work. This concept is tightly related to the requirements for the particular application and business process model considered.

The objective in the illustrative example was to show that the reengineering technique (and the models it adopts) is able to capture and describe the user's perception of the processes implemented by a Web application. The technique has been verified to enable the analyst to model the processes from the user perspective and, subsequently, to discuss and evaluate possible restructuring interventions that can improve the experience of using the services the application offers.

Thanks to the revised version of the UWA Organization and Execution Models it adopts, the reengineering process is able to represent the transactions implemented in a Web application with an high level of abstraction and to model them according to the way they are perceived by the user. The usefulness of the reengineering technique can be emphasized in the fact that it enables the analyst or the designer to:

Draw information that describes how the reverse-engineered Web transaction is experienced by the user, and how the application behaves when executing it;

Represent the recovered information in a clear and sharable way by drawing the models the technique adopts;

Evaluate the current design of the transaction from the user perspective, and identify undesirable shortcomings; and

Define possible solutions to the revealed shortcomings to produce a restructured "to-be" version of the transaction design that better meets user expectations and aids Web site evolution.

5. Summary

An approach to transaction design that concentrates on system- and data- centered issues usually considered by the database theory (e.g., data consistence, concurrency control, and state management) results in shortfalls in usability (intuitiveness, simplicity, effectiveness) of the services offered by the application by means of the transactions. In contrast, designing Web transactions with a user-centered approach can considerably improve the user's experience in using the application. A Web transaction can be reengineered to meet these objectives using the process described in this paper. The restructuring can be done without violating the business rules implied by the business process that the Web application is intended to implement.

To illustrate the transaction reengineering process, restructuring options were proposed for the recovered transaction design of Alitalia.it's travel reservation system. The design recovery procedure relies on 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 with or without tool support.

One area of future work that we see as of paramount importance is the development of a testing methodology to assess the efficacy of the restructured transaction design. 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 user. 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.

We intend to explore the possibility of exploiting established techniques from Web testing and usability research as a means of performing assessment of the reengineering process. The results of these studies will be published in a suitable public forum.

Acknowledgements

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

References

[1] Alitalia. Online at www.alitalia.it.

[2] Baresi, L.; Denaro, G.; Mainetti, L.; and Paolini, P. "Assertions to Better Specify the Amazon Bug." *Proceedings of the ACM International Conference on Software Engineering and Knowledge Engineering* (SEKE 2002: Ischia, Italy, 2002).

[3] Chikofsy, E.; and Cross, J. "Reverse Engineering and Design Recovery: A Taxonomy." *IEEE Software* 7(1):13-17, January 1990.

[4] Distante, D. "Reengineering Legacy Applications and Web Transactions: An extended version of the UWA Transaction Design Model." Ph.D. Dissertation, University of Lecce, Italy. June 2004.

[5] Distante, D.; Parveen, T.; and Tilley, S. "Towards a Technique for Reverse Engineering Web Transactions from a User's Perspective." In *Proceedings of the 12th*

IEEE International Workshop on Program Comprehension (IWPC 2004: June 24-26, 2004; Bari, Italy). Los Alamitos, CA: IEEE CS Press, 2004.

[6] G. Booch, J. Rumbugh, I. Jacobson, "The Unified Modeling Language User Guide", (Rational Corporation Software), Addison-Wesley.

[7] G. Rossi, H. Schmid, F. Lyardet, "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, 2003).

[8] H. Schmid, G. Rossi, "Modeling and Designing Processes in E-Commerce Applications." *IEEE Internet Computing*, January/February 2004.

[9] J. Bellows, "Activity Diagrams and Operation Architecture", CBD-HQ White paper, www.cbd-hq.com, Jan. 2000.

[10] Object Management Group (OMG), "Unified Language Modeling Specification, version 1.5", www.omg.org, Mar. 2003.

[11] Tilley, S.; Distante, D.; and Huang, S. "Design Recovery of Web Application Transactions." Submitted to *The 11th IEEE Working Conference on Reverse Engineering* (WCRE 2004: Nov. 9-12, Delft, The Netherlands). June 2004.

[12] UWA (Ubiquitous Web Applications) Project, "Deliverable D3 Requirements Investigation for Bank121 pilot application", http://www.uwaproject.org, 2001.

[13] UWA (Ubiquitous Web Applications) Project, "Deliverable D6: Requirements Elicitation: Model, Notation and Tool Architecture", www.uwaproject.org, 2001.

[14] UWA (Ubiquitous Web Applications) Project, "Deliverable D7: Hypermedia and Operation design: model and tool architecture", www.uwaproject.org, 2001.

[15] UWA (Ubiquitous Web Applications) Project, "Deliverable D8: Transaction design", www.uwaproject.org, 2001.

[16] UWA (Ubiquitous Web Applications) Project, "Deliverable D9: Customization Design Model, Notation and Tool Architecture", www.uwaproject.org, 2001.

[17] UWA (Ubiquitous Web Applications) Project, "The UWA approach to modeling Ubiquitous Web Application", www.uwaproject.org, 2001.