TRANSACTION SUPPORT FOR INTERACTIVE WEB APPLICATIONS

David Paul, Mark Wallis, Frans Henskens and Michael Hannaford
School of Electrical Engineering and Computer Science, University of Newcastle, NSW, Australia

Keywords: Web Services, transactions, Super Browser, ACID.

Abstract: In the Web Services environment, traditional ACID transactions are not always sufficient to support the activities that businesses would like to process. While Web Services transactions standards do exist, it is still difficult for an end-user to combine services from loosely-coupled providers into a single action to be performed. We describe the use of a "Super Browser" that enables users to more easily find and combine Web applications into a single activity that they can view and manipulate throughout its life-cycle.

1 INTRODUCTION

In the Web Services environment, traditional ACID transactions are not always sufficient to support the activities that businesses would like to process. Transactions that involve multiple service providers can run for long periods of time. This can result in negative side-effects when combined with traditional transaction-based concurrency control mechanisms. While Web Services transactions standards do exist, it is still difficult for an end-user to combine services from loosely-coupled providers so that they are used as a single action. Additionally, these existing standards are targeted primarily towards predefined transactions that are static and involve little or no direct human interaction.

This paper describes the use of a "Super Browser" that enables users to more easily find and combine Web applications into a single activity that they can view and manipulate throughout the activity's life-cycle. Initial support will allow the end-user to generate ad hoc transactions that contain multiple interactive Web application sessions. The design is extendable and supports the orchestration of both interactive and non-interactive Web Services.

The paper is organised as follows: Firstly a brief review of transactions is given, explaining the ACID properties and their typical reductions. A Super Browser design concept is then presented to support ad hoc transaction creation and orchestration, with a primary focus on integrating non-Web Service-based Web applications into a transactional structure. An outline is given for the support that a non-Web Service-enabled Web application must provide so it can participate in transactions managed by the Super Browser's transaction manager.

2 TRANSACTIONS

A transaction combines a group of independent actions into a single action with a set of predictable outcomes. Traditional transactions, where only a single system is involved, are well understood (Gray and Reuter, 1993). There is also a large body of theory (Garcia-Molina and Salem, 1987; Younas et al., 2000), behind techniques to support transactions in environments such as that of Web Services. A problem with these methods, however, is that the transactions are typically not dynamically configurable, especially with regard to human interaction.

2.1 Multidatabase Transactions

Traditionally, transactions are required to adhere to the ACID properties of Atomicity (ensuring that all actions in the transaction either complete successfully, or revert to a state where none of them were run), Consistency (ensuring that the system is not put into an illegal state), Isolation (letting concurrent transactions run as if they were the only transaction being processed), and Durability (ensuring that any completed transaction has its outcome recorded on a stable medium and cannot be undone, even by accidental events such as hardware or software failure). However, while transaction management in traditional systems typically offers an acceptable level of service, the same cannot be said for transactions achieved by combining services offered by multiple systems. Such multidatabase transactions often run for much longer periods of time than traditional transactions, so locking any data may block other transactions for an unacceptable length of time (Little, 2003). Because of this, the traditional ACID properties are typically reduced in strength, helping to ensure that the entire system maintains an acceptable level of service.

2.1.1 Reduction of Acid Properties

The main problem with Web Services transactions is that the cost to globally enforce the ACID properties traditionally needed for transactions is too great. Thus, there are many proposals to reduce the strength of some of the properties in these and similar systems (Garcia-Molina and Salem, 1987; Younas et al., 2000). Typically, consistency and durability are seen as required properties, whereas atomicity and isolation are the properties that are weakened.

Atomicity is intended to ensure that either a transaction completes successfully or the system is reverted to a state as if the transaction never ran. As the failure of a single transaction can affect the result of any other transaction that was processed while that transaction was running, atomicity is considered infeasible in the Web Services environment. Typically strict atomicity is replaced with a weaker version, most frequently semantic atomicity (Garcia-Molina, 1983).

For Web Services transactions, atomicity is also often further reduced (Little, 2003). Rather than requiring every service provider involved in a transaction to see the same result, it is sometimes useful to have some providers believe that the transaction has failed while others believe that it has succeeded. The main reason for this has to do with the definition of success. In Web Services transactions, multiple results may be seen as acceptable, so the failure of a single service provider may simply lead to a different acceptable state rather than to outright failure (Little, 2003).

The other property typically reduced in strength for Web Services transactions is isolation. While traditional optimistic concurrency control has been developed for Web Services transactions (Choi et al., 2005; Haller et al., 2005), and this completely supports isolation in the traditional sense, these techniques do nothing to solve the problem of the reduction of service levels that the strict support of isolation seems to demand. In fact, in order to keep acceptable levels of service, global isolation in Web Services transactions is typically completely ignored, instead relying on the local isolation of individual service providers to avoid most of the problems that a complete lack of isolation would cause. While this does improve performance of the overall system, ignoring isolation does cause some problems (Paul et al., 2007).

2.2 Interactive Web Applications and Web Services Transactions

As there are already three well-known standards for Web Services transactions (Ceponkus et al., 2002; Cox et al., 2004; Bunting et al., 2003), it may seem that there is little need for any further standards development. However, some processes require interactions unsupported by these standards. This is especially true when a user needs to interact with the transaction. The main problem is that Web Services are mainly seen to be for machine-to-machine interactions, when in many cases human interaction is either needed, or desired, to ensure correct results (den Haan, 2007).

While it is possible for individual Web Services to require or rely on user interaction, there is currently no easy way for a user to specify a transaction, other than writing a program specifically for that transaction. Currently, to combine Web Services offered by different providers, an end user must either write their own program (or process model) or use each service individually (eg through Web interfaces provided for each service). In the former case, the end user requires too much knowledge of Web Services to make this acceptable for people outside of the field. In the latter case, no transactional support is possible. The following section describes how a Super Browser supports abstraction over these difficulties to let end users easily and successfully combine multiple Web applications into a single transactional unit.

3 THE SUPER BROWSER

A Super Browser (Henskens, 2007) is a next-generation Web browser that in addition to standard Web browser functions, provides a consistent interface for applications to interact directly over the Internet. The Super Browser enables the user to find and combine services, run ad hoc transactions made up of these combined services, and manage the transactions currently being run by the Super Browser.

The remainder of this paper focuses on the creation of the Transaction Management facility within the Super Browser, and the establishment of ad hoc transactions.

3.1 Ad Hoc Transaction Creation and Tracking

The Super Browser provides a method for users to create a new ad hoc transaction. This ad hoc transaction later has particular Web Services associated with it manually by the user. The creation function requests that the user specify a descriptive name for the transaction. For example, the user may create a new transaction named the "Summer Holiday Booking" transaction that tracks the various Web Services involved in the organisation of holiday travel.

When the user creates a new ad hoc transaction, the Super Browser establishes the various data structures needed to track the progress of the transaction. The user then has access to a Transaction Viewer page that lists all the transactions that are currently pending within the Super Browser.

3.2 Identifying Transaction-enabled Web Applications

Typically, an interactive Web application does not allow a user to interact directly with any internal transactions that take place. For an interactive Web application to take part in a transaction managed by the Super Browser, an extension is provided to support three basic Web Service calls: *establishInteraction()*, *interactionStatus()*, and *rollback()*.

These three Web Service calls are exposed to the Super Browser using the existing WSDL (Christensen et al., 2001) and SOAP (Gudgin et al., 2007) standards.

If a Web application is extended to support these functions then it needs to notify the Super Browser that it is able to take part in a managed transaction. It does this by inserting a specific META tag into the head of the HTML provided by the Web application. The "LINK alternate" META tag described in the HTML 4.0 specification is used for this function. RSS feeds (RSS Advisory Board, 2007) currently use this same functionality for informing a standard Web browser that the Web application supports the RSS standard.

3.3 Adding a Service to an Existing Transaction

When the user requests to add a specific service to an existing transaction, the transaction manager in the Super Browser executes the *establishInteraction()* Web Service provided by the Web application. This function is passed the name of the specific service requested by the user. When the Web application receives this request, it establishes a Web session for that particular service, and passes back a unique Session ID and the URL which can begin this interaction to the Super Browser.

The Super Browser then takes this information and creates a new Web browser window. Within this window the browser automatically navigates the user to the provided URL and returns the Session ID to the Web application. This connects the user with the interactive Web application under the context of the transaction.

From here, the user can interact with the Web application as if it were not part of a transaction. When the user has finished using the service (e.g., when they have finished making their booking), the user or Web application closes the Web browser window. This window close action prompts the Super Browser to call the *interactionStatus()* function on the Web application, passing in the related Session ID. This function tells the transaction manager if the interactive session was successful, and whether it should mark that component of the transaction as complete.

At this stage, the user's interaction with the Web application is complete. Since the Web application is not truly transaction-aware, it has already committed the result of the user's interaction to its internal databases. For example, if the Web application in question was an Airline Booking service then at this stage the flight has been booked and the user has been charged.

3.4 Cancelling a Transaction

As the user continues to add various Web Services to a transaction, a point may be reached where the user is unable, or unwilling, to continue and wishes to either roll back part of the transaction or the transaction as a whole. This is made possible through use of the *rollback()* function. This *rollback()* function takes the Session ID described above as a parameter and reverses any internal commits that have already been performed. In the Airline example from above, this means that the rollback() function reverses the airline booking and refunds any payments back to the user, less any required compensation.

3.5 Commiting a Transaction

Finally, the Super Browser also allows the user to commit the transaction as a whole. Since each component Web Service is not transaction aware, this commit basically marks the transaction as completed and does not allow the user to add any additional Web Services to the transaction or roll back any of the components.

4 CONCLUSIONS

Many end-user activities are best implemented as transactions (Gray, 1981). Despite this, it is quite difficult for an end-user to create and manage Web Services transactions. We describe features of work in progress on a Super Browser that improves this situation. While the browser provides support for Web Services transactions standards, it also allows the ad hoc use of Web applications in a transaction-based context.

Allowing end-users to generate and manage their own transactions across multiple Web applications improves the user's capability to handle cases where a particular Web Service is unable to fulfil their request. Taking the Holiday booking scenario as an example, it has been shown that allowing the end-user to issue a roll back across a group of disconnected, heterogeneous Web Services provides an efficient way for the user to cancel services that have already been booked without having to manually interact with each Web application in turn.

The presented research is focused on enabling transaction support incorporating non-Web Service-based Web applications. The Super Browser concept, though, is extensible and allows for applications to be developed that can combine both server-to-server and user-interactive Web Services into a single transaction.

REFERENCES

Bunting, D., Chapman, M., Hurley, O., Little, M., Mischkinsky, J., Newcomer, E., Webber, J., and Swenson, K. (2003). *Web Services Composite Application Framework (WS-CAF) Ver 1.0*. Arjuna Technologies Ltd.

Ceponkus, A., Dalal, S., Fletcher, T., Furniss, P., Green, A., and Pope, B. (2002). *Business Transaction Protocol 1.0*. OASIS.

Choi, S., Jang, H., Kim, J., Kim, S. M., Song, J., and Lee, Y. (2005). Maintaining consistency under isolation relaxation of Web services transactions. In *Sixth International Conference on Web Information Systems Engineering (WISE'05)*, pages 245–257, NY, USA.

Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001). *Web Services Description Language (WSDL) 1.1*. Microsoft and IBM Research, W3C note March 2001 edition.

Cox, W., Cabrera, L. F., Copeland, G., Freund, T., Klein, J., Storey, T., and Thatte, S. (2004). *Web Services Transaction (WS-Transaction)*. BEA Systems Inc, IBM and Microsoft.

den Haan, J. (2007). SOA and human interaction. *The Enterprise Architect*.

Garcia-Molina, H. (1983). Using semantic knowledge for transaction processing in a distributed database. *ACM Transactions on Database Systems*, 8(2):186–213.

Garcia-Molina, H. and Salem, K. (1987). Sagas. In *ACM SIGMOD international conference on Management of data (SIGMOD '87)*, pages 249–259. ACM Press.

Gray, J. (1981). The transaction concept: Virtues and limitations. Technical report, Tandem Computers Incorporated.

Gray, J. and Reuter, A. (1993). *Transaction processing : concepts and techniques*. Morgan Kaufmann Publishers, San Mateo, California, USA.

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Nielsen, H. F., Karmarker, A., and Lafon, Y. (2007). *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. Microsoft, Sun Micosystems, IBM, Canon, Oracle and W3C, W3C recommendation April 2007 edition.

Haller, K., Schuldt, H., and Türker, C. (2005). Decentralized coordination of transactional processes in peer-to-peer environments. In *Conference on Information and knowledge management (CIKM '05)*, pages 28–35. ACM Press.

Henskens, F. A. (2007). Web service transaction management. In *International Conference on Software and Data Technologies (ICSOFT'07)*, volume PL/DPS/KE/MUSE, pages 112–119, Barcelona, Spain.

Little, M. (2003). Web services transactions: past, present and future. In *XML 2003*, Philadelphia, USA.

Paul, D., Henskens, F. A., and Hannaford, M. (2007). Isolation and web services transactions. In *Eighth International Conference on Parallel and Distributed Computing, Applications, and Technologies (PDCAT'07)*, pages 181–182, Adelaide, Australia.

RSS Advisory Board (2007). *RSS 2.0 Specification*, http://www.rssboard.org/rss-specification edition.

Younas, M., Eaglestone, B., and Holton, R. (2000). A review of multidatabase transactions on the Web: From the ACID to the SACReD. In *The 17th British National Conference On Databases (BNCOD 17)*, pages 140–152. Springer-Verlag.