White, P. and Grundy, J.C. Experiences Developing a Collaborative Travel Planning Application with .NET Web Services, In Proceedings of the 2003 International Conference on Web Services, Las Vegas, June 23-26 2003, CSREA Press.

Experiences Developing a Collaborative Travel Planning Application with .NET Web Services

Philip White[1] and John Grundy[1, 2]

[1]Department of Computer Science and [2]Department of Electrical and Electronic Engineering, University of Auckland, Private Bag 92019, Auckland, New Zealand
john-g@cs.auckland.ac.nz

Abstract

Web services have the potential to provide much more seamless, dynamic and open distributed applications than earlier technologies. We describe our experiences developing an integrated, collaborative travel planning application using .NET, C# and web services. This application provides a unified portal for customers and travel agents to plan, revise and book travel itineraries, with interaction with a wide range of travel provider systems (airlines, hotels, rental cars, trains etc). We describe the web services-based architecture of our prototype solution and discuss some of the key issues of using a web service-based approach for this application domain, outlining key areas for future research and development.

1. Introduction

Web services [18, 16, 10] provide an open, platform and language independent technology for building distributed systems. Such systems can be dynamic with the run-time discovery and integration of various services within an architecture. Many potential benefits of using a web service-based approach to building systems exist: a single integration technology is used to integrate all systems; systems can be dynamically discovered and integrated within an architecture; third-party remote services can be added or removed seamlessly; and multiple complementary services can be found and used to provide best overall system functionality, performance and reliability.

In previous work [4, 5, 12] we developed an architecture and several prototypes of collaborative travel planning applications. These used distributed components connected by TCP, CORBA or COM interfaces, adaptive user interfaces and architectures, multiple device user interfaces, and collaborative work-supporting services and infrastructures. While these prototype systems provided useful travel planning applications, a number of disadvantages existed: different architectures for integration and presentation components of the system had to be used; the architectures and integration technologies used were generally difficult to design and build; heterogeneous integration approaches and technologies had to be supported; and dynamic discovery and integration of remote travel planning provider systems was very difficult. These are all similar to problems found in similar systems using similar technological approaches [19, 17, 3, 8].

In this paper describe a new web services-based travel planning application prototype, built using the C# programming language and Microsoft .NET™ implemented web services. We firstly motivate this work with an overview of a desired travel planning application and some key issues developers must face when constructing such a system. We then outline our web services-based architectural solution and describe interesting aspects of our prototype application's design and implementation. We then summarise key experiences gained when developing and evaluating this prototype, outline the advantages and disadvantages of our web services-based solution, and discuss key areas for future research in developing web services-based applications.

2. Problem Domain

A collaborative travel planning application is one that supports customers (intending travellers) and travel agents in sketching, revising, specifying, booking and using a travel itinerary [4, 5]. Figure 1 (a) shows a summary of some of the important use cases that this system should support. This includes customer profile management; travel itinerary creation and maintenance; travel provider searches; travel booking (which may include multiple long transaction interactions with providers to book and pay for items); and itinerary usage during travel. The travel planner application must communicate with a variety of travel providers: airline, hotel, rental car etc. systems as well as third-party travel agents and electronic payment services. These may use a wide variety of different data formats, remote services and business processes.

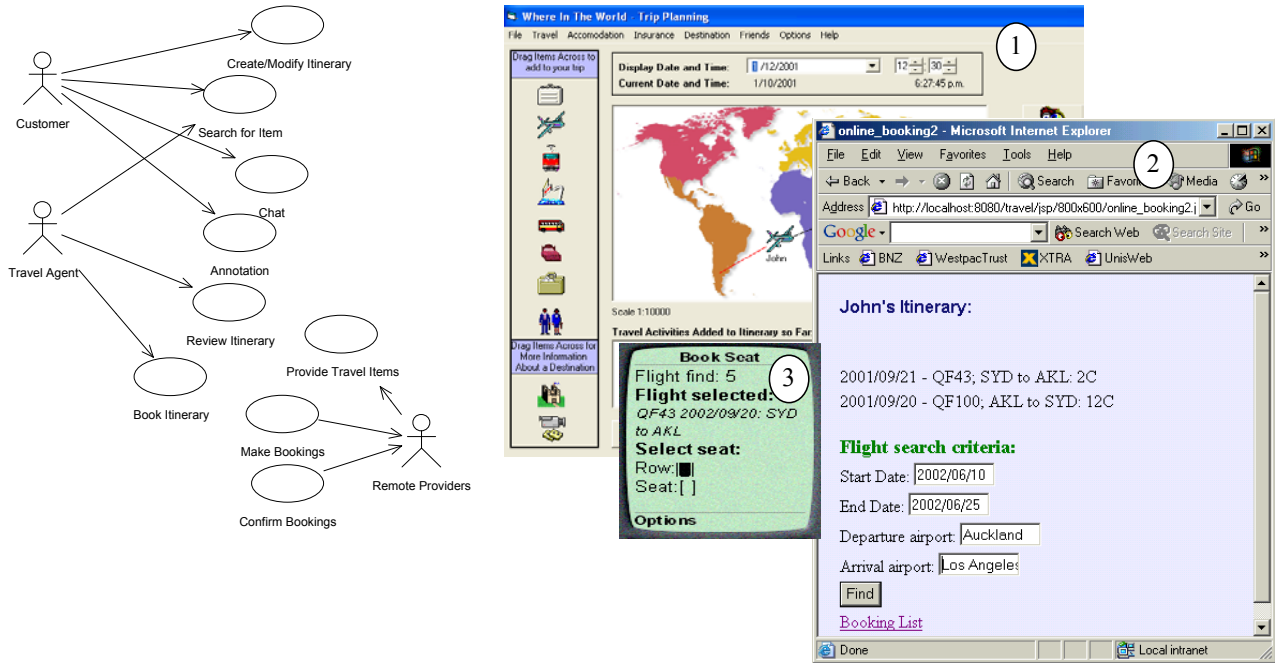

Figure 1. (a) Travel planner use cases; and (b) examples of travel planner user interfaces.

Figure 1 (b) shows some of the user interfaces that customers and/or travel agents may require to perform these tasks. View (1) shows a thick-client (VisualBasic.NET-implemented) travel planner that may be used by agents and some clients to search, build itineraries and book travel.View (2) shows a web-based interface for itinerary management, for customer usage during travel or thin-client interface using travel agencies. View (3) shows a mobile phone-hosted WAP interface for customer itinerary modification and itinerary viewing during travel.

A travel planning example application is often used by those describing web services technologies, but our example application also includes collaboration facilities (collaborative editing, chat, annotation and co-ordination facilities) and more stringent performance and reliability requirements. Unlike many exemplar travel planning applications used by others we aim to provide customers and agents the ability to use the system even if some remote provider systems are down, and to not greatly constrain the performance of the system when many remote systems must be interacted with.

Some of the key issues when developing such an application will include:

- Supporting communication between the travel planner and multiple remote travel provider systems that use different data formats, remote service interfaces, transactions and business processes
- Supporting dynamic location and integration to third party travel provider systems
- Providing good travel planner performance and reliability for customers and travel agents when communication with several remote, highly distributed systems is required
- Ideally providing a common, consistent infrastructure for both remote system integration and multiple user interface and collaborative work support infrastructure for the travel planning application

3. Architecture

We have developed an architecture for our collaborative travel planner prototype that uses web services as the key integration and architectural infrastructure. This architecture is quite different from previous travel planner prototypes we have developed [4, 5] in terms of its use of a homogeneous communications technology framework; its use of standardised dynamic discovery and integration support; its incorporation of flexible adaptors to support integration; and its use of data replication and long running business transactions to provide performance, reliability and scalability.

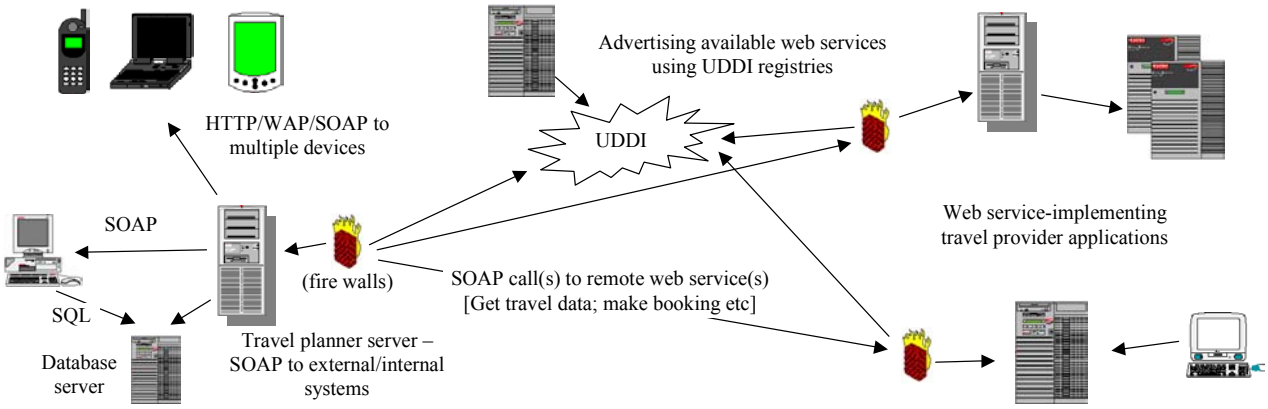

Figure 2. Architecture of our web-services based travel planner.

Figure 2 provides an outline of this architecture. The travel planner is comprised of an application server, database server (which can potentially be run on the same host machine if desired) and various thick- and thin-client interface devices (PCs, laptops, PDAs, WAP phones etc). The thick-client travel planner can directly communicate with the database via SQL or if preferred via a 3-tier architecture using SOAP messages to the travel planner server. The thin-client devices can also communicate with this server using SOAP (if supported), or alternatively the server can provide web servers providing HTTP, HTTPS and/or WAP services.

The travel planner application provides customer profile management, travel itinerary management, maintains a copy of available provider travel services, items (flights, cars, hotel rooms etc) and package deal information, and travel bookings made. Remote travel provider systems are located using UDDI registries which provide a description of the remote web services they provide. The travel planner application server communicates via SOAP messages with located provider web services via interface adaptors to obtain copies of their travel items, to place bookings and to confirm bookings. Remote providers can send messages back to the travel planner asynchronously e.g. quote about to expire, requested booking not longer available etc.

One of the key features of the interaction mechanism of our travel planner with the remote travel providers is the use of data replication and long running business transactions to achieve high performance, availability and scalability, a common issue with web service-based systems [10]. Figure 3 outlines how the travel planner locates, communicates with and supports long running transactions across remote provider systems.

The provider e.g. an airline advertises its available web services with a UDDI registry. The travel planner locates these and either using an existing adaptor e.g. ebXML adaptor to communicate with the provider or builds or configures one to translate its web service data and messages to/from the travel planner's. The travel planner then obtains a summary of flight information provided e.g. dates/times/destinations of flights available, caching it locally.

When a flight search is made this local cache is used. An optional asynchronous check may be made with the provider to check seat(s) are available for date requested and a tentative commitment to hold the seat(s) made by the provider. When a travel itinerary is booked, the travel planner server asks the provider for a final commitment for the seat(s). Subsequently, the provider may however inform the travel planner of e.g. flight change, seat no longer available, cost change (if not committed earlier).

This approach to integration of the travel planner and remote providers supports a range of desired characteristics from the previous section. The use of web service adaptors allows separation of travel planner server data formats and business logic from web service SOAP messages and different data/message formats and business processes in remote travel provider systems. The caching of summary information from providers allows customers and travel agents to make and change bookings with asynchronous communication with providers, supporting high response time and reliability of the travel planner functions. Even if a provider is down, users can modify itineraries and have the modifications attempted on the remote provider systems when they next become available. A "three-phase" long transaction with providers (asynchronously check desired item availability after find; commitment to item during booking; and subsequent asynchronous confirm/reject item after booking) provides high flexibility in interaction with provider systems, along with a scalable architecture and integration process.

4. Implementation

We have built a prototype travel planning application and some example travel provider systems to validate the architecture design from the previous section. We took an existing travel planner implemented with VisualBasic.NET and extended this by the use of C#-implemented .NET web service interfaces to support the communication, data replication, remote data update and long running business transactions with remote travel provider prototypes.

We implemented the remote provider systems using C# and .NET technologies, along with MS Access™ and SQL Server™ databases. The travel planner application server provides ASP.NET thin-client interfaces to a subset of itinerary and travel planning functionality for web browsers (HTTP/HTTPS) and wireless (WAP) devices.

Key features of the design of the travel planner prototype include the use of UDDI and WSDL to describe, register and look-up the remote web service interfaces of travel providers. SOAP web service messages are then used to communicate with these systems via an adaptor architecture which separates the travel planner business logic and data management from the details of the web service messages being exchanged. Adaptors may be fixed for a particular SOAP message set of business transactions of the provider, or may be configurable and able to translate several different provider message sets into the travel planner dialect.

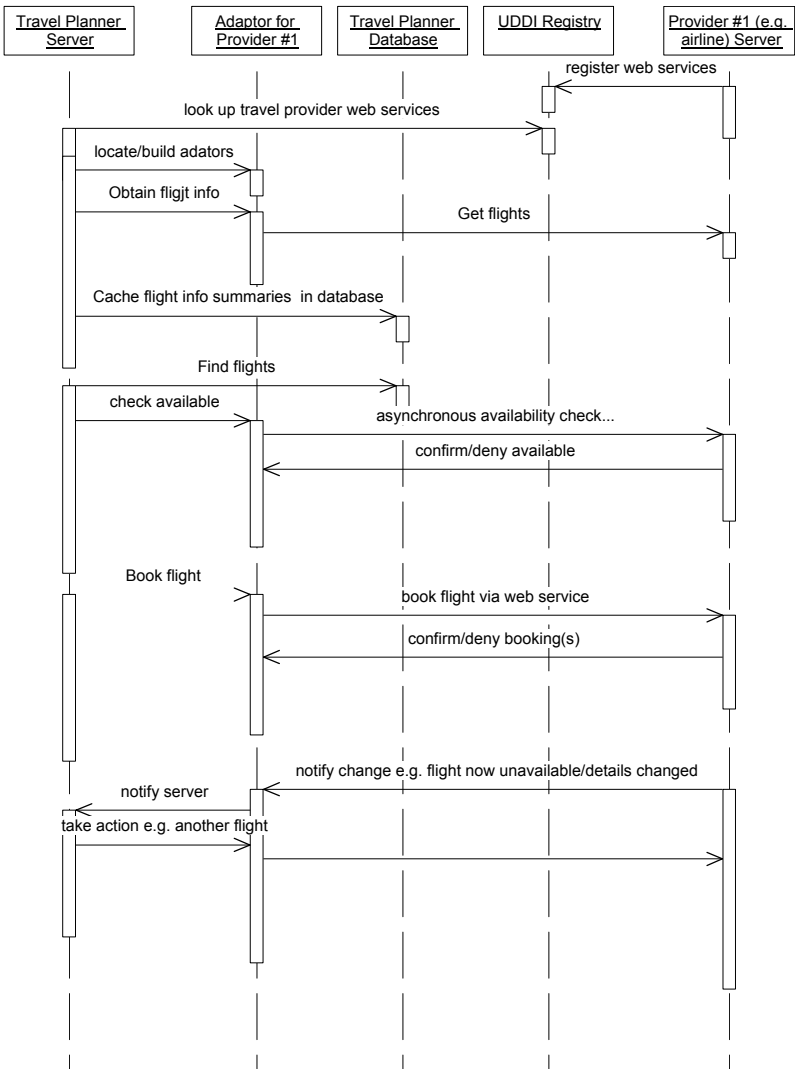

Figure 3. Data replication and remote update in the travel planner application.

Configuration can be manual or automated by examining WSDL descriptions of located provider web services. Message and data mapping are supported by value mappers, for simple data value mapping, and XSLT transformations for complex message and data translation.

Data synchronisation with remote travel providers is used to periodically refresh cached data and can support pulled data and pushed data (via asynchronous SOAP messages) data updates. A form of the Business Transaction Protocol (BTP) [14] is used to co-ordinate long transactions across the travel planner and multiple providers, as outlined in the previous section. Travel items can be asynchronously checked and commitment requested during either itinerary creation or during itinerary booking processes. Compensation transactions via web service calls can be run to un-commit booked items and notifications to change of status of booked items are supported.

Figure 4 shows some examples of the travel planner prototype in use. Views 1-3 are from the thick-client user interface where a travel agent is reviewing a travel plan and making a tentative hotel booking for a client. The customer (or travel agent) first selects the kind of travel item they wish to add or an existing travel item to modify (1). For example, they may choose to add a new hotel booking (2), and they specify date, time, room details etc (3). The date and customer information can be defaulted from the travel itinerary under construction and the interaction of the user with the travel plan (in view 1).

View 4 shows a customer booking of an airline seat via a thin-client, web browser-based interface. This interface provides the same functions as the thick-client interface, but uses a standard browser-based infrastructure and HTML-encoded screen. This has the advantage of not requiring any downloading nor installing of the travel planner application, as required if using the thick-client interface. The three views in 5 are an equivalent WAP-device implemented user interfaces for modifying an itinerary or viewing an itinerary during travel. These interfaces allow a customer who is on the move, or even has started their trip, to access and modify their travel itinerary information.

We have also added some collaborative work facilities similar to those in our previous work (chat, item annotation, collaborative editing, item locking and change notification) can all be provided via the travel planner server with appropriate interface extensions [5]. All of the interfaces – the thick-client, web browser-based and WAP-based, have a web services-based infrastructure. View 6 shows manual addition and configuration of a travel provider's web services via our UDDI registry client. This is used to update the travel planner UDDI registry with new or modified remote travel planning services e.g. to add a new hotel, airline, etc.

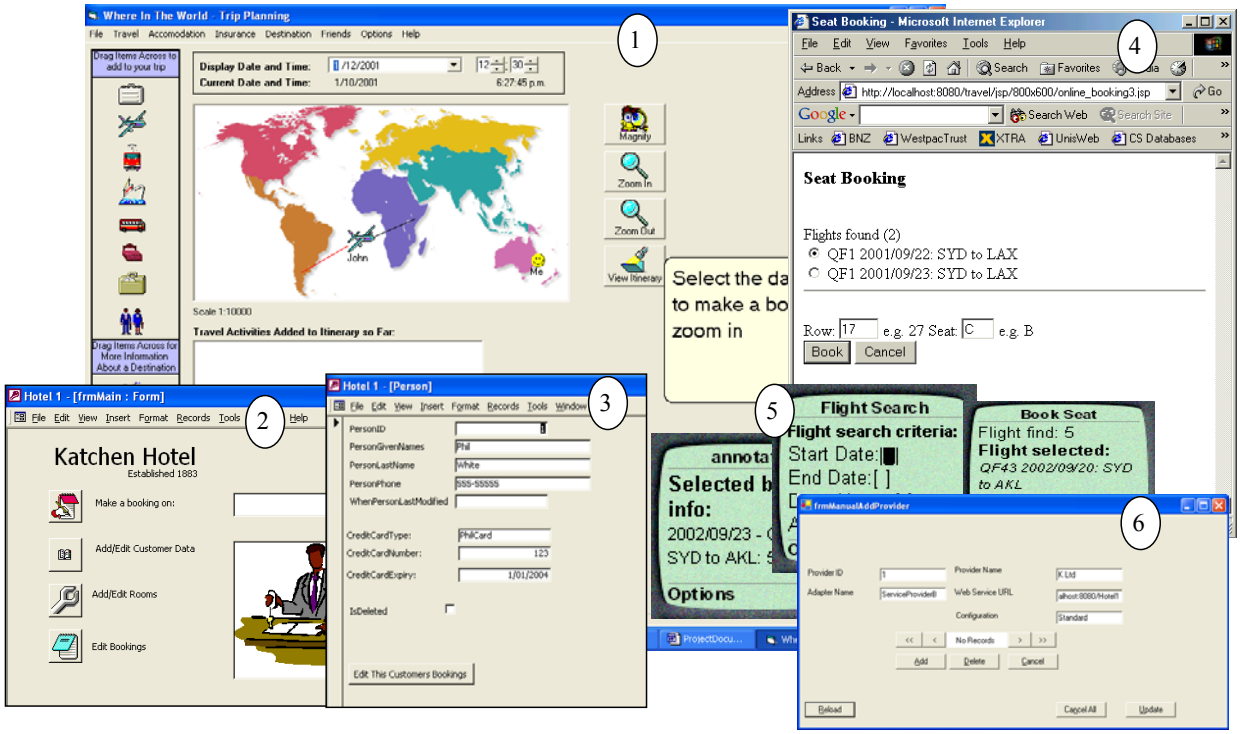

Figure 4. Examples of the travel planner in use.

We found .NET web services to provide a good infrastructure and technology for building these facilities. The existing thick-client travel planner had its database synchronised with remote systems via a .NET server, which used web services to communicate with the remote systems. We used a simple infrastructure to translate remote SOAP protocol messages to and from our server's protocol. We used UDDI registry entries to locate appropriate travel planning service providers and WSDL descriptions of these services to construct adaptors to communicate with them.

We developed a simple adaptor mechanism which allows properties of travel itinerary items (flights, hotel rooms, rental cars, train and bus rides, customer information and provider information) to be accessed and updated, with a set of design patterns used to implement simple property-based mappings between different formats for this data. In future XSLT-style message transformations would be necessary to implement complex mappings but we found this unnecessary for our travel planner prototype.

We have built simple web and WAP-based user interfaces in previous work [5] and used our infrastructure to integrate these with our web services-based travel planner architecture. As in previous work where we generally hard-coded these interfaces to work on a particular platform [12]. However, we plan to investigate the use of our new Adaptive User Interface Technology [] to build a single version of these thin-client user interfaces and with run-time adaptation to different display devices and users [6]. This would allow us to design and specify a single interface for e.g. booking hotel rooms, and provide an adaptation mechanism to support multiple device display of the interface and multiple user/task adaptation e.g. extra facilities for travel agency staff vs. ordinary customer users.

5. Discussion

A wide range of technologies and architectures have been developed for building systems such as the one described here. Commonly used technologies include CORBA and COM [15, 19], EDI and XML messaging [3, 11, 16], and data and message integration tools [8, 3, 2]. Recently various business process integration-based systems have been developed to support both the development and integration of heterogeneous systems [2, 1, 9].The main disadvantages with all of these approaches include the difficultly in building and adapting interfaces (CORBA, EDI); the complexity of translating data and message sets or agreeing on standards for messages (XML, EDI and ebXML); and the need to support heterogeneous integration technologies and wide ranges of different business processes (BPM).

We have carried out a combination of usability, performance and flexibility evaluations of our travel planner prototype. Test users have indicated that the cross-organisational business process embodied by the travel planner are appropriate to the travel planning tasks they wish to undertake. Performance evaluations have indicated that the travel planner prototype provides consistently high performance and availability even under very high loading (both travel planner users and remote application users). We have integrated several different travel provider systems with the travel planner, each using quite different travel item data formats, SOAP message formats and embodying different business processes (some immediate commit to requested items; some deferred commitment; some immediate payment and others invoiced payment).

The main limitations to date with our travel planning application prototype have related to issues of generalising the adaptors so that programming is seldom required to integrate new systems, suitably identifying advertised travel provider web services, and providing tailorable travel planning business process implementations in the travel planner itself. Generalising and providing automatically generated web service adaptors is important to support flexible integration infrastructure and more dynamic remote system interoperation. This will require improved advertising of functional and non-functional characteristics of web services, and improved cross-web service business transaction description than currently provided by WSDL, UDDI and BPEL4WS solutions [13].

We are currently looking at two further areas of research and development relating to the travel planner – improved web service design, characterisation and dynamic discovery and integration, and using web services to build adaptive, multi-device user interface and collaborative work support infrastructure. We are applying aspect-oriented software development techniques to the problem of designing, identifying and adapting to web services, and using a message transformation server architecture to address the problem of supporting adaptive user interfaces and collaborative work support with web services infrastructure. We are also working on developing a generic integration system using the data replication/data update/long running transaction model used in this work [7]. This will be used to provide not only a web services-based integration infrastructure but also to support seamless integration with non-web services legacy systems in the future.

6. Summary

We have designed a web service-based architecture for a collaborative travel planning application. This uses

web services as an integration infrastructure with remote travel provider systems and is being extended to use web services to provide multi-device interface and collaborative work infrastructure. A key feature of our prototype is its use of data replication, caching, remote update and long running business transactions with asynchronous messaging to support a high performance, reliable and scalable solution. We have developed a prototype travel planner with C# and .NET-based web services to realise this architecture.

References

1. Alvarez, M., Pan, A., Raposo, J., Cacheda, F., Vina, A. FINDER: a mediator system for structured and semi-structured data integration, In *Proceedings 13th International Workshop on Database and Expert Systems Applications*, IEEE. CS Press, 2002, pp.847-851.
2. eXcelon Corp, eXcelon B2B Integration Server White Paper, www.exceloncorp.com.
3. Goulde, M.A. Microsoft's BizTalk Framework adds messaging to XML. *E-Business Strategies & Solutions*, Sept. 1999, 10-14.
4. Grundy, J.C., and Hosking, J.G. Developing Adaptable User Interfaces for Component-based Systems, *Interacting with Computers*, Elsevier, May 2002.
5. Grundy, J.C. and Jin, W. *Experiences developing a thin-client, multi-device travel planning application*, In Proceedings of the 2002 New Zealand Conference on Human-Computer Interaction, , July 12-13, Hamilton, New Zealand.
6. Grundy, J.C. and Zou, W. An architecture for building multi-device thin-client web user interfaces, In *Proceedings of the 14th Conference on Advanced Information Systems Engineering*, Toronto, Canada, May 29-31 2002, Lecture Notes in Computer Science.
7. Grundy, J.C., Bai, J., Blackham, J., Hosking, J.G. and Amor, R. An architecture for efficient, flexible enterprise system integration, In *Proceedings of the 2003 International Conference on Internet Computing*, Las Vagas, June 23-26 2003, CSREA Press.
8. Gupta, A. Harinarayan, V. Rajaraman, A. Virtual database technology, *Proceedings of the 1998 14th International Conference on Data Engineering*, 23-27 Feb 1998, 297 – 301.
9. Hanson, J.E., Nandi, P., Kumaran, S. Conversation support for business process integration. In *Proceedings Sixth International Enterprise Distributed Object Computing Conference*, IEEE. CS Press, 2002, pp.65-74.
10. Litoiu, M. Migrating to Web services - latency and scalability. In *Proceedings Fourth International Workshop on Web Site Evolution*, IEEE CS Press, 2002, pp.13-20.
11. McGarr, M.S., Transforming business processes with EDI. *Electronic Commerce World*, vol.12, no.4, May 2002, pp.22-29.
12. Petrovski, A. and Grundy, J. Web-enabling an integrated health informatics system, In *Proceedings of the 7th Conference on Object-oriented Information Systems*, Calgary, Canada, August 27-30 2001, Springer LNCS, pp. 477-486.
13. Piccinelli, G., Emmerich, W., Zirpins, C., Schutt, K. Web service interfaces for inter-organisational business processes an infrastructure for automated reconciliation. In *Proceedings Sixth International Enterprise Distributed Object Computing Conference*, IEEE CS Press, 2002, pp.285-292.
14. Oasis Group, *OASIS Business Transaction Protocol*, Committee Specification 1.0, June 2002, www.oasis-open.org.
15. Sessions, R. *COM and DCOM: Microsoft's vision for distributed objects,* John Wiley & Sons 1998.
16. Sonh, E.J., Lee, H.S., Kwon, T.G. Design and implementation of a message service handler for ebXML. In *Proceedings of the Fourth International Conference on Enterprise Information Systems*, vol 2, ICEIS Press, pp.1064-1069.
17. Swatman, P.M.C., Swatman, P.A., Fowler, D.C. A model of EDI integration and strategic business reengineering. *Journal of Strategic Information Systems*, vol.3, no.1, 1994, pp.41-60.
18. Wiedemann, M. Web Services and collaborative commerce. *Information Management & Consulting*, vol.17, no.3, Aug. 2002, pp.57-60.
19. Wu, E. A CORBA-based architecture for integrating distributed and heterogeneous databases, In *Proceedings Fifth IEEE International Conference on Engineering of Complex Computer Systems*, IEEE CS Press, 1999, pp.143-152.