# Services Monitoring Model for Dynamic Workflow Changes

Leo Pudhota, School of IS, Curtin University of Technology, Perth, Australia 6845
Pudhotal@cbs.curtin.edu.au,
Elizabeth Chang, School of IS, Curtin University of Technology, Perth, Australia 6845
Change@cbs.curtin.edu.au

**Abstract:**
*Collaborative workflow management systems in logistic companies require strong information systems and computer support. These IT integration requirements have expanded considerably with the advent of e-business. This paper proposes service oriented model for monitoring dynamic workflow changes and strategy for implementation of these changes by isolation and segregation of services and business processes where we can monitor workflow and incorporate and integrate the changes in the workflow. This paper will also describe prototype implementation of such system which will be a useful tool for dynamic collaborative workflow management.*

## 1. Introduction

In this paper we discuss the design of workflow management system for dynamic business processes of large logistic consortia, Unpredictable situations occur as a result of changes in decisions made by the management and these changes need to be implemented at production level. The inability to deal with various changes greatly limits the applicability of workflow systems in real industrial and commercial operations. This situation raises problems in workflow design and workflow systems development. We propose workflow prototype implementation through service oriented architecture and system isolation for making changes to the existing workflow, we also propose services monitor to be able to trial these changes before being implemented.

## 2. Dynamic Workflow in Collaborative Environment

The advent of the web to bind organizations together, for carrying out sales over great distances and at any time, has created new modes for marketing and enabled partnerships, previously inconceivable within a wide array of businesses, as well as other human activities[1]. A consequence of this connectivity and information richness is that one is faced with an increasingly dynamic business environment and marketplace. This environment requires major form of collaborative workflow. A workflow is a sequence of activities that produces a result of observable value. A collaborative workflow is to focus on working together towards common goals. They can be small group of companies, project-oriented research teams, to widely dispersed industries with common interests. Effective use of collaborative workflow is now considered a vital element in the success of enterprises of all kinds. Workflow can be represented by sequence diagram, a collaboration diagram, Petri net or an activity diagram [5]. This IT support has expanded with the advent of e-commerce. However, with this advancement of B2B (Business to Business) and P2P (Partner to Partner) e-commerce [6], there has been an increasing tendency to set up consortia that represent several players in a given field. Such consortia consist of companies or organizations in a given field that get together and produce a single site or what appears to be single site in order to increase traffic through the site compared to other competitor's sites and/or extend beyond their region of operation, but a mere enumeration of all workers, activities and artefacts does not quite constitute a process. We need a way to describe meaningful sequences of activities that produce some valuable result, and to show interactions between processes.

## 3. Issues of Dynamic Workflow

Activities and artefacts do not quite constitute a process. We need a way to describe meaningful sequences of activities that produce some valuable result, and to show interactions between processes. Changes in collaborative workflow have to be incorporated into the integrated enterprise system; we have proposed a prototype of its working in our previous papers [8, 9, 10, 11].
In this paper we are concentrating on,
1. Design and Implementation of integrating solution for adaptation of changes in the new workflow into an already existing workflow.
2. Synchronization of new workflow to existing workflow.
Other issues like Management of data scattered over multiple origin systems/legacy systems, for example, a company will have consolidate data in one logical view with a unified architecture, thereby enabling data-source independence. Because application data continues to live and change in the origin systems, the new software layer must be able to retrieve origin data on the fly and also propagate changes back to the origin systems [17]. Provide support for transactions/interaction across multiple back-end systems.
These issues will help in having a uniform data processing environment for the whole enterprise, which would lead to changes and improvements in customer services, control of receivables and increase efficiency in communication, sales, marketing as well as minimization of warehouse stocks, streamlining inventory and logistics flows.
Provide control to Consortium management to monitor the collaborative enterprise's condition, its

stock, order and its general financial condition on a routine basis, this is indispensable to the management processes and enhances decision-making and changes which need to be taken on the short term and long term bases for the consortium to compete in the global market.

## 4. Service Oriented Framework to Support Backend Collaborative Workflow

In this paper we present a service oriented framework for collaborative logistic companies. The framework is divided in 3 sections 1. Business web services layer. 2. Services communication layer and 3. Process and transaction layer.

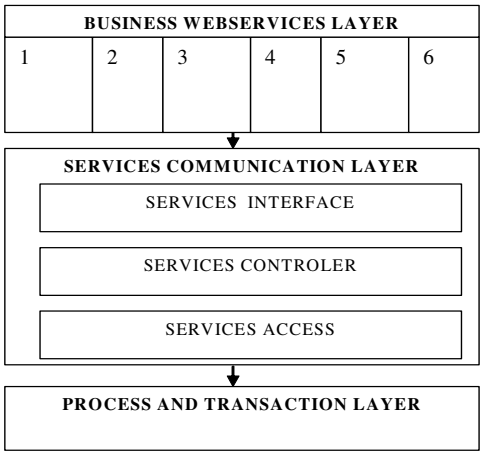


*Fig 1. Enterprise model Frame work*

In Business web application layer browsers interact with HTTP servers in their normal way taking advantage of any technologies that enhance this browser-to-web server link. For example secure socket layer communication protocols in browsers communicate with HTTP servers, which communicate with the Application Server. The Business web services layer has an underlying process for generating web application referred to by 1-2-3-4-5-6 in the figure 1, at run time, Services communication layer provides application's user interface, state management and provide an environment to use and create reusable components [7]. Enterprise model framework shown in figure 1, balances across one or more application server processes (also called instances) running on one or more machines. Once running, Enterprise service framework instances do not go away between user requests; they maintain themselves, their session's state for users, and their database connections. They are efficient, fast, and by definition redundant. It's the job of the HTTP server adaptor to communicate with a given HTTP server and forward requests to one or more application "instances" - an instance is a separate copy of a given application process. Enterprise services framework serving a few users may have only one instance. A large application may have tens or hundreds of instances running on one or more machines. If an application has more than one instance, the Services controller is essentially acting as load balancing agent. If an instance fails, it only affects that particular instance – all other instances and/or the site's web server is unaffected. The controller will forward requests over the network as easily as it will forward requests to applications running on the same box as the HTTP server. In fact, from a load sharing perspective, it is ideal for the HTTP server and Application servers to reside on separate boxes.

Since applications are server based, database access happens behind the firewall. Browsers need never make direct connections to a database server. Services access controls database connections so that they are highly secure (only accessible via actual application API), and conserved (that is, you never have more than one connection per instance regardless of the number of users supported - unless this is specifically something the developers desire). Process and Transaction layer works on underlying java foundation containing fundamental data structure, implementations and utilities used.

## 5. Conceptual Model of Web services Layer and Communication Layer

Referring to figure 1, we see each department has its own responsibility; however they are connected to each other. In a collaborative context, communication may have to be coordinated not only with in the organizations but also across organizations. Therefore a consortium may require synchronized coordination of activities of inter and intra organizational departments. This Conceptual Model provides an architectural separation of business functionality from technology implementation. This separation allows designers to use business rules defined in a UML model to drive two distinct steps in implementing such systems.

Step1. Create platform independent models in UML. The first model is a generic domain model, used to build a common understanding and vocabulary among warehouse Logistics domain experts.Step2. The domain model is then mapped into a representing warehouse logistic business. Each of the models includes a detailed set of UML Class Diagrams, Use Cases and associated Activity Diagrams describing the system [12]. This logical architecture of the web services frameworks is a programming building blocks of the largest granularity. Web services Frameworks is responsible in providing application's user interface and state management Since applications are server based, database access happens behind the firewall. Browsers need never make direct connections to a database server. Services access controls database connections so that they are highly secure (only accessible via actual application API), and conserved (that is, you never have more than one connection per instance regardless of the number of

users supported - unless this is specifically something the developers desire). Designers can use business rules as defined in previous section to define in a UML model. Using this business model, we can create one or more subsystems to represent the logical functions of each of the enterprise systems. This business model contains both the details of the business logic, as well as the mapping of the logic into the major subsystems. The business model forms the basis for managing all changes to the current systems. And the next step is System Integration using Conceptual Model of Platform Specific Models (PSM's), for each of individual systems to form enterprise system [12]. These models were each derived from one or more subsystems in the business model. System construction consists of customizing each of the enterprise systems, and creating the business logic. Business logic that spanned systems is constructed using components technology and deployed in the application server also called web service brokers, please refer figure1.

**6.0 Conceptual Model of Process and Transaction Layer**

This framework contains an underlying java Foundation made of fundamental data structure implementations and utilities used throughout the rest of Enterprise processes. Examples include arrays, dictionaries and formatting classes. These processes provide RDBMS independence for services persistence, provides object persistence transaction management, and provides services useful for web based presentation and deployment. It also provides an environment to use and create reusable components, it facilitates the use of true business objects in services oriented framework and handles storing and restoring objects to a data store and usually in a relational database. Since the business processes and objects created don't care about the underlying database or how their values are presented in user interfaces, they may be re-used over and over in any number of different web applications and can be maintained by developers. Web services framework also provides a persistence layer to maintain information at all time.

As a single process can produce a huge workflow map, the *sub workflow layers* allow the workflow to be broken down into more manageable sections. This also allows modularisation of commonly used functions – for example bulk notification activities rather than having them repeated throughout the main map or even several workflows maps or systems. This also makes them easier to manage and maintain. Subworkflow layer can be very useful to split your main process into its constituent elements – in a large process there is likely to produce a quicker initiation and processing. However, in some cases, the overhead of moving from a 'parent' workflow into a 'child' sub workflow can be a lot higher than the performance benefit of doing so, hence we need to plan the workflow carefully.

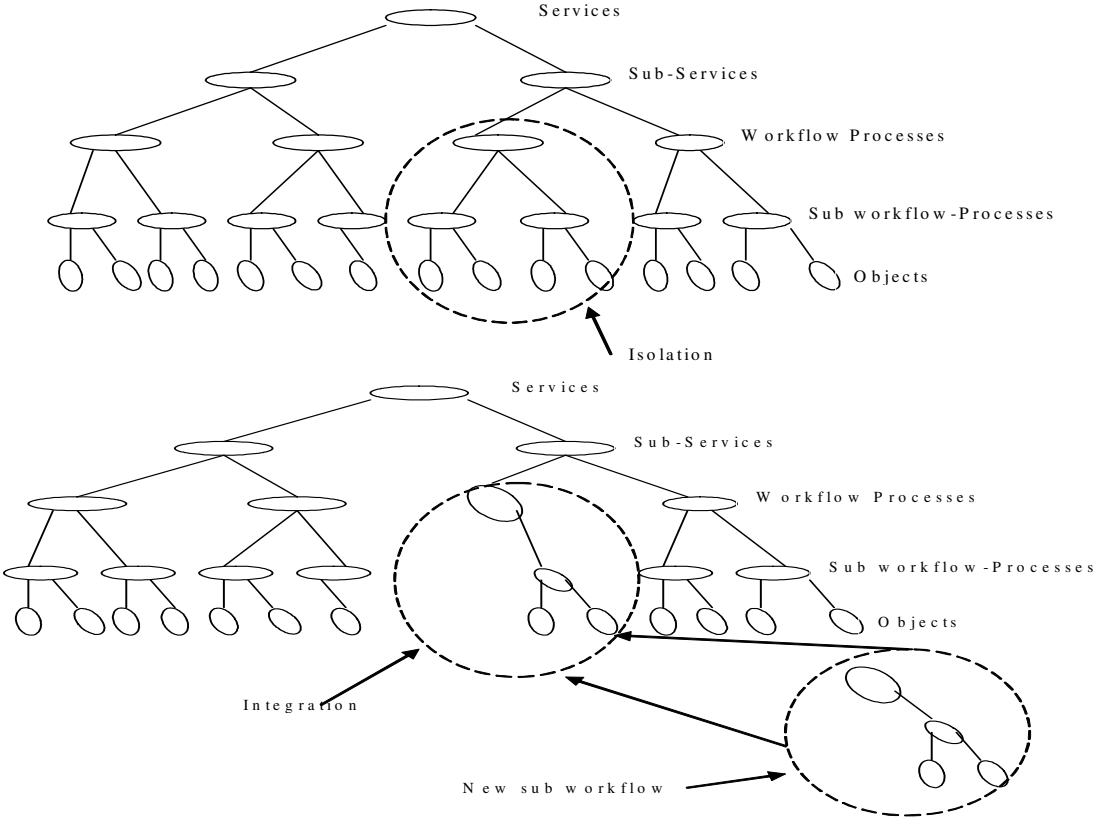


*Fig 2. Isolation replacement and integration of new workflow*

This type of architecture will help in bringing about main areas of changes like:

- Services Layer changes like criteria determining field colouration or visibility or edibility of a given field has changed, or a popup box is now required if some criteria is met.
- Data Changes / Mass Updates like Value Added Tax calculations need to be redone if VAT rate changes, customer name changes
- Business Logic changes like Logistic criteria changes, routing requirements change.
- Patches / Bug Fixes that need to be applied to many active workflows

First case could be handled simply by adding one or more JavaScript functions and some CSS directly onto the affected forms. However, this may result in large-scale repetition of code throughout the workflow system which would be difficult to manage and any changes would mean loading in new versions of the affected views. One approach is to suspend, correct and restart each workflow in sequence, although for large numbers of workflows this would be very time consuming [18]. We have prototyped implementation using Services Monitor.

## 7. Implementation

Large complex workflow processes are broken down into smaller workflows and sub *workflow layers* to be able to better manage and maintain each section. Some process activities may be repeated throughout the main map or even several workflows maps or systems. This allows modularization of commonly used functions and help in easy management by services Monitor described in detail below. Data store shown in Figure 7 aims to eliminate latency by allowing multiple applications to access a single physical data store directly. This architecture is suitable when applications and databases are located in the same data centre; this approach is more intrusive because we usually have to modify some applications to use a common schema. Reading data directly from a database is generally harmless, but writing data directly into an application's database risks corrupting the application's internal state. Although transactional integrity mechanisms protect the database from corruption through multiple concurrent updates, they cannot protect the database from the insertion of bad data. In most cases, only a subset of data-related constraints is implemented in the database itself [20]. To avoid this we include Services Monitor which is visual tool mapping software which is located between the applications and the data store/database,

Services monitor allows us to identify the workflow processes and sub workflow processes and objects stored in the data source, which need to be isolated and a new sub workflow which has to be integrated it also helps to create, edit, or delete existing data_store objects dynamically

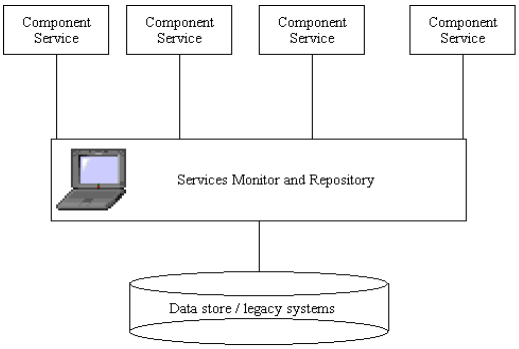


Fig 3. Implementation Framework

when connected to the data store. We can interact with the server data store using datastore diagrams incorporated in the service monitor. Datastore diagrams graphically represent the tables as of a normal database. These tables display the columns they contain, the relationships between the tables, and indexes and constraints attached to the tables. we can use data store diagrams to: View the tables in your database and their relationships. Perform complex operations to alter the physical structure of the database.

We can make changes freely in the datastore diagram without affecting the underlying datastore. When we modify a datastore object through a datastore diagram, the modifications made are not saved in the datastore until we save the table or the datastore diagram, Visualize the structure of your database tables and their relationships. Provide different visualizations of complex databases. Experiment with database changes without modifying the underlying database. Create new tables, indexes, relationships, and other constraints. Alter the structure of your database. Thus, we can experiment with "what if" and various workflow scenarios and also check if these changes made to the workflow can be integrated to the existing workflow, using a datastore design without having to permanently affect its existing design or data. During editing, we can experiment with different object definitions to see if proposed modification will affect the datastore. When we complete these modifications, we can either save our diagram/design or update the database to match the diagram, or we can discard it leaving the underlying database unchanged.

***Differentiating Database*** **Changes**

When editing of table in a datastore diagram has been done, an asterisk (*) appears after the table name in the title bar to indicate that the table contains changes to the workflow that have not yet been saved in the database.


This indicator appears as a result of a change made to the workflow objects in the datastore, represented as a column or index, in the table of the diagram/ design. When we add a modified table to another open diagram, the table appears there with its unsaved changes and an asterisk in its title bar. When you save the table or the diagram, the asterisk disappears.

***Identifying updated Diagrams***

Similarly, an asterisk (*) appears at the end of the diagram name in the title bar to indicate that the diagram contains workflow changes that are not yet been saved in the database.


This indicator appears whenever a datastore object in the diagram has an unsaved change or the diagram/design layout changed since last saved. When the diagram is saved, the indicator disappears.

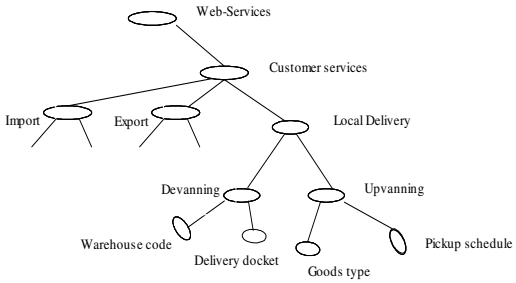


*Fig 4. Example of local delivery process*

For example, you can create a database diagram for customer services department that shows only tables that hold local delivery of goods information. We can create a diagram for this workflow part of the process that shows only those tables that are used in this specific workflow module, here we can make change to Devanning process to replace Warehouse code with Warehouse type code and Delivery docket with Delivery time. We can change the size, shape, and position of objects in the diagram without affecting their definitions in the database. When we save the datastore diagram, the layout of the diagram is preserved as well as any changes made to the object definitions in your diagram are also saved. So as to keep the whole consortium process running we propose exclusive locking mechanism; the locking level determines the size of the process that is locked. Performance and concurrency can also be affected by the locking level used, Exclusive locks are exclusive to the user till the changes are made without having to dissturb the overall workflow. Exclusive lock on a record means that part of the process is denyed access, there by that part of the workflow is isolated so that the required changes can be made only to that part of the process, one may choose some objects or even all of the workflow or sub workflow tasks to be associated with implicit invocation, Figure 5 shows the implementation of customer entity in detail and Figure 6 shows the data store changes on services monitor for customer object of enterprise services

**8. Conclusion:**

In this paper, we have discussed service oriented architecture for dynamic workflow systems. We have also discussed issues and frameworks service oriented enterprises systems and have come up with a prototype for monitoring these various approaches for dynamic adaptation of the changes to the existing workflow. We proposed new frameworks of such systems by the process of isolation and integration and have come up with a prototype of a working synchronization system, our future research will be to further test the implementation constraints of our prototype and propose changes as and when required.

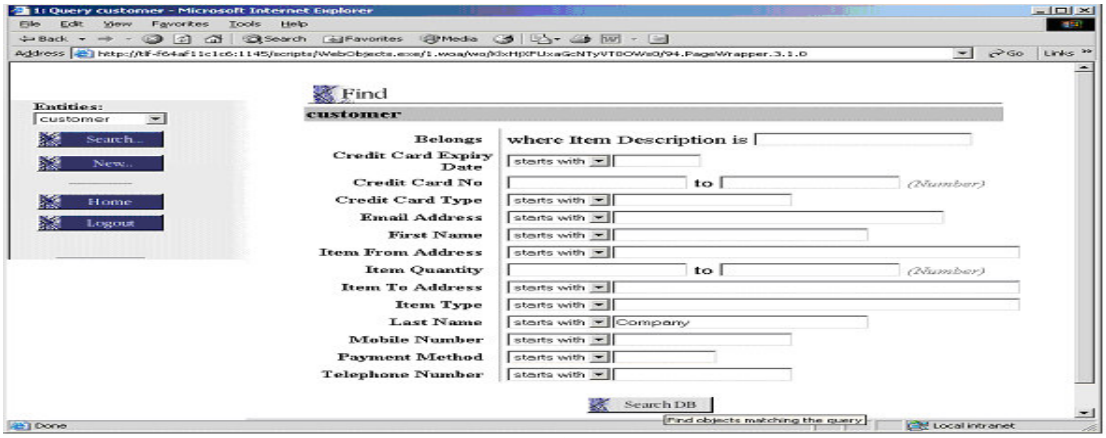


*Fig 5. Example of services*

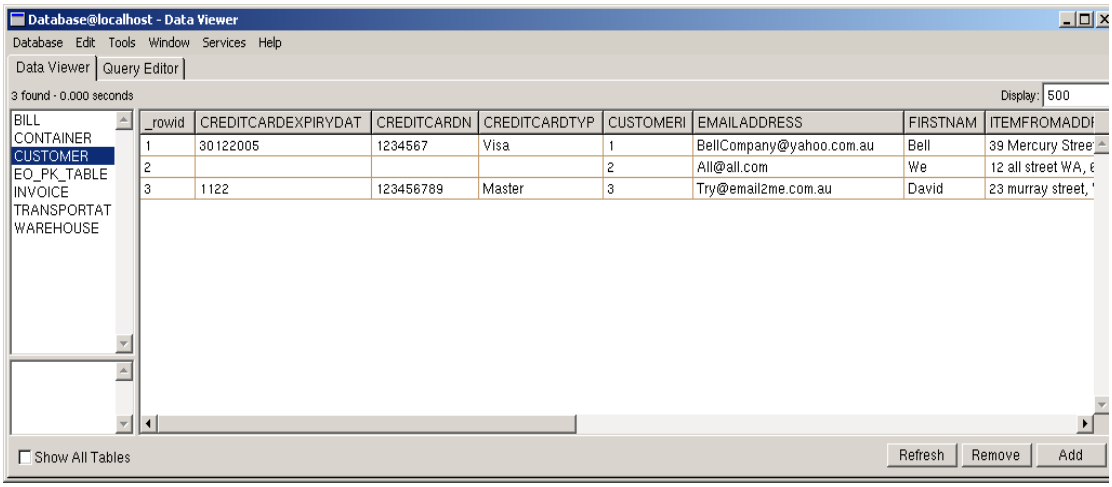


*Fig 6. Example of Services implementation and component integration*

**9. References:**

[1] Marshak, R.T (1994).: "Falling in Love with Distinctions", In "New Tools for New Times: The Workflow Paradigm", Future Strategies Inc.,

[2] Miers, D(1996): "The Workware Evaluation Framework", Enix Limited,

[3] Chang, E(2000): Requirement Specification of Logistic Manager for Software Engineering Project, Department of Computer Science and Software Engineering, The University of Newcastle,

[4] Haake, J.M., et al (2002): "Flexible Support for Business Processes: Extending Cooperative Hypermedia with Process Support".

[5] Denning, P.J (1994): "The fifteen level", In Proceedings of ACM SIGMETRIC Conference on Measurement & Modeling of Computer Systems.

[6] Sheth A (1996): "State-of-the-art and future directions", In Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems.

[7] David Neumann, (2004)An Introduction to WebObjects. Retrieved: July 30, from http://.mactech.com/articles/mactech/Vol.13/13.05/ WebObjectsOverview.

[8] Pudhota L, et al (2003). International Journal, Computer Science, System and Engineering, **"**Extension of Activity Diagrams for Flexible Business Workflow Modeling "volume 18 no3, UK.

[9] Pudhota L, (2004) "collaborative workflow management for logistics consortium" ICEIS Porto, Portugal. [10] Pudhota L, et al (2004) "Modelling the Dynamic Relationships between Workflow Components" ICEIS 2004 Porto, Portugal.

[11] Pudhota L, et al (2004) "E- Business technology adaptation through workflow mining" MSV June 2004 Las Vegas, Nevada, USA.

[12] Retrieved Feb. 10 2005, from: http://www.omg.org/mda/mda_files/UNextMDA4.pdf

[13] Ulieru. M, et al (2000) "The holonic enterprise: a model for Internet-enabled global manufacturing supply chain and workflow management", Canada

[14] Ulieru. M, et al. (2000) "Holonic metamorphic architecture for manufacturing" University of Calgary, Calgary, Canada.

[15] Brandenburger, A. M. et al (1996), Co-operation, Doubleday NY.Brennan, R. (2000), "Performance comparison and analysis of reactive and planning-control architectures for manufacturing", Robotics and Computer Manufacturing 16(2-3), pp. 191-200.

[16] Christensen, J.H. (1994), "Holonic Manufacturing Systems: Initial Architecture Standards Directions", Proceedings of the First European conference Manufacturing systems, European HMS Consortium, Hanover, Germany.

[17] Retrieved Feb. 10 2005, from: http://www.journee.com/n_hl_020703b.html

[18] Anastassia. A, et al (1998) "Scientific Workflow Management by Database Management" book title = "Statistical and Scientific Database Management" pages = "190-199",

[19] Clark, M., et al (2002). Web Services Business Strategies and Architectures. Expert Press.

[20] Retrieved Feb. 10 2005, from: http://www.persistence.com/technology/mapping.html