2006

Using Web services choreography to support an extensible and flexible system development process

M. H. Emami Khansari
University of Windsor

Using Web Services Choreography to Model Business Process in E-commerce

by

Sundaravadanam Menaka

A Thesis

Submitted to the Faculty of Graduate Studies and Research

Through the School of Computer Science

In Partial Fulfillment of the Requirements for

The Degree of Master of Science at the

University of Windsor

Windsor, Ontario, Canada

2006

Your file Votre référence
ISBN: 978-0-494-17055-7
Our file Notre référence
ISBN: 978-0-494-17055-7

Abstract

Traditionally, business websites were mainly used to access huge amounts of information, they are called Navigational Sites. A new generation of web applications, online auctions and e-commerce sites have emerged over the past few years that unlike navigational sites execute business process in them. The underlying web application modeling languages have not evolved to support these new application requirements. They treat a business process as a form of navigation. Due to the deficiency in conceptual modeling, the resulting web applications suffer from design and usability problems. Example of such design and usability problem is the famous "Amazon Bug" problem, a site evolved from poor conceptual modeling of business process.

Web modeling languages like WebML (Web modeling Language), OOHDM (Object Oriented Hypermedia design) are evolved from hypermedia models. And they pay attention only to hypermedia modeling and model business rules as a form of navigation, this is called business process emulation. Our approach is to use web service peer-peer language, such as WS-CDL to model business process in an e-commerce application. In this way we introduce a new layer that models all the business rules using WS-CDL In this approach the hypermedia model models only navigation using WebML and once business process is initiated from simple navigation the process layer defined using WS-CDL will execute the business rules. By constructing a case study to test this hybrid-modeling framework, we hypothesise that this newly released peer-peer collaborative

language for web-services can be used to model the concepts of business process. Thus we get a unique approach to model business process along with navigation. This approach is a proposed solution to the issues of business process emulation.

Keywords: WS-CDL, WebML, Modeling and Business Process

Acknowledgements

I would like to take this chance to offer my sincere gratitude to the faculty of the School of Computer Science at the University of Windsor who, during my Master's program, led me through the scientific world of Computer Science. I also would like to acknowledge the kind and encouraging support of Dr. Ezeife.

I offer my genuine appreciation to Dr. Ziad Kobti whose precious advice and comprehensive remarks not only helped me with my thesis work, but also led me to the professional work of scientific research. While his empirical attitude and scientific approach has already affected my strategy towards a structured and effective academic research, I am so flattered to have my thesis done under his wise supervision.

Here it is appropriate to recall the memory of Dr. Li whose expert guidance formed the fundamentals of this work.

I also thank Dr. Pathak for his support and technical suggestions his precious remarks about the correct features of a thesis paper.

I should also thank Mr. Gary Brown at pi4soa project for their constant support and consultancy during my work.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Introduction

Web conceptual modeling is a branch of web engineering which addresses the specific issues related to design and development of a large-scale web application [2]. Web conceptual modeling is a young discipline, which is gaining popularity among web developers and CASE tool vendors [4]. In order to provide efficient communication, web sites should be derived from conceptual models [18]. The web conceptual modeling languages like Object Oriented Hypermedia Design (OOHDM) [13], Web Modeling Language (WebML) [14] are an evolution of hypermedia models that pay attention mostly to the specification of data structures and navigation primitives. Now a days, web applications are transformed from simple navigation to process oriented applications like online auctions, online purchases, reservation systems which would involve complex processes running in parallel. This has raised the need for changes in the existing conceptual modeling methods because they emulate business process as a form of navigation [11].

A Business Process is a business transaction that takes place in a web site [32]. They use web pages as interfaces for triggering actions, example: an atomic action like adding a product to the shopping cart, this is achieved by pressing AddtoShoppingCart button on the Web page. This event triggers AddtoCart operation. This operation invokes shopping carts object add (product) operation thus changing the internal state of the shopping cart. The object state

is modified when browser navigates back and forth and hence the state of the object gets out of sync. When we checkout we end up in getting erroneous results [34].

Example of Business Process Emulation:

The user wishes to buy some music CD's and browses through the catalogue to select what he wants to buy. Figure 1.1 shows simple browsing.

Step 1:

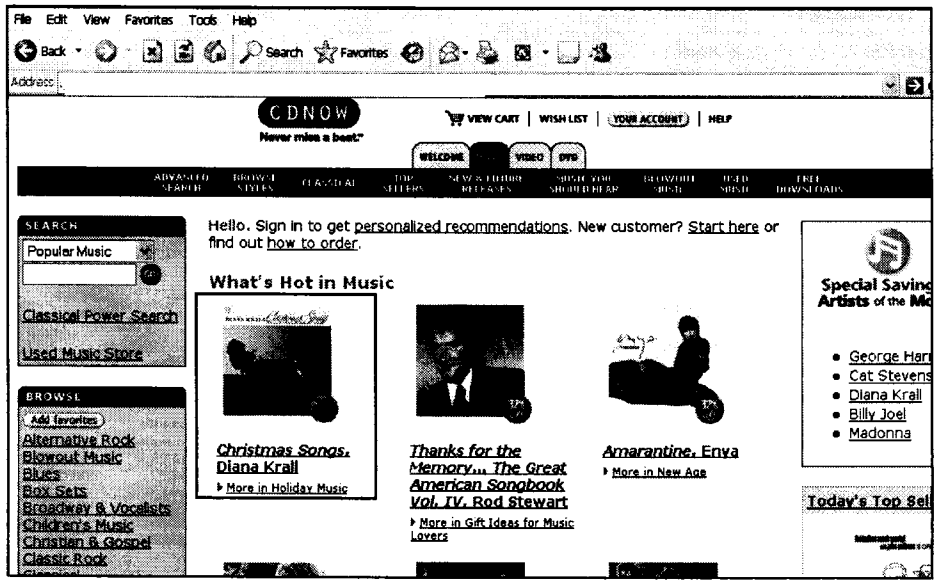


Figure 1.1: Simple browsing music catalogue

He chooses Diana Krall "*Christmas songs*" Audio CD and add's them to the shopping cart. After choosing the first CD the application presents to the user the shopping cart with selected items. The shopping cart page is shown in figure 1.2

Step 2:

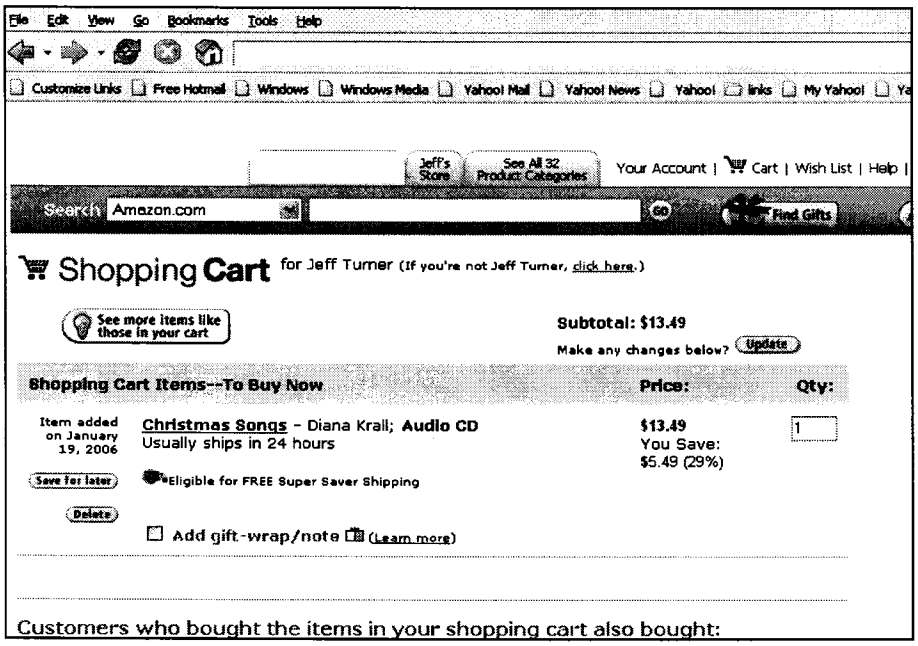


Figure 1.2: Shopping cart Page

Step 3:

If the user decided to add one more Audio CD say " The Girl In The Other Room" by Diana krall, he could use add to cart in the page shown in fig 1.3 and check it out for purchase.

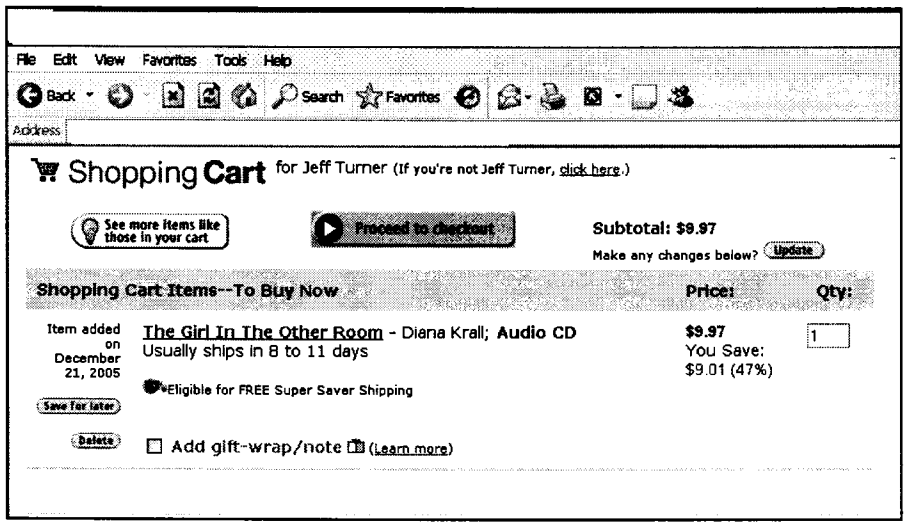


Figure 1.3: The buyer adds one more CD

This skeleton makes the user move to the shopping cart page which now contains 2 Audio CD's as shown in the figure 1.4

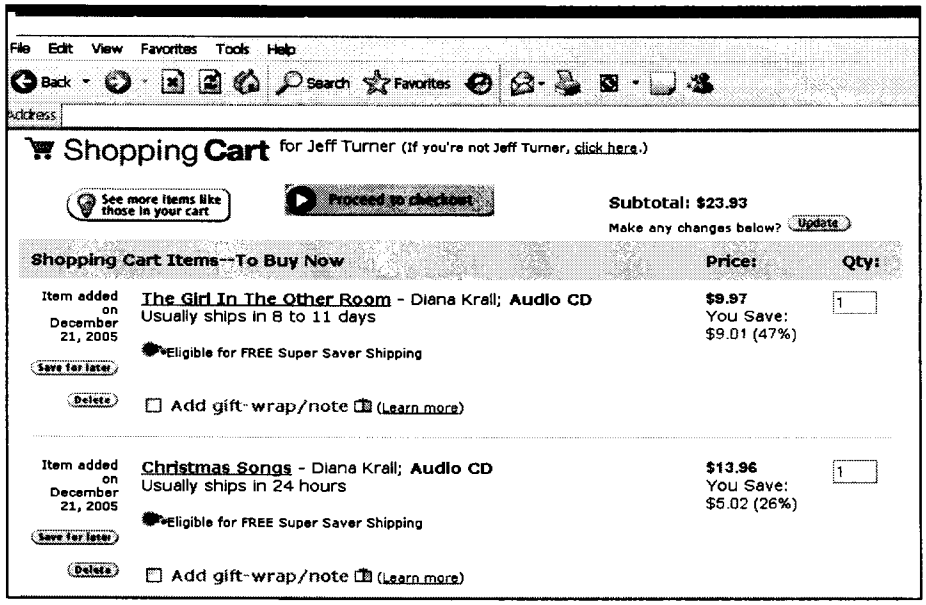


Fig 1.4: Shopping cart with two selected items

Step 4:

Thinking of the price, the user changes his mind and decides that he does not want to buy the second Audio CD. The rigorous user would delete the CD from the shopping cart, using the delete button and then he would checkout the shopping cart with just one product. This is explained in the figure 1.5

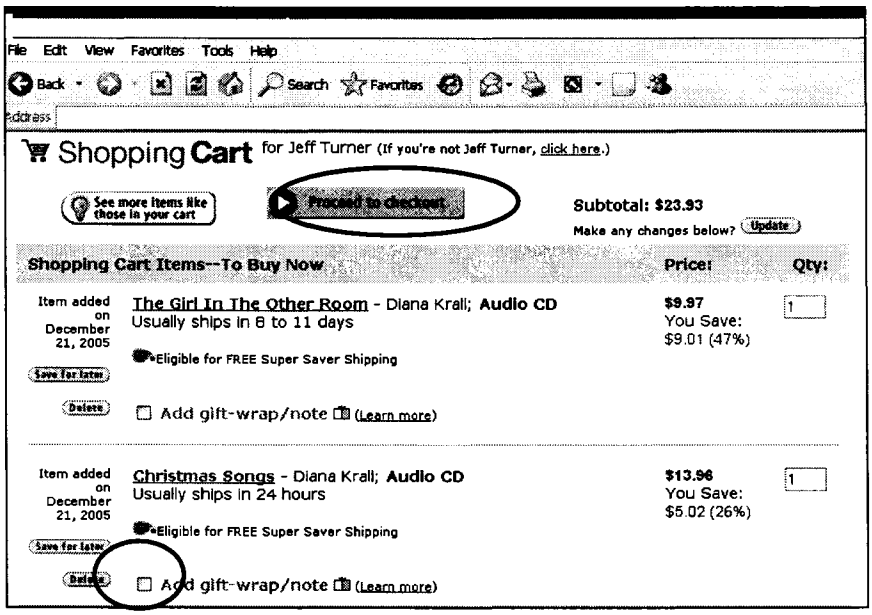


Figure 1.5: The User Decides Not To Buy The Second Cd

Step 5

In contrast a "free navigator" [50] would roll back to previous shopping cart as shown in figure 1.6 and 1.7 and then he would check out this page/cart.

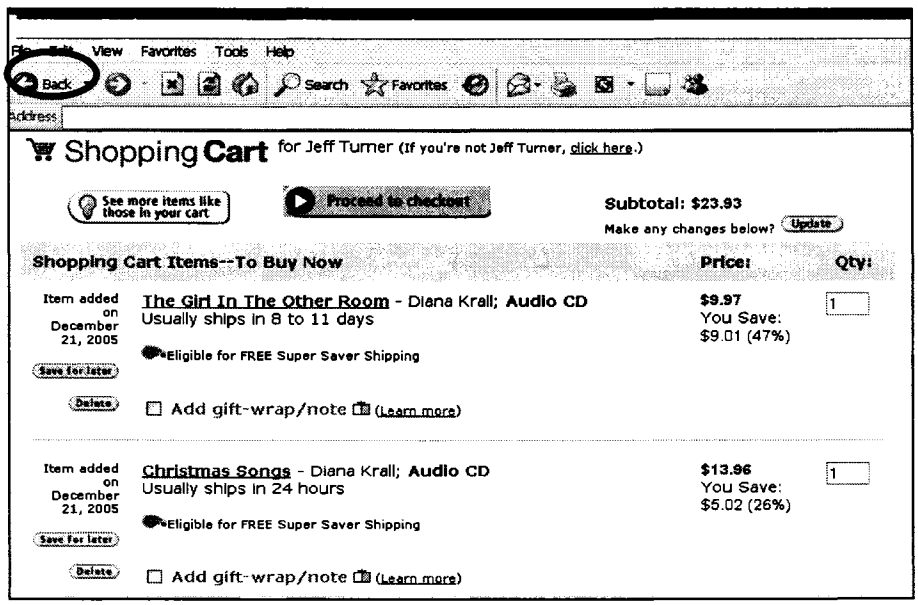


Figure 1.6: Buyer Uses Back Button

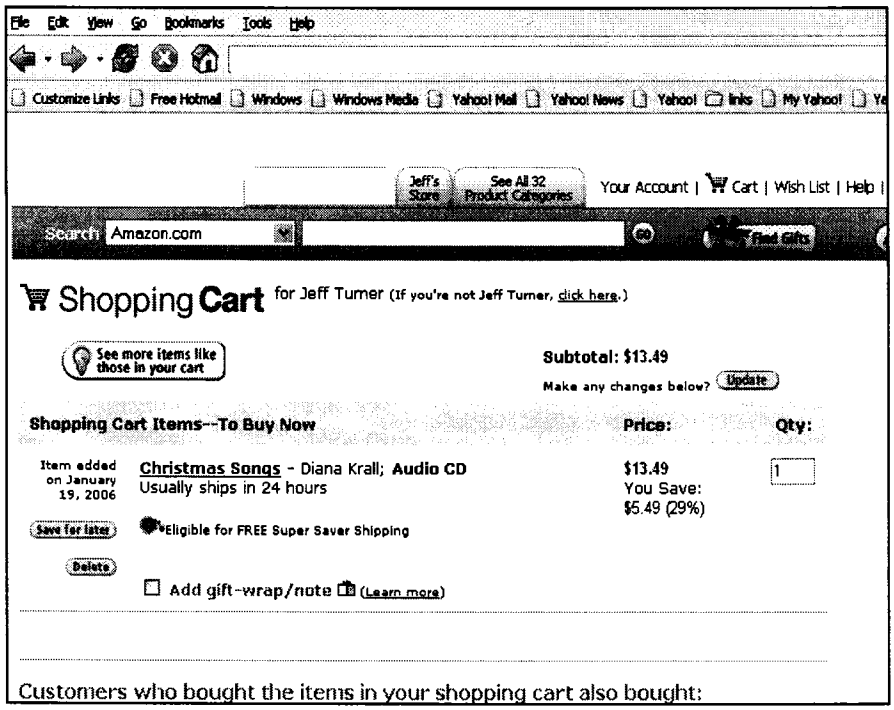


Figure 1.7: Buyer Uses Back Button To Reach A Page That Has One CD

When the buyer proceeds to check out from the page shown in 1.7 he ends up in buying two Audio CD's instead of one because the process state is still holding two CD's.

Thus, my emulating business process along with navigation of the process flow/ control flow is lost.

When emulating business process as a form of navigation the following problems are expected to happen [11]:

- Even a simple control flow is lost. This would lead the user through an unconditional linked sequence of navigational nodes.
- The user could get disoriented and fail to return to complete the process.
- Another problem is there is no way to define what it will mean to

leave the process nodes.

Hence, a business process should be separated from navigation primitives. The features of business process are [11]:

- A process is driven through its activities. A set of activities and its control flow defines a process
- A process keeps its state internal and is time stamped.

In this work we describe how a conceptual modeling language (WebML)[14] can be supported to model a workflow driven hypertexts with the help of a web service workflow language (WS-CDL) [1]. The Workflow Management Coalition (WFMC) describes workflow as the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action according to a set of procedural rules [19]. With the help of the workflow meta model, the design space of WebML is extended and our goal was to see if WS-CDL could be used to model workflow driven hypertext in WebML.

We are interested in:

- Modeling business process using web services peer-peer language.
- Propose a model using WS-CDL, WebML to overcome the problems of business process emulation.
- To demonstrate the ability of a newly released peer-to-peer collaborative language for web services in defining process from a global viewpoint where ordered message exchanges results in accomplishing a common business goal.

1.2 Background Study

1.2.1 What Is E-Commerce?

E-commerce is a business-to-business (B2B) initiative aimed at communicating business transaction documents on Internet and any on-line transaction of buying and selling via Electronic Data Interchange (EDI) [35].

1.2.2 E-Commerce Site

Business transactions take place in an e-commerce site. This would require a complex and stable web site. When taken an e-commerce site we would need to tackle the most complex web design, one that involves finding the right mix of aesthetics, brand identity and interactivity [8]. This then has to fit into a technical tangle of database, customer service and fulfillment systems.

1.2.3 Requirements of E-Commerce Site

A new generation of web applications has emerged over the past few years ever since the advent of e-commerce [20]. Unlike navigational sites, which mainly allow access to huge amounts of information, these new e-commerce sites execute business processes. Hence the standard requirement for any stable e-commerce application is the inclusion of both navigational elements and business processes.

1.2.3.1 Business Process

A business process is a business transaction that requests information from or changes the data in a database. It is a specific event in a chain of structural business activity. The event typically changes the state of data or a product and generates some type of output [32]. An example of a business process includes receiving orders, invoicing, shipping products, updating employee

information, or setting a marketing budget. Business processes occur at all levels of an organization's activities and include events that the customer sees and events that are invisible to the customer.

1.2.3.2 Advantage of Business Process Modeling

Business modeling enables to realize an advantage over the competitors. By accurately knowing how the business runs, one can enforce efficient processes, and change not-so-efficient processes to make the company more cost-effective and adaptable to change over the competitors [8]. Modeling processes cannot be done with just flowcharts and spreadsheets because they offer only a limited view. These limited views give false readings of how things are running [36]. In an e-commerce system we have business and process model. Business model is about interactions among business partners and process model focuses on operational and procedural aspects of business communication. Thus, a business model answers "*what*" in an e-commerce system and the process model answers "*how*" in an e-commerce system [37].

1.2.4 Web Modeling

Web modeling (model-driven Web development) is a branch of Web engineering which addresses the specific issues related to design and development of large-scale Web applications [2]. In particular, it focuses on the design notations and visual languages that can be used for the realization of robust, well-structured, usable and maintainable Web applications. Designing a data-intensive Web site amounts to specifying its characteristics in terms of various orthogonal abstractions such as [14]:

- Structural Model

- Composition model
- Navigational Model
- Presentation model

1.2.4.1 Need for Web Modeling

As the size and complexity of applications increases, a systematic approach is needed that would help in dealing with complexity [18]. By web modeling a developer can formalize their knowledge about the web applications in a high-level platform independent approach. For any e-commerce application a healthy navigation structure is the key success in hypermedia application. The user should understand where he/she could go and how he/she can reach a desired point and can consequently be benefited the most from the application. Building the interface of a web application is also complex; not only do we need to specify which interface objects should be implemented, but also the way in which of those interface objects will interact with the rest of the application. We also need to distinguish when an interface action causes navigation to another page, or it is just a local interface effect (for example activating a pop-up menu), etc.

Design methods provide high-level abstraction and mechanisms aimed at solving most of the previously mentioned problems. Object Oriented Hypermedia Design (OOHDM) [13], Web Modeling Language (WebML) [14], Unified Modeling Language (UML) [22] are the most popular modeling languages for designing web applications at the conceptual level.

1.2.5 Overview of Web Application Modeling

Modeling is a visual process used for constructing and documenting the design

and structure of an application [2]. Use of modeling tools gives developers a high-level view of what could amount to thousands of individual lines of code. "Modeling can be introduced at any point in an existing project, as most modeling tools will read existing code, creating a visual model based on that code" [20]. Most of the modeling languages follow the same structure. Each step focuses on a particular design concern. Classification, aggregation and generalization/specialization are used throughout the process to enhance abstraction power and reuse opportunities [21]. The Table 1.1 below summarizes the steps, products, mechanisms and design concerns in Web development.

Activities	Products	Formalisms	Mechanisms	Design Concerns
Requirements Gathering	Use Cases, Annotations	Scenarios; User Interaction Diagrams; Design Patterns	Scenario and Use Case Analysis, Interviews, UID mapping to Conceptual Model	Capture the stakeholder requirements for the application.
Conceptual Design	Classes, sub-systems, relationships, attribute perspectives	Object-Oriented Modeling constructs; Design Patterns	Classification, aggregation, generalization and specialization	Model the semantics of the application domain
Navigational Design	Nodes, links, access structures, navigational contexts, navigational transformations	Object-Oriented Views; Object-Oriented State charts; Context Classes; Design Patterns; User Centred Scenarios	Classification, Aggregation, generalization and specialization.	Takes into account user profile and task. Emphasis on cognitive aspects. Build the navigational structure of the application

Abstract Interface Design	Abstract interface objects, responses to external events, interface transformations	Abstract Data Views; Configuration Diagrams; ADV-Charts; Design Patterns	Mapping between navigation and perceptible objects	Model perceptible objects, implementing chosen metaphors. Describe interface for navigational objects. Define lay-out of interface objects
Implementation	Running application	Those supported by the target environment	Those provided by the target environment	Those provided by the target environment

Table: 1.1: Steps in Web Development [21][Daniel Schwabe]

1.2.5.1 Requirements Gathering

The first step is to gather the stakeholder requirements. To achieve this, it is necessary to first identify the actors (stakeholders) and the tasks they must perform. Next, scenarios are collected (or drafted), for each task and type of actor. The scenarios are then collected to form a Use Case, which is represented using User Interaction Diagrams. These diagrams provide a concise graphical representation of the interaction between the user and the system during the execution of a task.

1.2.5.2 Conceptual Design

In this step a conceptual model of the application domain is built using well-known object-oriented modeling principles, augmented with some primitives such as attribute perspectives, multiple valued attributes. Conceptual classes may be built using aggregation and generalization/specialization hierarchies. There is no concern for the types of users and tasks, only for the application domain semantics. A conceptual schema is built out of sub-systems, classes

and relationships.

1.2.5.3 Navigational Design

Here we describe the navigational structure of a hypermedia application in terms of navigational contexts, which are induced from navigation classes such as nodes, links, indices, and guided tours. Navigational contexts and classes take into account the types of intended users and their tasks. Nodes represent logical "windows" (or views) on conceptual classes defined during domain analysis. Different navigational models may be built for the same conceptual schema to express different views on the same domain. Links are derived from conceptual relationships. By defining the navigational semantics in terms of nodes and links, we can model movement in the navigation space (i.e., the subset of nodes with which users can interact at any given moment) independently of the conceptual model. The navigational model may evolve independently from the conceptual model, simplifying maintenance.

1.2.5.4 Abstract Interface Design

The abstract interface model is built by defining perceptible objects (e.g. a picture, a city map, etc.) in terms of interface classes. Interface classes are defined as aggregations of primitive classes (such as text fields and buttons) and recursively of interface classes. Interface objects map to navigational objects, providing a perceptible appearance. Interface behaviour is declared by specifying how to handle external and user-generated events and how communication takes place between interface and navigational objects. Hence once the navigational structure has been defined, it must be made perceptible to the users through the application interface, which is done in which step by

defining an abstract interface model. This means which interface objects the user will perceive and in particular the way in which different navigational objects will appear, which interface objects will activate navigation, the way in which multimedia interface objects will be synchronized and which interface transformations will take place.

1.2.5.5 Implementation

Implementation maps interface objects to implementation objects and may involve elaborated architectures, e.g., client-server, in which applications are clients to a shared database server containing the conceptual objects.

1.2.6 Overview of Web Modeling Languages

Information modeling has gained more importance [5]. The most famous information modeling languages are Object Oriented Hypermedia Design (OOHDM) [13], Web Modeling Language (WebML) [6] and adoption of Unified Modeling Language (UML) [22].

1.2.6.1 Comparative Study of Various Modeling Languages

A complex web application is said to be functionally complex and information rich. The existing modeling languages focuses modeling at a relatively low-level, they failed to address higher-level aspects such as business process modeling [5]. UML [42] focuses on functional aspects and WebML concentrates on informational aspects of a web system. Business process is more functionally rich. When the e-commerce web application consists of both functional elements like business process and informational elements like navigational elements there is no one modeling language that fulfills both the requirements.

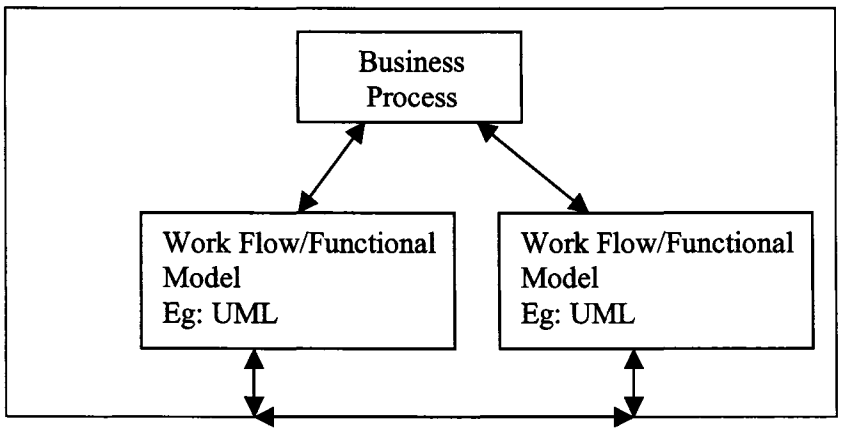


Figure 1.8: WIED [5] Modeling language for Web Application

We tried to evaluate the various modeling languages based on how they modeled functional aspects and informational aspects of a complex e-commerce application.

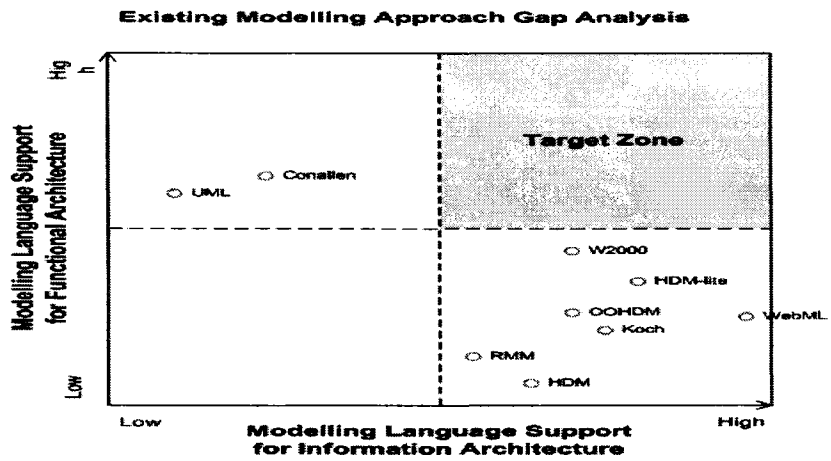


Figure 1.9: WIED [5] Comparative Study of All Modeling Language

UML does not provide a clear connection between business models and the functional design of systems. The problem with web information model is that they only capture aspects such as the content viewpoint & navigational structures. To connect these two sets of models, we need to model not only the

information itself but also the relationship between the underlying content and the user perceived views of those contents, the interactions with these views and the ways in which the information is represented to the users. This is much more complex than data modeling since representing informational contents is also important. We could use UML to represent the functional aspects but they are not effective in representing these informational aspects. Similarly, we could use WebML for informational modeling but not so effective in modeling functional aspects. This raises the need for a unique language to represent both functional and informational parts of a complex web site.

1.2.7 Limitations of Web Modeling Languages

Web applications include both navigation and business process, but the design methods treat business process as just another kind of navigation [11]. Unlike navigational sites, which mainly allow access to huge amounts of information, these new applications- portals, reservation services, auctions, e-commerce sites, and so on support and execute business processes. The web modeling methods has not yet evolved to support these new applications requirement. The web application that's been generated from these modeling methods suffers from design and usability problems. They therefore generate erroneous results from business-process executions.

1.2.7.1 Hypermedia-Based Navigation

The hypermedia paradigm provides easy access to information resources. It considers a web application as a set of nodes linked through URLs. Hypermedia-based navigation has evolved from pure navigation to advanced navigation and even business-process emulation.

1.2.7.2 Pure Navigation

A user can access any node linked to any other node by clicking the related links and opening the target pages; this navigation process can repeat itself infinitely.

Semantics of simple navigation are:

- The page that's present in the browser determines the navigational state. Thus, the user can change the current state via navigational links or browser's back and forward buttons.
- The user can decide freely which node to visit next.
- The browser cashes the history.

OOHDM [13], WebML [5]–web application design methods describe navigational aspects in similar ways. WebML uses Structural Model, Composition and Navigational models to describe possible navigation.

1.2.7.3 Advanced Navigation

Advanced navigations are not only read only applications. They have web pages to trigger various actions. For example, adding a product to shopping cart, this calls an application domain object. This object performs an operation thus changing the state of the object. Advanced navigation can change an applications internal state. Unlike "pure" navigation, the web application's state is determined by the current node displayed by the browser and by the state of the objects associated to the nodes. Since the process is dependent on the objects status associated to the current node. It might get out of sync when user changes of the nodes by using the browser's backward button. This would lead to the famous problems so-called "Amazon bug problem".

1.2.7.4 Business Process Emulation

Business process is nothing but a predefined activity sequence. For example for a shopping cart check out process the predefined sequences are:

- Log in
- Conform items
- Enter shipping address
- Select delivery options
- Select payment methods etc.

Only after executing these activities we can complete these processes successfully.

1.2.7.5 Problems with Business Process Emulation

When we emulate business process in navigation, this would lead to several problems. For example the checkout process discussed in 5.2.3 the process is a control flow of strict sequence navigational nodes.

- If the designer wants to let the user navigate to other pages that are not related to the current process the user could get disoriented and might fail to return to complete the process.
- Another problem is that we cannot define what it means to leave the process nodes: should the process be aborted or remain in same state for later navigation back to it.
- Third problem is that user can create inconsistent state in the process by exploring other pages.
- The ultimate problem lies when the user uses the browser back button during a business process. It's difficult to answer how the system

would behave when the user uses the back button in the middle of a process.

Hence its gets clear that the developers can't model and represent business process adequately using hypermedia primitives and navigational semantics.

1.2.7.6 Features of Business Process

- Business process drives users through its activities. It defines the set of activities to be executed and the possible control flow among them.
- The process keeps its state internally, and it, alone can change the state in response to the user's actions; pressing the browser back should not affect the process state.

1.3 Overview of Problem Background

In the last years web has been the premier platform for application development [38]. Business process is essential when we build complex applications over message-based paradigms [2]. There is a lack of well founded software engineering methods for encoding business process within web applications because current web modeling is data-centric and do not cover the hypertext front-end [37]. Since web is becoming a vehicle for implementing B2B applications this raises the need for extending web conceptual modeling from data-centric application to data and process centric applications [4]. The conceptual modeling of web applications cannot simply pile up existing techniques of hypermedia design (borrowed from the hypermedia/web communities) with methods and notations for "traditional" operation modeling (borrowed from the information systems or software engineering communities). "The crucial point is to integrate and extend

models, design methodologies and techniques, in order to meet new design challenges" [36].

Conceptual modeling for web proposed so far is an evolution of hypermedia models [35]. WebML since 1998 is being used for specifying high-level specification of data centric web applications and automatic generation of their implementation code [5]. WebML is good for web applications making extensive use of database, while such specifications adequately cover the needs of applications developed within a single organisation and WebML do not cover well the integration of web applications within externally provided business logic [39]. Even UML that is good at Process Modeling when trying to model web-based system Navigation and process are poorly modeled because Object Oriented (OO) methods based on UML [42] profile do not capture many essential ingredients of web-based systems [22].

1.4 Motivation of Thesis

From the previous study we understand that large volumes of business transactions take place in an e-commerce site and these web conceptual modeling languages treat business process as a form of navigation. Because of these deficient conceptual modeling languages the resulting web applications suffer from design and usability problems and therefore generate erroneous results. This thesis focuses on the above problem and aims to solve the problem by using a workflow language to model business processes in web applications in order to provide a stable modeling approach that would combine business process along with navigation is an e-commerce application.

1.5 Thesis Outline

The rest of the thesis is organized as follows: Chapter 2 explains the preliminaries of the conceptual modeling language the WebML and its limitations followed by Chapter 3 explaining about a workflow language WS-CDL and its features. Chapter 4 reviews related work on web conceptual modeling. Chapter 5 on experimental design. Chapter 6 on implementation and test results analysis and finally chapter 7 focuses on conclusion and future work.

Chapter 2: WebML (Web Modeling Language)

WebML (Web Modeling Language) since 1998 is a modeling language for designing web applications [6]. WebML at the conceptual level can specify the concepts of complex web sites. The four orthogonal views of WebML are:

- Structure Model
- Composition Model
- Navigational Model
- Presentation Model

The concepts of WebML are associated with a graphic notation and a textual XML Syntax [12]. These specifications are independent of the client side language and the server side platform. The entire design methods are fully implemented in web design tools like Torrisoft, Web Ratio [25]. A more complete and formal definition of WebML can be found at http://webml.org and in [14].

2.1 Data Design

2.1.1 Structural Model

The data content of the site are expressed in terms of relevant entity-relationship model. WebML does not propose yet another language for data modeling but is compatible with classical notations like the E/R model [40], the ODMG object-oriented model [41], and UML class diagrams [42].

The Structural Model Primitives are:

- **Entity**: a class of objects in the application domain

- **Attribute**: a property of an entity
- **Relationship**: a connection between entities
- **IS-A hierarchy**: for classification and grouping

```
<ENTITY id="Album">
  <ATTRIBUTE id="title" type="String"/>
  <ATTRIBUTE id="cover" type="Image"/>
  <ATTRIBUTE id="year" type="Integer"/>
    <RELATIONSHIP id="Album2Artist" to="Artist" inverse="ArtistToAlbum"
          minCard="1" maxCard="1"/>
  <RELATIONSHIP id="Album2Track to="Track" inverse="Track2Album"
          minCard="1" maxCard="N"/>
</ENTITY>

<ENTITY id="Artist">
  <ATTRIBUTE id="firstName" type="String"/>
  <ATTRIBUTE id="lastName" type="String"/>
  <ATTRIBUTE id="birthDate" type="Date"/>
  <ATTRIBUTE id="birthPlace" type="String"/>
  <ATTRIBUTE id="photo" type="Image"/>
  <ATTRIBUTE id="biographicInfo" type="Text"/>
  <RELATIONSHIP id="Artist2Album" to="Album" inverse="Album2Artist"
          minCard="1" maxCard="N"/>
</ENTITY>
```

Figure 2.1: Example of E-R Model

Thus Entities are containers of data elements, and relationships, enable the semantic connection of entities.

2.2 Hypertext Model

2.2.1 Composition Model

After the data design, it's the hypertext design. This aims at providing the hypertext topology of the web site i.e., the organization of content by basic units and links between them, and unit composition within pages. Hypertext design is based on five types of *content units*, shown in Table 2.1.

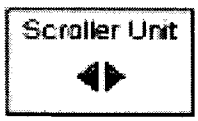
UNIT NAME	VISUAL NOTATION	DESCRIPTION
Data		Shows data about a single entity instance.
Multi-data		Shows data about all entity instance
Index		Shows a list of properties (also called descriptive keys) of a given set of entity instances. A user click on an index entry causes one instance to be selected.
Scroller		Provides commands for scrolling through objects in a set, for example through all the instances in an entity. Scrolling commands allow moving to the first, the last, the previous, and the next element in the set, and cause one instance of the set to be selected and displayed.
Entry		Shows a form with several fields for collecting input by users. This input may consist of conditions used for performing searches over instances of an entity, or of parameters to be supplied to operations, like content updates, login, or generic external operations.

Table 2.1: Content units of the WebML composition model

Links relate units and express navigation on the Web site. They also transform information from unit to another one. Content units can be composed into pages, which represent the abstraction of a self-contained region of the screen (e.g.: delivered to the user as an HTML page).

The goals of the hypertext model are:

- They model at a high level the front-end of a dynamic web application and interactions with the back end business logic and data
- They use a simple visual notation
- Automatic generation of dynamic page templates and data access and

manipulation queries.

After applying the concepts of the hypertext we can achieve the **site view**.

The hypertext is served as **pages** to the users. Each hypertext nodes has **content units** and they are linked via **links**.

2.2.2 Navigational Model

2.2.2.1 Links

2.2.2.1.1 Contextual Links

A contextual link is an oriented connection between two units. The source unit and the target unit. They are rendered by means of a submit button.

The contextual links helps in

- The user move from one place to another
- Transport information from one place to another

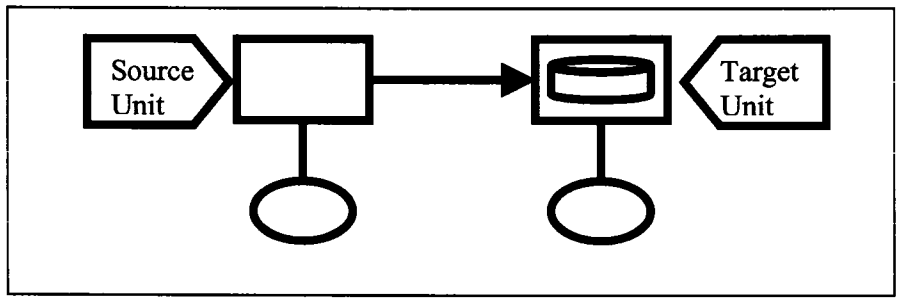


Figure 2.2: Example of Contextual Link

2.2.2.1.2 Non Contextual Links

A non-contextual link is a link between pages no context information is being transferred. The user traverse from one page to another via anchor.

2.2.2.1.3 Automatic Link

An automatic link passes some default information to the destination unit, immediately after the display of the source unit, without the user intervention.

2.2.2.1.4 Transportation Link

A transportation unit has a default content that is passed to the target unit

immediately after the display of the source unit, without user intervention.

2.2.2.2 Page

The page helps in delivering the granularity of units together. Hence it's a container for one or more pieces of information shown to the user at the same time. A page can have sub pages. The user navigates the site made of pages.

2.2.2.2.1 Home Page

This is the main page/first page of the site the user should see. Each site view must contain a home page.

2.2.2.2.2 Landmark Pages:

A landmark page is a globally visible page. A user can jump to them from anywhere in the site view.

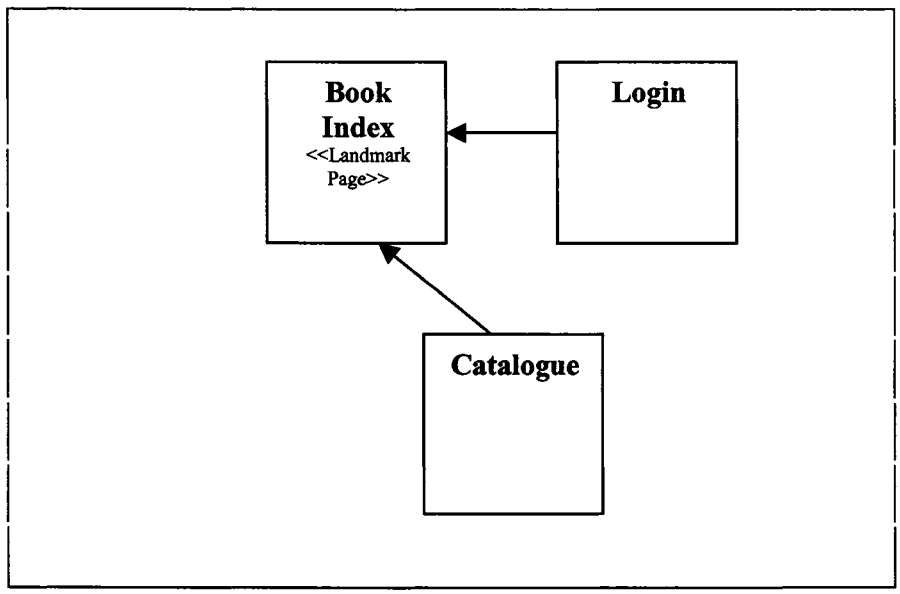


Figure 2.3: Example of a Page

2.2.2.3 Site View

A WebML site view comprises of a set of pages and links, with associated presentation styles, all reachable from a designated home page. All the site views of a given Web application share the same structural schema, which

represents at high level the data sources underlying the site.

Figure 2.4 shows the hypertext fragment of a web site. This illustrates the expressive power of WebML as a model for abstracting and conceptualizing access mechanism of a web site.

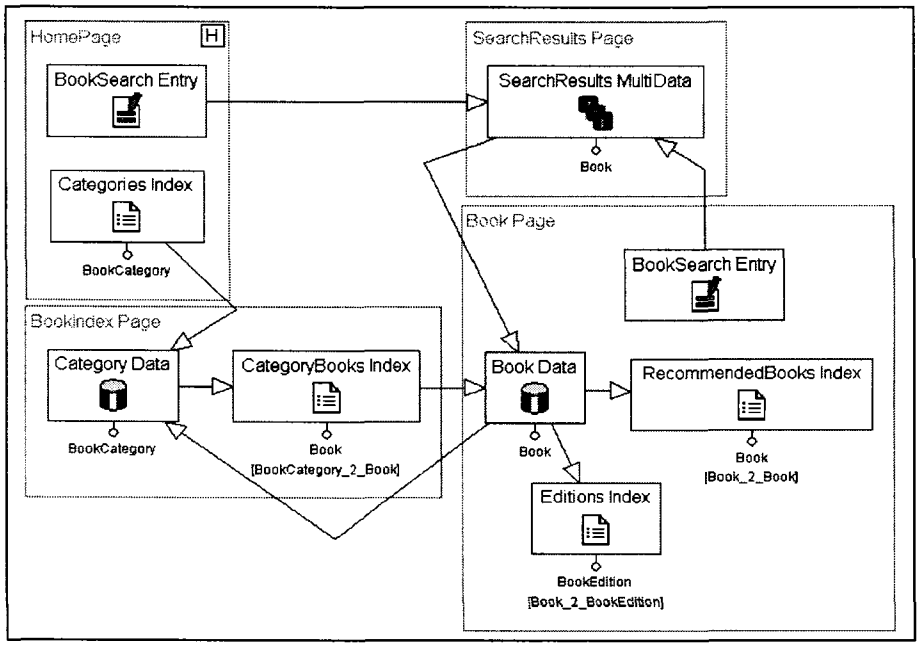


Figure 2.4: Example Of Hypertext Model

2.2.2.4 Presentation Model

It expresses the layout and graphic appearance of pages, independent of the output device and of the rendition language, by means of an abstract XML syntax. Presentation specifications are either page-specific or generic. In the former case, they dictate the presentation of a specific page and include explicit references to page content (e.g., they dictate the layout and the graphic appearance of the title and cover data of albums); in the latter, they are based

on predefined models independent of the specific content of the page and include references to generic content elements (for instance, they dictate the layout and graphic appearance of all attributes of a generic object included in the page).

Figure 2.5 represent the presentation model of the book page of a Web site, which shows the actual rendition on the Web of some conceptual elements included in the Book page of the diagram of Figure 2.5

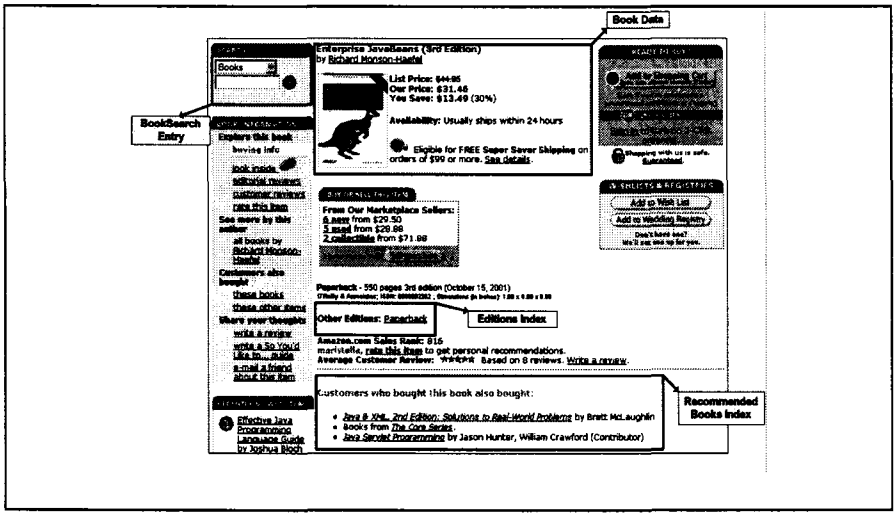


Figure 2.5: Example Of a WebML generated Web site

2.3 Defects Of WebML

Web applications include both navigation elements and business process [36], but WebML treats business process as just another kind of navigation. The problems faced

By the approach of WebML are

- When emulating business process with navigation the problem arises if the designer wants to let the user to navigate to other

pages that are not related to a particular process sequence, the action takes the user away from the process node. Hence, the user gets disoriented and fails to complete the process.

- Another problem is there is no definition if the user leaves the process nodes. Should the process be completed or it has to be resumed when the user again gets back to the process.
- WebML is good for pure navigation and for websites that has low-level business logics. When designing a complex business process it fails.

2.4 Open Issues In WebML

1. WebML [14] is currently being extended to support a new type of pages for performing operations and updating the site content. Work is ongoing on the translation of WebML specifications into WML-based Asp templates, thereby providing evidence that the model-driven approach of WebML is particularly effective in supporting multi-device web sites.

2. Formally defining the semantics of the new introduced constructs, identifying the possible changes to the hypertext computation logic the extensions require for.

3. Extension of WebML proposed in paper [23] would be implemented on WebRatio [25], he current WebML case tool. This will entail both a modification of the site designer component, to enable the model extensions, and of the automatic generation of the

application from the specifications. Changes are not hard, due to the extensibility of the WebRatio architecture.

4. Modeling an extension for WebML (for example a SendMail operation), in the context of the development of a complex workflow application, aimed at supporting the workflow among a large computer company and it's resellers and partners [16].

5. Ability to respond to asynchronous, external events (such as deadlines, errors, exceptions and so on). The only event that can be trapped in WebML is the navigation along the link. A possible way to work around this is to create guards in the system (such as event monitors and time managers) or in the database (such as triggers), devoted to detect asynchronous events and react to them.

6. In general, existing web-design models and methods could be easily extended with a few primitives (e.g. conceptual-level business rules, asynchronous messaging, exception handling, conditional constructs for expressing fork or join conditions), making modeling of web-based workflow management system more effective.

This document represents a WebML [14] features for designing data intensive web applications. Most of the information presented in this document are derived from Webratio [25], a WebML Case tool. In particular, the WebRatio tool suite comprises Site Designer, for editing the WebML specifications of the structural, hypertext, and personalization models; Presentation Designer,

for visually defining presentation style sheets [25]; Site Manager, for site administration and evolution. The architecture is completed by a Template Generator, which transforms WebML specifications into Java Server Page (JSP) templates running on top of relational DBMSs for data storage. Code generation is based on standard XML technology (XSL) and therefore WebRatio can be easily extended to support template generation in more than one markup language and for multiple server-side scripting engines.

Chapter 3: Overview of Web Service Language (WS-CDL)

B2B applications are based on interactions between people and organizations or between organization and organization [24]. Web Services provide a means to deal with these aspects. Web Service Choreography and Orchestration languages allow us to express behavioural policies between involved entities. They can be used to model Business Interactions in a System. In particular, we consider that they can be used not only to describe behavioural rules but also for designing and testing if the involved entities move with system specification.

This chapter is mainly to describe how a workflow language can be used to model business interactions in e-commerce applications. Hence we can use these languages to extend our e-application modeling languages.

3.1 Overview of Web Service Technology

In figure 3.1 we can see the layout of web service architecture. In 2004 W3C working group submitted their latest version their work defined the Web Services Architecture. [26].

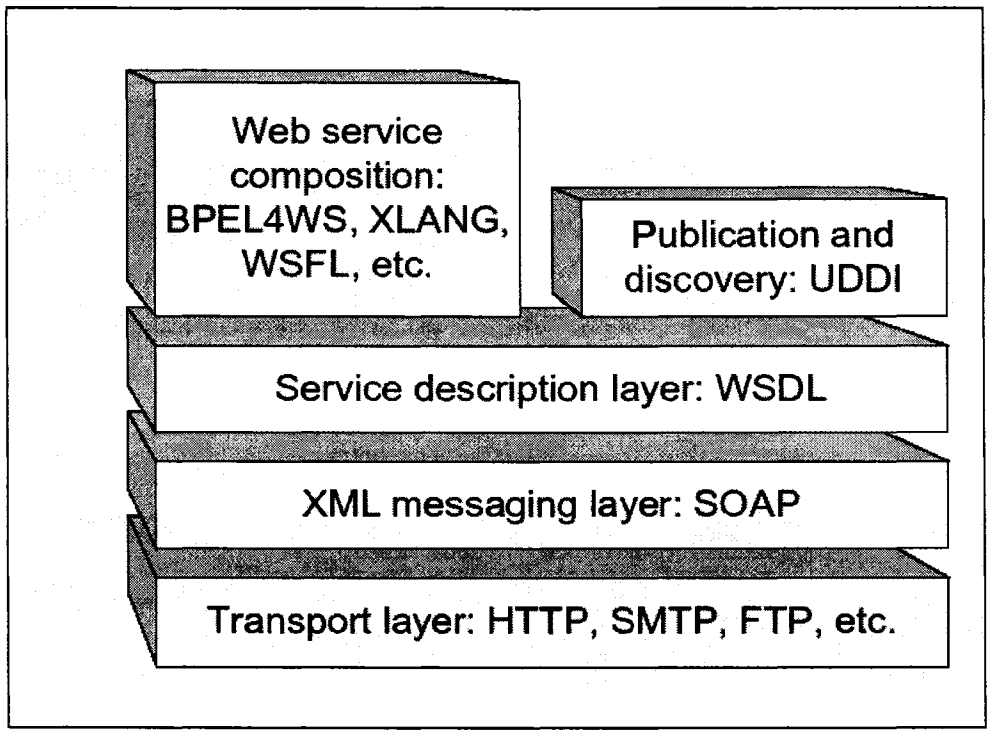


Figure 3.1: Web Service Architecture [26]

The bottom layer is the transport layer, which is the same layer as in network protocol. All the messages and packages are transferred over the network in this layer. HTTP, SMTP and FTP are different network protocols for specific applications, for example, FTP is used to transfer files, and SMTP is used to transfer emails. SOAP [27] (Simple Object Access Protocol) a W3C recommendation published in 2003 is an XML messaging layer which defines the basic format of a message and the basic delivery options independent of programming language, operating system, or platform. It is an XML-based protocol for the exchange of information in a decentralized, distributed environment. WSDL (Web Service Description Language) [28] describes the static interface of a web service. Microsoft and IBM submitted the first draft to the W3C in 2001. It defines the protocol and the message characteristics of end

points. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows for the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types, which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding, and collection of ports defines a service [29].

UDDI (Universal Description Discovery & Integration) is the definition of a set of services supporting the description and discovery of: (1) businesses, organizations, and other web services providers, (2) the web services they make available, and (3) the technical interfaces, which may be used to access those services. UDDI is working as a "Yellow pages" for web services.

Web services composition languages like BPEL4WS [43], WS-CDL [1] build directly on top of WSDL. They both provide and/or use one or more WSDL services. A WSDL service is composed of ports that provide operations. Each operation involves the transfer of messages, and it could be one-way (request or respond) and two-way (request-response, solicit-response). WSDL services and the corresponding operations combine together to provide composed services. To combine such WSDL [28] services a process model is needed to specify the order in which the operations are executed. A web services composition language provides the means to specify such a process model. An

important difference between WSDL and a language like BPEL4WS is the consideration of states. WSDL is almost stateless because the language has no idea about the states in-between operations. The only state notion supported is the state in-between sending and receiving a message in a request-response or solicit-response operation. Any technology supporting a web services composition language will have to record states for processes that are more complex than a simple request-response. Only by recording the state it is possible to determine what should be done, thus enabling long-lived business transactions. In the development of languages like BPEL4WS [43], WSFL [44], XLANG [45] and WS-CDL [1] this is the most important issue to be considered.

3.2 Web Service Work Flow Languages

Web Service composition languages are normally divided in two main divisions namely orchestration languages and choreography languages. Orchestration means the centre of the system through which all the information passes. Whereas Choreography govern the rules of the interactions between the parts involved in the system.

3.2.1 WS-CDL vs. BPEL4WS

WS-CDL [1] aims to constrict the behaviours of the Web Services involved in the system ruling the exchange of their messages instead of BPEL4WS [43] that allows the design of a central entity, which carries out an activity invoking other services. The essential behaviours of choreography and orchestration are highlighted in order to describe the role they could play for designing system based on components. The following is the list of features that list the

differences between BPEL4WS and WS-CDL.

3.2.1.1 Executable processes:

WS-CDL is a descriptive language and does not provide specifications for its execution.

In this sense it is only a descriptive document without any direct computational purpose. On the contrary BPEL4WS assumes the existence of engines able to animate the specifications. A so-called orchestration engine executes BPEL4WS code and it is an essential part of the system, which can invoke interactions and respond to requests [24].

3.2.1.2 Interactions designing

WS-CDL provides a top view of the system focusing on the interactions between the participants. It is a definition of the rules, which governs messages exchanged among the parties involved in the choreography. All the interactions have to be fulfilled between the two entities and there is no notion of a centralized entity, which carries out the activity. On the contrary BPEL4WS is always cantered on the orchestrate engine, which drives all the interactions allowing service synchronization.

3.2.1.3 Activity State:

In WS-CDL the state of the activity is distributed among the entities [24]. BPEL4WS [43], as said before, is centred on the orchestrate engine which is the entity which manages the communications and which stores all the state of the activity it is carrying out.

3.3 Reason to work on WS-CDL

WS-CDL is a better match over BPEL4WS in designing e-commerce

requirements.

- In orchestration it is always the center of the system through which information passes.
- Due to the nature of e-commerce systems, they require more than one orchestration engine running in parallel; this increases the complexity of the system.
- WS-CDL has the potential to aid the design of application via a refinement process starting from the document description.
- WS-CDL is a good candidate to model

Hence we decided to choose WS-CDL to model Web-ML concepts to aid business process design in web modeling methods.

3.4 Overview of WS-CDL

WS-CDL is a language to describe interoperable peer-to-peer collaboration between parties. It is not an "executable business process language" or an implementation language [1]. It does not depend on a specific business process implementation language. Thus, it can specify truly interoperable, collaboration between any type of party regardless of the supporting platform or programming model used by the implementation languages or the hosting environments. Each party adhering to WS-CDL could be implemented using a "Business Process executable language" or by "General purpose programming language". WS-CDL is based on Pi Calculus. Pi Calculus is based on the following concepts

- Automation over a set of actions like Act has four ingredients

- A set of states Q={q0, q1, ...}
- A start state q0
- A set of transitions which are triplets (q,a,q') members of Q x Act x Q
- A subset F of Q called the accepting states

- Theoretically, a business is deterministic. So it complies with the rule that

  For each pair of state and action (q,a) there is at most one transition (q,a,q')

- Reconfiguration: messages can include channel-names
- Bisimulation: The ability of Pi Calculus in supporting channel transmission makes it a good choice for bisimulation. Pi Claculus Bisimulation is considered to specify the behavior of a Web Service and determine whether the implementation is equivalent to the specification.

3.4.1 WS-CDL Model Overview

When two parties have to collaborate they have to establish relationship between them. Their collaboration takes place in a jointly agreed set of ordering and constraint rules. The main purpose of their collaboration between parties is information exchange. WS-CDL Model consists of the following Entities [3].

3.4.1.1 Participant Types, Role Types and Relationship Types

In Choreography information is always exchanged between parties.

3.3.1.1.1 Role Type

This is the behavior a party should exhibit in order to collaborate with other parties

3.4.1.1.2 Relationship type

This describes the mutual commitment that must be made between two parties for them to collaborate successfully.

3.4.1.1.3 Participant Type

This is nothing but grouping together those parts of the observable behavior that must be implemented by the same logical entity or organization.

3.4.1.2 Information Types, Variables and Tokens

3.4.1.2.1 Variables

Variables contain information about objects in collaboration. This information could be like information exchange or the information of role types.

3.4.1.2.2 Token

They are alias, they are used to reference parts of variables.

3.4.1.2.3 Information Type

Both variables and tokens have types that define the structure of a variable or the token reference.

3.4.1.3 Choreography

They define collaborations between interacting parties

3.4.1.4 Choreography Life-line

They express the progress of collaboration. Collaboration is established between parties, then work is performed within it and finally it completes either normally or abnormally

3.4.1.5 Choreography Exception Blocks

This specifies what additional interactions should occur when choreography behaves in an abnormal way.

3.4.1.6 Choreography Finalize Blocks

When a choreography instance has been successfully completed, it goes to the finalization actions that conform, cancel or otherwise modify the effects of its completed actions.

3.4.1.7 Channels

This is the point of collaboration between parties by specifying where and how information is exchanged.

3.4.1.8 Work Unit

They are the constraints that must be fulfilled for making progress.

3.4.1.9 Activities

They are lowest level components of the Choreography that perform the actual work.

3.4.1.10 Ordering Structures

Ordering Structures combine activities with other Ordering Structures in a nested structure to express the ordering conditions in which information within the Choreography is exchanged.

3.4.2 Overview of WS-CDL Package:

Below is the details description of the WS-CDL package details:

```
<Package  Name="ncname"  Author="xsd: string"?
   version="xsd:string"?
TargetNamespace="uri"xmlns="http://www.w3.org/2004/12/ws-
chor/cdl">
   InformationType*
   Token*
   TokenLocator*
   RoleType*
   RelationshipType*
   participantType*
   ChannelType*
   Choreography-Notation*
</Package>
```

Figure 3.2: Example of WS-CDL Package [3]

3.4.3 Detail Definition of Package Level Constructs

3.4.3.1 Role Type

They are the observable behavior a party exhibits in order to collaborate with other parties. For example the "Buyer" Role Type is associated with purchasing of goods or services and the "Supplier" Role Type is associated with providing those goods or services for a fee.

Syntax:

```
<RoleType name="ncname">
   <Behavior name="ncname" interface="qname"? />+
</roleType>
```

Figure 3.3: Example of a Role type

Here the roletype element has an attribute name for specifying a distinct name for each roletype. Within the roleType element, the behavior element specifies a subset of the observable behavior a party exhibited. A Role Type MUST contains one or more behavior elements.

3.4.3.2 Relationship Types

A Relationship Type identifies the Role Types and Behaviors. For example

- A "Logistics Provider" Relationship Type between the Supplier and the Shipper, and
- A "Goods Delivery" Relationship Type between the Buyer and the Shipper

Syntax

```
<relationshipType name="ncname">
   <role type="qname" behavior="list of ncname"? />
   <role type="qname" behavior="list of ncname"? />
</relationshipType>
```

Figure 3.4: Example of a Relationship type

The attribute name is used for specifying a distinct name for each relationshipType. A relationshipType element MUST have exactly two Role Types defined.

3.4.3.3 Participant Types

It groups together the role types that must be implemented by the same logical schema.

Syntax

```
<participantType name="ncname">
   <role type="qname" />+
</participantType>
```

Figure 3.5: Example of participant type

Example

Let's take the following example. A specific collaboration requires the following relationships types, role types.

Role types

- Buyer
- Shipper
- SellerforBuyer
- SellerforShipper
- Shipper

Relationship Type

1. Buyer-Seller: This involves 2 role types Buyer and SellerforBuyer

2. Seller-Shipper: This involves 2 role types Shipper and SellerforShipper

```
<participantType name="Broker">
   <role type="tns:SellerForBuyer" />
   <role type="tns:SellerForShipper" />
</participantType>
```

Figure 3.6: Example of choreography

Where the Participant Type "Broker" which also implements the "SellerForShipper" Role Type belonging to a "Seller-Shipper" Relationship Type implements the "SellerForBuyer" Role Type belonging to a "Buyer-Seller" Relationship Type.

3.4.3.4 Channel Types

A channel is the point of collaboration between the parties. The channel information is passed between parties for subsequent collaborations.

- A channel must describe the "Role Type" and the reference type of a party.
- A channel may be passed around from one party to another in an information exchange.

Syntax:

```
<channelType  name="ncname"
              usage="once"|"unlimited"?
              action="request-respond"|"request"|"respond"? >
   <passing  channel="qname"
             action="request-respond"|"request"|"respond"?
             new="true"|"false"? />*
   <role  type="qname"  behavior="ncname"? />
        <reference>
      <token name="qname"/>
        </reference>
        <identity>
      <token name="qname"/>+
        </identity>?
</channelType>
```

Figure 3.7: Example of channelType

The attribute name is for giving a distinct name to the channel. All the other optional attributes within the channel type entity are usage: to specify if the channel usage is just once or unlimited. And the action is a request or a response to a request or request-response. The optional element `passing` describes the Channel Type(s) of the Channel(s) that are passed, from one party to another.

Each channel should have a role type that it should refer to. Reference element is required to identify for dynamically determining where and how to send or receive information to or into the party. The OPTIONAL element identity MAY be used for identifying an instance of an entity implementing the behavior of a party and for identifying a logical conversation between parties.

Example:

```
<channelType name="RetailerChannel">
    <passing channel="ConsumerChannel" action="request" />
    <role type="tns:Retailer" behavior="retailerForConsumer"/>
    <reference>
    <token name="tns:retailerRef"/>
    </reference>
    <identity>
    <token name="tns:purchaseOrderID"/>
    </identity>
</channelType>
```

Figure 3.8: Example of ChannelType

"RetailerChannel" that realizes a point of collaboration with a Retailer. The Channel Type identifies the Role Type of the Retailer as the "Retailer". The information for locating the Retailer is specified in the reference element, whereas the instance of a process implementing the Retailer is identified for correlation purposes using the identity element. The element passing allows only a Channel of "ConsumerChannel" Type to be passed in a request information exchange through a Channel of "RetailerChannel" Type.

3.4.3.5 Information Type:

Information Type is used to describe the type of information used within choreography.

Syntax:

```
<informationType name="ncname"
                 type="qname"?|element="qname"?
                 exceptionType="true"|"false"? />
</informationType>
```

Figure 3.9: Example Of InformationType

The attribute `name` is used for specifying a distinct name for each informationType element declared within a Choreography Package. The OPTIONAL attributes `type` and `element` describe the type of information used within Choreography as a WSDL 1.1[28] Message Type.

Example:

Example1:
```
The informationType "purchaseOrder" refers to the WSDL 1.1
Message type "pns:purchaseOrderMessage"
<informationType name='purchaseOrder"
type="pns:purchaseOrderMessage"/>
```
Example2:
```
The informationType "customerAddress" refers to the WSDL 2.0
Schema element "cns:CustomerAddress"
<informationType name="customerAddress"
element="cns:CustomerAddress"/>
```
Example 3:
```
The informationType "intType" refers to the XML Schema type
"xsd:int"
<informationType name="intType" type="xsd:int"/>
```
Example 4:
```
The informationType "OutOfStockExceptionType" is of type
Exception Type and refers to the WSDL 2.0 fault name
"cwns:OutOfStockExceptionType"
<informationType name="OutOfStockExceptionType"
 type="cwns:OutOfStockExceptionType" exceptionType="true"/>
```

Figure 3.10: Various Examples of InformationType

3.4.3.6 Variable

Variable is used to capture information about objects in choreography. The syntax of variable is variable name, and type.

3.4.3.7 Tokens

Token is an alias of a variable. Variable contain values whereas Tokens contain only a specific value of importance. All Tokens must have information type. For example "Order ID" could be of Information type int, counter, alphanumeric.

Syntax:

```
<token  name="ncname"  informationType="qname" />
```

Figure 3.11: Example of a Token

3.4.3.8 Token Locator:

They provide a query mechanism to locate the Token in the Choreography document.

Syntax:

```
<tokenLocator  tokenName="qname"
               informationType="qname"
               part="ncname"?
               query="XPath-expression" />
```

Figure 3.12: Example of a TokenLocator

3.4.4 Choreography

Choreography is re-usable. The top level Choreography is called the Root Choreography that does not share its information with other top level Choreographies. The re-usable choreographies are the one at the low level. They also called enclosed choreographies [1].

Requirements of Choreography [1][3]:

- Choreography MUST contain at least one Relationship Type
- Choreography MUST contain an *Activity-Notation*. The Activity-Notation specifies the actions of the Choreography that perform the actual work.

- Choreography can recover from exceptional conditions by defining one *Exception Block*
- An enclosed Choreography that has successfully completed MAY need to provide finalization actions that confirm, cancel or otherwise modify the effects of its completed actions
- Choreography can also be coordinated. *Choreography Coordination* guarantees that all involved Roles agree on how the Choreography ended.

Chapter 4: Previous/ Related Work

This chapter reviews some of the previous and current work done in our thesis area. Our thesis is directly related to Web Service composition languages, New issues in web conceptual modeling languages like WebML and ongoing work in WebML.

4.1 Web Service Composition Languages

Processes being built today need the business agility to quickly adapt to customer needs and market conditions [8]. This would include incorporating new customers, partners, or suppliers used in a process. A single standard is desired that can manage both EAI and B2B interactions involving web services. Web services orchestration or choreography is about providing an open, standard-based approach for connecting web services together to create higher-level business processes. In general they are called composition languages for web services. Standards such as BPEL4WS [42], WS-CDL [1], XLANG [44], WSFL [43] etc are used to express the logic of composite Web Services.

In July 2002 Jean-Jacques Dubray released the key elements for choreography. The key elements of choreography are [3]:

Composition Features

- Ability to define choreography
- Definition of choreography external observable behaviors
- Life cycle management (e.g. Creation, Termination etc.)
- Message passing interaction between services (e.g. receive, invoke etc)

- Behavior definition (e.g. Sequencing, looping, concurrent execution etc)
- Scooping rules
- Activity

Association

- Linkage between Web Services
- References to Web Services

Message Exchange

- Conversation management
- Correlation and their life cycle management

State Management

The two main aspects WS Choreography are:

Process interface-Interact with other Web Services

Process Execution-Internal Behavior Web Services

4.1.1 Composition Languages In Supporting E-Commerce Applications

A broad spectrum of electronic commerce applications is currently available on the Web, providing services in almost any area one can think of [45]. These new trends in Electronic commerce (e-commerce) would be involving lot more complex business process than the traditional sites. Electronic commerce applications need technologies to support the business processes [46]. It requires a specification of business processes with a business interaction and connection-oriented perspective in order to compose them from the existing assets of the enterprise, partners, and suppliers.

In 2004 [Mario Bratvetti] suggested that web service composition languages provide a means to deal with these aspects. Web service composition languages like WS-CDL [1] and BPEL4WS [42] allow us to express behavior policies between the involved entities; it is considered that they can be used not only for describing behavioral rules or business rules but also for designing and testing whether the involved entities move in according to system specifications.

BPEL4WS [42] can be executed using an orchestration engine and they introduce invocation, concurrent and synchronization primitives for flowing information among different Web Services and carry out a main activity. Each process would need one orchestra engine to execute it. Due to the nature of e-commerce systems they require more than one orchestration engine running in parallel, thus increasing the complexity of the system. Therefore, a top view system description is necessary in order to fulfill and program correctly the different engines involved.

WS-CDL, which is a draft document of W3C, it fixes the rules of the interactions between the parts involved in the system [46]. It has the potentiality on one side to aid the design of applications via a refinement process starting from the WS-CDL description, and on the other side to allow the verification of the compatibility of already available services willing to participate in choreography described in WS-CDL. Choreography is a sort of contract between the parts (which could be companies or single applications)

in order to rule their interactions. So, we can imagine that each part will design their own application and then verify its correctness exploiting the sentences of the WS-CDL document drawn up.

In 2003 Jim Amsden of IBM and others introduce a UML profile [58] which supports modeling with a set of semantic constructs for automated business processes that allow BPEL4WS processes to be modeled using UML. It takes processes defined in the Unified Modeling Language (UML) and generates the corresponding BPEL and WSDL files to implement that process. And in 2003 Koichi TERAI of Japan [59] and his fellow Researchers proposed a framework for Web Services coordination based on business models. They have used J2EE platform, this provides interface to the client application and EJB container managing the business logic.

4.2 New Issues In Web Conceptual Modeling Languages

Designing and maintaining Web applications is one of the major challenges for the software industry of the year 2000 [6]. S. Ceri, P. Fraternali and A. Bongio presented a paper on Web Modeling Language (WebML) in WWW9 Conference, Amsterdam on May 2000. According to them WebML is a notation for specifying complex Web sites at the conceptual level. WebML enables a high-level description of a Web site under distinct orthogonal dimensions: its data content (structural model), the pages that compose it (composition model), the topology of links between pages (navigation model), the layout and graphic requirements for page rendering (presentation model), and the customization features for one-to-one content delivery (personalization

model). All the concepts of WebML are associated with a graphic notation and a textual XML syntax. WebML specifications are independent of both the client-side language used for delivering the application to users, and of the server-side platform used to bind data to pages. They can be effectively used to produce a site view implementation in a specific technological setting. WebML is built on several proposals for hypermedia and web design language like OOHDM [13] etc.

OOHDM [13], a methodology for designing hypertext share with WebML the orthogonal design dimensions like conceptual design, navigational design, interface design and implementation. In 1999 J. Conallen proposed how web applications can be modeled using UML. To support web application modeling using UML, he suggested a formal extension mechanism to allow practitioner to extend the semantics of the UML. UML models server side aspects of a web page with one class and the client side aspect with another, they both can be distinguished by using the UML extension mechanism [47]. The main disadvantage of UML in modeling web applications is that they provide low-level abstractions; it has only syntax and no specific web semantics and diagrams of realistic web applications tend to become quickly unmanageable because web pages are not object-oriented [48].

These web-modeling languages have some modeling problems especially the way they model business processes [11]. Hans Albrecht Schmid and his co-workers in 2004 submitted a paper in IEEE, this paper stated web application

design methods focus mainly in hypermedia-based navigation and neglect business processes. And they treat business process as a kind of navigation. This would result in some design problems; usability problems and execution of this business process will lead to some erroneous results. They introduced business processes as "first class citizens" in the modeling and design of Web applications. They extended the Object-Oriented Hypermedia Design Method (OOHDM) by processes so that it allows a clear specification and easy design of Web applications embodying business processes.

Earlier web sites were mainly read only applications that fetch huge amounts of information. Ever since the advent of e-commerce web based systems like online booking system, online auction system, the requirements of web applications have changed they are not just simple information retrieval web applications but they are hypermedia based distributed applications [35]; they blend navigation and business processes together. This new coupling raises a number of novel design issues like how to blend the two discrete elements together [49].

In 2000 L.Baresi demonstrated the new requirements of these e-applications. From the example http://cdnow.com the author demonstrated the new requirements of these process oriented web sites, which includes; user while navigating between catalogues of records can filter the products of interest, they can bookmark these products and include them in the shopping bag. They can inspect the content of the bag by navigating from bag to selected records;

they can exclude some of the products previously chosen and evaluate the cost and by providing additional information can complete the buying transaction [49]. Conceptual modeling of this web application is not just the union of two activities performed in isolation rather it is an integration of the two facets of design (navigation/hypermedia and operation). When integrating operation and navigation; designers often face problems; they need to answer questions like [49]: How do information structures and navigation structures support operations? How do operations affect information and navigation structures? How do navigation and operations interface? And how are the user tasks related to both navigation and operations? This addition of services to the web applications should be addressed during the design to deliver quality web applications [50]. G. Denaro and his co-workers proposed a new reference model for web applications. Because if applications are poorly modeled; if they emulate business process as a form of navigation they will generate erroneous results. This can be demonstrated by an example. The user wishes to buy some music CD's and browses through the catalogue to select what he wants to buy. Figure 4.1 shows simple browsing.

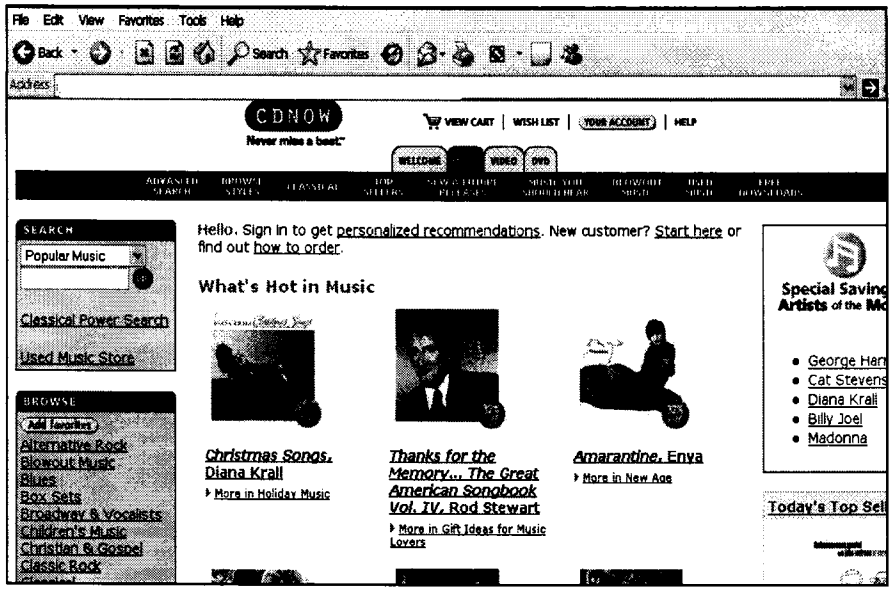


Fig 4.1: Simple browsing music catalogue

He chooses Diana Krall "*Christmas songs*" Audio CD and add's them to the shopping cart. After choosing the first CD the application presents to the user the shopping cart with selected items:

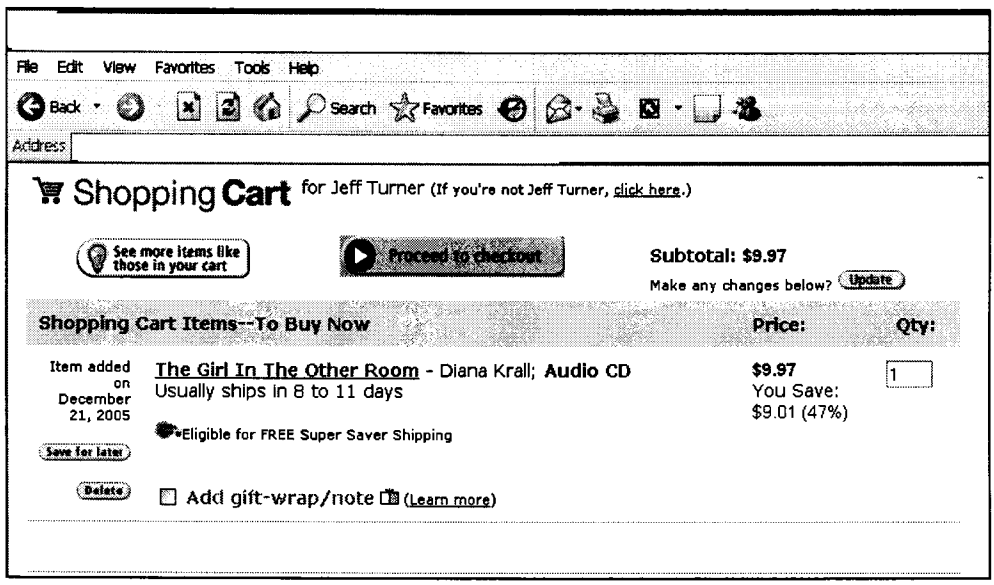


Fig 4.2 shopping cart with selected items

If the user decided to add one more Audio CD say " The Girl In The Other Room" by Diana krall, he could use add to cart in the page shown in fig 4.2 and check it out for purchase. This skeleton makes the user move to the shopping cart page which now contains 2 Audio CD's as shown in the figure 4.3

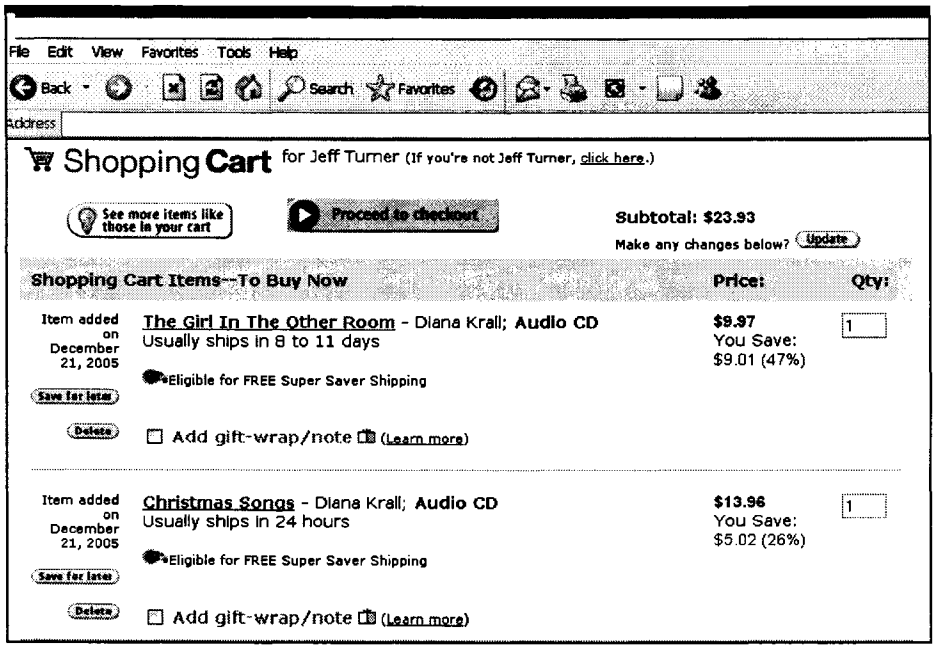


Fig 4.3: Shopping cart with two selected items

Thinking of the price, the user changes his mind and decides that he does not want to buy the second Audio CD. The rigorous user would delete the CD from the shopping cart, using the delete button and then he would checkout the shopping cart with just one product. In contrast a "free navigator" [50] would roll back to previous shopping cart as shown in figure 4.2 and then he would check out this page/cart. He now ends up in buying two Audio CD's instead of one because the process state is still holds 2 CD's. Thus my emulating business process along with navigation the process flow is lost.

Many researchers have contributed for extending conceptual modeling languages like WebML, OOHDM, W2000 etc to support modeling business processes.

4.3 Research Contribution In Extending Web Conceptual Modeling Languages

In 2000 [A. Bongio, S. Ceri, P. Fraternali and A. Maurino.Modeling] showed web-modeling abstractions, which integrate data-entry and operation invocation in WebML. They extended the navigation model one for operation activation (OK LINK) and one for operation failure (KO LINK). Thus WebML was extended with data entry and operation units, for gathering information from clients and invoking arbitrary operations. Adaptive web is a new research area. It addresses the personalization of the web based on each users experience. Stefano Ceri proposed a high-level model based on WebML (web modeling language) for the specification of web applications which takes in account all possible manners a user could interact with the application for supplying contents and for gathering data. For this purpose an Event-Condition-Action (ECA) model was proposed. This model captured arbitrary number of clicking behaviors in a web application.

P. Fraternali, M. Brambilla in 2003 at SEBD presented the WebML approach to integrate traditional data-intensive Web applications with remote service invocation and workflow capabilities. Two sets of new WebML primitives were proposed: (i) for describing Web service publication and consumption (ii) for describing business process implementation within Web applications [51].

In 2003 S. Ceri, M. Brambilla, P. Fraternali and S. Comai extended a declarative model and language for specifying data intensive web applications in order to model complex interactions between applications and remote processes [52]. Their model is based on WebML and implemented in WebRatio a WebML case tool.

M. Brambilla and his co-workers at Sigmod industrial '05 presented how data-intensive and process intensive web applications can be integrated through web services [53]. They have exploited WebML as the conceptual modeling tool for model verification and visual data marshalling and automatic code generation. Thus, this applied method is based on a declarative model for specifying data-intensive web applications that enact complex interactions, driven by users, with remote process implemented as services. P. Fraternali and his co-workers in 2003 addressed a model-driven development of Web applications that integrate hyper textual navigation, content publishing and management, and interaction with remote Web Services [54]. Their proposed approach relies on an extension of the Web Modeling Language (WebML), a visual notation for the design of data-intensive web applications, with primitives for capturing various forms of interaction with Web services, including one-way and request-response operations, asynchronous messaging, and long running conversations.

There are some architectural issues rising from the integration of data-intensive web applications and web services [54]. Dr. S. Ceri and his co-workers contributed on how WebML could be extended to integrate web

services that involve complex processes with web applications that are only data oriented. They developed two orthogonal extensions for the inclusion of web services inside web applications, accompanied by suitable protocols for message exchange, and the empowerment of Web modeling primitives with workflow capabilities. The combined use of these two features gives WebML enough expressive power for specifying complex Web service interactions [55].

N. D'Elia and his co-workers in 2004 demonstrated that new primitives must be put in place to implement workflows describing business processes in web modelling [56]. They proposed workflow-enabling primitives for Web applications, and a high level approach to the management of exceptions that occurs during execution of processes. They also presented a classification of exceptions that can occur inside workflow-based Web applications, and recovery policies to retrieve coherent status and data after an exception. Marco Brambilla in 2003 at the international conference on conceptual modeling [57] presented a pragmatic approach to incorporate classical process modeling primitives within a web-modeling framework. He applied his methodologies to an industrial case, the Acer Business Portal. They proposed an extension of the WebML data model with process metadata and WebML hypertext model with process enactment primitives. The modeling approach he proposed was implemented using WebML case tool the WebRatio.

Chapter 5: Our Approach & Design

This section presents our approach to the problem of business process emulation.

5.1 Problem Statement

E-commerce modeling languages fail to model complex business process. Due to poor design, the resulting code mostly do not match the requirements or they generate erroneous results.

5.2 Hypothesis

Web Service composition languages could be used to improve this behaviour.

5.3 Thesis Statement

We have three main goals in our thesis work.

- Modeling business process using web services peer-peer language.
- Propose a model using WS-CDL, WebML to overcome the problems of business process emulation.
- To analyze the ability of this newly released language, which motivates us for further research in finding a permanent solution to the above-mentioned problem.

5.4 Development Process of Web Application

In the development process of a web applications the following steps are usually followed. Figure 5.1 shows the development process in a web application.

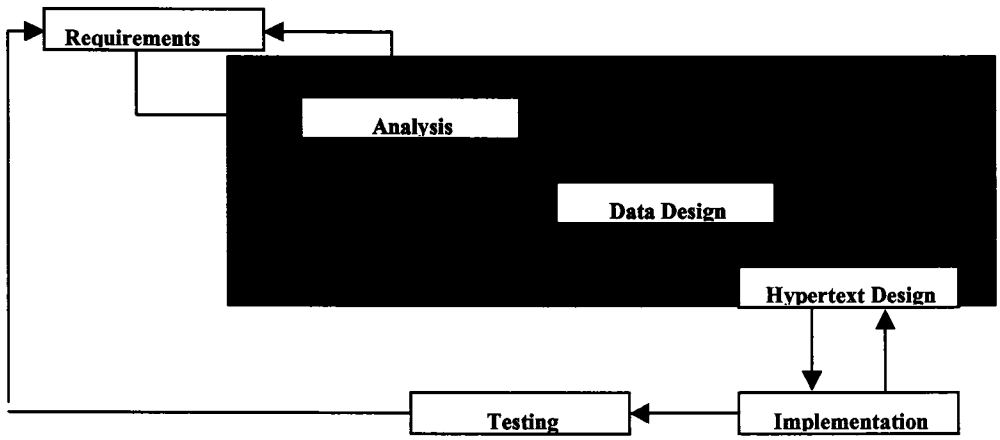


Fig 5.1: Phases in the development of our e-commerce applications

For some of these activities, we adopt widely consolidated standards and notations. In particular user groups and use cases were modeled in UML. The data design and hypertext design in WebML. We used UML sequence diagrams to model functional requirements. These aspects are consolidated and addressed only to show how conceptual design space is extended to adopt complex functional requirements and the system output matches the requirements.

5.5 Overview of Our Model

Our Model can be explained using a Case Study. Our Case Study is a running example called the "Online Bargain Store". This application aims at providing online bargain and purchase of an item. This system is said to compose of the buyers, sellers, shippers, credit check agency etc. The ultimate goal of this system is to allow flexible communication between the buyer and seller especially when they bargain for a product using different data exchange

formats, communication and technological support. This whole e-business site would involve and serve various user groups including the site managers, administrators, sellers, buyers, shipping agency, credit check agency etc. But in our research we consider or have developed only a small subset of the real requirements to test our thesis statement. This portal must be able to make buyers and sellers negotiate on the price and settle with a final value price. The navigational aspects are captured in WebML and the business requirements in WS-CDL.

Algorithmic Approach to Our Model

1.Identify Business Process from functional requirements.

2. Identify the Actors/ Use Cases to model the process

3. Map the actors to choreography roles, participants

4. Model the Use Cases in Choreography activities, which perform the actual execution of a process

5. Each Role executes a specific activity & Define the choreography layer with business rules

6. Parse the Scenario file/ Test Cases to verify and validate these business rules defined in choreography & Verify the Choreography Channel because information is sent and received via a channel

7. This will test the choreography layer for control flow and information flow

8. Based on the outcome of the system further actions are carried out.

5.5.1 Requirements Specification

Requirements collection identifies the general picture of the application [30].

At the end of the collection phase we were able to identify the main business actors that will use this system, the use cases to be supported and the functional requirements and constrain required.

5.5.1.1 Requirements Collection

1. The customer interacts with the seller to determine the price of a product.
2. They bargain on the price
3. When the price is accepted the customer orders the goods
4. Seller requests the credit check to verify his credit card information
5. Once approved, the seller requests a delivery to the shipper
6. Shipper communicates direct with the buyer and informs the buyer of the delivery details.

5.5.1.2 Identification of User Groups

The first objective of requirements collection is to establish who the users are and then cluster them into groups characterized by homogeneous goals and behaviours [30]. Each user is associated to a distinct site view. In our case we have identified two groups of actors one internal to the system and the other external to the system. Fig 5.2 shows the user taxonomy of the application.

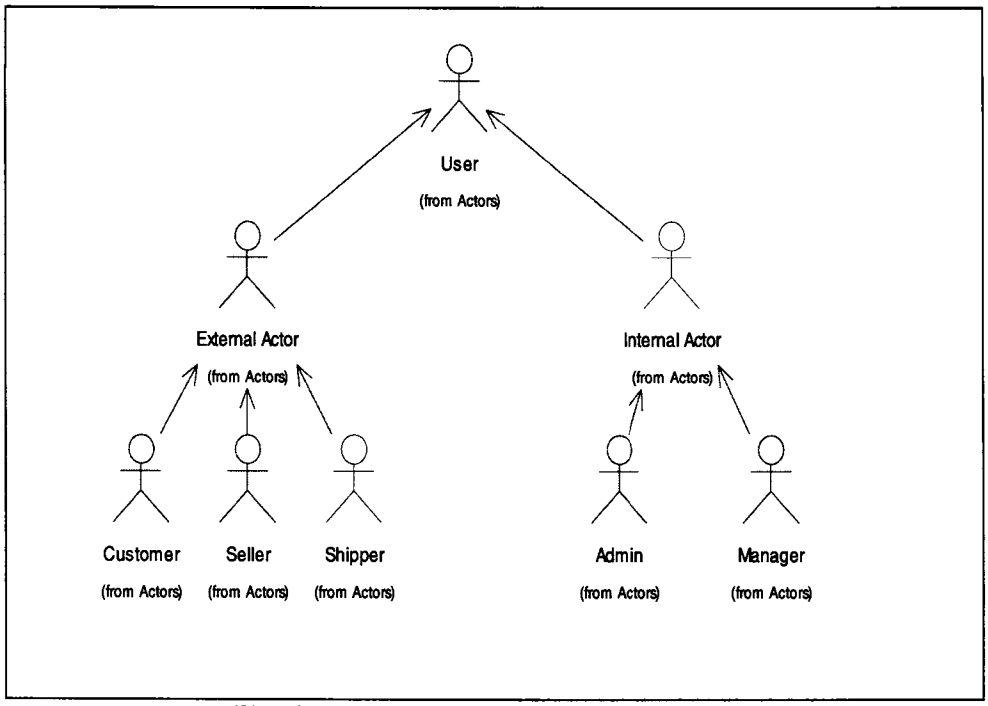


Fig 5.2: User Group Taxonomy of the application

5.5.1.3 Use Case Specification

A use case is a unit of interaction with the application by users of a given group [30]. From the requirements the following use cases are identified. Login, catalogue browse, purchase, add new items, modify items, add a new category and many more.

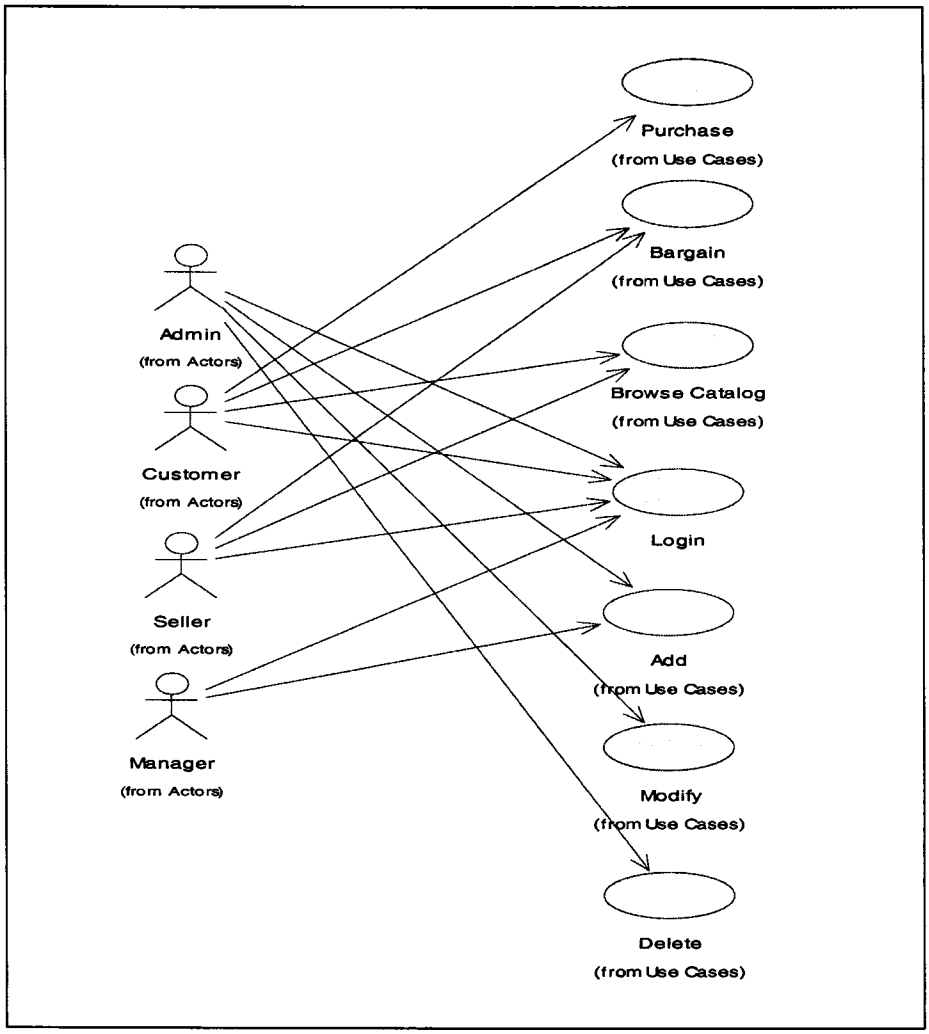


Figure 5.3: System Use Case Specification

For research purpose we have taken into consideration only specific requirements. Each and every use case is associated with a specification sheet, which includes elements like name, definition, pre-condition; the condition that must be satisfied before performing the use case, post- condition; the condition that must be true after performing the use case, work flow, the steps to be performed for successful execution of use case.

5.5.1.4 Process Requirements

Process requirements state the structure of a process and how the users of the

application execute the function. We make use of the UML activity diagrams to enumerate the relevant process and to express the flow of activities within a specific use case.

According to the work flow model a process and can be internally structured using a variety of constructs [31]: sequence of activities, AND split, AND-joins, OR-splits, OR-joins, iterations for repeating the execution of one or more activities, pre- and post-conditions. Figure 5.4 shows the process model specifying the way in which the bargain process takes place.

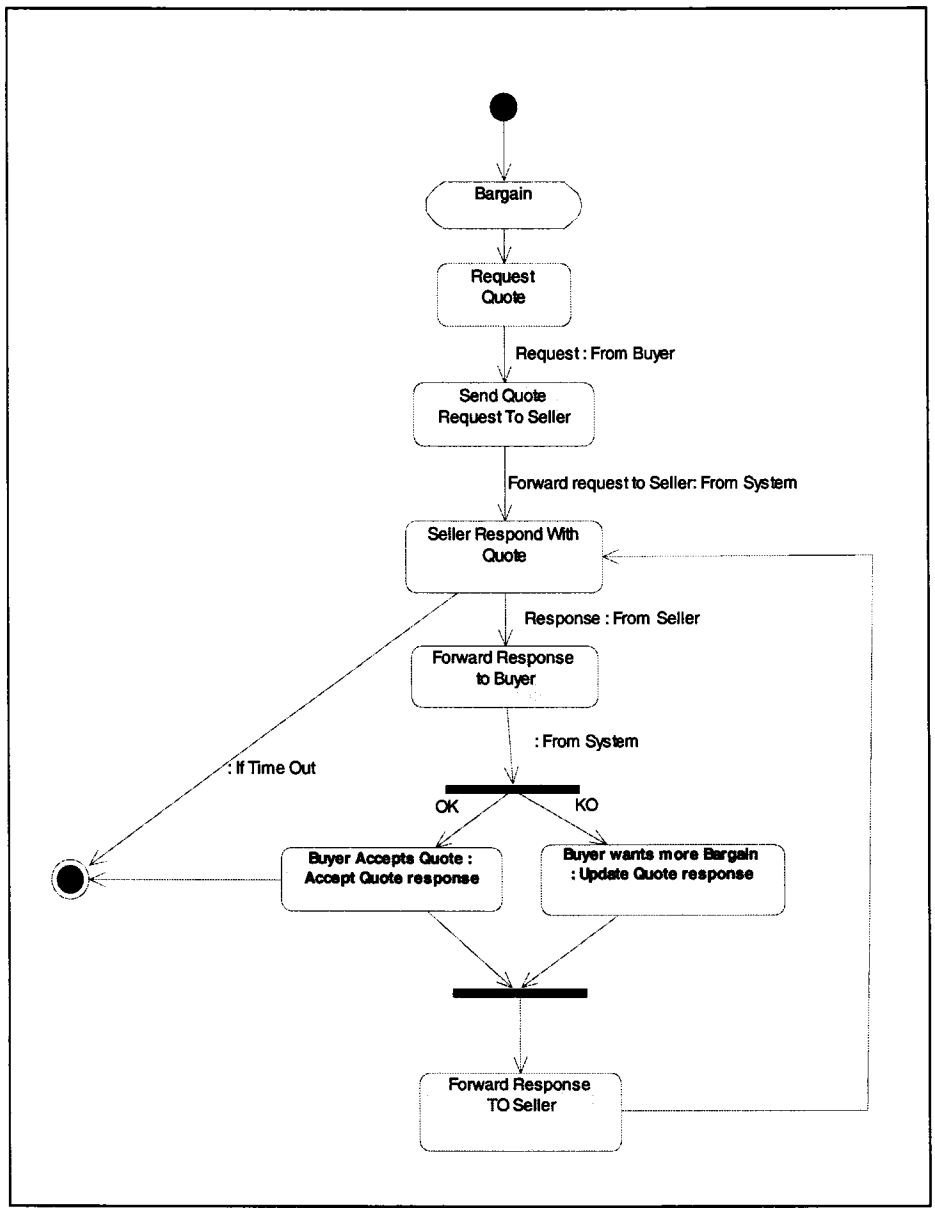


Figure 5.4 The process model of the Bargain Process

The buyer browses for products. Once he identifies his products of interest he engages in a bargain with the seller. This is initiated by sending a request quote for the product he selected to the seller. The seller on the other hand responds for quote request. If the buyer is happy with the quote specified by

the seller he responds with accept quote response. Else he sends update quote response. This process is time stamped. If the response runs out of time the process ends.

5.5.2 Choreography

As discussed earlier choreography model describes collaboration between services.

A choreography model is supposed to capture interactions, control flow, data flow, message correlation, time constraints, transactional dependencies between parties of collaboration [3] etc. From the requirements we have collected we can identify four services (S).

S1. Buyer/Customer

S2. Seller

S3. Shipper

S4. Credit Check

The "Activities" in the choreography represent the actual business activities. We have shown choreographies in the form of UML sequence diagrams. Figure 5.5 shows all the four services involved in the choreography and the way they send and receive messages.

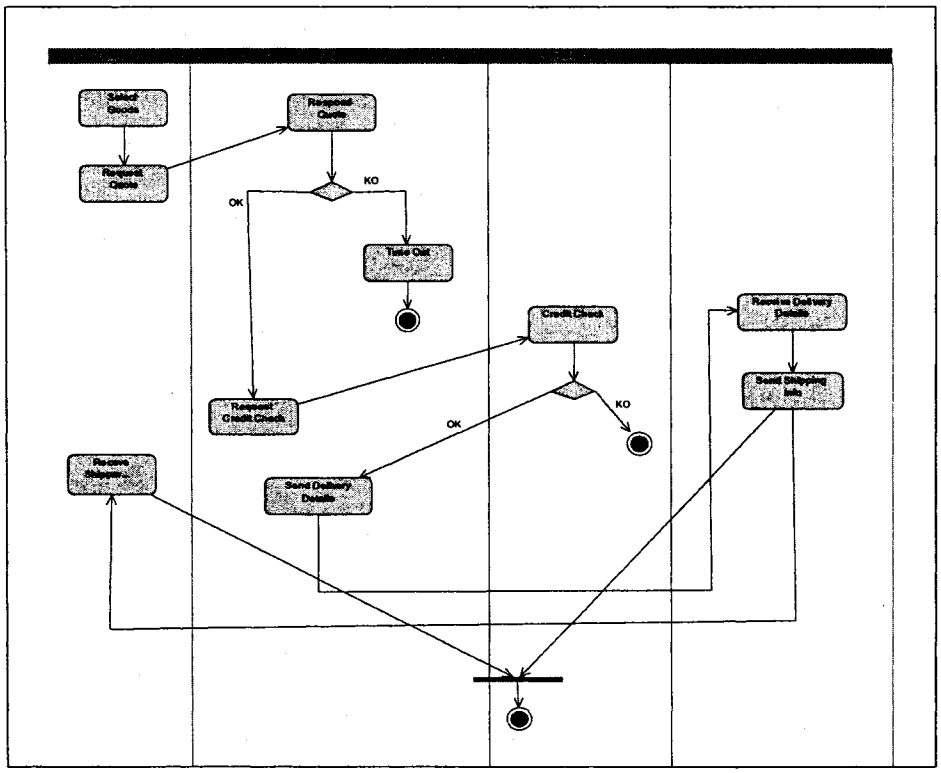


Figure 5.5: Choreography Interaction between parties

Choreography constitutes an agreement between parties. When the buyer identifies his product of interest he requests for quote from the seller. Then the buyer and seller engage in bargain choreography. When bargain process is completed successfully without any time delay, the seller invokes for credit check service. If the credits check service approves the buyer the seller would then invoke shipping service to send delivery details to the buyer. The buyer receives the shipping information and the services ends.

5.5.3 Behavioural Interface

Unlike the choreography the behavioural interface concentrates only on a single participant. In the real world scenario a B2B (Business to Business) interaction a role in choreography might have multiple numbers of behavioural interfaces. For example figure 5.5 shows the behavioural interface of the role

type "buyer"

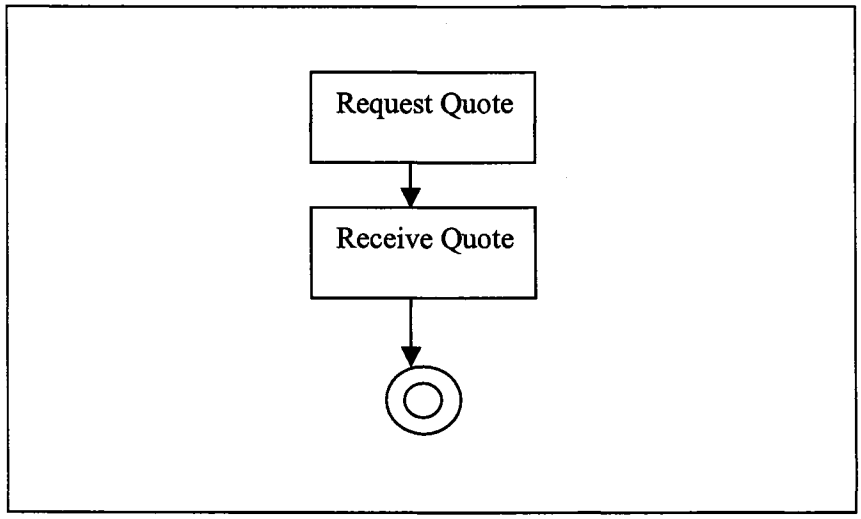


Figure 5.6: Buyer Behavioural Interface

Similarly, the behavioural interface of a seller, shipper, and credit check can also be generated.

5.5.4 Conceptual Design

Conceptual design is nothing but data design and hypertext design. In WebML data design or the Structural Schema is represented by simple E-R Model (Entity-Relationship model).

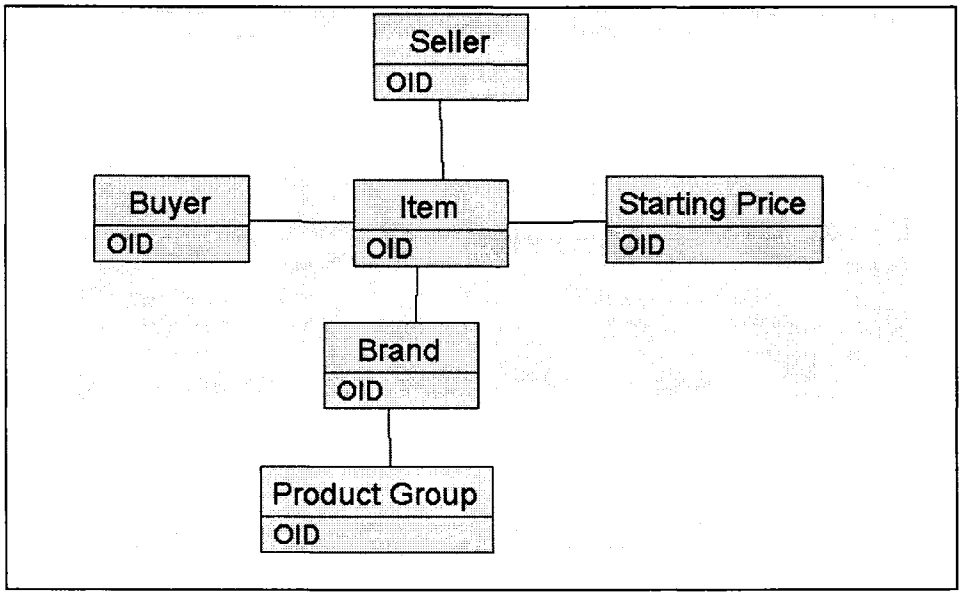


Figure 5.7: Conceptual Design

Each seller is associated with one or more items. Each item is associated with a bargain price. Each specific item is derived from a specific brand and brands are derived from product group.

5.5.4.1 Mapping WebML Structural Schema with choreography

The WebML data model describes the domain objects. In our approach we have taken the meta data model for workflow and have represented it in WS-CDL. Figure 5.8 describes the meta data model for workflow.

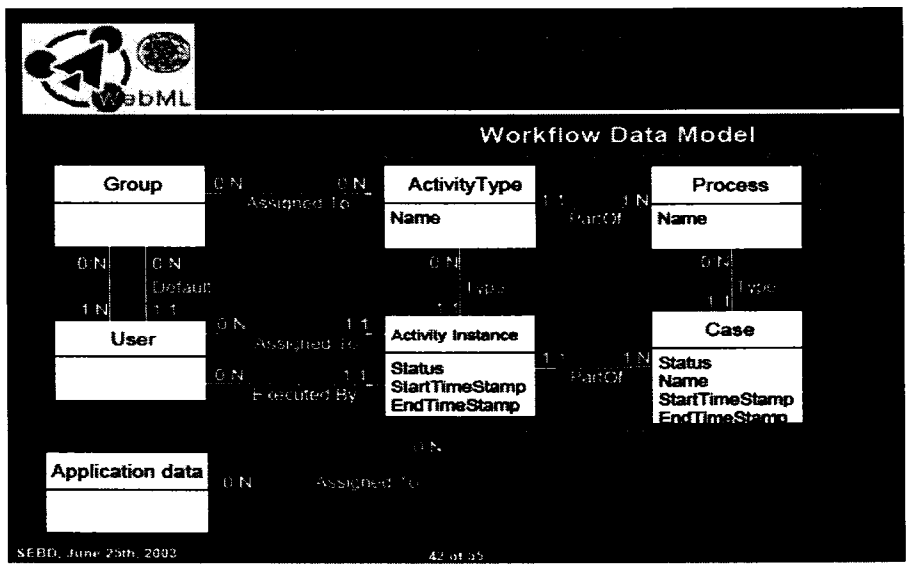


Figure 5.8: Meta-Model for Workflow [Ref: 2]

Each process is associated to an activity type to represent the activities that can be executed in a process. Case denotes the instance of a process and the activity instance is the occurrence of an activity. Each activity state can be active, inactive or completed. Group and user represent the workflow actors.

In our model we have taken this meta-model and have represent the ideas in WS-CDL. Figure 5.9 shows how WS-CDL represent the workflow Meta-

model.

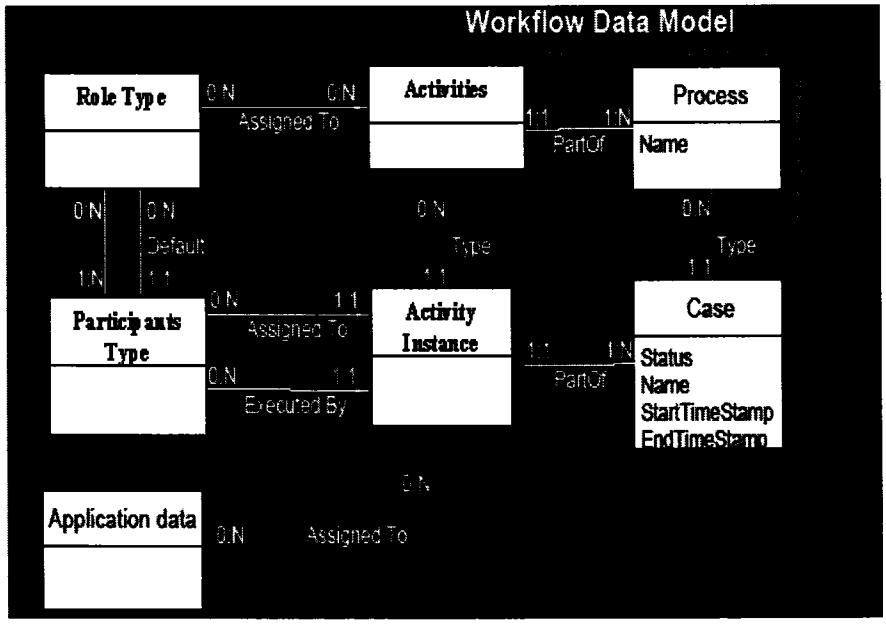


Figure 5.9: WS-CDL Meta Model

A role type enumerates the observable behaviour a party exhibits in order to collaborate with other parties [1]. In our example we say "Buyer", "Seller" as Role Types. Participant type identifies the set of roles implemented by the same logical entity [1]. In other words we mapped participant type as the group in the workflow Meta-model. The relationship type in WS-CDL represents the relationship between the roles. Activities in WS-CDL are the lowest level components that execute the process. Activity instance is derived from activity, which denotes the start and the end of the activity. Fig: 5.10 represent the application data model for the application purchase.

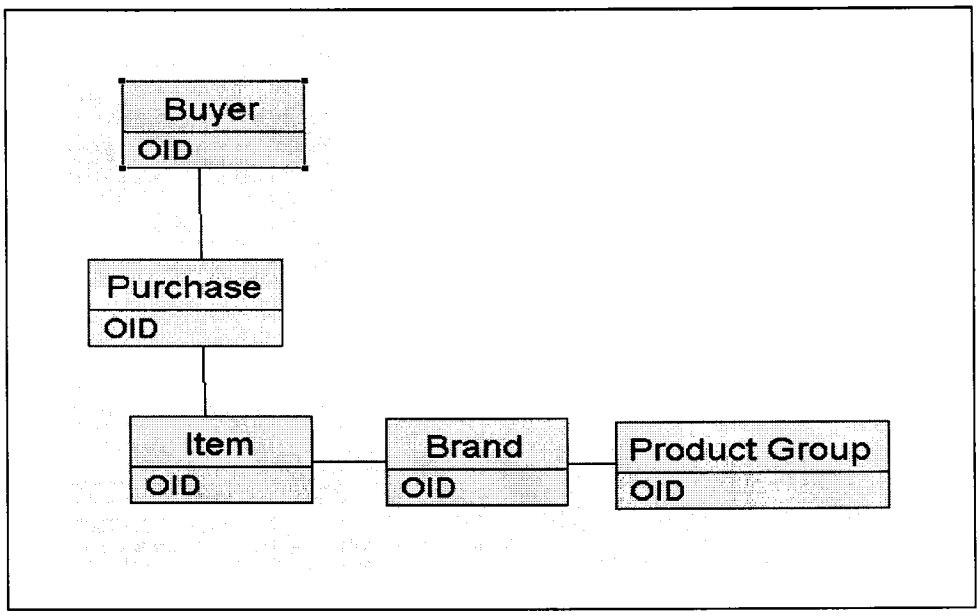


Figure 5.10: Application Model for Purchase Process

The buyer browses the catalogue of products and finds his item of interest. The purchased application has a start time stamp and end time stamp. This is because the buyer has to wait until the seller contacts the credit check service, approves the buyer to buy. Figure 5.11 shows how it is related to the WS-CDL workflow Meta-model.

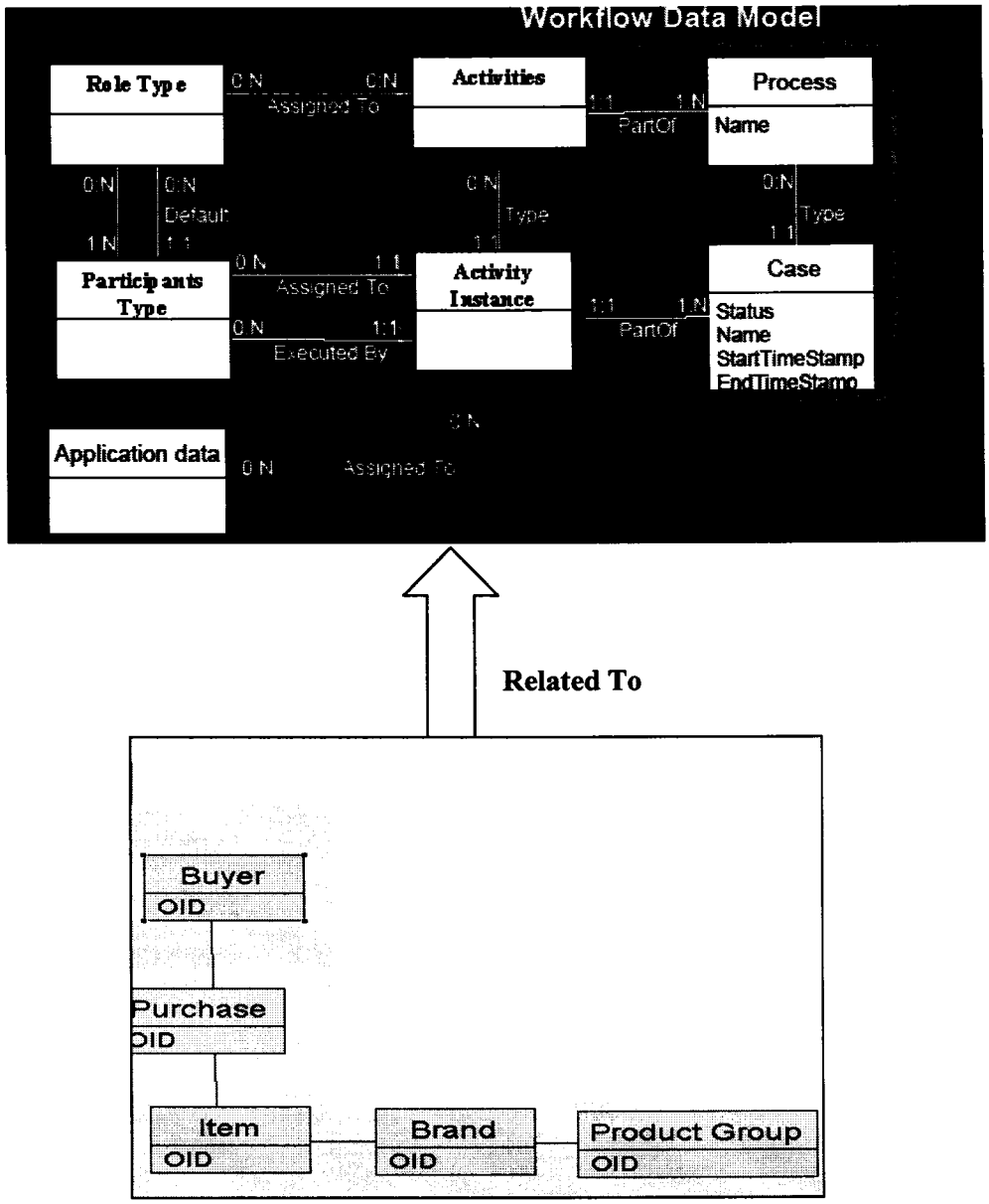


Figure 5.11: Data Model and WS-CDL Work Flow

In WS-CDL we represent "Buyer" as a role type. The buyer role is associated with purchase function. Only a set of role type can execute this function, for example bargain function is executed by roles "Seller" and "Buyer". To complete the function a set of activities has to be executed. Purchase is related to activity instance, which has a Start Timestamp and an End Timestamp.

5.5.5 Hypertext Model

The purpose of hypertext model is to describe one or more hypertext that can be published in the site. Different hypertext defines a so-called site view. In our case study we have two site views, "The Buyer Site View" and "The Seller Site View". Table 5.1 and Table 5.2 is the partial specification sheet describing the interface requirements for executing those site views.

Site View Name	The Buyer Site View
Description	In order to access this site view the buyer has to first login. He can then browse the complete product catalogue, selects his item of interest, bargain, check status of his bargain, purchase and logout.
User Group	Buyer
Use Cases	"Login", Purchase", "Bargain", "Browse"

Table 5.1: Site View Specification Sheet for "The Buyer Site View"

Site View Name	The Seller Site View
Description	In order to access this site view the seller has to first login. He can then bargain with the buyer, check his status of bargain. Once bargain is completed he can invoke credit check.
User Group	Seller
Use Cases	"Login", "Bargain", "invoke credit check"

Table 5.2: Site View Specification Sheet for "The Seller Site View"

5.5.8.1 Mapping WebML hypertext model with Choreography

Simple navigation is straightforward. A business process is initiated from a simple navigation, for example the buyer browsing the catalogue is a simple navigation. When he proceeds to checkout or initiates bargain, sequence of activities has to be executed to complete the process. Our model dedicates a portion of the hypertext to execute these processes. The process is enclosed between operations like Initiate WS-CDL and terminates WS-CDL. This portion is dedicated to execute workflow data. Figure 5.12 and 5.13 describes how WS-CDL models business process

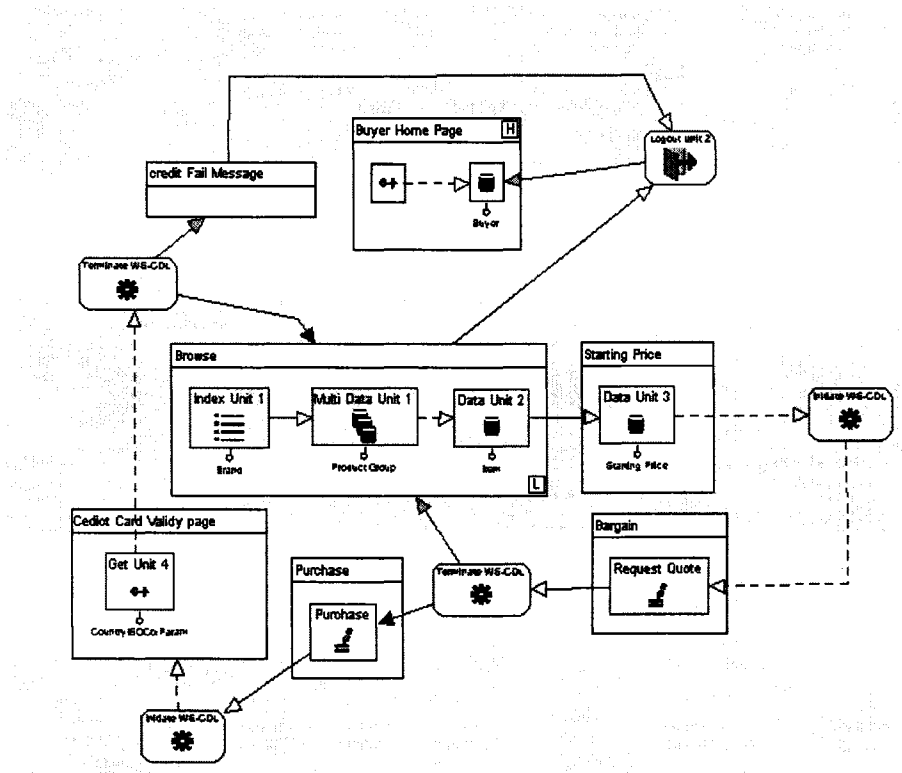


Figure 5.12: Buyer Site View, Pure Navigation with WS-CDL Business Process

To access this site the buyer has to first login. Once he logs in he can browse the catalogue of products. This is pure navigation and is represented using simple WebML implemented using Web Ratio. Once he finds his item of interest he starts his bargain. The bargain process starts with initiate WS-CDL and terminates WS-CDL. Fig 5.13 represents WS-CDL process in detail.

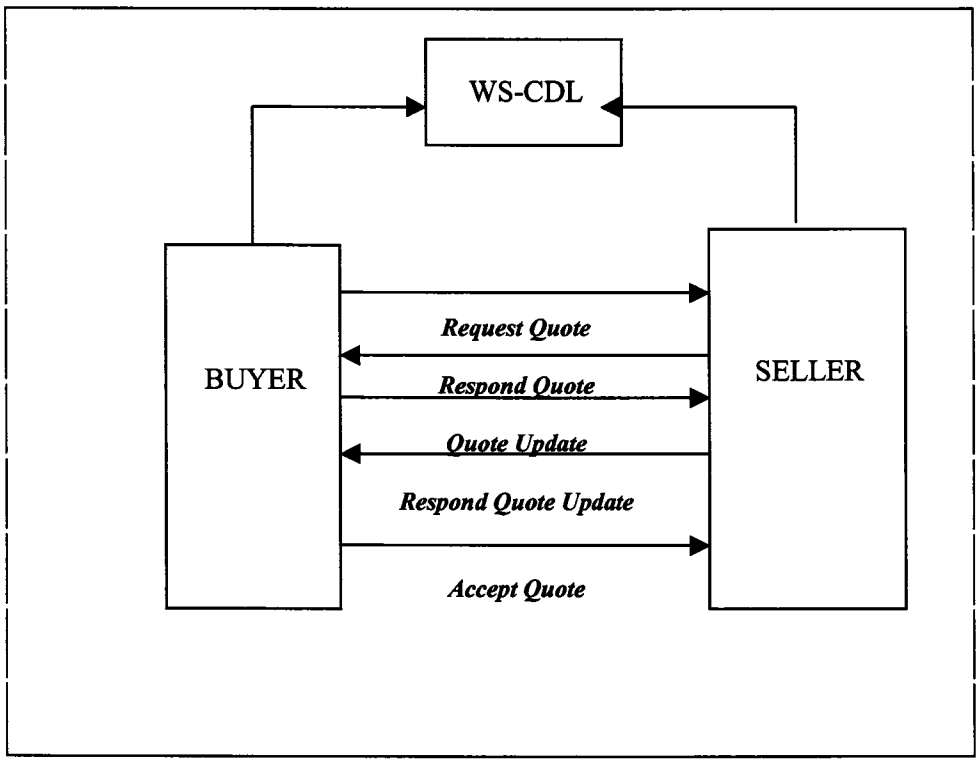


Figure 5.13: Initiate WS-CDL and Terminate WS-CDL

If the quote has been accepted the buyer sends accept quote response. Else update quote, the seller responds for the update quote. If bargain is successful the navigation would lead to checkout process else it will lead back to browse page/ Home page. Fig: 5.14 represent the seller's site view. The seller invokes two services the credit check service and the shipping service. The process is enclosed within the initiate WS-CDL and terminates WS-CDL. Hence, in our

model we have tried to achieve business process along with navigation using a newly released web service language WS-CDL.

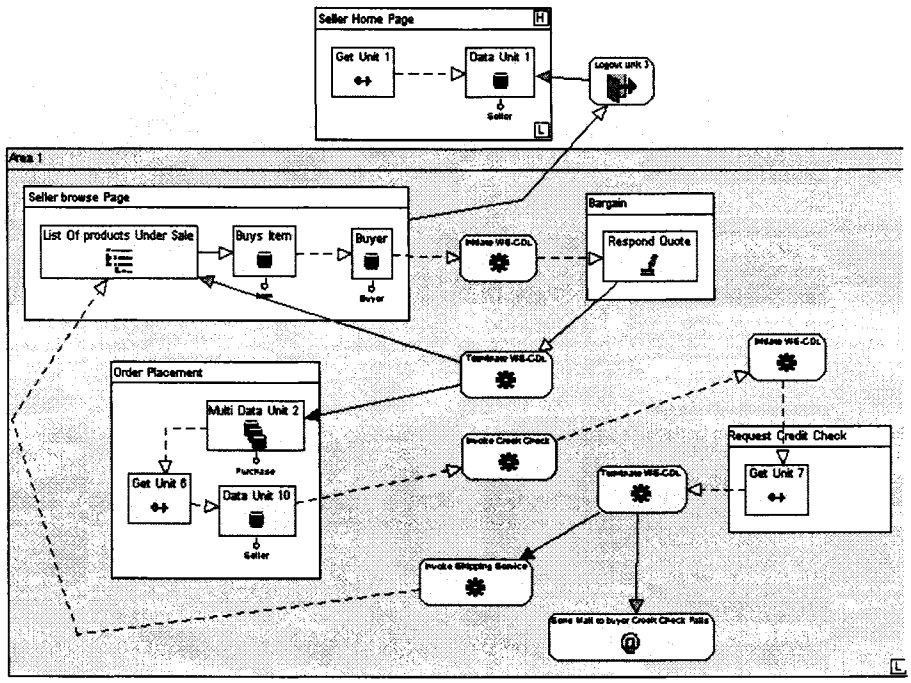


Figure 5.14: Seller Site View, Pure Navigation with WS-CDL Business Process

5.6 Where WS-CDL Fits?

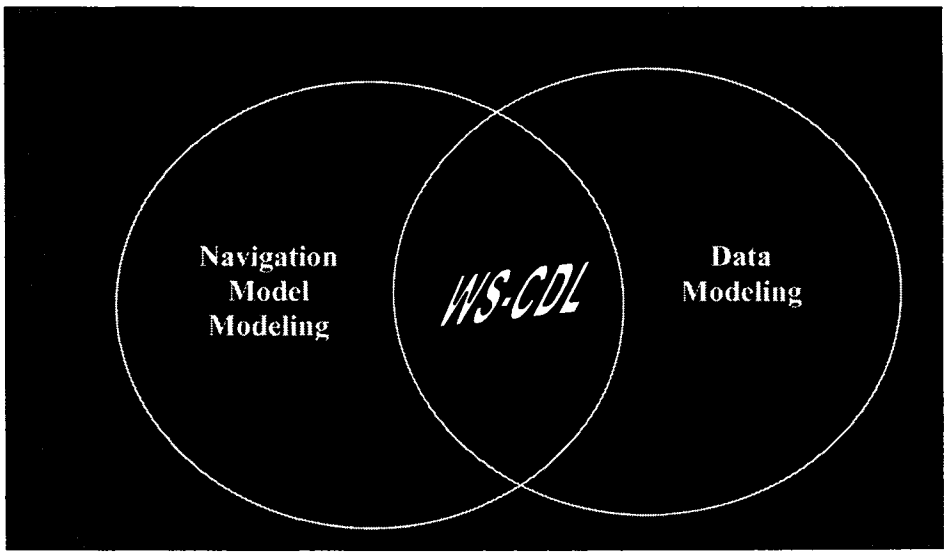


Figure 5.15: WS-CDL for Conceptual Modeling

5.8 Implementation

No other current modeling approach has used WS-CDL to model a business process. A Language meant for Web Service collaboration is being used for modeling business process. This choreography layer is the inventory of all business rules. The advantage of this approach is we get a centralized modeling approach to store all rules and this is the advantage over the present modeling approaches. We test and verify for this business processes. We used eclipse based WS-CDL editor called the pi4soa editor for testing WS-CDL process models. We created the choreography files, the .cdl file from the defined roles, participants, activities to perform the actual business process and test the business process using various scenario files we generated using the pi4soa editor.

Chapter 6: Experimental Results & Analysis

This section presents the experimental design and the result analysis for evaluating the performance of our model. We test for information flow and control flow from the choreographies generated. We ran many kinds of experiments to evaluate this system. All the experiments performed are based on the same case study and the design model proposed in chapter 5.

6.1 Testing & Verification

6.1.1 Defining the Choreography Layer for our Case Study

We verified the buyer and the seller services. Each service has participant type. The buyer belongs to the participant type "Buyer" and the seller belongs to the participant type "Seller". Each participant plays a role with a specific In our case study we have the "BuyerRoleType" which has the "BuyerBehavior" and the role type "SellerRoleType" has the behavior "SellerBehavior". When the buyer and the seller communicate, messages are exchanged over a specific channel. We developed the "Buyer2Seller" channel for the buyer and the seller to communicate during a session and the "4BuyerChannel" to inform the details to the buyer during a session. In our example we make bargain as one session, which is time stamped. The coordination between the participants is the main part of our choreography. Interactions are the main elements of a choreography describing the major jobs being performed. Each interaction consists of message exchanges "exchange details" to complete a specific task. We specified the information being exchanged in each message and the channel over which this exchange is performed is specified. Table 6.1 defines the choreographies needed to complete

the bargain process. The definition for the choreography and the roles that take part in each choreography is defined.

NO	Choreography	Definition	Roles
1	BuyerSeller.cdl	Root Choreography that co-ordinates with the other choreographies	Buyer, Seller
2	CreateTable.cdl	SqlGenerator	System
3	RequestQuote.cdl	Initial quote request	Buyer
4	RespondQuote.cdl	Send initial quote response	Seller
5	BuyReq.cdl	Buyer request for quote	Buyer
6	BuyRes.cdl	Buyer responds for quote send by the seller	Buyer
7	SellerRes.cdl	Seller responds for quote request	Seller
8	AcceptQuote.cdl	Send accept quote response by the Buyer	Buyer
9	UpdateQuote.cdl	Send update quote response by the Buyer	Buyer
10	AcceptReq.cdl	Accept the transaction request	Buyer, Seller
11	AcceptRes.cdl	Accept the transaction response	Buyer, Seller

Table 6.1: Choreographies Needed to Complete Bargain Process

6.1.2 Test Cases to Verify the Choreographies

For testing our case study, we generated various choreographies. These choreographies have to be verified and tested. In order to test and verify our choreography we generated various test cases. They are called test scenarios. Each choreography for verification purposes will have two test scenarios. A valid test scenario and an invalid test scenario. We have 11 choreographies to verify our case study. We generated 2 scenarios for each choreography one is valid and the other is invalid, hence we have a total of 22 test cases to verify and validate our

choreographies. Table 6.2 and 6.3 defines the various test scenarios, their definition and the choreographies associated to it.

NO	Test Case Name	Choreography Name	Description
1	BuyerSeller-valid.scenario	BuyRes.cdl	This is the root choreography. This co-ordinate with all the other choreographies. A valid scenario sends a "AcceptQuoteResponse"
2	BuyerSeller-invalid.scenario	BuyRes.cdl	This is the root choreography. This co-ordinate with all the other choreographies. An in-valid scenario sends a "UpdateQuoteResponse"
3	TableCreate-valid.scenario	CreateTable.cdl	A scenario to connect to database. A valid scenario sends a "ConnectionEstablished Response"
4	TableCreate-invalid.scenario	CreateTable.cdl	A scenario to connect to database. An invalid scenario sends a "ConnectionFailedResponse"
5	ReqQuoteSend-valid.scenario	RequestQuote.cdl	Initial request for quote message passed to the seller. Response from seller: "ResponseQuoteSend-valid" respond within the timestamp
6	ReqQuoteSend-invalid.scenario	RequestQuote.cdl	Initial request for quote message not passed to the seller. Response from seller: "ResponseQuoteSend-invalid" response.
7	ResponseQuoteSend-valid.scenario	RespondQuote.cdl	Initial response for quote from the seller.
8	ResponseQuoteSend-invalid.scenario	RespondQuote.cdl	Initial response for quote not been able to send from the seller.
9	BuyReqSend-valid.scenario	BuyReq.cdl	Request for quote message passed to the seller. Response from seller: "ResponseQuoteSend-valid" respond within the timestamp
10	BuyReqSend-invalid.scenario	BuyReq.cdl	Request for quote message not passed to the seller. Response from seller: "ResponseQuoteSend-invalid" response.

Table 6.2: Table to Define Various Test Scenarios

NO	Test Case Name	Choreography Name	Description
1	BuyResSend-invalid.scenario	BuyerSeller.cdl	This is the root choreography. This co-ordinate with all the other choreographies. An in-valid scenario sends a "UpdateQuoteResponse"

2	BuyResSend-valid.scenario	BuyerSeller.cdl	This is the root choreography. This co-ordinate with all the other choreographies. A valid scenario sends a "AcceptQuoteResponse"
3	SellerResSend-valid.scenario	SellerRes.cdl	For all valid request that is send to the seller service from the buyer service, it sends a "ResponseQuoteSend" response
4	SellerResSend-invalid.scenario	SellerRes.cdl	In this scenario the seller does not receive his quote request from the buyer
5	AcceptQuoteRes-valid.scenario	AcceptQuote.cdl	The buyer accepts the quote. The final response is "AcceptQuoteRes"
6	AcceptQuoteRes - invalid.scenario	AcceptQuote.cdl	The buyer does not accept the quote, the response is "UpdateQuoteRes"
7	UpdateQuoteRes-valid.scenario	UpdateQuote.cdl	If the buyer does not accept the quote. The response is "AcceptQuoteRes-invalid"
8	UpdateQuoteRes-invalid.scenario	UpdateQuote.cdl	The buyer accepts the quote. The final response is "AcceptQuoteRes-valid"
9	AcceptTranReq-valid.scenario	AcceptReq.cdl	Both the buyer and seller accept the transactions and send the request to each other to complete the process. Response is "TransactionAcceptValid"
10	AcceptTranReq-invalid.scenario	AcceptReq.cdl	Both the buyer and seller accept the transactions and send the request to complete the process. If the request not reached The response is "TransactionAccept Invalid"
11	AcceptTranRes-valid.scenario	AcceptRes.cdl	Both the buyer and seller accept the transactions and send the request to each other to complete the process. "TransactionAcceptValid" response is send
12	AcceptTranRes-invalid.scenario	AcceptRes.cdl	Both the buyer and seller accept the transactions and send the request to each other to complete the process. "TransactionAcceptValid" response is not send

Table 6.3: Table to Define Various Test Scenarios

6.2 Experimental Results

From the case study discussed in chapter 5, we can say we have taken a non-trivial business process, the online bargain between the buyer and seller. To complete this transaction multiple request/response messages should be exchanged between the buyer and the seller. We have modeled this functional requirement in WS-CDL. The outcome of this system should be either "AcceptQuote" or "UpdateQuote" operation or an "Invalid" response.

When it is an AcceptQuote operation it means that the transaction is completed successfully. For the UpdateQuote operation it means that the buyer needs more bargain. If it is an invalid operation then it means the operation failed. This null service endpoint leads us to an invalid response. Based on this we have tested our model. All the valid scenario union would lead to a valid response and the invalid scenario union would lead to an invalid response.

Our approach would be if the result has to be valid then all the scenarios should be valid. If the result is invalid then all or at least one of the scenarios should be invalid. Based on this we have developed and tested our model. And the tables below summarize the experimental results. Table 6.4 describes control flow and the information flow of the in-valid scenario and table 6.5 describes the control flow and the information flow of the valid scenario.

PID	Description	Service	Completed	Operation	Final Result
Null	Null	Null	X	Null	Null
1	Initial request For Quote	Buyer	X	"RequestQuote"	Completed Initial Request For Quote
2	Update Quote Response	Buyer	X	"UpdateQuote"	Completed First Update Quote From Buyer
3	Accept Quote	Buyer/Seller	X	"AccepQuote"	Invalid response

Table 6.4: In-Valid Scenarios Results

PID	Action	Participant	Type	Channel	Completed
1/0	Message Event	Buyer	Request Send	C1	Message handled
1/1	Message Event	Seller	Request received	C1	Message handled
½	Message Event	Seller	Response Send	C1	Message handled
1/3	Message Event	Buyer	Response received	C1	Message handled

Table 6.4.1: Initial Request For Quote Results

PID	Action	Participant	Type	Channel	Completed
2/0	Message Event	Buyer	Request Send	C1	Message handled
2/1	Message Event	Seller	Request received	C1	Message handled
2/2	Message Event	Seller	Response Send	C1	Message handled
2/3	Message Event	Buyer	Response received	C1	Message handled

Table 6.4.2: First Update Quote Results

PID	Action	Participant	Type	Channel	Completed
3/0	Message Event	Buyer	Request Send	C1	Message handled
3/1	Message Event	Seller	Request received	C1	Message handled
3/2	Message Event	Seller	Response Send	C1	Message handled
3/3	Message Event	Buyer	Response received	C1	Message handled

Table 6.4.3 In-Valid Accept Quote Results

PID	Description	Service	Completed	Operation	Final Result
Null	Null	Null	X	Null	Null
1	Initial request For Quote	Buyer	X	"RequestQuote"	Completed Initial Request For Quote
2	First Update Quote Response	Buyer	X	"UpdateQuote"	Completed First Update Quote From Buyer
3	Second Update Quote Response	Buyer	X	"UpdateQuote"	Completed Second Update Quote From Buyer
4	Accept Quote Response	Buyer/ Seller	X	"AcceptQuote"	Finally the quote is accepted

Table 6.5: Valid Scenarios Results

PID	Action	Participant	Type	Channel	Completed
1/0	Message Event	Buyer	Request Send	C1	Message handled
1/1	Message Event	Seller	Request received	C1	Message handled
½	Message Event	Seller	Response Send	C1	Message handled
1/3	Message Event	Buyer	Response received	C1	Message handled

Table 6.5.1: Initial Request For Quote Results

PID	Action	Participant	Type	Channel	Completed
2/0	Message Event	Buyer	Request Send	C1	Message handled
2/1	Message Event	Seller	Request received	C1	Message handled
2/2	Message Event	Seller	Response Send	C1	Message handled
2/3	Message Event	Buyer	Response received	C1	Message handled

Table 6.5.2: First Update Quote Results

PID	Action	Participant	Type	Channel	Completed
3/0	Message Event	Buyer	Request Send	C1	Message handled
3/1	Message Event	Seller	Message Received	C1	Message handled
3/2	Message Event	Seller	Invalid Response	C1	Message handled
3/3	Message Event	Buyer	Invalid Response	C1	Message handled

Table 6.5.3: Second Update Quote Results

PID	Action	P articipant	Type	Channel	Completed
4/0	Message Event	Buyer	Request Send	C1	Message handled
4/1	Message Event	Seller	Message Received	C1	Message handled
4/2	Message Event	Seller	Invalid Response	C1	Message handled
4/3	Message Event	Buyer	Invalid Response	C1	Message handled

Table 6.5.4: Valid Accept Quote Results

6.3 Result Analysis

We have tested the Business Process for control flow and the information flow. When emulating the business process with navigation the control flow and the information is lost. This would lead to erroneous results. Hence we have chosen to test for these two parameters. The two services exchange their information via a channel. We have named the channel as C1 for the buyer and the seller to communicate. Choreography is based on pi-calculus hence we a starting state, a set of transition and an end state. The system takes in an input and produces an output. Based on the output further actions are carried out. We tested for two main things: the system generates a valid response to complete the bargain process; the system results in an invalid output so that the process ends. Figure 6.1, 6.2 illustrates this. This entire test was based on the functional requirements discussed in chapter 5.

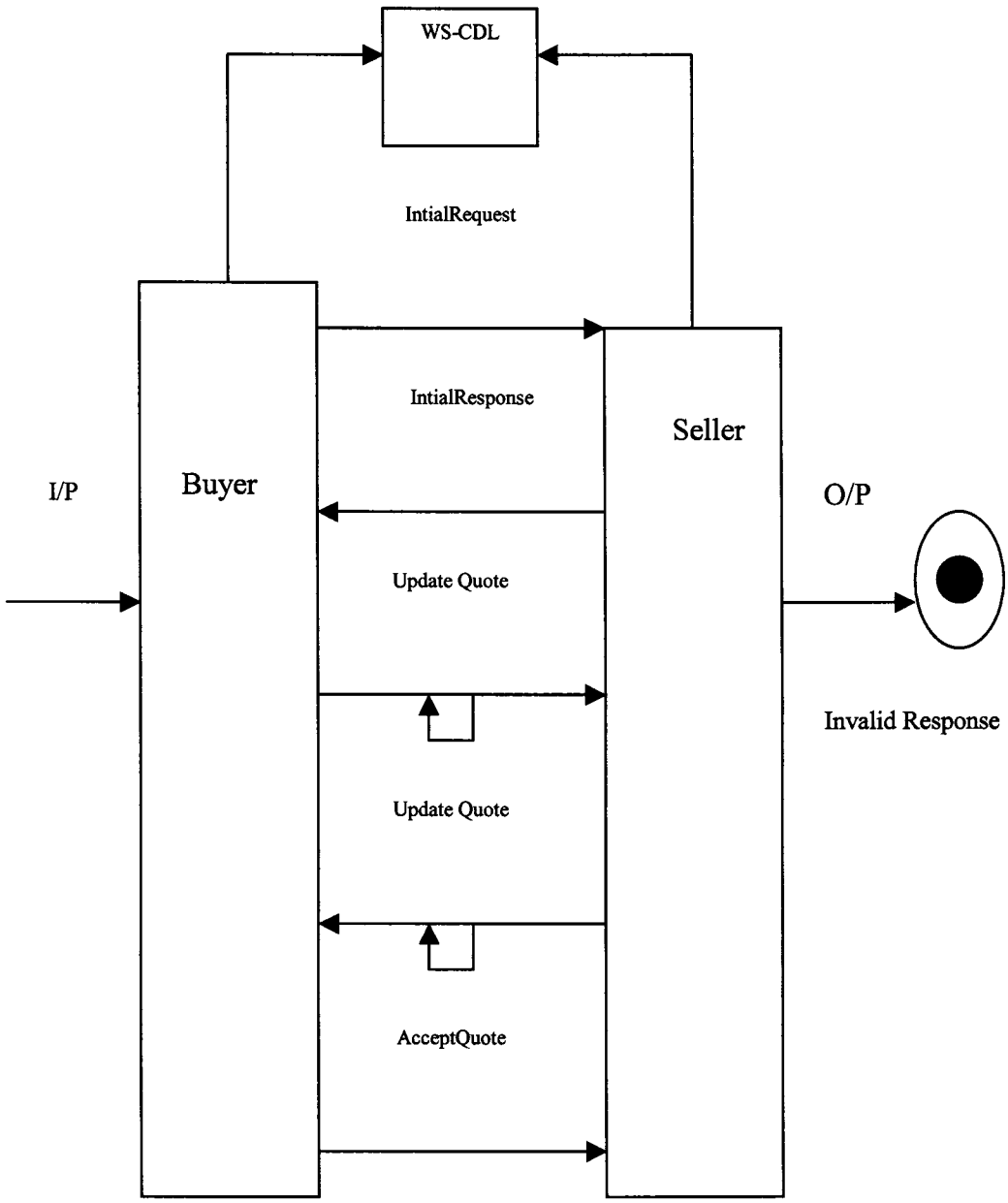


Figure 6.1: To demonstrate the Invalid Output From the System

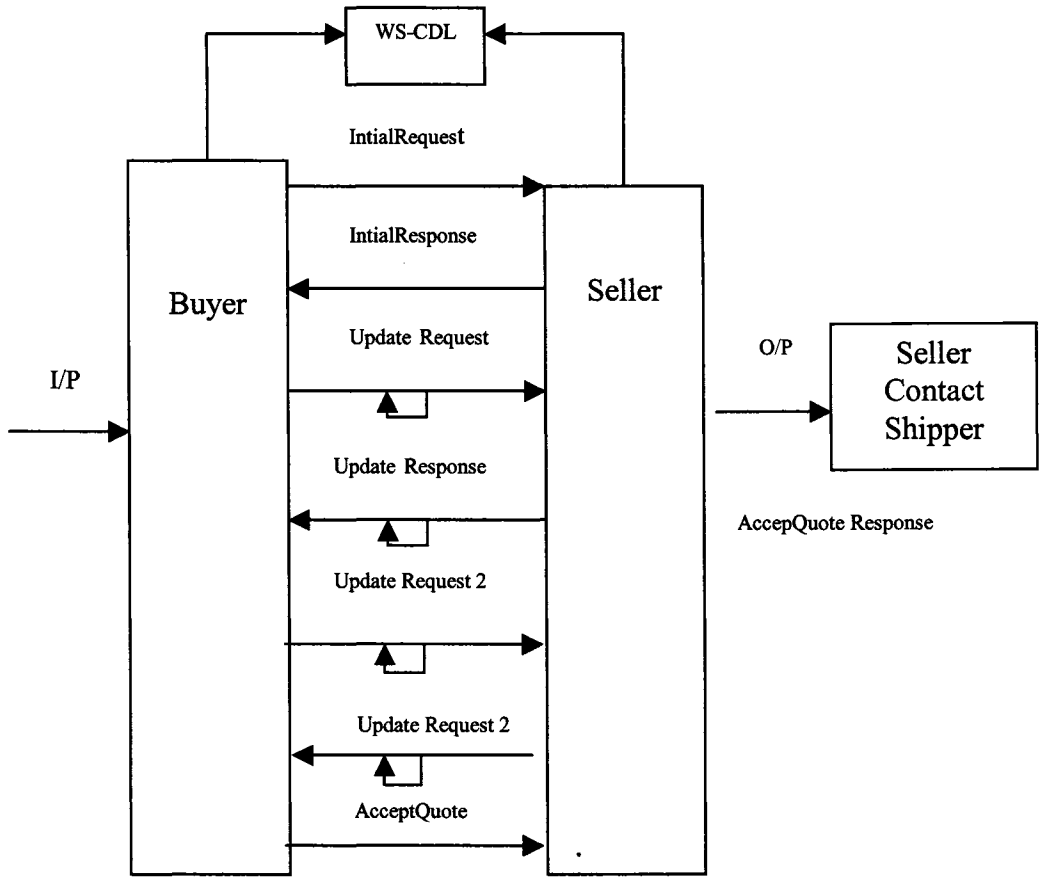


Figure 6.2: To demonstrate the Valid Output From the System

From the outcome of the system further actions are carried out for example in an in-valid scenario when we receive an invalid response the system ends the bargain process without completing the transaction. In a valid scenario the system generates the "AcceptQuote" response. When the seller receives this response he contacts the credit check agency for completing the transaction with the buyer. Thus further actions are carried out based on the output of the system tested. By this way the control flow of the system is verified.

Extensive evaluation, testing and verification of the system show that the functional requirements of any business process can be fulfilled using WS-CDL to model the process.

Chapter 7: Conclusion & Future Work

7.1 Conclusion

There are 4 contributions of this work:

- A novel approach to represent business process and navigation are based on WS-CDL and WebML.
- Proposing a solution to the problems faced by current modeling methods, the way in which they model business process by emulating them along with navigation.
- Extensive experiments to prove the potentiality of this newly released peer-peer choreography language for web services in modeling business process in e-commerce applications.
- Successful practical implementation of the choreography layer where business process is exclusively designed and tested using the pi4soa tool.

The experimental results have shown that WS-CDL being a peer-peer web service collaborative language is also good at modeling business process. Based on these experimental results we can say that instead of having a web-modeling language to model business process and navigation, they can be used to model only navigation and informational aspects of a web site and the business requirements of the web site can be modeled using WS-CDL. By this approach we can overcome the limitations of a web modeling language as they emulate business process as a form of navigation. By this way we would have one unique model, to model business process along with navigation.

7.1.1 Limitations of our Approach

Though we have proposed a model to represent business process and navigation based on WS-CDL and WebML and demonstrated the ability of WS-CDL to model business process to support the model we proposed, the model has the following limitations: WS-CDL is a newly released peer-peer language for web service collaboration and the language editors are not fully developed. The pi4soa tool we used for WS-CDL is the first WS-CDL editor in the market. This editor as of the current date does not support automatic java code generation from the choreography models developed. Hence we are not able to develop a front end to represent a presentation model and deploy it to see how effective business process is working when modeled in WS-CDL along with navigation primitives.

7.2 Future Work

This model can be extended to include all the aspects of web modeling. Hence we would have one unique framework for all the modeling languages to overcome the business modeling approach. With the help of this framework it would be an ideal way to model all the requirements of an e-commerce site in a simpler way. E-commerce is growing these days; this leads to new modeling requirements like b2b interaction, web sites to support web service collaboration etc. Good modeling approach is necessary to meet these new requirements of e-commerce.

References

1- Web Services Choreography Description Language Version 1.0 W3C Working Draft 12 October 2004. http://www.w3.org/TR/2004/WD-ws-cdl-10-20041012/

2- Conceptual Modeling Of Web Applications Featuring e-Services and Workflow Join work with: Stefano Ceri, Aldo Bongio, Sara Comai, Maristella Matera, Loana Manolescu, Roberto Acerbis, Stefano Butti, Emanuele Molteni.
25th, March 2005. http://webml.org/webml/upload/ent5/1/sebd03_tutorial.pdf

3- A Critical Overview of the Web Services Choreography Description Language (WS-CDL) By Alistair Barros, Marlon Dumas, Phillipa Oaks 18th March 2005 http://www.bptrends.com.

4- Extending hypertext conceptual models with process-oriented primitives By Marco Brambilla, Milano Italy, 15th December 2005 http://www.webml.org/webml/upload/ent5/1/er2003paper230_brambilla.pdf

5- **WIED: A Web Modelling Language for Modelling Architectural-Level Information By** Rachatrin Tongrungrojana and David Lowe Faculty of Engineering, University of Technology Sydney, P.O. Box 123 Broadway Sydney, NSW 2007. http://jodi.tamu.edu/Articles/v05/i02/Tongrungrojana/

6- S. Ceri, P. Fraternali, and S. Paraboschi, "Web Modeling Language (WebML): A Modeling Language for Designing Web Sites," *Proc. 9th Int'l World Wide Web Conf.,* Elsevier, 2000, pp 137—157

7- L. Baresi, F. Garzotto, and P. Paolini. "From Web Sites to Web Applications: New Issues for Conceptual Modeling," *Proc. Int'l Workshop on the World Wide Web and Conceptual Modeling,* LNCS 1921, S.W. Liddle, H.C. Mayr, and B. Thalheim, eds. Springer, 2000, pp. 89—100

8- H.A. Schmid, and G. Rossi, "Designing Business Processes in E-Commerce Applications," *Proc EC-Web 02,* LNCS 2455, Springer, 2002, pp. 353-362.

9- L. Baresi, et al., "Assertions to Better Specify the Amazon Bug," *Proc. 14th Int'l Conf. Software Eng. and Knowledge Eng.,* ACM Press, 2002, pp. 585,592.

10- D. Schwabe, and G. Rossi, "An Object-Oriented Approach to Web-Based Application Design," *Theory and Practice of Object Systems,* vol. 4, no. 4, 1998, pp. 207—225

11- Modeling and designing processes in e-commerce applications Schmid, H.A.; Rossi, G. Internet Computing, IEEE Volume: 8 Issue: 1 Jan-Feb 2004 Page(s): 19- 27

12- WebML+ in a nutshell: Modeling architectural-level information flows By David Lowe, Rachatrin Tongrungrojana Faculty of Engineering, University of Technology, Sydney http://www2003.org/cdrom/papers/poster/p003/p3-Lowe.html

13- Schwabe, D. and Rossi, G. The Object-Oriented Hypermedia Design Model. *Communications of the ACM*, 38 (8). 45-46.

14- Ceri, S., Fraternali, P. and Bongio, A., Web Modeling Language (WebML): a modeling language for designing Web sites. in *Proceedings of WWW9 Conference*, (Amsterdam, 2000).

15- Modeling Data **Entry and Operations in WebML:** S. Ceri, P. Fraternali, A. Bongio, **A.** Maurino: "Modeling data entry and operations in WebML". WebDB 2000, Dallas, 2000

16- S. Ceri, F. Daniel, M. Matera. "Extending WebML for Modeling Multi-Channel Context-Aware Web Applications". Proc. of the WISE - MMIS'03 Workshop (Mobile Multi-channel Information Systems), Roma, December 2003, IEEE Press

17- F. M. Facca, S. Ceri, J. Armani, V. Demaldé, Building Reactive Web Applications. Poster at WWW2005, Chiba, Japan

18- Winter, R., Strauch, B., Conceptual Modeling of Large Web Sites, in: Proceedings of Mehdi Khosrowpour (Hrsg.): Challenges of Information Technology Management in the 21st Century, Idea Group Publishing, Hershey (USA), 2000, pp. 17

19- http://jdj.sys-con.com/read/36492.htm

20- Modeling Web Applications by Paul Allen, Joseph Bambara. Date: Jul 19, 2002 http://www.telemidia.puc-rio.br/oohdm/oohdm.html

21- Baresi, L., Garzotto, F. and Paolini, P. (2001) "Extending UML for Modeling Web Applications". In *Proceeding of the 34th Hawaii International Conference on System Sciences*, Hawaii, pp. 1285-1294

22- Stefano Ceri, Florian Daniel, Maristella Matera: Extending WebML for Modeling Multi-Channel Context Aware Web Applications.

23- Mario Bravetti, Claudio Guidi, Roberto Lucchi, and Gianluigi Zavattaro: Supporting e-commerce systems formalization with choreography

languages 2005 ACM Symposium on applied science.

24- http://www.webratio.com/sv1.do

25- Web Services Architecture W3C Working Group Note 11 February 2004 http://www.w3.org/TR/ws-arch/

26- SOAP Version 1.2 Part 1: Messaging Framework W3C Recommendation 24 June 2003 http://www.w3.org/TR/soap12-part1/

27- Web Services Definition Language (WSDL) 2.0 W3C Working Draft 3 August 2004. http://www.w3.org/TR/wsdl20/

28- Joerg Becker and Michael zur Muehlen and Marc Gille. Workflow Application Architectures: Classification and Characteristics of Workflowbased Information Systems. Workflow Handbook 2002, Layna Fischer, 2002, Future Strategies, Lighthouse Point, FL, p39-50.

29- Designing Data-Intensive Web Applications (The Morgan Kaufmann Series in Data Management Systems) (Paperback) by Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, Maristella Matera.

30- Workflow Management Coalition the Workflow Reference Model Document Number TC00-1003 Document Status - Issue 1.1 19-Jan-95 Author: David Hollingsworth

31- H.A schmid, " Business entity components and business process components," *J. object-oriented programming, vol. 12, no.6, 1999.*

32- H.A schmid, A.Cristaldi, and G.Jacobson, "A Business Process Component Framework," *proc 7th Int'l Conf. Object-oriented Information Systems* (OOIS 01), Springer, 2001, pp. 513-522.

33- L. Baresi et al., "Assertions To Better Specify the Amazon Bug," *Proc. 14th Int'l Conf. Software Eng. and Knowledge Eng., ACM Press, 2002, pp. 585,592.*

34- Luciano Baresi, Franca Garzotto, and Paolo Paolini "From Web Sites to Web Applications: New Issues for Conceptual Modeling", Piazza Leonardo da Vinci 32, 20133 Milano, Italy

35- G. Rossi, D. Schwabe and F. Lyardet: Web Application Models are more than Conceptual Models, in [8], pp.239-252.

36- Stefano Ceri, Aldo Bongio, Marco Brambilla, Sara Comai, Ioana Manolescu "Conceptual Modelling of Web Applications, Workflows and Web Services" WISE Tutorial, Roma, Dec 10th 2003

37- M. Brambilla, S. Ceri, S. Comai, P. Fraternali "Model-driven development of Web Services and hypertext applications" Dipartimento di Elettronica e Informazione P.zza L. da Vinci, 32 I-20133 Milano, Italy

38- Marco Brambilla, Stefano Ceri, Sara Comai, Piero Fraternali, Ioana Manolescu "Model-driven Specification of Web Services Composition and Integration with Data-intensive Web Applications" Dipartimento di Elettronica e Informazione P.zza L. da Vinci, 32 I-20133 Milano, Italy

39- P. P. Chen, The Entity-Relationship Model, Towards a Unified View of Data, ACM-Transactions on Database Systems, 1:1, 1976, pp. 9-36.

40- R. G. G. Cattell, Douglas K. Barry, and Dirk Bartels (Eds.), The Object Database Standard: ODMG 2.0, Morgan-Kaufmann Series in Data Management Systems, 1997.

41- G.Booch, I. Jacobson, and J. Rumbaugh, The Unified Modeling Language User Guide, The Addison-Wesley Object Technology Series, 1998.

42- Business Process Execution Language for Web Services Version 1.1 By IBM 05 May 2003 by Satish Thatte, Microsoft http://www-06.ibm.com/developerworks/library/ws-bpel/

43- WSFL: Web Service Flow Language 1.0 May 2001 By Prof. Dr.Frank Leymann, Distinguished Engineer Member IBM Academy of Technology, IBM Software Group http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

44- XLANG: Web Services for Business Process Design 2001 Microsoft Corporation By Satish Thatte, Microsoft http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

45- **Integrating and customizing heterogeneous e-commerce.** The International Journal on Very Large Data Bases Volume 10, Issue 1 (August 2001) Pages: 16 – 38 Year of Publication: 2001 ISSN: 1066-8888

46- **A web services-based business interactions manager to support electronic commerce applications** ACM International Conference Proceeding Series; Vol. 113 Proceedings of the 7th international conference on Electronic commerce Xi'an, China Pages: 435 - 445 Year of Publication: 2005 ISBN: 1-59593-112-0

47- J. Conallen: Modeling Web Application Architectures with UML, Communications of the ACM, 42:10, Oct. 1999, pp. 63-70.

48- Conceptual Modeling of Web Applications, Workflows and Web Services

Workflows and Web Services Piero Fraternali DEI, Politecnico di Milano WISE Tutorial, Roma, Dec 10th 2003.

49- L. Baresi, F. Garzotto, P. Paolini, "From Web Sites to Web Applications: News Issues for Conceptual Modeling", Proc. Int'l Workshop on the World Wide Web and Conceptual Modeling, LNCS 1921, S.W. Liddle, H.C. Mayr, B. Thalheim, eds. Springer, 2000, pp. 89-100

50- L. Baresi, G. Denaro, L. Mainetti, and P. Paolini "Assertions to Better Specify the Amazon Bug" Dipartimento di Elettronica e Informazione Politecnico di Milano Piazza L. da Vinci 32, I20133 Milano.

51- P. Fraternali, M. Brambilla, Conceptual modeling of Web applications featuring e-services and workflows, Tutorial at SEBD 2003, Cetraro, Italy

52- I. Manolescu, S. Ceri, M. Brambilla, P. Fraternali, S. Comai: "Exploring the combined potential of Web sites and Web services", poster at WWW'03, Budapest, Hungary

53- M. Brambilla, S. Ceri, P. Fraternali, R. Acerbis, A. Bongio. Model-driven Design of Service-enabled Web Applications. Sigmod Industrial '05, Baltimore, June 13-16 2005

54- M. Brambilla, S. Ceri, S. Comai, P. Fraternali, I. Manolescu: "Model-driven Development of Web Services and Hypertext Applications", SCI2003, Orlando, Florida, July 2003

55- M. Brambilla, S. Ceri, S. Comai, P. Fraternali, I. Manolescu. "Model-driven Specification of Web Services Composition and Integration with Data-intensive Web Applications", IEEE Bulletin of Data Engineering, December 2002

56- M. Brambilla, N. D'Elia. "Exception Handling within Workflow-based Web Applications", Web Engineering 4th International Conference, ICWE 2004, Munich, Germany, 2004 – Proceedings, LNCS, Springer

57- M. Brambilla, Extending hypertext conceptual models with process-oriented primitives, ER 2003 (International Conference on Conceptual Modeling), October 2003, Chicago

58- UML Modelling of Automated Business Processes with a Mapping to BPEL4WS by Version 1.1 June 9th 2003 Jim Amsden, Tracy Gardner, Catherine Griffin, Sridhar Iyengar

59- Coordinating Web Services based on business models Koichi TERAI, Noriaki IZUMI, and Takahira YAMAGUCHI . Pages: 473 - 478 Year of Publication: 2003 ISBN: 1-58113-788-5

Appendix

A: Pi4SOA Details

This section gives the detail about the Pi4SOA. This tool is the first WS-CDL editor available in the market. Pi4 Technologies offers open source tools, consulting and professional open source support to leverage the power of the *Pi Calculus*. The *Pi Calculus*, invented by Professor Robin Milner et al, forms the basis for the W3C's WS-CDL. Professor Robin Milner, Dr. Kohei Honda and Dr Nobuko Yoshida are all invited experts contributing to WS-CDL.

WS-CDL is the first open source solution that we are offering to the community at large. It provides the necessary tools to describe and police blueprints for complex distributed IT architectures as well as for describing cross-domain business protocols (e.g. FIX, fpML, SWIFT, etc). Figure below is a snap shot of our working environment in pi4soa:

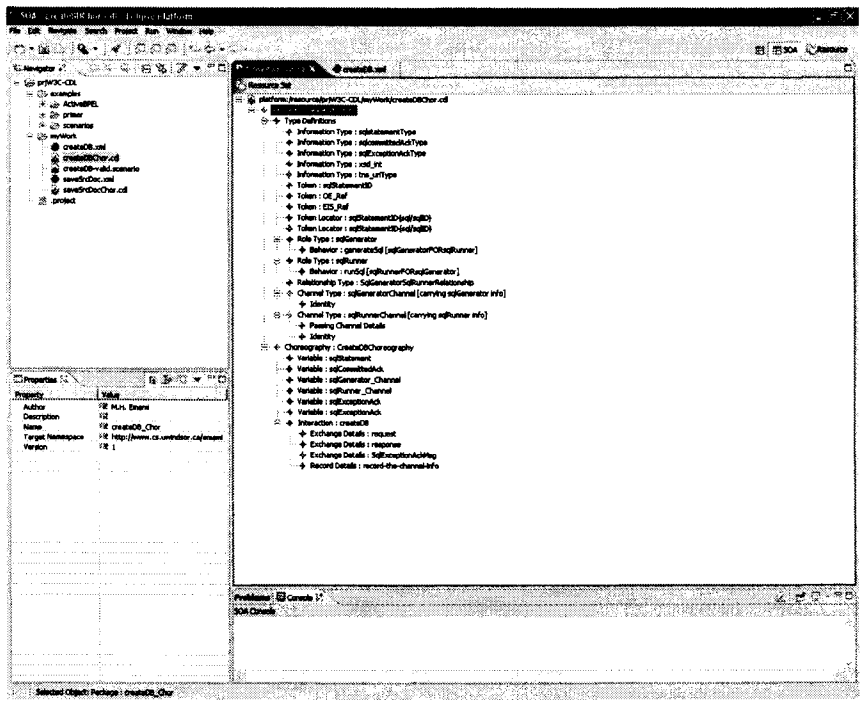


Figure: Pi4SOA Tool Snapshot

Installation Details of Pi4SOA

System Requirements

Eclipse: tested with version 3.0.1, 3.0.2 and 3.1

This can be downloaded from the Eclipse website: http://www.eclipse.org

EMF (Eclipse Model Framework): tested with version 2.0.1 and 2.1

This can be downloaded from the Eclipse website: http://www.eclipse.org/emf

GEF (Graphical Editor Framework): tested with version 3.0.1 and 3.1

This can be downloaded from the Eclipse website: http://www.eclipse.org/gef

JDK1.4.2 or **JDK1.5**

Downloading And Installation

The release can be downloaded from the SourceForge project, at http://www.sourceforge.net/projects/pi4soa.

- Once the ZIP file has been downloaded, the contents should be extracted into your Eclipse installation's home directory.
- When the contents have been unpacked, then you should restart your Eclipse Application and verify that the plugins have been correctly installed.
- Simply click on the WS-CDL Eclipse icon, or run eclipse.exe in the installation directory.

Verifying the Installation

To verify Pie4SOA has been properly installed in your Eclipse follow the steps below

- Selecting the "About Eclipse Platform" item on the "Help" menu can do this.

- This will display a window with some buttons at the bottom. Press the button labelled "Plug-in Details".
- This will display a list of plugins that are currently visible within your Eclipse environment, and should now include a set of plugins from www.pi4soa.org with the appropriate version number.

Reporting Problems in Installation

If a problem occurs while using the pi4soa tools, then please consult the "bugs" and "support requests" sections on the pi4soa sourceforge project: http://sourceforge.net/projects/pi4soa.

B: Sample Code

Buyer Seller Main Choreography

```
<?xml version="1.0" encoding="ASCII"?>
<org.pi4soa.cdl:Package xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:org.pi4soa.cdl="http:///org/pi4soa/cdl.ecore" name="Buyer"
author="Sundar6" version="1"
targetNamespace="http://www.cs.uwindsor.ca/sundar6">
 <initialBehaviorDescription name="ExampleRFQPattern">
  <activityTypes xsi:type="org.pi4soa.service.behavior:Sequence">
   <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="RequestForQuote:rfqReq" channelName="channel1"
operationName="requestForQuote" serviceType="ServiceB"
channelType="channelType1"/>
   <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="RequestForQuote:rfqResp" channelName="channel1"
operationName="requestForQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
   <activityTypes xsi:type="org.pi4soa.service.behavior:While">
    <preConditions
xsi:type="org.pi4soa.service.behavior:LookaheadMessage"
messageActivity="//@initialBehaviorDescription/@activityTypes.0/@activity
Types.2/@activityTypes.0"/>
    <postConditions
xsi:type="org.pi4soa.service.behavior:LookaheadMessage"
messageActivity="//@initialBehaviorDescription/@activityTypes.0/@activity
```

```
Types.3"/>
      <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="UpdateQuote:updatedQuoteReq" channelName="channel1"
operationName="updateQuote" serviceType="ServiceB"
channelType="channelType1"/>
      <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="UpdateQuote:updatedQuoteResp" channelName="channel1"
operationName="updateQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
      <reEvaluateCondition
xsi:type="org.pi4soa.service.behavior:XPathCondition" expression="true"/>
    </activityTypes>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="AcceptQuote:acceptReq" channelName="channel1"
operationName="acceptQuote" serviceType="ServiceB"
channelType="channelType1"/>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="AcceptQuote:acceptResp" channelName="channel1"
operationName="acceptQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
   </activityTypes>
 </initialBehaviorDescription>
 <serviceTypes name="ServiceB">
  <operations name="requestForQuote">
   <request canInitiateSession="true"/>
   <response/>
  </operations>
  <operations name="updateQuote">
   <request canInitiateSession="true"/>
   <response/>
  </operations>
  <operations name="acceptQuote">
   <request canInitiateSession="true"/>
   <response/>
  </operations>
 </serviceTypes>
</org.pi4soa.service.behavior:ServiceDescription>

<?xml version="1.0" encoding="ASCII"?>
<org.pi4soa.cdl:Package xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:org.pi4soa.cdl="http:///org/pi4soa/cdl.ecore" name="Seller"
author="Sundar6" version="1"
targetNamespace="http://www.cs.uwindsor.ca/sundar6">
 <initialBehaviorDescription name="ExampleRFQPattern">
  <activityTypes xsi:type="org.pi4soa.service.behavior:Sequence">
100    <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
```

```
label="RequestForQuote:rfqReq" channelName="channel1"
operationName="requestForQuote" serviceType="ServiceB"
channelType="channelType1"/>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="RequestForQuote:rfqResp" channelName="channel1"
operationName="requestForQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
    <activityTypes xsi:type="org.pi4soa.service.behavior:While">
     <preConditions
xsi:type="org.pi4soa.service.behavior:LookaheadMessage"
messageActivity="//@initialBehaviorDescription/@activityTypes.0/@activity
Types.2/@activityTypes.0"/>
     <postConditions
xsi:type="org.pi4soa.service.behavior:LookaheadMessage"
messageActivity="//@initialBehaviorDescription/@activityTypes.0/@activity
Types.3"/>
     <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="UpdateQuote:updatedQuoteReq" channelName="channel1"
operationName="updateQuote" serviceType="ServiceB"
channelType="channelType1"/>
     <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="UpdateQuote:updatedQuoteResp" channelName="channel1"
operationName="updateQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
     <reEvaluateCondition
xsi:type="org.pi4soa.service.behavior:XPathCondition" expression="true"/>
    </activityTypes>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="AcceptQuote:acceptReq" channelName="channel1"
operationName="acceptQuote" serviceType="ServiceB"
channelType="channelType1"/>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="AcceptQuote:acceptResp" channelName="channel1"
operationName="acceptQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
   </activityTypes>
  </initialBehaviorDescription>
  <serviceTypes name="ServiceB" serviceProvider="true">
   <operations name="requestForQuote">
    <request canInitiateSession="true"/>
    <response/>
   </operations>
   <operations name="updateQuote">
    <request canInitiateSession="true"/>
    <response/>
   </operations>
   <operations name="acceptQuote">
    <request canInitiateSession="true"/>
```

```
    <response/>
   </operations>
  </serviceTypes>
</org.pi4soa.service.behavior:ServiceDescription>
```

Valid Scenario

```
<?xml version="1.0" encoding="ASCII"?>
<org.pi4soa.cdl:Package xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:org.pi4soa.cdl="http:///org/pi4soa/cdl.ecore" name="buyer"
author="Sundar6" version="1"
targetNamespace="http://www.cs.uwindsor.ca/sundar6">

   <activityTypes xsi:type="org.pi4soa.service.behavior:Sequence">
    <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="RequestForQuote:rfqReq" channelName="channel1"
operationName="requestForQuote" serviceType="ServiceB"
channelType="channelType1"/>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="RequestForQuote:rfqResp" channelName="channel1"
operationName="requestForQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
    <activityTypes xsi:type="org.pi4soa.service.behavior:While">
     <preConditions
xsi:type="org.pi4soa.service.behavior:LookaheadMessage"
messageActivity="//@initialBehaviorDescription/@activityTypes.0/@activity
Types.2/@activityTypes.0"/>
     <postConditions
xsi:type="org.pi4soa.service.behavior:LookaheadMessage"
messageActivity="//@initialBehaviorDescription/@activityTypes.0/@activity
Types.3"/>
     <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="UpdateQuote:updatedQuoteReq" channelName="channel1"
operationName="updateQuote" serviceType="ServiceB"
channelType="channelType1"/>
     <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="UpdateQuote:updatedQuoteResp" channelName="channel1"
operationName="updateQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
     <reEvaluateCondition
xsi:type="org.pi4soa.service.behavior:XPathCondition" expression="true"/>
    </activityTypes>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="AcceptQuote:acceptReq" channelName="channel1"
operationName="acceptQuote" serviceType="ServiceB"
channelType="channelType1"/>
```

```
    <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="AcceptQuote:acceptResp" channelName="channel1"
operationName="acceptQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
   </activityTypes>
  </initialBehaviorDescription>
  <serviceTypes name="ServiceB">
   <operations name="requestForQuote">
    <request canInitiateSession="true"/>
    <response/>
   </operations>
   <operations name="updateQuote">
    <request canInitiateSession="true"/>
    <response/>
   </operations>
   <operations name="acceptQuote">
    <request canInitiateSession="true"/>
    <response/>
   </operations>
  </serviceTypes>
</org.pi4soa.service.behavior:ServiceDescription>

<?xml version="1.0" encoding="ASCII"?>
<org.pi4soa.cdl:Package xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:org.pi4soa.cdl="http:///org/pi4soa/cdl.ecore" name="Seller"
author="Sundar6" version="1"
targetNamespace="http://www.cs.uwindsor.ca/sundar6">
  <initialBehaviorDescription name="ExampleRFQPattern">
   <activityTypes xsi:type="org.pi4soa.service.behavior:Sequence">
    <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="RequestForQuote:rfqReq" channelName="channel1"
operationName="requestForQuote" serviceType="ServiceB"
channelType="channelType1"/>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="RequestForQuote:rfqResp" channelName="channel1"
operationName="requestForQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
    <activityTypes xsi:type="org.pi4soa.service.behavior:While">
     <preConditions
xsi:type="org.pi4soa.service.behavior:LookaheadMessage"
messageActivity="//@initialBehaviorDescription/@activityTypes.0/@activity
Types.2/@activityTypes.0"/>
     <postConditions
xsi:type="org.pi4soa.service.behavior:LookaheadMessage"
messageActivity="//@initialBehaviorDescription/@activityTypes.0/@activity
Types.3"/>
```

```
      <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="UpdateQuote:updatedQuoteReq" channelName="channel1"
operationName="updateQuote" serviceType="ServiceB"
channelType="channelType1"/>
      <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="UpdateQuote:updatedQuoteResp" channelName="channel1"
operationName="updateQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
      <reEvaluateCondition
xsi:type="org.pi4soa.service.behavior:XPathCondition" expression="true"/>
    </activityTypes>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="AcceptQuote:acceptReq" channelName="channel1"
operationName="acceptQuote" serviceType="ServiceB"
channelType="channelType1"/>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="AcceptQuote:acceptResp" channelName="channel1"
operationName="acceptQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
   </activityTypes>
 </initialBehaviorDescription>
 <serviceTypes name="ServiceB" serviceProvider="true">
  <operations name="requestForQuote">
   <request canInitiateSession="true"/>
   <response/>
  </operations>
  <operations name="updateQuote">
   <request canInitiateSession="true"/>
   <response/>
  </operations>
  <operations name="acceptQuote">
   <request canInitiateSession="true"/>
   <response/>
  </operations>
 </serviceTypes>
</org.pi4soa.service.behavior:ServiceDescription>
```

Invalid Scenario

```
<?xml version="1.0" encoding="ASCII"?>
<org.pi4soa.cdl:Package xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:org.pi4soa.cdl="http:///org/pi4soa/cdl.ecore" name="buyer"
author="Sundar6" version="1"
targetNamespace="http://www.cs.uwindsor.ca/sundar6" participant="buyer">
<org.pi4soa.service.behavior:ServiceDescription xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
```

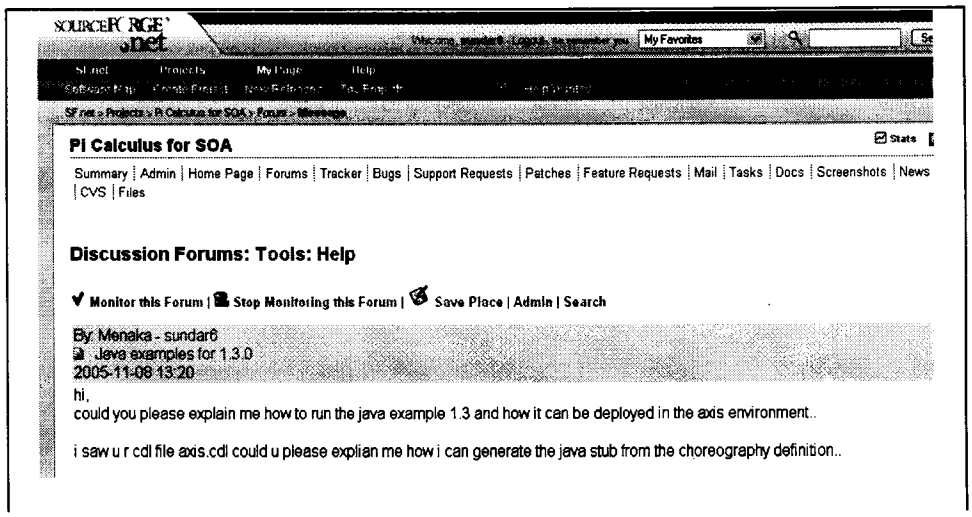
```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:org.pi4soa.service.behavior="http:///org/pi4soa/service/behavior.ecore"
name="simplerfq" version="1.1" participant="Buyer">
 <initialBehaviorDescription name="ExampleRFQPattern">
  <activityTypes xsi:type="org.pi4soa.service.behavior:Sequence">
   <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="RequestForQuote:rfqReq" channelName="channel1"
operationName="requestForQuote" serviceType="ServiceB"
channelType="channelType1"/>
   <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="RequestForQuote:rfqResp" channelName="channel1"
operationName="requestForQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
   <activityTypes xsi:type="org.pi4soa.service.behavior:While">
    <preConditions
xsi:type="org.pi4soa.service.behavior:LookaheadMessage"
messageActivity="//@initialBehaviorDescription/@activityTypes.0/@activity
Types.2/@activityTypes.0"/>
    <postConditions
xsi:type="org.pi4soa.service.behavior:LookaheadMessage"
messageActivity="//@initialBehaviorDescription/@activityTypes.0/@activity
Types.3"/>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="UpdateQuote:updatedQuoteReq" channelName="channel1"
operationName="updateQuote" serviceType="ServiceB"
channelType="channelType1"/>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="UpdateQuote:updatedQuoteResp" channelName="channel1"
operationName="updateQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
    <reEvaluateCondition
xsi:type="org.pi4soa.service.behavior:XPathCondition" expression="true"/>
   </activityTypes>
   <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="AcceptQuote:acceptReq" channelName="channel1"
operationName="acceptQuote" serviceType="ServiceB"
channelType="channelType1"/>
   <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="AcceptQuote:acceptResp" channelName="channel1"
operationName="acceptQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
  </activityTypes>
 </initialBehaviorDescription>
 <serviceTypes name="ServiceB">
  <operations name="requestForQuote">
   <request canInitiateSession="true"/>
   <response/>
105 </operations>
```

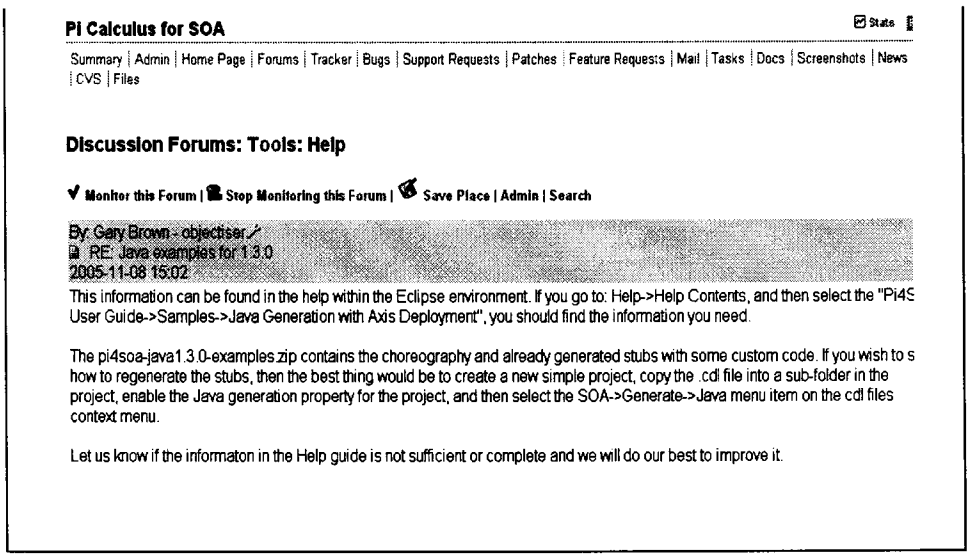
```
    <operations name="updateQuote">
      <request canInitiateSession="true"/>
      <response/>
    </operations>
    <operations name="acceptQuote">
      <request canInitiateSession="true"/>
      <response/>
    </operations>
  </serviceTypes>
</org.pi4soa.service.behavior:ServiceDescription>

<?xml version="1.0" encoding="ASCII"?>
<org.pi4soa.cdl:Package xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:org.pi4soa.cdl="http:///org/pi4soa/cdl.ecore" name="seller"
author="Sundar6" version="1"
targetNamespace="http://www.cs.uwindsor.ca/sundar6" participant="seller">
  <initialBehaviorDescription name="ExampleRFQPattern">
    <activityTypes xsi:type="org.pi4soa.service.behavior:Sequence">
      <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="RequestForQuote:rfqReq" channelName="channel1"
operationName="requestForQuote" serviceType="ServiceB"
channelType="channelType1"/>
      <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="RequestForQuote:rfqResp" channelName="channel1"
operationName="requestForQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
      <activityTypes xsi:type="org.pi4soa.service.behavior:While">
        <preConditions
xsi:type="org.pi4soa.service.behavior:LookaheadMessage"
messageActivity="//@initialBehaviorDescription/@activityTypes.0/@activity
Types.2/@activityTypes.0"/>
        <postConditions
xsi:type="org.pi4soa.service.behavior:LookaheadMessage"
messageActivity="//@initialBehaviorDescription/@activityTypes.0/@activity
Types.3"/>
        <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="UpdateQuote:updatedQuoteReq" channelName="channel1"
operationName="updateQuote" serviceType="ServiceB"
channelType="channelType1"/>
        <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="UpdateQuote:updatedQuoteResp" channelName="channel1"
operationName="updateQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
        <reEvaluateCondition
xsi:type="org.pi4soa.service.behavior:XPathCondition" expression="true"/>
      </activityTypes>
```

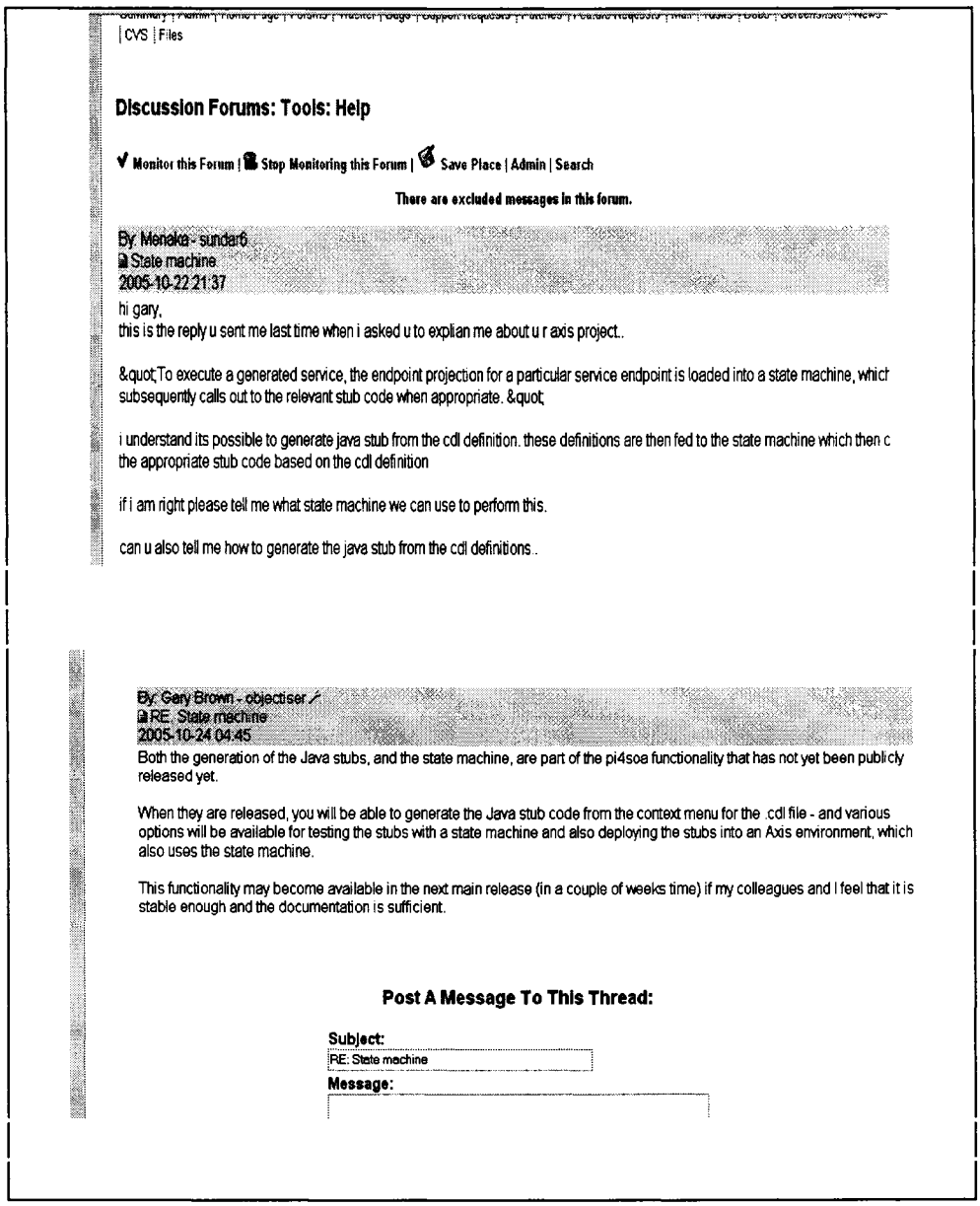
```
    <activityTypes xsi:type="org.pi4soa.service.behavior:Receive"
label="AcceptQuote:acceptReq" channelName="channel1"
operationName="acceptQuote" serviceType="ServiceB"
channelType="channelType1"/>
    <activityTypes xsi:type="org.pi4soa.service.behavior:Send"
label="AcceptQuote:acceptResp" channelName="channel1"
operationName="acceptQuote" serviceType="ServiceB"
messageType="Response" channelType="channelType1"/>
  </activityTypes>
 </initialBehaviorDescription>
 <serviceTypes name="ServiceB" serviceProvider="true">
  <operations name="requestForQuote">
   <request canInitiateSession="true"/>
   <response/>
  </operations>
  <operations name="updateQuote">
   <request canInitiateSession="true"/>
   <response/>
  </operations>
  <operations name="acceptQuote">
   <request canInitiateSession="true"/>
   <response/>
  </operations>
 </serviceTypes>
</org.pi4soa.service.behavior:ServiceDescription>
```

C: Technical Communication

This section includes some of the technical emails between W3C Choreography Team members and me. It also includes parts of my communications with pi4soa project administrator in their official forum at thesourceforge.net web site.







| CVS | Files

Discussion Forums: Tools: Help

Monitor this Forum | Stop Monitoring this Forum | Save Place | Admin | Search

There are excluded messages in this forum.

By: Menaka - sundar6
State machine
2005-10-22 21:37

hi gary,
this is the reply u sent me last time when i asked u to explian me about u r axis project..

"To execute a generated service, the endpoint projection for a particular service endpoint is loaded into a state machine, whicl subsequently calls out to the relevant stub code when appropriate. "

i understand its possible to generate java stub from the cdl definition. these definitions are then fed to the state machine which then c the appropriate stub code based on the cdl definition

if i am right please tell me what state machine we can use to perform this.

can u also tell me how to generate the java stub from the cdl definitions..

By: Gery Brown - objectiser
RE: State machine
2005-10-24 04:45

Both the generation of the Java stubs, and the state machine, are part of the pi4soa functionality that has not yet been publicly released yet.

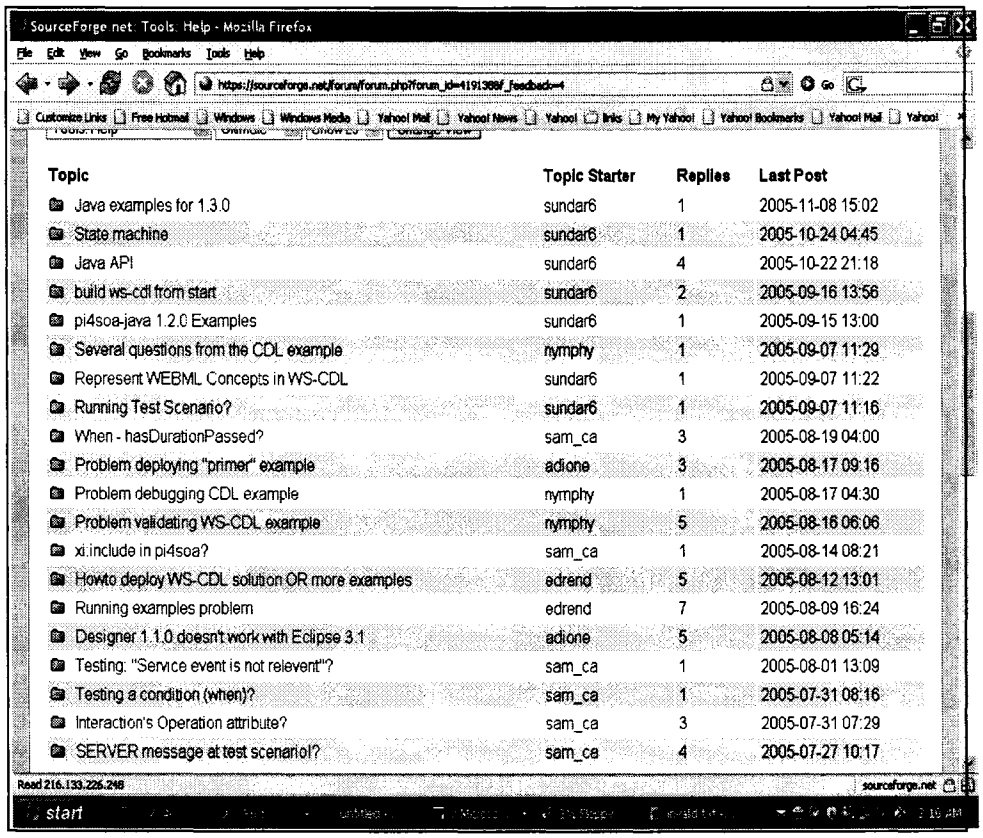
When they are released, you will be able to generate the Java stub code from the context menu for the .cdl file - and various options will be available for testing the stubs with a state machine and also deploying the stubs into an Axis environment, which also uses the state machine.

This functionality may become available in the next main release (in a couple of weeks time) if my colleagues and I feel that it is stable enough and the documentation is sufficient.

Post A Message To This Thread:

Subject:
RE: State machine
Message:

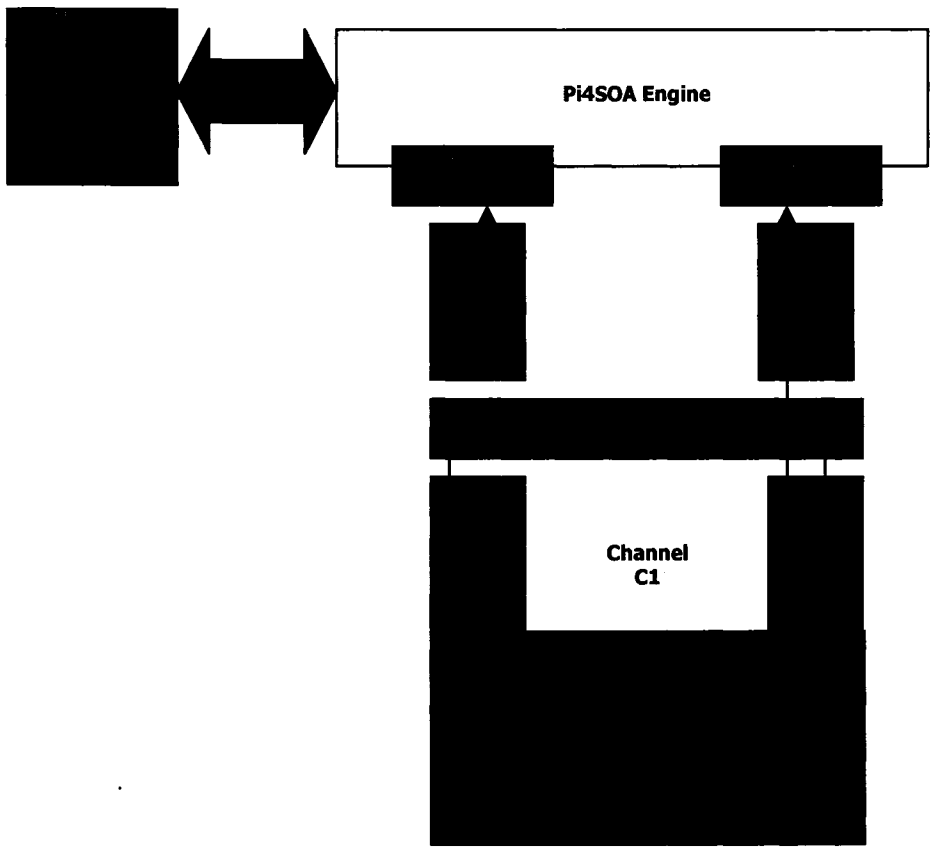
I have used the name "Sundar6" in all my emails. The following is the list of questions I have posted.



D: Algorithmic Details

This gives us the algorithmic details or the step-by-step procedure about the way in which we have implemented our whole model. Each step is explained in detail in words in each chapter. We have tested and verified for the chorography layer in our model. This choreography layer is the inventory of all business process.

Sample Input: Invalid.Scenatio File



Choreography Execution Steps:

1.Parse the Scenario File

2.Verify and retrieve the associated choreography files

3.Read the choreography file to retrieve the details about the message to be sent

4.Send message according to the choreography file & Receive message according to the choreography file.

Sample Output:

INFO: <processing id="null" text="Test Scenario [name=null,description=null]" />
Jan 20, 2006 2:29:00 AM org.pi4soa.service.test.ScenarioTester info
INFO: <processing id="1" text="Event Group [description=Initial request for quote messages]" />
Jan 20, 2006 2:29:00 AM org.pi4soa.service.test.ScenarioTester info
INFO: <processing id="1/0" text="Message event" />
Jan 20, 2006 2:29:00 AM org.pi4soa.service.tracker.impl.LoggerServiceTrackerImpl record
INFO: <serviceStarted name="simplerfq" participant="Buyer" version="1.1" sessionId="simplerfq0" />
Jan 20, 2006 2:29:00 AM org.pi4soa.service.tracker.impl.LoggerServiceTrackerImpl record
INFO: <sentMessage sessionId="simplerfq0" ><message operation="requestForQuote" type="request" serviceType="null" serviceURL="ref[url=null]" /><channel name="channel1" ><identity>c1</identity></channel></sentMessage>
Jan 20, 2006 2:29:00 AM org.pi4soa.service.test.ScenarioTester info
INFO: <completed id="1/0" text="Message handled" />
Jan 20, 2006 2:29:00 AM org.pi4soa.service.test.ScenarioTester info
INFO: <processing id="1/1" text="Message event" />
Jan 20, 2006 2:29:00 AM org.pi4soa.service.tracker.impl.LoggerServiceTrackerImpl record
INFO: <serviceStarted name="simplerfq" participant="Seller" version="1.1" sessionId="simplerfq1" />
Jan 20, 2006 2:29:00 AM org.pi4soa.service.tracker.impl.LoggerServiceTrackerImpl record
INFO: <receivedMessage sessionId="simplerfq1" ><message operation="requestForQuote" type="request" serviceType="null" serviceURL="ref[url=null]" /><channel name="channel1" ><identity>c1</identity></channel></receivedMessage>
Jan 20, 2006 2:29:00 AM org.pi4soa.service.test.ScenarioTester info

INFO: <completed id="1/1" text="Message handled" />
Jan 20, 2006 2:29:00 AM org.pi4soa.service.test.ScenarioTester info
INFO: <processing id="1/2" text="Message event" />
Jan 20, 2006 2:29:00 AM
org.pi4soa.service.tracker.impl.LoggerServiceTrackerImpl record
INFO: <sentMessage sessionId="simplerfq1" ><message
operation="requestForQuote" type="response" serviceType="null"
serviceURL="ref[url=null]" /><channel name="channel1"
><identity>c1</identity></channel></sentMessage>
Jan 20, 2006 2:29:00 AM org.pi4soa.service.test.ScenarioTester info
INFO: <completed id="1/2" text="Message handled" />
Jan 20, 2006 2:29:00 AM org.pi4soa.service.test.ScenarioTester info
INFO: <processing id="1/3" text="Message event" />
Jan 20, 2006 2:29:00 AM
org.pi4soa.service.tracker.impl.LoggerServiceTrackerImpl record
INFO: <receivedMessage sessionId="simplerfq0" ><message
operation="requestForQuote" type="response" serviceType="null"
serviceURL="ref[url=null]" /><channel name="channel1"
><identity>c1</identity></channel></receivedMessage>
Jan 20, 2006 2:29:00 AM org.pi4soa.service.test.ScenarioTester info
INFO: <completed id="1/3" text="Message handled" />
Jan 20, 2006 2:29:00 AM org.pi4soa.service.test.ScenarioTester info
INFO: <completed id="1" text="Event Group [description=Initial request for
quote messages]" />

Note:
The complete Scenario File will execute the choreography file and the messages are sent and received as per the instructions in the choreography file.

Thus the entire process can be modeled in the form of a request and response.

Vita Auctoris

Name:	Menaka Sundaravadanm
Place OF Birth:	Chennai, India
Year OF Birth:	1981
Education:	University of Windsor Ontario, Canada Department of Computer Science Master of Science (2003-2006)
	University of Madras Chennai, India Department of Computer Science & Engineering Bachelor of Science (1998-2002)