University of Crete

School of Sciences and Engineering

Computer Science Department

Applying Service Oriented Architecture Techniques to facilitate RosettaNet-Driven Business Interactions

By Evangelos Papathanasiou

Master's Thesis

Heraklion, October 2011

University of Crete Computer Science Department

Applying Service Oriented Architecture Techniques to facilitate RosettaNet-Driven Business Interactions

By

EVANGELOS PAPATHANASIOU

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science

Author: ____

Evangelos Papathanasiou, Department of Computer Science

Heraklion, October 2011

Applying Service Oriented Architecture Techniques to facilitate RosettaNet-Driven Business Interactions

Evangelos Papathanasiou

Master Thesis

University of Crete – School of Sciences and Engineering

Computer Science Department

Abstract

Over the past years, modern businesses rely on the Internet because they need high level of interoperability, which enables them to conduct their business transactions in a dynamic and efficient way, without the need for the development of private or special solutions for all commercial cooperation. This interoperability focuses on the automation of business processes in the supply chains. It provides the ability to find easily and quickly potential business partners, information on their services and business processes they support. It also concludes short-term business relationships that involve making a specific number of electronic transactions. Achieving interoperability is only possible by applying inter-enterprise frameworks or B2B frameworks are commonly called. The most common B2B framework is provided by RosettaNet. RosettaNet standards allow manufacturers, distributors, sellers and end users to benefit from the Internet to exchange business documents across the entire supply chain on a global basis. RosettaNet framework focuses on the development of common public business processes are called Partner Interface Processes (PIPs). PIP is a specific sequence of steps required to execute a transaction between two partners in the supply chain. It is essentially a dialogue, based on XML, conducted between systems. RosettaNet business interactions will be modeled and implemented based on Service Oriented Architecture techniques. Business processes arise from PIPs are designed and modeled in BPMN by a provided empirical mapping. PIPs interactions are developed as web services. The use of more than one PIP web services is developed as Choreography and is implemented in BPEL. The use of Web Services for implementing PIPs will lead to more flexible interactions between organizations, as new services will easily replace the existing ones.

Εφαρμόζοντας Service Oriented Architecture Τεχνικές για τη διευκόλυνση τωνεπιχειρησιακων αλληλεπιδράσεων με βάση την RosettaNet

Ευάγγελος Παπαθανασίου

Μεταπτυχιακή Εργασία

Πανεπιστήμιο Κρήτης – Σχολή Θετικών & Τεχνολογικών Επιστημών

Τμήμα Επιστήμης Υπολογιστών

Περίληψη

Τα τελευταία χρόνια, σύγχρονες επιχειρήσεις βασίζονται στο Διαδίκτυο, επειδή χρειάζονται υψηλό επίπεδο διαλειτουργικότητας, η οποία τους επιτρέπει να διεξάγουν τις επιχειρηματικές τους συναλλαγές με ένα δυναμικό και αποτελεσματικό τρόπο, χωρίς την ανάγκη για την ανάπτυξη του ιδιωτικού ή ειδικές λύσεις για κάθε εμπορική συνεργασία. Η διαλειτουργικότητα επικεντρώνεται στην αυτοματοποίηση των επιχειρηματικών διαδικασιών στις αλυσίδες εφοδιασμού. Παρέχει τη δυνατότητα να βρίσκουν εύκολα και γρήγορα πιθανούς επιχειρηματικούς εταίρους, πληροφορίες για τις υπηρεσίες τους και τις επιχειρηματικές διαδικασίες που υποστηρίζουν. Επίσης συνάγειβραχυπρόθεσμες επιχειρηματικές σχέσεις που να συνεπάγονται ένα συγκεκριμένο αριθμό των ηλεκτρονικών συναλλαγών. Η επίτευξη της διαλειτουργικότητας είναι δυνατή μόνο με την εφαρμογή διεπιχειρησιακών πλαισίων ή B2B πλαισίων όπως συνήθως αποκαλούνται. Το πιο κοινό B2B πλαίσιο παρέχεται από την RosettaNet. Τα RosettaNet πρότυπα επιτρέπουν στους κατασκευαστές, διανομείς, πωλητές και τελικούς χρήστες να επωφελούνται από το Διαδίκτυο για την ανταλλαγή επιχειρηματικών εγγράφων σε όλη την εφοδιαστική αλυσίδα σε παγκόσμιο επίπεδο. Το RosettaNet πλαίσιο επικεντρώνεται στην ανάπτυξη δημόσιων κοινών επιχειρηματικών διαδικασιών που καλούνται Partner Interface Processes (PIPs). Η PIP είναι μια συγκεκριμένη ακολουθία βημάτων που απαιτούνται για την εκτέλεση μιας συναλλαγής μεταξύ των δύο εταίρων στην αλυσίδα εφοδιασμού. Πρόκειται ουσιαστικά για ένα διάλογο, με βάση την XML, η οποία διεξάγεται μεταξύ των συστημάτων. Οι RosettaNet αλληλεπιδράσεις των επιχειρήσεων έχουν διαμορφωθεί και υλοποιηθεί με βάση Service Oriented Architecture τεχνικές. Οι επιχειρηματικές διεργασίες προκύπτουν από PIPs έχουν σχεδιαστεί και μοντελοποιηθεί με BPMN σύμφωνα με μία εμπειρική μεθοδολογία που παρέχεται. Οι PIPs αλληλεπιδράσεις έχουν αναπτυχθεί ως web services. Η χρήση περισσοτέρων του ενός web services από PIP αναπτύσσονται ως χορογραφία και υλοποιούνται σε BPEL. Η χρήση των Web Services για την υλοποίηση PIPs θα οδηγήσει σε πιο ευέλικτες αλληλεπιδράσεις μεταξύ των οργανισμών, αφού οι νέες υπηρεσίες θα μπορούν να αντικαταστήσουν εύκολα τις ήδη υπάρχοντες.

Table of Contents

Table of Figures

Table of Tables

Acknowledgements

I would like to thank my supervisor Christos Nikolaou for his assistance and support through this thesis work. I would also like to thank both Prof. D. Plexousakis and Researcher FORTH-ICS K. Maggoutis for being members of my board of enquiry, for reading my work and providing valuable comments and feedback about it.

Special thanks to my friend and colleague, Mariana Karmazi, for her invaluable support in the last months of this work. I would like also to thank my friend Manolis Daskalakis that stayed by my side in the difficult times, and Pantelis Petridis for his piece of advice that provide me in last months. I also wish to thank my close friends for their love and patience over the past few years and for helping me pull through hard times. Finally, I would like to thank the members of the Transformation Services Laboratory.

I dedicate this work to my family. Their love and belief in me have been driving force in my life.

1. Introduction

The growth of the Web is revolutionizing the way businesses interact with their partners and customers. Millions of organizations are moving or have already moved their main operations to the Web to take advantage of the potential of more automation, efficient business processes, and global visibility [1]. The Web offers a unique opportunity for businesses to take a central stage in the fast growing economy [2]. In this business environment, innovation of organizational processes and products is a major business challenge and critical for firm success [3]. In the past, business organizations focused on reducing costs and improving quality to gain a competitive advantage. Today, companies must be able to innovate at the global frontier and create and commercialize a stream of new products and processes that shift the technology frontier, progressing as fast as their rivals catch up [4].

Numerous organizations started using the Web as a means to automate relationships with their business partners. B2B relationships have elicited the formation of alliances in which businesses joined their applications, databases, and systems to share costs, skills and resources in offering value-added services. The ultimate goal for businesses is therefore to have inter- and intra-enterprise applications evolve independently, yet allow them to effectively and conveniently use each other's functionality.

An important challenge in B2B is interaction. Interaction is defined as consisting of interoperation and integration with both internal and external enterprise applications. This has been a central concern because B2B applications are composed of autonomous, heterogeneous, and distributed components. This interoperability focuses on the automation of business processes to the supply chains, and the ability to easily and quickly find potential business partners, information on their services and business processes they support, and conclude short-term business relationships that involve making a specific number of electronic transactions [5].

However, the achievement of interoperability is possible only by application of standard inter-enterprise frameworks - or B2B frameworks such as are commonly called. In recent years a large number of B2B frameworks have been developed, most of which are based on XML [6]. The dynamics of this language lies in the exceptional opportunities offered to represent data in a structured manner that is understandable by both humans and from machines. This is very crucial because the B2B frameworks are trying not only to standardize the form of commercial documents and messages, but also to describe and automate entire business processes.

Such a framework, which is studied on this thesis, is provided by RosettaNet[7]. RosettaNet aims at standardizing product descriptions and business processes in information technology supply chain applications and connectors). RosettaNet focuses on three key areas of standardization to automate B2B interactions. First, the vocabulary needs to be aligned. Second, the way in which business messages are wrapped and transported must be specified. Third, the business process governing the interchange of the business messages themselves must be harmonized and specified. RosettaNet’s PIPs (Partner Interface Processes) are pre-defined XML-based conversations. A conversation consists of a set of business documents (e.g., purchase order, purchase order acknowledgment) and message exchange logic (e.g., the sequencing of the actions that take place during a product quote request). A PIP is defined using a combination of textual and graphical (UML-based state machine) representations. At the communication layer, common Internet transport protocols are supported. At the content layer, RosettaNet uses an XML-based schema as document content model. The use of a vertical ontology (i.e, common vocabulary with information technology supply chain domain) contributes to solving the problem of semantic heterogeneity. At the business process layer, RosettaNet focuses on providing a common basis for B2B public interactions via PIPs. Partners perform the integration of PIPs with internal business processes.

However, nowadays there is not any open source implementation of RosettaNet PIPS. The use of RosettaNet PIPS stays between the organizations, which decide which PIPs to use to fulfill their inner needs. Based on the information and the guidelines are retrieved by RosettaNet, this thesis focuses on designing and developing RosettaNet PIPs based on the Web services paradigm and principles of the Service Oriented Architecture (SOA). The thesis proposes a practical methodology to model RosettaNet PIPs in BPMN. Moreover, it proposes the implementation of the exchange of messages between business entities of each as Web Services. Finally it demonstrates the exposed Web Services as BPEL Orchestrations and Choreographies in which the business entities exchange messages based on PIPS in order to complete their transactions.

INTRODUCTION

The thesis target is to provide an implementation of RosettaNet PIPs. This implementation is divided into two areas. The first area is focusing on the business connections, procedures and messages that exist between the participants. Examining relationships between the participants takes out the operational details and describes the way that these relationships between the participants take place. This kind of information is depicted in the business process level, where all these relationships that are described in each PIP are finally modeled in BPMN. The second area is the core implementation of RosettaNet PIPS as Web Services. The use of XML standards either for the development of RosettaNet PIPS such as WDSL and SOAP will enable systems to withstand technology evolution and changes without costly redesign or reconfiguration. Furthermore, an easy integration of newly web services that provides new and better functionality is feasible.

The solution outlined in this thesis offers significant advantages. Service-Oriented Architecture (SOA) provides a way to address problems related to the integration of heterogeneous applications in a distributed environment, as the implementation is based on existing standards such as WSDL, BPEL.

The remainder of this work is organized as follows: In the next section is presented the background theory about RosettaNet Standards, focusing on RosettaNet PIPs structure. In Chapter 3, the Business Process Modeling is discussed, concentrating on the BPMN‘s standard Principles. Chapter 4 introduces the Web service technologies by describing prevailing standards in the Web services area such as WSDL, SOAP, UDDI and BPEL, standards that this thesis is based on. It describes Service Oriented Architecture. In Chapter 5, an experiential methodology introduced in order to model RosettaNet PIPs in BPMN. Chapter 6 demonstrates the implementation of RosettaNet PIPs as Web Services. It provides the methodology is followed to implement such services. In Chapter 7, an Automobile scenario is described based on the implemented RosettaNet PIPs web services. It demonstrates a composition of web services that implement three orchestrations, which exchange messages between them as a choreography. The last chapter discusses the work done in this master thesis and proposes future work.

2. RosettaNet Framework's Perspective

Before we demonstrate the work has that been done in this thesis, we should understand the “tools”, which were used in order to accomplish this subject. This chapter introduces the readers with the ideas and the perspective of very popular standards related to supply chain, business management processing and information technology, and RosettaNet standards. RosettaNet standards are briefly described in the following sections. Great emphasis is placed on RosettaNet PIPs, which is the major building material for this thesis.

2.1. RosettaNet Consortium

RosettaNet is an independent, self-funded, non-profit consortium of companies, which was founded in June 1998 by major Computer and Consumer Electronics, Electronic Components, Semiconductor Manufacturing, Telecommunications and Logistics companies. Dedicated to develop open standards for e-commerce, those integrate business processes between supply chain partners in high technologies. RosettaNet standards enable manufacturers, distributors, sellers and end users to benefit from the Internet to exchange business documents to along the entire supply chain on and a global basis. The RosettaNet standards are based on XML and define message guidelines, interfaces for business processes, and implementation frameworks for interactions between companies. Mostly addressed is the supply chain area, but also manufacturing, product and material data and service processes are in scope.

The association of Rosettanet includes companies such as IBM, Netscape, Oracle, SAP, Cisco Systems, Intel, Compaq, etc. The modeling activity of RosettaNet is divided into three sections relating to standards for data formatting in operational procedures and protocols for exchanging messages. In this chapter we will examine in details each of these three modeling divisions that constitute the RosettaNet framework [7].

2.2. RosettaNet standards

Users for users develop RosettaNet standards, referred as Partner Interface Process (PIP). RosettaNet messages are global, universally accepted, and proven through successful implementations. They also provide a common language for electronic business transactions and the foundation for integrating critical processes among partners within the global trading network.

RosettaNet standards allow trading partners to improve communication with each other as well as offer more collaboration with the exchange of business information electronically. In particular, supply chain companies are able to facilitate speed, efficiency, and reliability, while reducing company costs. Through RosettaNet standards, these trading partners are able to realize gains associated with reducing inventory, improving order-processing time and customer satisfaction with regard to product cycle time-to-market. RosettaNet industry standards provide business frameworks that allow individual companies to enhance the interoperability of business processes across the global supply chain. These standards transcend proprietary solutions in the marketplace. In fact, RosettaNet leverages existing protocols, guidelines and specifications to quickly create standards for efficient business communication across multiple platforms, applications and networks. RosettaNet standards are global and open. They prescribe how to implement collaborative business processes between supply-chain trading partners using networked applications. These specifications include the business process definitions and technical elements for interoperability and communication. To determine the data format, the RosettaNet framework includes two dictionaries, a dictionary of operational conditions (Business Dictionary) and a dictionary of technical terms (Technical Dictionary). These dictionaries define a common set of properties for standard operational procedures of the relevant commercial documents .The sequences of steps occurring in the performance of operational processes between partner companies specified in RosettaNet framework through Partner Interface Processes or PIPs. The sequences are also provided and characterized as Business Process Choreography. Examples of such processes are the order management, distribution of information about new products, etc. In addition to choreography, a RosettaNet PIP includes the specifications for the structure and content of business documents exchanged (XML DTDs [8] and guidelines), and the restrictions imposed on interactions between trading partners. These restrictions relate to safety and performance and timing. It should be noted that PIPs are public proceedings (public processes) as related only to interactions between trading partners. Today there are about 100 PIPs defined. The PIPs are at an implementation level represented by DTD and XSD [9] files. This improves the possibilities for building and validation using standard XML techniques.[10] Finally, the RosettaNet includes RosettaNet Implementation Framework or RNIF [11], which provides the technical infrastructure required for the conduct of trade, namely the implementation and execution of PIPs. RNIF determines the form and content of messages and safety measures

RosettaNet Framework's Perspective

Figure 1 shows the three different levels of the RosettaNet standard.

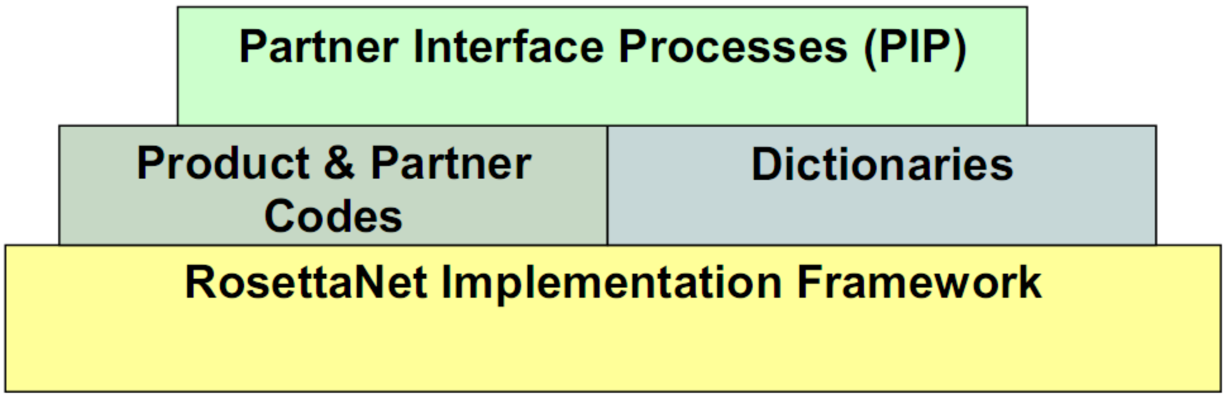

Partner Interface Processes (PIP)
Product & Partner Codes
Dictionaries
RosettaNet Implementation Framework

Figure 1: Overview of the RosettaNet Standard.

A short introduction to the building blocks is presented below followed by a more detailed description of the parts [12]:

- PIPs

RosettaNet Partner Interface Processes® (PIPs®) define business processes between trading partners.

- PIP Directory

The PIP® Directory provides you with faster access to the PIP information you are seeking.

- RosettaNet Implementation Framework

The RosettaNet Implementation Framework (RNIF) Core Specification is the packaging, routing, and transport of all PIP® messages and business signals.

- Dictionaries

RosettaNet dictionaries provide a common set of properties for PIPs®. The RosettaNet Business Dictionary designates the properties used in basic business activities. RosettaNet Technical Dictionaries provide properties for defining products. RosettaNet Business Dictionary: designates the properties used in basic business activities. It defines the Business Properties, Business Data Entities and, Fundamental Business Data, Entities in PIP Messages. RosettaNet Technical Dictionary (RNTD): provide properties for defining products. This dictionary, coupled with the RosettaNet Business Dictionary, provides a common vocabulary for conducting e-business, eliminating confusion in the procurement process due to companies' uniquely defined terminology. The RNTD eliminates the need for partners to utilize separate dictionaries when implementing multiple PIPs, allowing it to be used in a variety of supply chain applications.

- Product & Partner Codes

Product and partner codes in RosettaNet standards expedite the alignment of business processes between trading partners.

- Engineering Information Management

Engineering Information Management Foundational Program was formed with the initiation to develop technical specifications that enable efficient implementation of EIM business processes based on the combination of RosettaNet Automated Enablement (RAE) and RosettaNet Dictionary (RNTD, RDA) technologies. Two specifications are created from this effort: EIPS XSD Specification and TPIR-PIP for Engineering Information Specification.

- Multiple Messaging Services

Multiple Messaging Services (MMS) addresses the support of RosettaNet XML business messages and business-to-business (B2B) collaboration over horizontal message handling systems. Web Services, AS/2 and ebMS were identified as the three pre-dominant messaging systems for which specifications have been derived for RosettaNet Business Message transport.

- Message Control and Choreography

Message Control and Choreography (MCC) enables the RosettaNet PIP architecture to support integrated multi-PIP and multi-party business interactions and to enable long-running processes in dynamic trading networks.

- Trading Partner Implementation Requirements

Trading Partner Implementation Requirements enable trading partners to constrain schema-based PIP and view, respond to and create RosettaNet PIPs without requiring backend integration.

Figure 2 contrasts electronically conducted trade with the conventional made between people, and presents the levels (green) of the contribution of the RosettaNet framework in E-business [13]. As shown in the picture is, the RosettaNet dictionaries provide a framework of words, the RNIF plays the role of grammar and PIPs represent the dialogue [14].

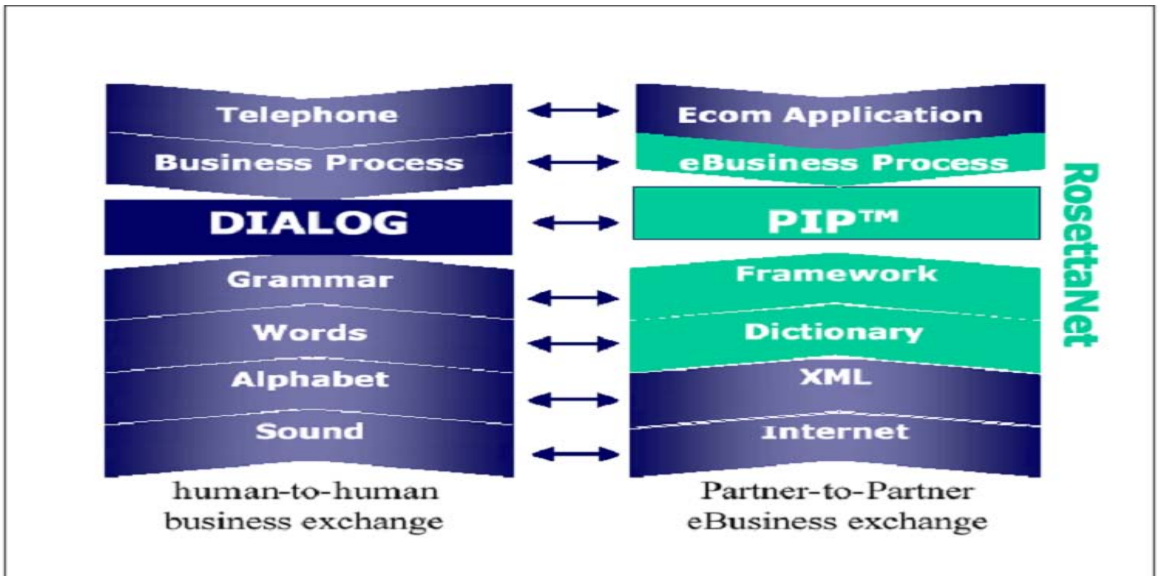


Figure 2: Contribution of RosettaNet framework at e-business

2.2.1. PIPs

A Partner Interface Process (PIP) defines business processes between trading partners. The PIPs, as shown in Figure 3, operate as a "bridge" among the private business processes [14]. PIP is a specific sequence of steps required to execute a transaction between two partners in the supply chain. It is essentially a dialogue, which is based on XML [6], conducted between systems. The specifications for each PIP, in particular, include:

- A guide specification that covers the whole PIP process (process definition, initial and final state descriptions companies' roles, limitations and monitoring activities) and on the documents related thereto;
- The XML format of the messages of PIP,
- A guide for structuring messages PIP includes the structure of the message given to him in a hierarchical form.

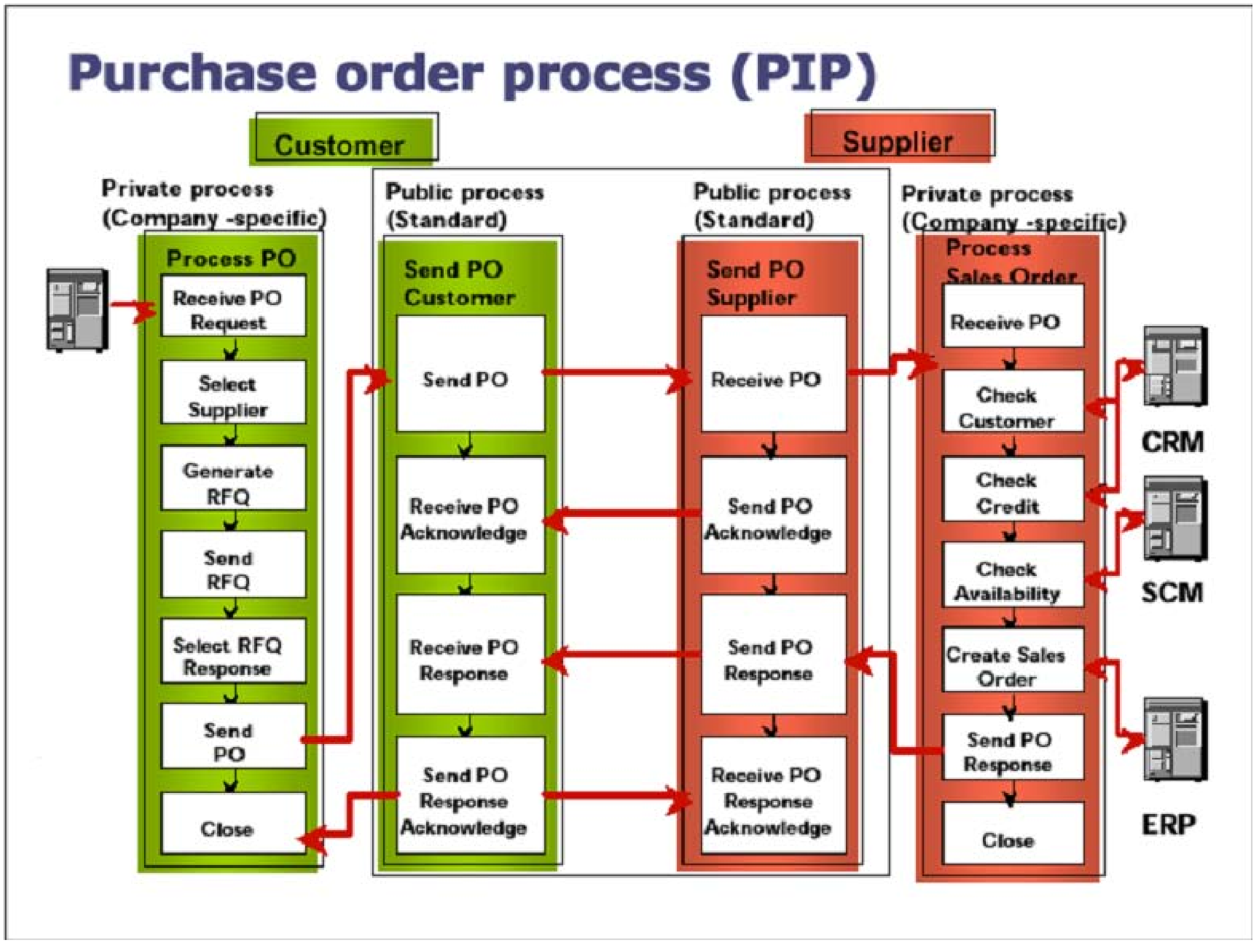


Figure 3: The RosettaNet PIPs as a bridge between private business processes.

PIPs fit into eight clusters, or groups of core business processes, that represent the backbone of the trading network. Each cluster is broken down into segments, which are cross-enterprise processes involving more than one type of trading partner. Within each segment are individual PIPs. The PIP architecture also has another two different levels, which are Activities and Actions. In these levels different aspects of the PIP is described.

According to the RosettaNet consortium there is four criteria's that a PIP has to possess [11]:

- Provide a measurable business outcome or output.
- Contain non-proprietary business processes.
- Include more than one role interaction.
- Stand as discrete units of work that can be attached and built into other PIPs to achieve a larger business outcome.

Some examples of PIPs are presented in Table 1 . All PIPs,

RosettaNet Framework’s Perspective

Website which belong in the Rosetta Net’s framework, will be easily found at Rosetta Net’s Website.

PIP Number	PIP Name
PIP 2A1	Distribute New Product Information
PIP 2A2	Query Product Information
PIP 2A9	Query EC Technical Information
PIP 3A2	Request Price and Availability
PIP 3A3	Transfer Shopping Cart
PIP 3A4	Manage Purchase Order
PIP 3A6	Distribute Order Status
PIP 3A7	Notify of Purchase Order Acceptance
PIP 3B2	Notify of Advance Shipment
PIP 3B4	Query Shipment Status
PIP 4B1	Allocate Inventory
PIP 5C1	Distribute Product List
PIP 5C2	Request Design Registration

Table 1 - Examples of PIPs

The PIPs are divided in to eight clusters each containing a different unit of B2B communication [15] which the most important we see it at Figure 4: PIPs ' Clusters:

- Cluster 0: RosettaNet Support.

It holds administrative functionality.

- Cluster 1: Partner Product and Service Review.

Holds PIPs for information gathering, maintenance, distribution for development of new business partners profiles product information subscriptions.

- Cluster 2: Product Information.

Holds PIPs for distribution and periodic updates of product and design information, including product change notices and detailed technical specifications.

- Cluster 3: Order Management.

It supports the full order-management business area, from price and delivery quoting through purchase order initiation, status reporting, and management. Order invoicing, payment, and discrepancy notification are also managed using this cluster of processes.

- Cluster 4: Inventory Management.

It enables inventory management, including collaboration, replenishment, price protection, reporting, and allocation of constrained products.

- Cluster 5: Marketing Information Management.

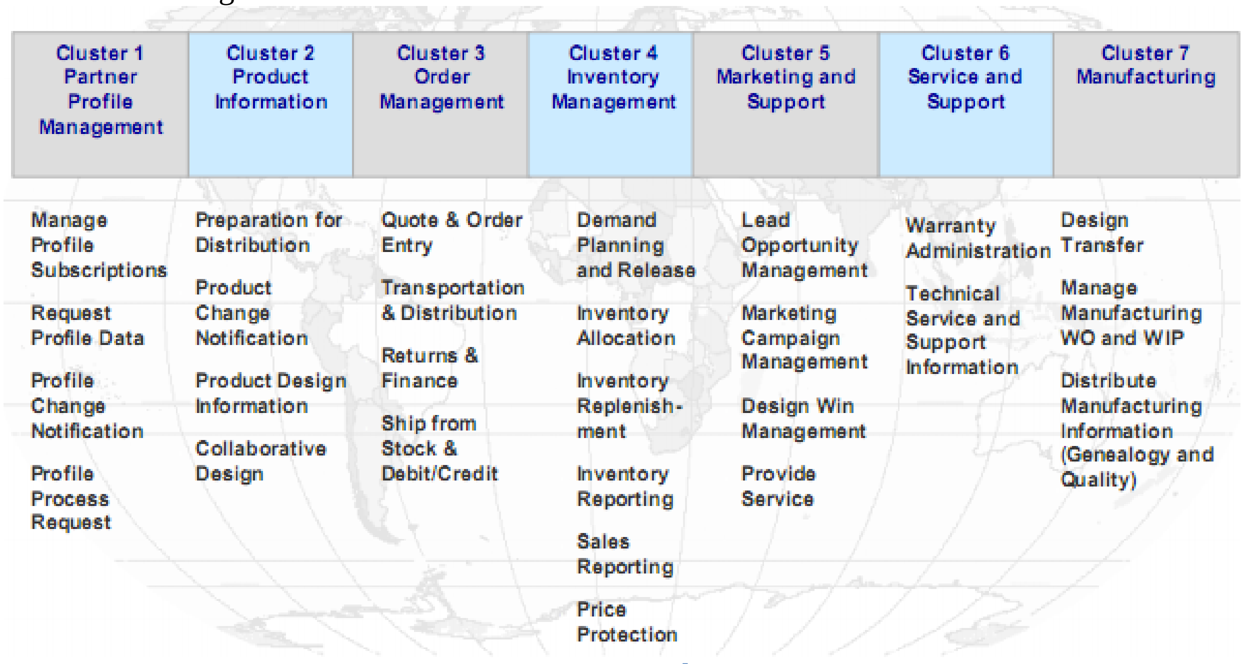
It enables communication of marketing information, including campaign plans, lead information, and design registration.

- Cluster 6: Service and Support.

It provides post-sales technical support, service warranty support, and asset management capabilities.

- Cluster 7: Manufacturing

It enables the exchange of design, configuration, process, quality, and other manufacturing floor information to support a virtual manufacturing environment.



Cluster 1 Partner Profile Management	Cluster 2 Product Information	Cluster 3 Order Management	Cluster 4 Inventory Management	Cluster 5 Marketing and Support	Cluster 6 Service and Support	Cluster 7 Manufacturing
Manage Profile Subscriptions	Preparation for Distribution	Quote & Order Entry	Demand Planning and Release	Lead Opportunity Management	Warranty Administration	Design Transfer
Request Profile Data	Product Change Notification	Transportation & Distribution	Inventory Allocation	Marketing Campaign Management	Technical Service and Support Information	Manage Manufacturing WO and WIP
Profile Change Notification	Product Design Information	Returns & Finance	Inventory Replenish-ment	Design Win Management		Distribute Manufacturing Information (Genealogy and Quality)
Profile Process Request	Collaborative Design	Ship from Stock & Debit/Credit	Inventory Reporting	Provide Service		
			Sales Reporting			
			Price Protection			

Figure 4: PIPs ' Clusters

Each cluster is then divided into two or more segments. Each segment holds PIPs with similar functionality. The following describes in detail each segment of each cluster [16]:

- Cluster 0: RosettaNet Support.

  - Segment 0A: Administrative. Notification of failure.

- Segment 0C: Testing. Asynchronous Test Notification, Asynchronous Test Request / Confirmation, Synchronous Test Notification. And Synchronous Test Query / Response.

▪ Cluster 1: Partner Product and Service Review.

- Segment 1A: Partner Review. Provides the ability to share information, such as locations and contacts, and send and receive acknowledgement of receipt.
- Segment 1B: Product and Service Review. Enables suppliers to manage product information available to partners using a subscription process as well as establish and maintain lists of products that given partners are authorized to sell and lists of partners that sell given products

▪ Cluster 2: Product Information.

- Segment 2A: Preparation for Distribution. Enables distribution of product resources, including sales catalog and basic technical information, and obtainment of extended product information.
- Segment 2B: Product Change Notification. Enables update of product resources.
- Segment 2C: Product Design Information. Enables release and update of product engineering design information.

▪ Cluster 3: Order Management

- Segment 3A: Quote and Order Entry. Allows partners to exchange price and availability information, quotes, purchase orders and order status, and enables partners to send requested orders, or shopping carts, to other partners.
- Segment 3B: Transportation and Distribution. Enables communication of shipping- and delivery-related information with the ability to make changes and handle exceptions and claims.
- Segment 3C: Returns and Finance. Provides for issuance of billing, payment and reconciliation of debits, credits and invoices between partners as well as supports product return and its financial impact.

▪ Cluster 4: Inventory Management

- Segment 4A: Collaborative Forecasting. Enables standardization of collaborative order and sales forecasting between supply-chain partners.

- Segment 4B: Inventory Allocation. Lets sellers inform buyers of product allocation determined through the evaluation of preset criteria, such as demand forecast and availability, and allows buyers to respond manually if negotiation is needed.

- Segment 4C: Inventory Reporting. Lets buyers provide daily inventory reports to sellers, allows sellers to notify buyers when reports are reflected in their records and enables sending of discrepancy reports.

- Segment 4D: Inventory Replenishment. Facilitates inventory replenishment managed by buyer, seller or both -- via a pull signal from the buyer or push notification from the seller that triggers an order, change, ship or return.

- Segment 4E: Sales Reporting. Lets buyers provide periodic sales reports to product providers, allows providers to notify buyers when reports are reflected in their records and enables sending of discrepancy reports.

▪ Cluster 5: Marketing Information Management.

- Segment 5C: Design Win Management (EC). Enables design registration with suppliers, including requests for design-win registration at different phases of the design cycle and award of win based on predefined criteria.

- Segment 5D: Ship from Stock and Debit (EC). Supports creation and use of marketing programs and distribution of pricing incentives to product distributors.

▪ Cluster 6: Service and Support

- Segment 6A: Provide and Administer Warranties, Service Packages, and Contract Services. Enables registration and product warranty support.

- Segment 6C: Technical Support and Service Management. Enables support requesters to submit request for support and provides the ability for an authorized service provider to submit a claim to the warranty provider for a completed request for support.

- Cluster 7: Manufacturing

  - Segment 7B: Manage Manufacturing WO & WIP. Enables the release, management and the exchange of factory production information.

  - Segment 7C: Distributed Manufacturing Information. Distributed manufacturing information to support product improvements, Quality and warrantee entitlement.

Thus, a PIP is characterized from a number that has three parts: a number, a letter and another number, for example 3A4. The first number indicates the team. Thus the PIP 3A4 is in the cluster 3. The letter represents the section PIP 3A4 belongs to section A in the cluster 3. The second number is a serial number that completes the number of PIP. Thus, PIP 3A4 is the fourth PIP section A for Cluster 3.

In Figure 5 is presented a public business process, which consists of seven PIPs [17]. As shown in Figure 5, a buyer asks to be informed by a vendor for the price and availability of certain products (PIP 3A2). Having received no reply, the buyer initiates an order form (PIP 3A4). The seller on the other hand, after acknowledging that he had received this request sends the invoice (PIP 3C3) to the buyer. Then the seller sends a request to transfer (PIP 3B1) to the carrier. The carrier initiates the transfer process and informs the buyer about the state of transportation (PIP 3B3). After receiving the ordered goods, the buyer sends the seller a notice of receipt (PIP 4B2). Finally, the seller prepares a report charging and sends the buyer (PIP 3C5).

RosettaNet Framework’s Perspective

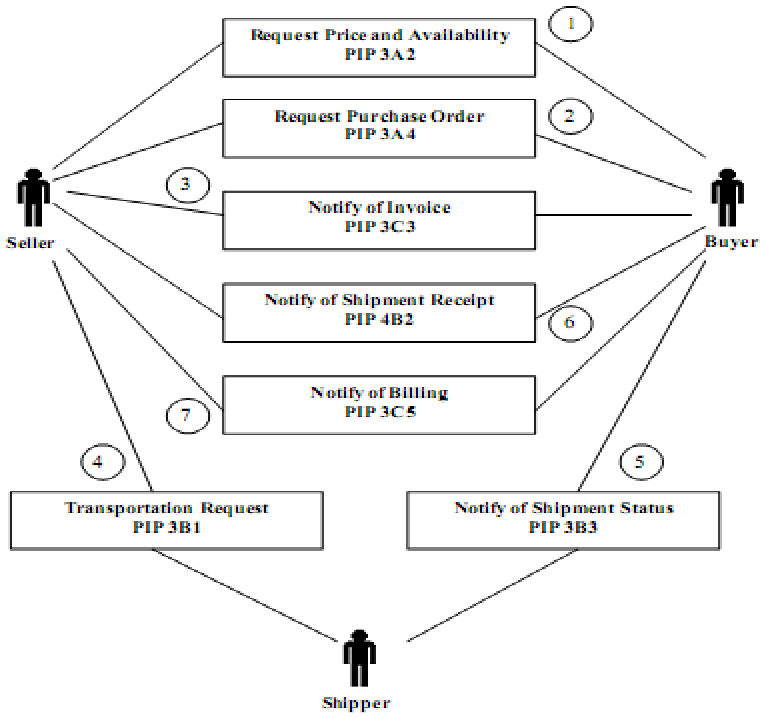


Figure 5: Example of public business process consisting of RosettaNet PIPs.

The next level is the actual PIPs that in turn are divided into activities and action. The PIPs define the dialog in a server/client-based manner. The messages exchanged between the buyer and seller is divided in to activities and actions. Below is an example of how everything is structured from cluster to action level.

For Example:

Cluster 3: Order Management

Segment A: Quote and Order Entry
PIP 3A1: Request Quote
 Activity: Request Quote
 Action: Quote Request Action
Action: Quote Confirmation

The difference between an activity and an action is that the action is the business messages sent between the different trade partners. An activity defines the function and which actions are required in that function [18].

Finally, the documentation of PIP specifications is divided in to three main parts describing the different aspects of the PIP. These parts are referred to as Business Operational View (BOV), Functional Service View (FSV) and Implementation Framework View (IFV) [18].

2.2.1.1. Business Operational View

The BOV describes a PIP from a business perspective. This includes an informal textual description of the application context of the PIP and an UML activity diagram [19]visualizing the PIP. In that diagram the roles of the business partners involved are represented by swim-lanes and an activity node is inserted for every business document to be exchanged. The first activity of such a diagram is stereotyped with a Business Transaction Type according to UMM [20] and the business documents to be exchanged are visualized as object flows [21]. Finally, the BOV specifies start and end states of a PIP execution and Business Process Activity Controls like Time to Perform for the overall PIP. The most informative part of the BOV is probably the business process flow diagram that illustrates process as a kind of state machine. The buyer and seller in the Figure 6 are referred to as partner roles [18].

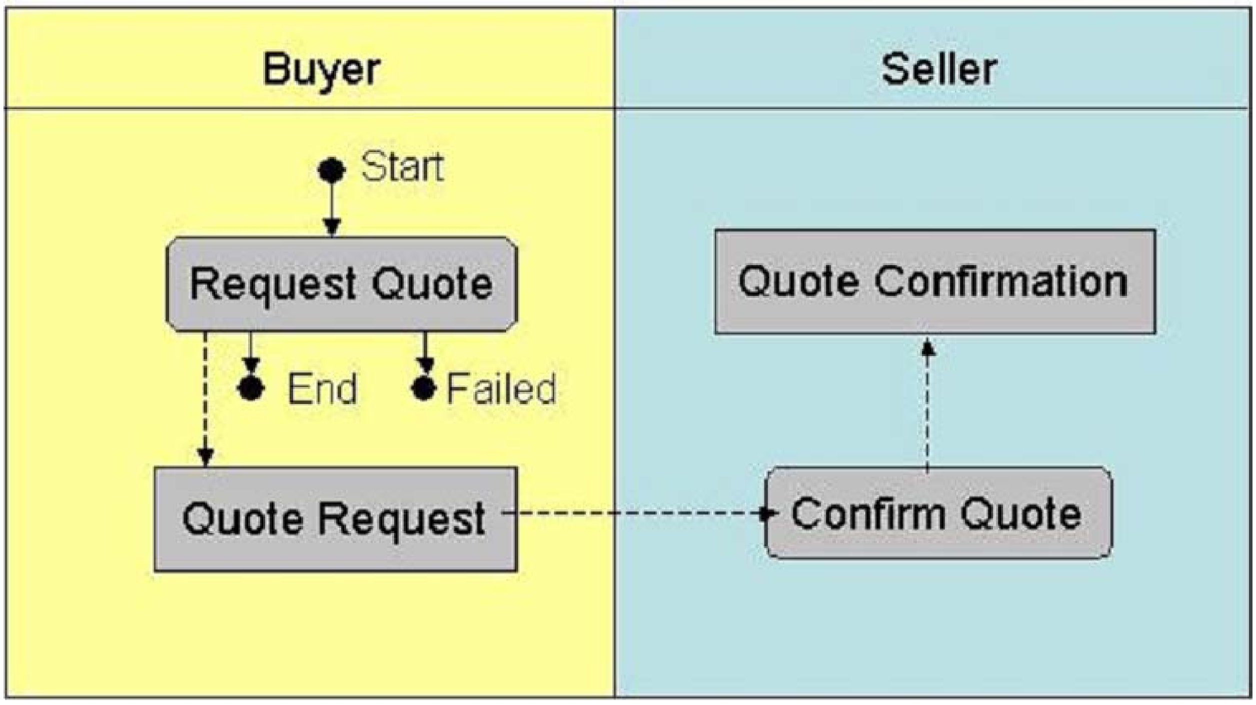


Figure 6: Business Operational View flow chart.

2.2.1.2. Functional Service View

Functional Service View (FSV) originates from a BOV and describes the interactions between the network component services in a PIP see Figure 7andFigure 8. It includes all transaction dialogs in the PIP and they are described by a network component design and a transaction dialog specification. For each role of the BOV a component is defined in the FSV that is responsible for exchanging business documents as Actions and control messages as Signals. The exchange of each message is detailed by Message Exchange Controls and finally, the intended order of message exchange is represented by an UML sequence diagram.

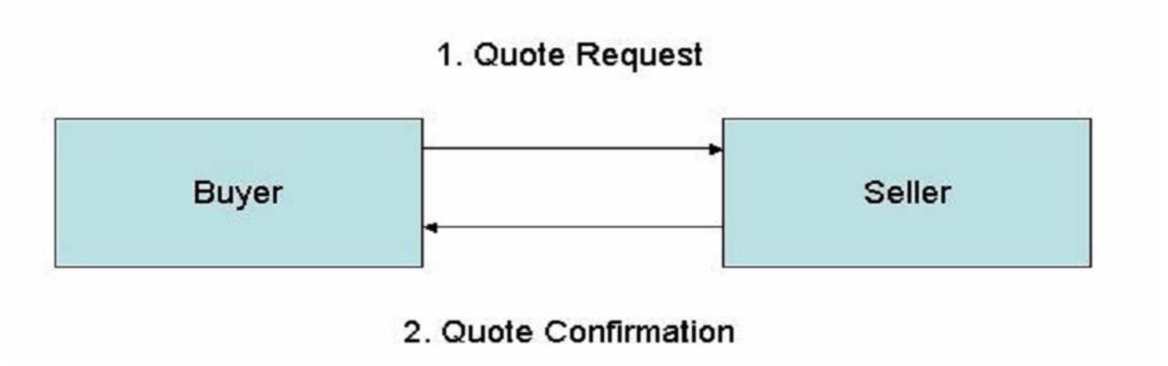


Figure 7: Functional Service View action chart.

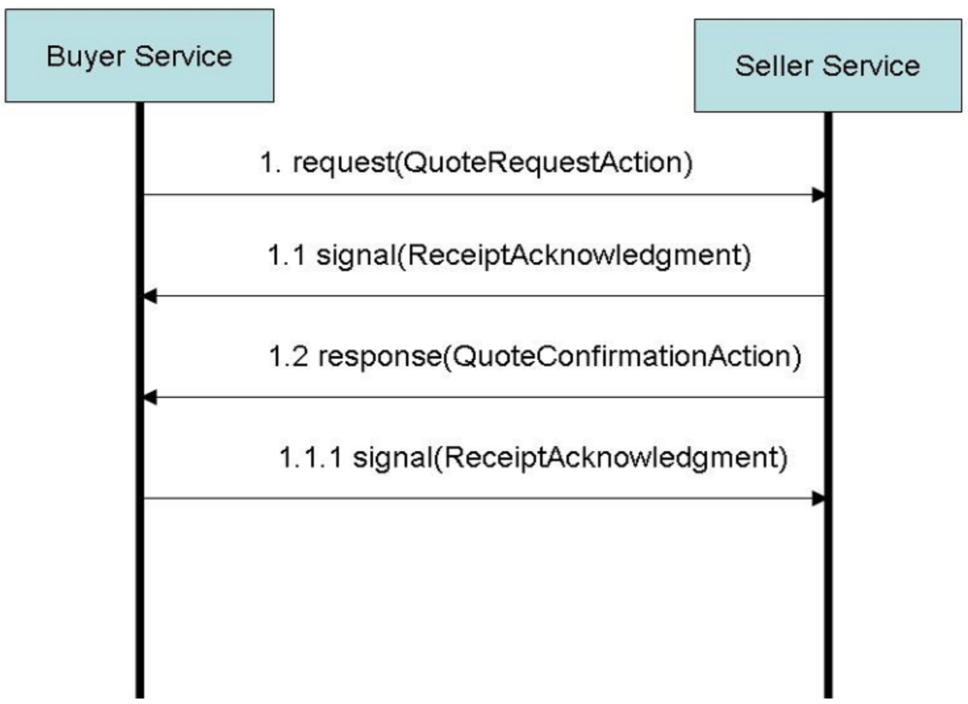

Buyer Service
Seller Service
1. request(QuoteRequestAction)
1.1 signal(ReceiptAcknowledgment)
1.2 response(QuoteConfirmationAction)
1.1.1 signal(ReceiptAcknowledgment)

Figure 8: Functional Service View flow chart.

2.2.1.3. Implementation Framework View

The IFV specifies the action message formats and communication requirements between RosettaNet services according to the RosettaNet Implementation Framework (RNIF). It specifies if the business messages require a digital signal and if they need to be transported with a Secure Socket Layer (SSL). The main task of the IFV is the detailed specification of the business documents to be exchanged which is done in a XSD-file. Moreover the IFV specifies encryption details of the messages to be exchanged.

2.2.2. Dictionaries, Product and Partner Codes

A big source for confusion in B2B communication is the diversity of terminology amongst organizations. This has caused a lot of problems when earlier efforts to automate business processes have been made. To deal with this problem the RosettaNet organization has provided two different dictionaries to specify a valid vocabulary for e-business. There is one dictionary for business terminology (BD) and one for technical terminology (TD). Both the BD and TD are apart from the human readable version represented in XML and DTD files [10].

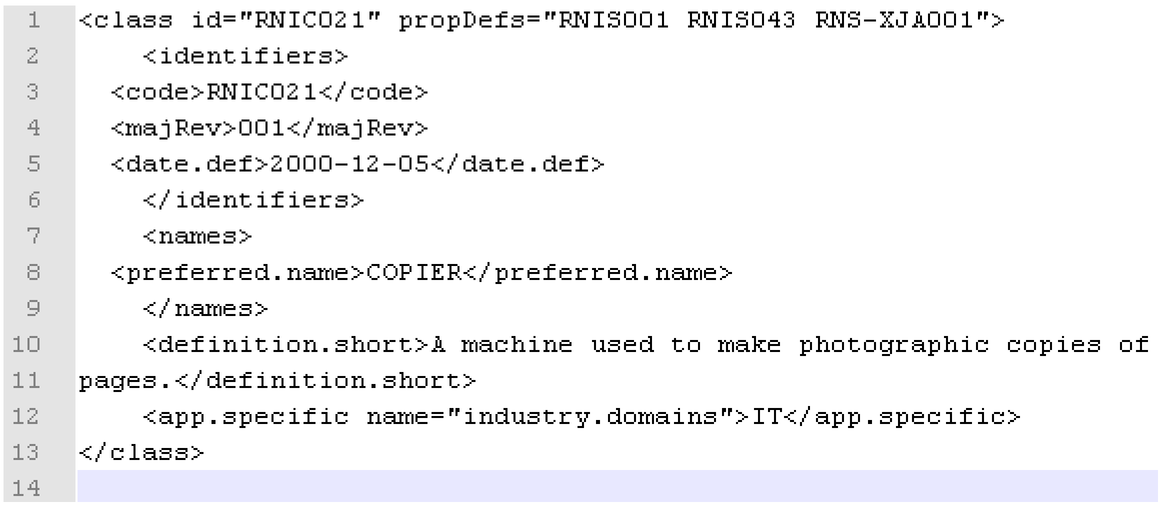
The RosettaNet Business Dictionary (RNBD) defines the following different business terminologies. Since this dictionary is supposed to cover most of the terminology used for well over a 100 business processes it is quite extensive. Below is a list of the entities that make up the RNBD [22]. There is an example of entities presented in Table 2.

Business Properties	
Example	
Name	Discount Amount
Definition	The financial amount representing a reduction to the total amount due.

Entity Instances	
Example	
Entity	Global Country Code
Instance	SE
Definition	Sweden

Table 2: Examples of RNBD entities

The RosettaNet Technical Dictionary (RNTD) is far more extensive then the RNBD since it has to hold a lot more information. At first every industry had their own TD but now there is one central TD to cover all industry needs. The RNTD is designed to support unambiguous and automated electronic exchange of production information [11]. This is achieved by standardizing the semantics used to describe product characteristics and information. In Figure 9, we can see an example of a definition of a photocopier:



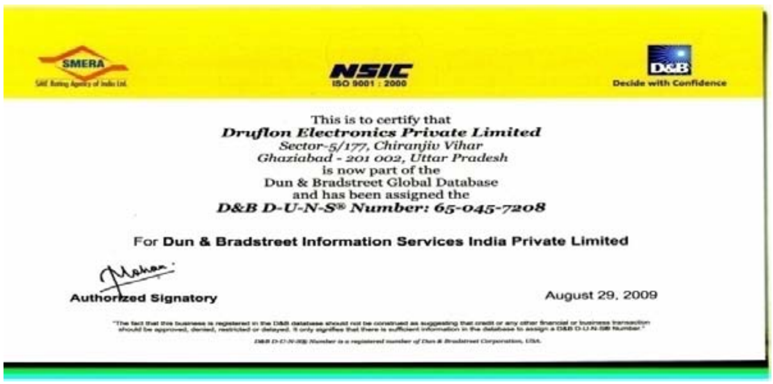
```
<class id="RNIC021" propDefs="RNIS001 RNIS043 RNS-XJA001">
    <identifiers>
  <code>RNIC021</code>
  <majRev>001</majRev>
  <date.def>2000-12-05</date.def>
    </identifiers>
    <names>
  <preferred.name>COPIER</preferred.name>
    </names>
    <definition.short>A machine used to make photographic copies of
pages.</definition.short>
    <app.specific name="industry.domains">IT</app.specific>
</class>

```

Figure 9: definition of photocopier

For each component it is defined which class it belongs to and how its properties are measured and by which unit of measurement [23].

Additionally, RosettaNet product and partner codes are one of the efforts to align business processes between trading partners. The dictionaries are created to act in conformity with the product and partner codes. The following definitions are taken from the RosettaNet homepage:

- Global Company Identifier — RosettaNet specifies the Data Universal Numbering System (D-U-N-S®) for Global Company Identifier in its PIPs. The nine-digit D-U-N-S Number is a worldwide standard for company identification, distinguishing unique business locations around the globe [24]. In Figure 10there is an example of a certificate of an authorized D-U-N-S.



SMERA
NSIC
ISO 9001 : 2000
D&B
Decide with Confidence

This is to certify that
Druflon Electronics Private Limited
Sector-5/177, Chiranjiv Vihar
Ghaziabad - 201 002, Uttar Pradesh
is now part of the
Dun & Bradstreet Global Database
and has been assigned the
D&B D-U-N-S® Number: 65-045-7208

For **Dun & Bradstreet Information Services India Private Limited**

[illegible]
Authorized Signatory

August 29, 2009

"The fact that this business is registered in the D&B database should not be construed as suggesting that credit or any other financial or business transaction should be approved, denied, restricted or delayed. It only signifies that there is sufficient information in the database to assign a D&B D-U-N-S® Number."

D&B D-U-N-S® Number is a registered number of Dun & Bradstreet Corporation, USA.

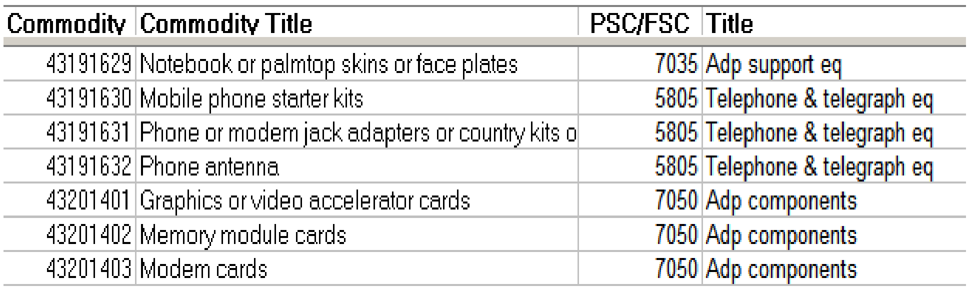
Figure 10: Authorized D-U-N-S number.

- Global Product Identifier — RosettaNet specifies the Global Trade Item Number (GTIN) for Global Product Identifier in its PIPs. The GTIN is a worldwide multi-industry standard for trade-item identification. GTINs are 14-digit numbers that uniquely and globally identify products and services [25]. A GTIN number is presented in Figure 11.

4 512345 678906 >

Figure 11: GTIN number

- Global Class Identifier — RosettaNet specifies the United Nations/Standard Product and Services Code (UN/SPSC) for Global Class Identifier in its PIPs. The UN/SPSC is an open, global commodity code standard for classifying products and services. Items are classified using numbers derived from the system's five-level hierarchy in which two digits are assigned at each level [26]. In Figure 12 is shown a table of UN/SPSC numbers.



Commodity	Commodity Title	PSC/FSC	Title
43191629	Notebook or palmtop skins or face plates	7035	Adp support eq
43191630	Mobile phone starter kits	5805	Telephone & telegraph eq
43191631	Phone or modem jack adapters or country kits o	5805	Telephone & telegraph eq
43191632	Phone antenna	5805	Telephone & telegraph eq
43201401	Graphics or video accelerator cards	7050	Adp components
43201402	Memory module cards	7050	Adp components
43201403	Modem cards	7050	Adp components

Figure 12: Sample of UN/SPSC number.

2.2.3. RosettaNet Implementation Framework

Next to the PIPs the RosettaNet Implementation Framework (RNIF) is the most important part of the RosettaNet standard. The RNIF is a central document for organizations that plan to implement a RosettaNet solution. RosettaNet makes use of a number of other standards like MIME and HTTP and the relations between these standards are described. The functionality of RNIF can be divided into three main categories: business message packaging, protocol stack and security [11].

2.2.3.1. RosettaNet Business Message

The heart of RNIF is the RosettaNet Business Message, which groups the operational content of the message (business payload), the compounds of the header and other elements such as the optional digital signature, which must be transferred as units between the two sides interact. RosettaNet business message is independent of transport protocols.

RNIF sets usage of the Multipurpose Internet Mail Extensions (MIME) multipart / related to the basic folder structure that encapsulates the elements of an operational Rosettanet message. Furthermore using the Secure Multipurpose Internet Mail Extensions (S / MIME) v.2 multipart / signed and applications/pkscs7 mime and envelope-data for digital signature and embedding content, respectively.

The basic components of a RosettaNet message, as shown in Figure 10, enclosed in a MIME file that contains the headers and business content. As it is also shown in Figure 13, part of the contents (payload) provides support for optional attachments. The three headings (Preamble, Delivery and Service) are separate instances of XML documents, which set out shapes from RNIF specifications. There are three types of headers:

- Preamble: This header identifies the standard with which this message structure is compliant.

- Delivery Header: This header identifies message sender and recipient and message instance information. This information is placed separately from the Service Header, in order to allow access to the information by a Hub when the Service Header is encrypted.

- Service Header: This header identifies the PIP, the PIP instance, the activity, and the action to which this message belongs.

These headers are used by the recipient to determine that it is a RosettaNet business message and what version of the RNIF that is used, the context of the message and to identify the sender for authentication and authorization.

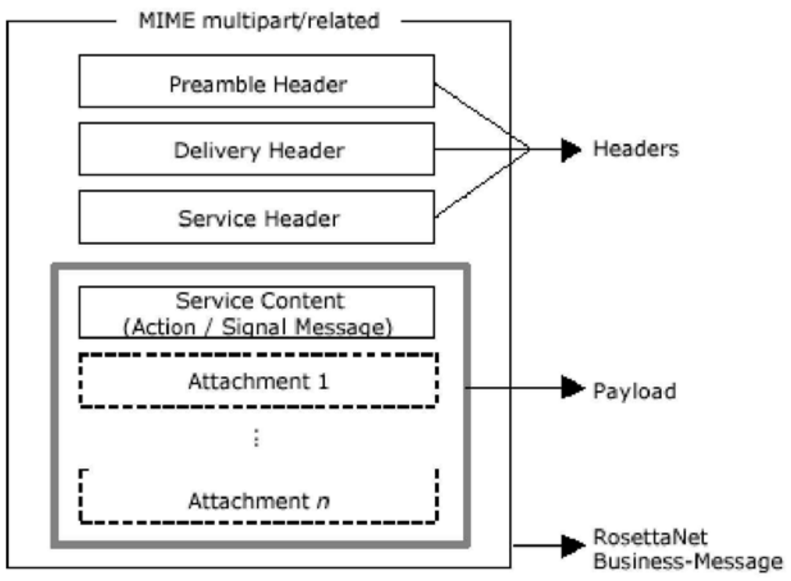


Figure 13: Basic components of the RosettaNet business message.

2.2.3.2. Stack Protocols

The RosettaNet business messages passed between the two extremes RosettaNet contact (endpoints).RNIF supports HTTP over SSL (HTTPS) and SMTP, but allows inclusion of other transport protocols such as FTP. The protocol stack in a RosettaNet end communication occurs in Figure 14.
The specifications define formats for RNIF Messages Recognition Taken (Receipt Acknowledgement) and Messaging Exception (Exception Messages), which are called Operational Signals (Business Signals). Operational signals are used to ensure reliable message transfer. Also, RNIF define standardized sequences of exchanged messages including time constraints and the flow of messages in RosettaNet PIPs. The flow is the logical sequence of steps required to execute a PIP and contains exceptions or error conditions that can occur at any step of execution. There is basically four ways of communication used by the RosettaNet standard and that is:

- Asynchronous Single-Action Activity: One message is sent and received by a trading partner. No timeout requirements have to be fulfilled.

- Asynchronous Two-Action Activity: A request is sent to a trading partner and a response is sent from the trading partner back to the initiator. No timeout requirements have to be fulfilled.

- Synchronous Single-Action Activity: One message is sent and received by a trading partner. Timeout requirements have to be fulfilled.

- Synchronous Two-Action Activity: A request is sent to a trading partner and a response is sent from the trading partner back to the initiator. Timeout requirements have to be fulfilled.

One aspect prior is Timeout. Timeout is specified for different actions in a PIP and it corresponds to the limit of time the trade partner has to produce a reply to a business message.

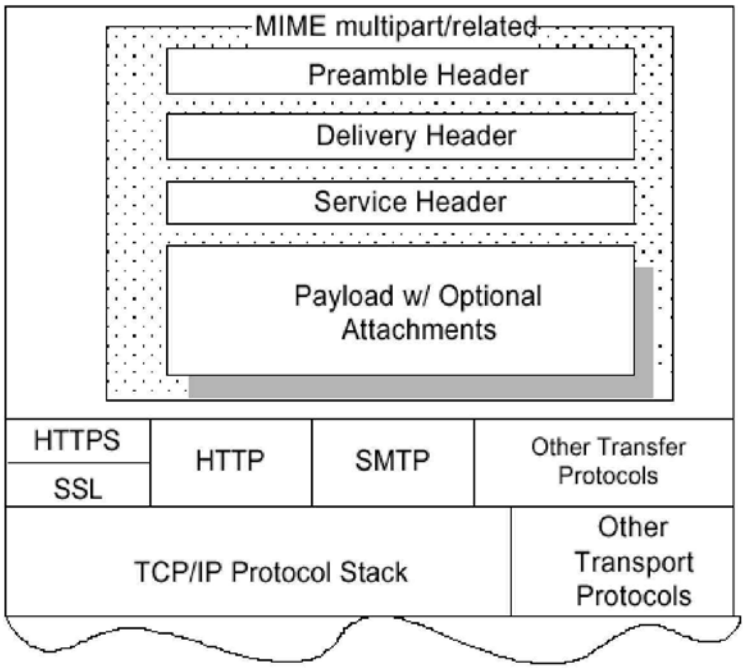


Figure 14: Stack Protocol

2.2.3.3. Security

The specifications include RNIF answers for authentication, authorization, encryption and non-waiver is essential to conduct secure electronic transactions over the Internet.

Authentication is the process of verifying that a partner has a specific identity. RNIF determines the use of digital signatures that are compatible with the requirements of S / MIME N.2 and associated digital certificates issued by a trusted third entity.

Authorization is the permission granted to the sending side to send a particular message. RNIF determines for the purpose of enabling the use of digital signatures that are compatible with the requirements of S / MIME N.2 and associated digital certificates.

Non-renunciation is to ensure that a person or a service actually sent or received a message. For example, the sender of a message cannot deny that he sent that message. This is known as non-renunciation of the source and content. Similarly, the message recipient cannot deny that it received the message. The case was classified as not making the waiver. Digital signatures are attached to messages sent. Their purpose is to ensure the non-waiver of both the source and the content and the reception. The latter, in particular, is achieved through recognition of a signed receipt. The use of digital signatures is also helpful in detecting unauthorized modifications of messages, thus ensuring data integrity.

Finally, the RNIF ensures confidentiality through encryption using S / MIME files for Trade Documents. Confidentiality requires that only the parties involved in a business process is able to read the documents exchanged.

2.3. Summary

Focusing on electronic business interfaces in the supply chain of IT and electronic components; RosettaNet is trying to develop a joint operational environment in which companies in these industries can work together efficiently. Future plans of RosettaNet include expansion to other supply chains.

RosettaNet framework provides two dictionaries, one for business and one for technical term, to identify properties that relate to products, companies and trades. These dictionaries are the basis for the PIPs, which describe sequences of steps for carrying out individual transactions. Also, for the exchange of RosettaNet PIP message is designated an implementation framework, which provides the technical infrastructure required for implementation of PIPs.

This chapter has introduced us to RosettaNet standards and highlights the basic concepts and perspectives. The knowledge of these concepts is crucial for better understanding the RosettaNet standards, which this thesis is based on. The next chapter will discuss the Business Process Management visualized by Business Process Modeling Notation (BPMN).

3. Business Process Modeling Notation

This section studies Business Process, focusing on visualizing processes through graphical notations. It briefly demonstrates the principles of Business Process Modeling and presents Business Process Modeling Notation (BPMN). BPMN is a crucial part of this thesis as it is used to visualize RosettaNet PIPs ‘message exchange between the involved sides. In this chapter is emphasized the concepts and the designing elements of BPMN.

3.1. Business Process Management

Business process management (BPM) is a methodical approach to comprise an organization's progress more effective, well organized and more adaptive to the continuous changes of the environment. The term business process refers to a set of activities that should be achieved in order to serve a specific organizational goal. BPM targets on collimating all aspects of an organization with what clientele needs and wants [27].The target of BPM is the reduction of the errors and lack of communication caused by human and concentrating investors on the requirements of their roles. BPM takes the role of a management entity, which is concerned with upholding and enhancing an organization's operations. Studies implied that BPM makes organizations to increase customer satisfaction, improve product quality, and raise delivery speed and reduce time-to-market speed [28]. BPM is considered as a link of the line-of-business and the IT department within a company.

BPM provides three different kinds of BPM frameworks. Horizontal frameworks are generally emphasized on technology and the reuse, and coping with the design and the development of business processes. Vertical BPM frameworks concentrate on particular sets of coordinated tasks based on pre-built templates that can be easily configured and deployed. Full-service BPM suites have five basic components [29]:

- Process discovery and project scoping
- Process modeling and design
- Business rules engine
- Workflow engine
- Simulation and testing

In Figure15 is demonstrated a BPM suite, which shows how business process management (BPM) tools can be used to implement business processes through the orchestration of activities between people and systems [30].

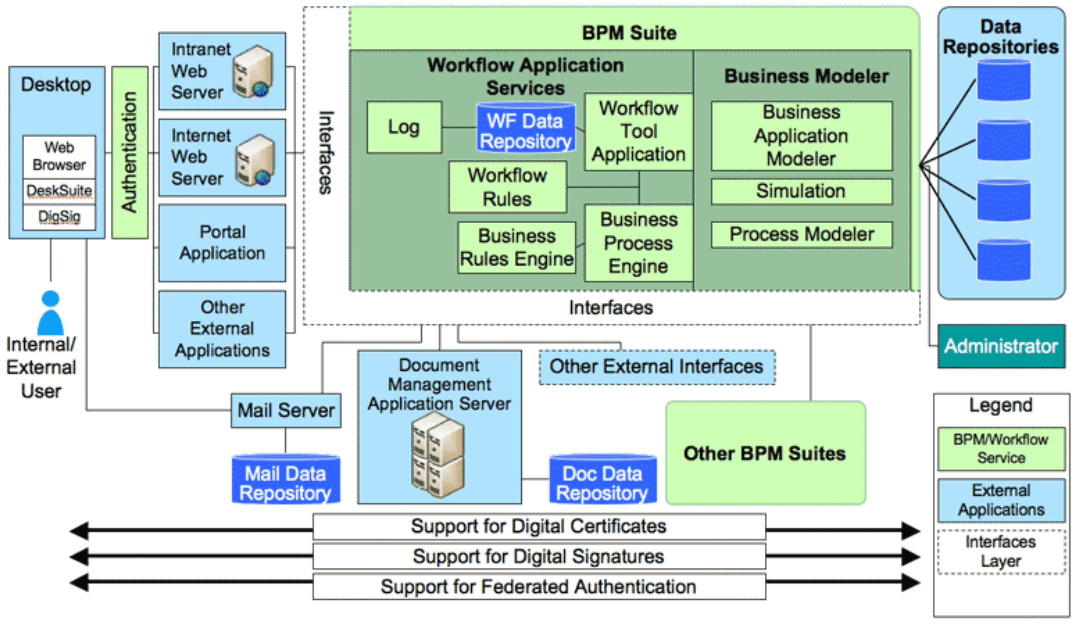


Figure 15: Example of Business Process Management (BPM) Suite.

In order to align the aspects of an organization, BPM offers some actions. These actions can be grouped into six categories such as vision, design, modeling, execution, monitoring, and optimization. Every aspect of these activities is very crucial, because it allows organizations and industries continually streamline and optimize their procedures to ensure that they are tuned to their market need.

Thereafter, it will be illustrated the assets of Business Process Modeling as they are a constructing tool for this thesis.

3.2. Business Process Modeling

The term business process refers to a collection of interrelated, structured activities or tasks that produce a specific product or service (serve a certain goal) for a certain client or clients. A business process can be broken down into several sub-processes, which have their own properties, but nevertheless contribute to the objective of the central process. The analysis of business processes typically includes the mapping of processes and sub-processes to the level of an activity. The business process model is a model of one or more business processes and determines ways in which the different functions to fulfill the objectives of an organization. Such a model remains an abstract concept and depends on the intended use. It is able to describe the workflow or the integration between business processes. It can be manufactured at multiple levels. The workflow (workflow) is a display a series of functions that are the work of a man, a simple or complex mechanism, a group of people or an organization of people or machines.

The Business Process Modeling (BPM) used in the fields of systems engineering and software engineering and is the process of schematic processes of an enterprise so they can be analyzed and improved. Business analysts and managers who try to improve the quality and effectiveness of business processes usually perform the modeling. The optimization often requires the use of information technology, so they can create successful models. The modeling could be created with the Modeling Tools. Business modeling tools give companies the ability to model business processes, implement and execute these models in perfection. As a result, business modeling tools can provide transparency in business processes, as well as the concentration of corporate business models and executive sizes. As they give the opportunity of modeling and simulation, tools allow user to have a pre-implementation. Optimization is available after the performance, based on analysis of actual figures as executed.

The crucial part of BPM is the ability to model the process in a graphic way. The graphics are called business modeling diagrams. The business modeling diagrams are:

- Use case diagrams

- Activity diagrams

There are some also business modeling techniques which are:

- Business Process Modeling Notation (BPMN)
- Cognition enhanced Natural language Information Analysis Method (CogNIAM)
- Extended Business Modeling Language (xBML)
- Event-driven process chain (EPC)
- ICAM DEFinition (IDEF0)
- Unified Modeling Language (UML), extensions for business processes such as Eriksson-Penker

Finally, the software suite of business modeling provides programming interfaces (web services application program interfaces) that allow business applications to be built in such a way so as to stimulate the operation of BPM. This element is often referred to as the driving force of the BPM software suite. These programming languages for BPM include some models and specific languages:

- BPMN
- Business Process Execution Language (BPEL),
- Web Services Choreography Description Language (WS-CDL).
- XML Process Definition Language (XPDL),
- Architecture of Integrated Information Systems (ARIS),
- Java Process Definition Language (JBPM),

Other technologies related to business modeling include the model-driven architecture and service-oriented architecture.

In the next section we discuss the basic principles around Business Process Modeling Notation (BPMN) that will be the basis for our business processes, which describe Rosettanet Pips.

3.3. Business Process Modeling Notation Perspective

The Business Process Management Initiative (BPMI) has developed a standard Business Process Modeling Notation (BPMN). BPMN succeed in becoming widely accepted as a graphic notation open standard for service-oriented architecture and process modeling. The BPMI Notation Working Group's primary goal of the BPMN effort was to create and give an open standard that is easily understandable and useable by all users who are involved, from the analysts who design the initial plans of the processes, to the IT developers who are responsible for implementing and deploying those processes and the business users who monitor those processes and their results. Furthermore, BPMN has achieved to bridge the gap between the business process design and process implementation by mapping BPMN diagrams to Business Process Execution Language (BPEL). Moreover, BPMN is capable of modeling orchestrations and choreographies because of the providing great support for modeling the process patterns. Finally, BPMN provides two versions. The version, which is used in this work, is BPMN 1.2.The other version is BPMN 2.0, which is in BETA version. BPMN 1.2 is an established modeling standard fully operational.

3.3.1. BPMN language elements

BPMN is based on a flowcharting technique, which is responsible for creating graphical models for business process tasks and activities. So a Business Process Model is a network of graphical objects, which are activities and the flow controls that define their order of performance.

BPMN's basic elements as defined [31]are grouped in four categories: Flow Objects, Connecting Objects, Swimlanes and Artifacts. Flow Objects and Connecting Objects define the model of the process, while Swimlanes and Artifacts give additional information in the process model for the user. These elements enable the easy development of simple diagrams that will look familiar to most business analysts (e.g., a flowchart diagram). The elements were chosen to be distinguishable from each other and to utilize shapes that are familiar to most modelers.

3.3.1.1. Flow Objects

There are three categories of Flow Objects that are the Activities, the Events and the Gateways. These elements define the behavior of a business model. Let's take a look of its element separately.

Activity

Activities are elements, which are used to visualize the tasks and the actions that are performed inside a process by humans or machines. Activities can be atomic or compound. Compound activities are composed of many other activities. Activities are categorized in two types, which are Task and Sub-Process. Activity is represented by a rounded-corner rectangle, while Sub-Process is distinguished by a small plus sign in the bottom center of the shape as we see at Figure 16.

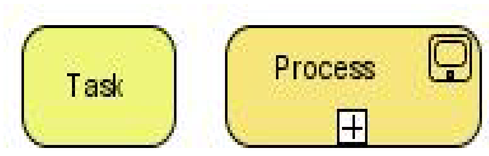


Figure 16: Task and Sub-Process

Event

Events take place and happen through the process and they cause something to trigger or they cause something to happen. Examples of triggers or results are timers, incoming messages, defined rules or more complicated results or triggers, which are composites from two or more of them. Events are circles with open centers to allow internal markers to differentiate different triggers or results. There are three types of Events, depending on when they take place in the process: Start, Intermediate, and End. See Events in the Figure 17 below.

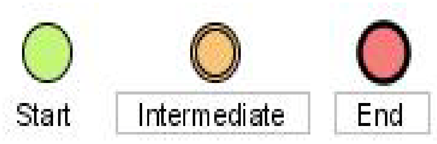


Figure 17: Events

Gateway

Gateways are modeling elements that are used to control how flows interact as they converge and diverge within a Process. All types of Gateways are diamonds (see theFigure18 below). Different internal markers indicate different types of behavior. All Gateways both split and merge the flow. Thus, they will determine traditional decisions, as well as the forking, merging, and joining of paths.

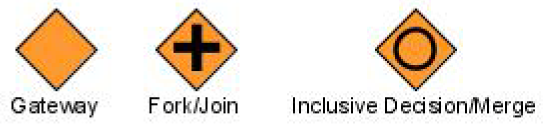


Figure 18: Gateways

3.3.1.2. Connecting Objects

The Flow Objects are connected together through the Connecting Objects in order to make up the fundamental structure of a business process. There are three different ways to connect such as Sequence Flow, Message Flow and Association (Figure 19).A Sequence Flow visualizes and describes the order (the sequence) that activities will be performed in a Process. A Message Flow is used to model the exchange of messages between two separate participants. Participants are visualized by pools, which are reported to the next section. Activities, which are in the same pool, cannot pass on between them through Message Flows. Message Flows can also synchronize or trigger the start of a process. An Association is used to associate data objects and Artifacts, which will be discussed in the coming sections, with flow objects. Associations are used to demonstrate the inputs and outputs of activities.

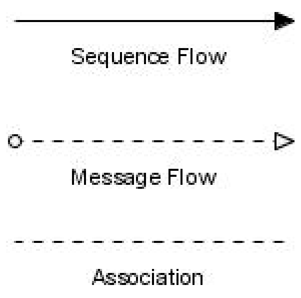


Figure 19: Connecting Objects

3.3.1.3. Swimlanes

In order to group the modeling elements, BPMN provides two constructs: Pools and Lanes (see Figure 20). A Pool is used to represent a participant (organization) in a process. It also contains the activities that they are done by different participants. A Lane is a sub-partition within a Pool. Each Lane depicts a different department of the organization.

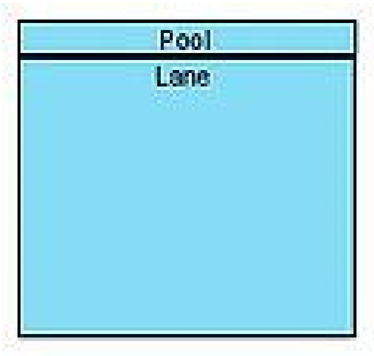


Figure 20: Pool and Lane

3.3.1.4. Artifacts

Artifacts are designed to give additional information about the processes without changing and affecting the sequence and message flow. Artifacts are divided into three categories: Data Objects, Groups and Text Annotations (see Figure 21).Data Objects are a mechanism to show information such as documents or messages that are inputs or outputs in specific flow objects. They are connected to activities through Associations. Groups are grouping flow objects depending on the similarity between them. Text Annotations are a mechanism to assist the reader to understand the BPMN Diagram by giving information to the related objects.

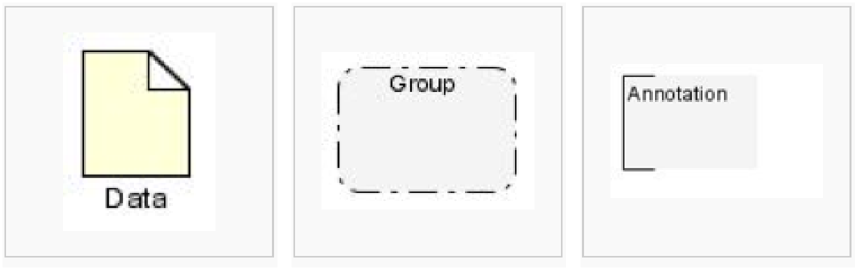


Figure 21: Data Object, Group, and Text Annotation.

3.3.2. BPMN Model Types

BPMN is constructed in such a way in order to support various types of modeling, depending on the wide variety of information and audiences. It covers many types of modeling and allows the creation of process segments as well as end-to-end business processes, at different levels of fidelity. According to the purposes and the goal that must be succeed in the business process modeling, BPMN provides three types of model that can be designed, developed and analyzed: Private (internal) business processes, Abstract (public) processes, and Collaboration (Public) B2B processes. The basic idea of modeling a business process is to start designing the high-level activities and then adding more details in the process to lower levels by separating them with different diagrams. On the other hand, BPMN gives the opportunity to each model developer to apply his methodology. Next, it is represented in briefly way the BPMN model types.

Collaboration B2B process demonstrates the interactions between two or more entities or organizations. These interactions are a sequence of activities that show the message exchange patterns between the organizations took part. A B2B process can be assumed as two or more abstract processes, which are communicating each other. The actual processes are likely to have more activities and detail than what is shown in the collaborative B2B processes. This model refers to the global relationships between the organizations. Collaboration processes usually are modeled within a Pool and the different organizations' interactions are shown as Lanes within the Pool.

Private business processes are the basic workflow processes, which are showing the internal interactions from a specific organization. They define the activities that are invisible to the public but they can show interaction with external organizations. The Pool demonstrates the barriers of an organization and different activities can be classified into Swimlanes. So the Sequence Flow of the Process is in the Pool cannot override the barriers of the Pool. On the other hand Message Flow can cross the Pool boundary to show the existing interactions between other internal processes.

Abstract (public) processes represent the interactions between a private business process and another organization. Thus, in such a model are included only activities that communicate outside the internal business process. However, the activities of the private business process are not demonstrated in the abstract process. Thus, the abstract process shows to the outside world the sequence of messages that are required to interact with that business process.

In this work, we will focus on these three models in order to succeed visualize and model RosettaNet PIPS. RosettaNet PIPs, as they were described in the previous chapter, include relationships between organizations and relationships between single organizations.

3.4. Summary

Business process management focuses on making organizations more efficient, productive and profitable by providing tools to analyze and model tasks and activities that execute. The use of Business Process Modeling is one of the most powerful ways to re-organize business entities. Business Process Modeling provides several languages and graphical notations in order to achieve modeling business processes. A widely accepted open standard graphic notation for business process modeling and service architecture (SOA) is BPMN. The primary goal of the BPMN effort was to provide a notation that is readily understandable by all business users, from the business analysts who create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and, finally, to the business people who will manage and monitor those processes. There are two Version of BPMN 1.2 and 2.0. However, in this work BPMN 1.2 is used because it is a stable, in opposition to 2.0, which is a BETA version. BPMN provides a great variety of graphical notations and descriptive basic models, which support the designing and modeling of business processes through organizations.

The knowledge of BPMN is mandatory in order to visualize RosettaNet PIPS and create the suitable business processes. In next chapter it will be introduced the technologies of Web Services and BPEL.

4. Technologies of Web Services

This chapter introduces the principles and the concept of Service Oriented Architecture (SOA) and Web Services. As Web evolves, new technologies and standards come along that they will be discussed in details, showing the main implications. Great significance is placed on the Web Services Technologies and BPEL as well, which are the building materials of this thesis, as they demonstrate and use the implementation of RosettaNet PIPS according to the instructions given by the standard.

4.1. Service Oriented Architecture (SOA)

Service-oriented architecture (SOA), which is an evolution of distributed computing, is the underlying structure supporting communications between services. Thus, an application's business logic or individual functions are modularized and presented as services for consumer or client applications. SOA also provides a framework, which is capable of matching what is necessary for entities. So SOA defines how two entities, such as programs, interact in such a way as to enable one entity to perform a unit of work on behalf of another entity. Service interactions are defined using a description language. Each interaction is self-contained and loosely coupled, so that each interaction is independent of any other interaction [32].
Service-oriented architecture is not a new thing and it can be implemented with various technologies. The first service-oriented architecture for many people in the past was with the use DCOM or Object Request Brokers (ORBs) based on the CORBA specification. Other common technologies are Web Services, Service Component Architecture (SCA), and Enterprise JavaBeans (EJB).
The most common and newest technology implementing Service oriented Architecture is Web Services. Although there are some conflicts and limits, implementing SOA with Web Services is the most conceptual solution, as it can deliver adjustable, robust and reusable services, ready to fulfill every need that appears. Finally, such implementation offers a simply approach on integrating systems because it provides universal connectivity [33].

4.2. Web Services

Web Services is an extremely innovating and evolutionary technology, as it offers integration between applications through web. Web Services could be referred as the “Holy Grail” of Distributing computing. A highly accepted definition, by Websevice.org, states, “Web Services are encapsulated, loosely coupled contracted functions offered via standard protocols”. Its word of this definition has a complex meaning. “Encapsulated “defines that the implementation of the function is never seen from the outside. “Loosely coupled” states that changing the implementation of one function does not require change of the invoking function. “Contracted” shows that the descriptions of the function’s behavior are available to public, where it is explained how to bind to the function as well as its input and output parameters. With the term protocols is define the way that Web Services communicate. Services are defined and described by the use of the Web Services Description Language (WSDL) [34]standard and the Simple Object Access Protocol (SOAP) [35], which is the communication protocol. The main idea is that a Web service is a software application that can be accessed remotely via Internet using different XML documents. Examples of Web Services based systems, are booking tickets, auctions, stock trading, online reservations and weather reporting.

The fundamental basis of Web Services is three XML- based standards. These three standards are responsible for deploying, describing and defining a web service. These standards are the Simple Object Access Protocol (SOAP), WSDL, and he Universal Description Discovery and Integration (UDDI) [36]. These standards are proposed from two organizations: OASIS [37] and W3C [38].

As it was discussed previous Web service is a software application that can be accessed remotely using different XML-based languages. Normally, a URL, just like any other Web site, identifies a Web service. In the traditional client server world, Web Services are different from ordinary Web sites in the type of interaction that they can provide. Most Web sites are designed to provide a response to a request from a person. The person either types in the URL of the site or clicks on a hyperlink to create the request. This request takes the form of a text document that contains some fairly simple instructions for the server. These instructions are limited to the name of a document to be returned or a call to a server-side program, along with a few parameters. On the other hand, Web Services are based on Web Service Reference Model [39]. Web Service Reference Model consists of three different components, which co-operate between them. The Service Registry, which has the published services’ description and acts like a broker between Service Provider and Service Requestor (client). The

Service Provider, which implements the Web Service and publishes it to the Service Registry. The Service Client, which asks the Service Registry to find a suitable Service Provider in order to bind it and invoke the Web Service. The Figure 22 shows the relationships and the terms of collaboration between the entities.

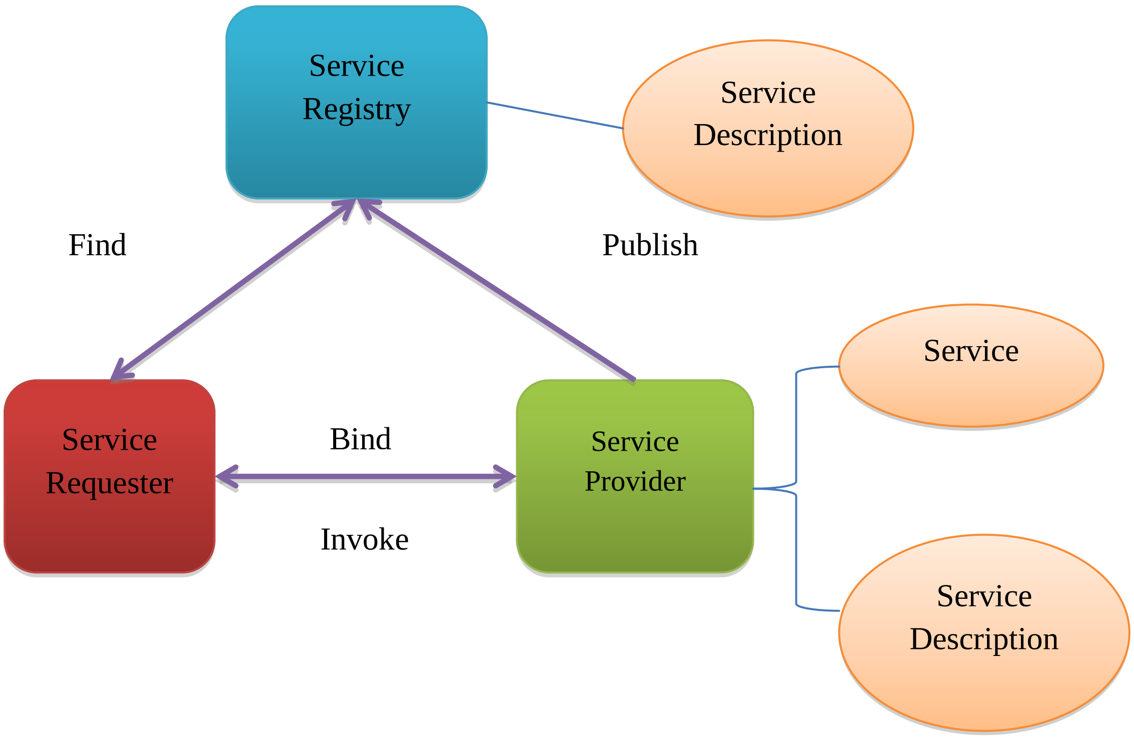


Figure 22: The Web Service Reference Model

The Web Service Stack Protocols helps the reader to make clearer the Web Service Architecture. It provides the building block of how web services are implemented, invoked, published and used. As it is shown in Figure 23 there are five crucial stack layer levels. The five layers, which are shown in Figure 23, are: Communication, Messaging, Description, Discovery and Process. Each layer gives certain functions to Web Services in order to make more efficient and easier the communication and interoperation among Web Services and Web Service Clients. The Stack in its full form provides full integration between Web Services applications. In general terms, a web service is a software component or service that has been published on the World Wide Web with the discovery procedure through UDDI, to describe services that are exposed using SOAP and took input or return output through the WSDL signatures, an XML-based message encoding that is typically transported using HTTP. The Process layer refers to complex interaction between Web Services such as coordination, co-operation or composition. The other three layers refer provide ways to improve the safety, the functionality and the quality of the Web Service [40].

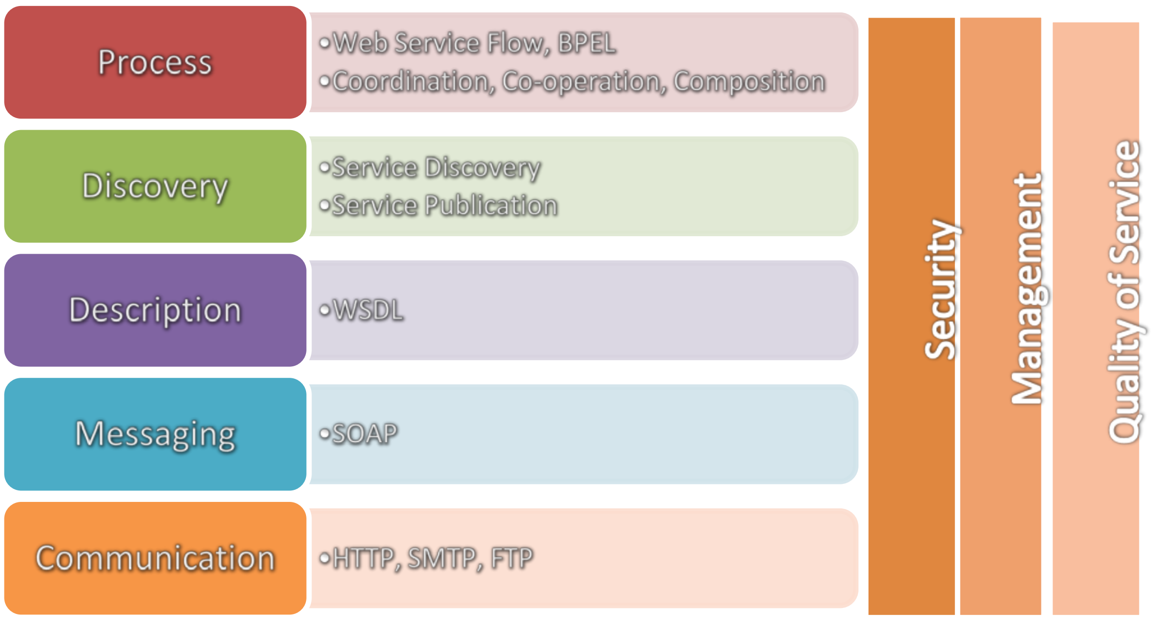


Figure 23: Web Service Protocols Stack

Other key standards, which are not in the Figure 23, are WS-I standards.WS-I standards are developed by Web services Interoperability Organization [41]. The importance of these standards is obvious, as they are responsible for the integration and interoperability of Web Services. The stated goal of WS-I is providing guidelines to accelerate the adoption of Web services by assisting in the selection and interpretation of Web services specifications, and in the development of common best practices for their usage in the development, deployment, and integration of business applications.

Finally, the Web Service implementation based on the discussed standards is supported by two different frameworks: .NET and J2EE [42]. This work is implemented on J2EE. The next section introduces Simple Object Access Protocol (SOAP).

4.2.1. Simple Object Access Protocol (SOAP)

Simple Object Access Protocol (SOAP) is a simple XML-based communication protocol between software applications, which lets the exchange of data (information) via HTTP. In simple words SOAP is a protocol for accessing Web Services.

A SOAP message is a typical XML document, which contains the following four elements. An Envelope element is used to identify the XML document as a SOAP message. Header element has the header information. A Body element contains call and response information. A Fault element contains errors and status information.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-
envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>
```

Figure 24: SOAP message elements

The SOAP message elements are demonstrated in Figure24.

The SOAP body of a message element depends on the style of Web service. The body of an RPC (remote procedure call) style SOAP message is constructed in a specific way, which is defined in the SOAP standard. It is built around the assumption that you want to call the web service just like you would call a normal function or method that is part of your application code. The message body contains an XML element for each parameter of the method. These parameter elements are wrapped in an XML element, which contains the name of the method that is being called. The response returns a single value (encoded in XML), just like a programmatic method. On the other hand, a document style web service contains no restrictions for how the SOAP body must be constructed. It allows you to include whatever XML data you want and also to include a schema for this XML.

The SOAP protocol is language and platform independent because of XML-based nature, so it allows application communication running on different operating systems, with different technologies and programming languages.

4.2.2. Web Service Description Language (WSDL)

The Web Service Description Language (WSDL) is a XML-based language, which describes Web Services' interfaces. A WSDL document is an XML document that provides all the information that you need to connect to the Web service. When a web service is implemented, a WSDL document is created, which is published in a Service Registry. Web Service Clients can find the WSDL files and use the services. WSDL specifies the location of the service and the operations (or methods) the service exposes. WSDL document is divided into six major elements:

The definitions element must be the root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.

The type's element describes all the data types used between the client and server. WSDL is not tied exclusively to a specific typing system, but it uses the W3C XML Schema specification as its default choice. If the service uses only XML Schema built-in simple types, such as strings and integers, the type's element is not required. A full discussion of the type's element and XML Schema is deferred to the end of the chapter.

The message element describes a one-way message, whether it is a single message request or a single message response. It defines the name of the message and contains zero or more message part elements, which can refer to message parameters or message return values.

The port Type element combines multiple message elements to form a complete one-way or round-trip operation. For example, a port Type can combine one request and one response message into a single request/response operation, most commonly used in SOAP services. Note that a port Type can (and frequently does) define multiple operations.

The binding element describes the concrete specifics of how the service will be implemented on the wire. WSDL includes built-in extensions for defining SOAP services, and SOAP-specific information therefore goes here.

The service element defines the address for invoking the specified service. Most commonly, this includes a URL for invoking the SOAP service.

Instable are explained briefly.

<definitions>	Root WSDL Element
<types>	What data types will be transmitted
<message>	What messages will be transmitted
<port Type>	What functions will be supported
<binding>	How will the messages be transmitted? What SOAP-specific details are there
<service>	Where is the service located

Table 3: WSDL elements

The Figure shows WSDL's elements structure.

```
<definitions>

<type> Definitions of types</type>

<message>…</message>

<portType>….</portType>

<binding>…..</binding>

<service>….</service>

</definitions>
```

Figure 25: WSDL's elements structure

4.2.3. Universal Description, Discovery and Integration (UDDI)

UDDI is a platform-independent framework for describing services, discovering businesses, and integrating business services by using the Internet. The reason that UDDI is acceptable to all the vendors mentioned previously is that it is built on the same SOAP standards that ordinary Web services are. This means that a registry can be written in and accessed by any computer language running on any hardware platform running any operating system. The UDDI itself is configured as a replicated collection of Web services. All the public directories replicate information posted on any of them. This ensures that you can access all the public Web services by accessing only one of them. The information in a registry is composed of three types of entries: white, yellow, and green pages. Each entry has specific information. The white pages contain the basic contact information. The yellow pages contain taxonomy information. The green pages contain instructions for specific client development to invoke the specific Web Service.

UDDI registries work just like other Web services. All the APIs in the UDDI specification are defined in XML, placed inside SOAP envelopes, and sent over HTTP. In addition, client requests that entail modifying data are required to be secured and authenticated.

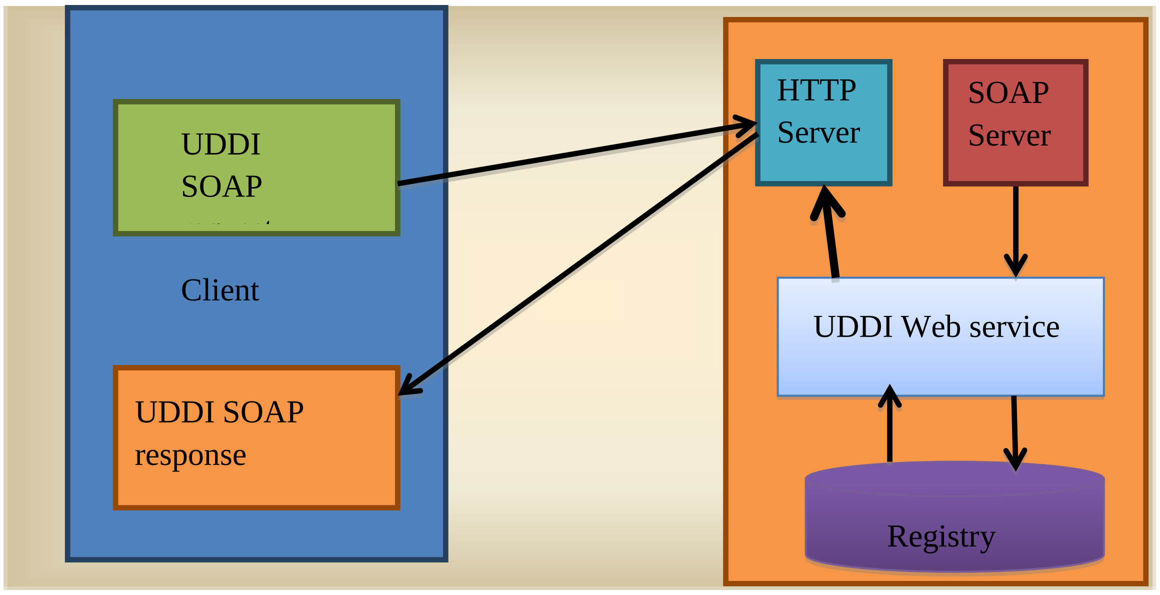


Figure 26: UDDI registry architecture

How does UDDI operates is shown in Figure 26.

4.2.4. Web Services Addressing (WS-Addressing)

WS-Addressing or Web Services Addressing is a specification of transport-neutral mechanisms that allow web services to communicate addressing information. It essentially consists of two parts: a structure for communicating a reference to a Web service endpoint, and a set of Message Addressing Properties which associate addressing information with a particular message [43].WS-Addressing is a standardized way of including message routing data within SOAP headers. Instead of relying on network-level transport to convey routing information, a message utilizing WS-Addressing may contain its own dispatch metadata in a standardized SOAP header. The network-level transport is only responsible for delivering that message to a dispatcher capable of reading the WS-Addressing metadata. Once that message arrives at the dispatcher specified in the URI, the job of the network-level transport is done.

An Endpoint Reference (EPR) is an XML structure encapsulating information useful for addressing a message to a Web service. This includes the destination address of the message, any additional parameters (called reference parameters) necessary to route the message to the destination, and optional metadata (such as WSDL or WS-Policy) about the service.

4.3. Business Process Execution Language (BPEL)

Combining Web services to create higher level, cross-organizational business processes requires standards to model the interactions. Several standards are working in that way. SOA provides, through its standards, the ability to create complex Web Services by connecting single. A single web service could be referred as an abstract business process. Using the SOA's ability, many web services can be connected together and create many combined business processes [44].

Web services offer standards-based mechanisms for addressing this issue. However, existing methods for creating business processes are not designed to work with cross-organizational components. Nor are these methods flexible enough to handle the technical interfaces that Web services introduce.

The terms orchestration and choreography describe two aspects of creating business processes from composite Web services.

4.3.1. Orchestration

In this case there is a central controller process, which controls and co-ordinates all the Web Services involved in the application. This central process can be a Web Service a well. The point to note in this case is that all other Web Services don't really know that they are participating in a higher-level business process. How the participating Web Services will be called, what will be the control flow, what all transformation will take place. Only the central controller process knows these all things. The other Web Services simply honor the requests whenever called. The Figure 27 makes it quite easier to become familiar with the overall process.

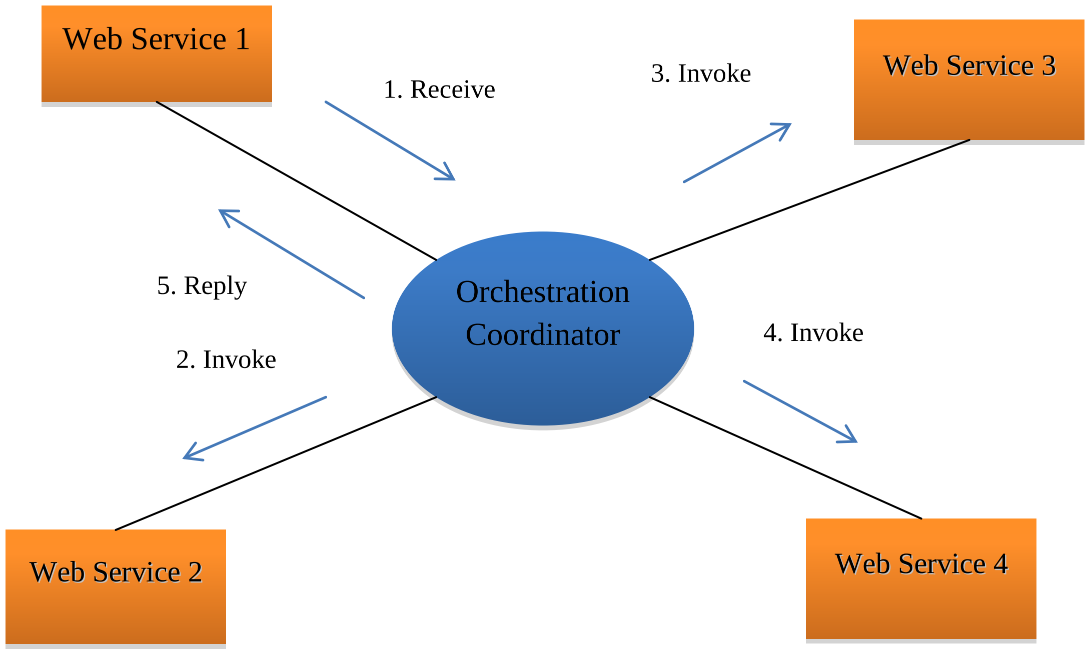


Figure 27: Orchestration of Web Services

4.3.2. Choreography

Here, there is not any central controller process and hence all the participating Web Services know when to call, whom to interact, when to execute operations. Choreography can be visualized just like a collaborative effort of many participating Web Services and since there is not any controller hence all the Web Services need to know the actual business process and things involved in it like message exchanges, time of call. Find below a diagram depicting a typical Choreography process. The Figure makes it quite easier to become familiar with the overall process.

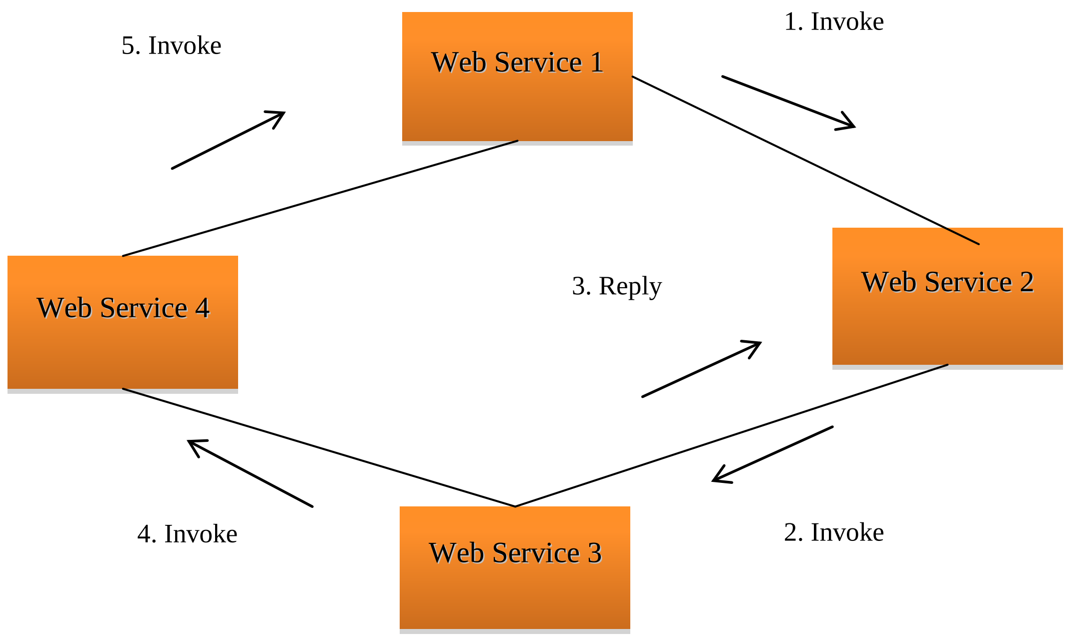


Figure 28: Choreography of Web Services

4.3.3. Orchestration versus Choreography

Orchestration has a central controller process and all other participating Web Services don't know about the actual business process. Only the controller process calls them and they don't know anything about other Web Services involved in the application. Whereas Choreography doesn't have any controller process/service and all the participating Web Services know the actual business process and they are well aware of which all Web Services they need to interact with, when to execute the operations [45].

In order to succeed in creating, defining and executing complex business process with Web services, WS-BPEL (BPEL) is used.

4.3.4. BPEL's Perspective

The Business Process Execution Language for Web services is an OASIS'[37] standard executable language that specifies business processes that are composed of Web services as well as exposed Web services[46]. BPEL is a XML-based workflow language and it is oriented by combining IBM's Web Service Flow Language (WSFL) and Microsoft's XLANG.

BPEL works like an orchestration mechanism, which was discussed above. It describes the internal and external information exchanges. BPEL deals explicitly with the functional aspects of business processes: control flow (branch, loop, parallel), asynchronous conversations and correlation, long running nested units of work, faults and compensation. BPEL directly addresses these business process challenges: coordinating asynchronous communication between services, correlating message exchanges between parties, implementing parallel processing of activities, manipulating data between partner interactions, supporting long running business transactions and activities, and providing consistent exception handling.

BPEL opens a completely new way or at least enhanced way, for software development for mainstream business applications to allow a programmer to describe a business process that will take place across the Internet. BPEL provides an XML-based grammar for describing the logic to control and coordinate Web services participating in a process flow. This grammar can be interpreted and executed by a BPEL orchestration engine, which is controlled by one of the participating business parties. The engine coordinates all of the activities in the process, and controls the system's corrective activities when exceptions occur. BPEL builds on the foundation of XML and Web services. It uses an XML-based language that supports the Web services technology stack, including SOAP, WSDL, UDDI, WS-Reliable Messaging, WS-Addressing, WS-Coordination, and WS-Transaction.

BPEL offers a nice model to abstract orchestration logic from the participating services, and configuration using BPEL over (hard core) coding of service interactions is enticing. However, there is processing overhead and infrastructure expense, so BPEL might not be the best choice for simple orchestrations. BPEL supports two different ways of describing business processes that support orchestration and choreography: Executable processes and Abstract process.

Executable processes allow you to specify the exact details of business processes. They follow the orchestration paradigm and can be executed by an orchestration engine. Executable business processes model actual behavior of a participant in a business interaction.

Abstract business protocols allow specification of the public message exchange between parties only. They do not include the internal details of process flows and are not executable. They follow the choreography paradigm.

BPEL is used to model the behavior of both executable and abstract processes. The scope includes: Sequencing of process activities, especially Web Service interactions, correlation of messages and process instances, recovery behavior in case of failures and exceptional conditions, bilateral Web Service based relationships between process roles.

A BPEL process specifies the exact order in which participating Web services should be invoked, either sequentially or in parallel. With BPEL, you can express conditional behaviors. For example, an invocation of a Web service can depend on the value of a previous invocation. It can also construct loops, declare variables, copy and assign values, define fault handlers, and so on. By combining all these constructs, you can define complex business processes in an algorithmic manner.

4.3.5. BPEL Construct Elements

A BPEL process can be synchronous or asynchronous. A synchronous BPEL process blocks the client (the one which is using the process) until the process finishes and returns a result to the client. An asynchronous process does not block the client. Rather it uses a callback to return the result (if any). Usually we use asynchronous processes for longer-lasting processes and synchronous for processes that return a result in a relatively short time. If a BPEL process uses asynchronous web services, the process itself is usually also asynchronous.

A BPEL business process, in order to begin, receives a request. To fulfill it, the process then invokes the involved web services and finally responds to the original caller. Because the BPEL process communicates with other web services, it relies heavily on the WSDL description of the web services invoked by the composite web service. A BPEL process is described in an XML file with extension. bpel. A BPEL process always starts with the process element. Inside process there must be at least one activity either a primitive or a structured one. A BPEL process consists of steps. Each step is called an activity. BPEL supports primitive and structure activities. Primitive activities represent basic constructs and are used for common tasks, such as those listed in the Table 4 below.

<invoke>	Invoking other web services
<receive>	Waiting for the client to invoke the business process through sending a message (receiving a request)
<reply>	Generating a response for synchronous operations
<assign>	Manipulating data variables
<throw>	Indicating faults and exceptions
<wait>	Waiting for some time
<terminate>	Terminating the entire process
<exit>	immediately terminate the running process
<empty>	no-op instruction
<compensate>	invoke compensation on all completed child scopes in default order
<compensateScope>	invoke compensation on one completed child scope
<extensionActivity>	wrapper for language extension

Table 4: Primitive Activities

Structure activities are used to combine primitive activities to define complex algorithms inside a process. These structures are listed in the Table 5 below.

<flow>	defines a set of activities that will be invoked in parallel
<pick>	associates activities with events and waits until an event is triggered.
<sequence>	defines a set of activities that will be invoked in an ordered sequence
<while>	defines loops
<switch>	implements branches
<if-elseif-else>	Implements conditionally branching
<repeatUntil>	defines loops like <while>
<scope>	splits process into parts
<compensationHandler>	defines which activities will happen when something goes wrong
<faultHandler>	catches errors
<eventHandler>	defines the steps of the process when a certain event occur

Table 5: Structure activities

Each BPEL process will also declare variables, using <variable>, and define partner links, using <partnerLink>.BPEL calls the links to all parties it interacts with partner links. Partner links can be links to web services that are invoked by the BPEL process. Partner links can also be links to clients, which invoke the BPEL process. Each BPEL process has at least one client partner link, because there has to be a client that invokes the BPEL process. Usually a BPEL process will also have at least one invoked partner link, because it will most likely invoke at least one web service (usually more than one). Invoked partner links may, however, become client partner links—this is usually the case with asynchronous services, where the process invokes an operation. Later the service (partner) invokes the callback operation on the process to return the requested data. BPEL treats clients as partner links for two reasons. The most obvious reason is support for asynchronous interactions. The second reason is based on the fact that the BPEL process can offer services. More than one client can use these services, offered through port types. The process may wish to distinguish between different clients and offer them only the functionality they are authorized to use.

For its client a BPEL process is recognized like any other web service. When BPEL process is defined, a new web service is actually defined that is a composition of existing services. The interface of the new BPEL composite web service uses a set of port types, through which it provides operations like any other web service. To invoke a business process described in BPEL, the resulting composite web service is invoked.

4.3.6 BPEL Engines

To execute BPEL executable processes we need an orchestration server. Orchestration servers provide a run-time environment for executing BPEL business processes. BPEL is strongly related to web services and to the modern software platforms that support web service development, particularly to Java 2 Enterprise Edition (J2EE) and

Microsoft .NET. BPEL engines are Active VOS Enterprise from Active Endpoints, ODE engine from Apache Software Foundation, Oracle BPEL Process Manager, Intalio and others.

4.4. Summary

This chapter introduces the basic concepts of Service Oriented Architecture. Main principles of Web Services' standards were discussed briefly. These standards will be the basis for this work proposed implementing message exchange patterns based on RosettaNet PIPs. The following section will introduce modeling RosettaNet PIPS in BPMN.

5. Modeling RosettaNet PIPS in BPMN

This section studies Rosettanet PIPs, focusing on the business connections, procedures and messages that exist between the participants. As it was mentioned before, a Partner Interface Process (PIP) defines business processes between trading partners. The necessity to model these business processes, is crucial. Studying relationships between the participants takes out the operational details and describes the way that these relationships between the participants take place. This kind of information is depicted in the business process level, where all these relationships that are described in each PIP, modeled in BPMN. This section contributes to analyze a PIP, providing a methodology to transform a PIP model to collaboration B2B process. The methodology consists of instructions to design and model a Partner Interface Process based on each PIP specification. The creation of B2B process is depicted by BPMN modeler tool of eclipse [47]. Next section provides information about Partner Interface Process.

5.1. Partner Interface Process Specifications

The PIP standardizes public (business)process automation by standardizing business documents, the sequence of sending these documents, and the physical attributes of the messages that define the quality of service. PIP also defines the messaging system used to send and receive these documents. The business documents are delivered to business services defined by each trading partner.

A PIP defines exactly two roles for the trading partners that participate in the business process. The business process is divided into one or more business activities. The messages (business documents) exchanged between the roles during the business activities are called action messages. The PIP specifications for a business process define the structure of the action messages and the sequence in which the messages are sent between roles. The exchange sequence of action messages for a specific PIP is described in the specification guide for that PIP .The UML activity diagram illustrates the activities of each PIP. The PIPs follow one or more business transaction patterns, such as request-response or notification. Depending on whether one or two action messages exist in a PIP, a RosettaNet PIP can also be classified as an one action PIP or a two-action PIP. A request-response business transaction pattern results in a two-action PIP, whereas a notification business transaction pattern results in a one-action PIP. The attributes of messages sent between the trading partners are also specified. These attributes are related to the quality of service.

5.2. PIP to BPMN translation method

Assuming the information taken from the specification of each PIP, the modeling of PIP in BPMN is feasible. As the PIP specification cannot give a detailed approach about what exactly happens in each activity, so it is difficult to create an automated methodology to transform PIP in BPMN. A feasible target is to provide thumb rules and instructions in order to succeed in modeling a PIP. In this work, such a methodology is provided for a manual designing and modeling.

The PIP defines two roles for the trading partners that participate in the business process. In BPMN the two roles are translated as the pools of the business process. In each pool, each partner executes each task. The business activities, that PIP has, are the BPMN’s tasks or sub-processes depending on the complexity of the described activity. The messages (business documents) exchanged between the roles during the business activities are referring to BPMN’s data objects. The physical attributes of messages, which are connected to time as execution time or acknowledgement time, are the intermediate time events of BMPN. In the Table 6 is showing the directly translated elements of the recommended method.

PIP specification elements	BPMN elements
Business roles	Pools
Activities	Tasks or Sub-processes
Business documents	Data Objects
Time attributes	Intermediate time Events

Table 6: BPMN direct transformation rules

On the other hand, some parts of the PIP cannot be transformed directly. The modeling of these parts depends on the judge of the developer and the instructions that given through the specification. In the next section, an example of PIP modeled in BPMN.

5.3. Modeling PIP in BPMN Example

The example will describe the transformation PIP3A4, Request Purchase Order. PIP3A4 defines the roles buyer and seller. The business process is divided into one or more business activities. In PIP3A4, Request Purchase Order, the business activities are Request Purchase Order and Confirm Purchase Order. The Request Purchase Order business activity sends a Purchase Order Request from the buyer to the seller. The seller activates the Confirm Purchase Order business activity and sends a Purchase Order Confirmation to the buyer, who acknowledges, at the line level, if the purchase order is accepted, rejected, or pending. In the Figure 29, it is showing the Business Process Scope Diagram, which is provided by the PIP's specification.

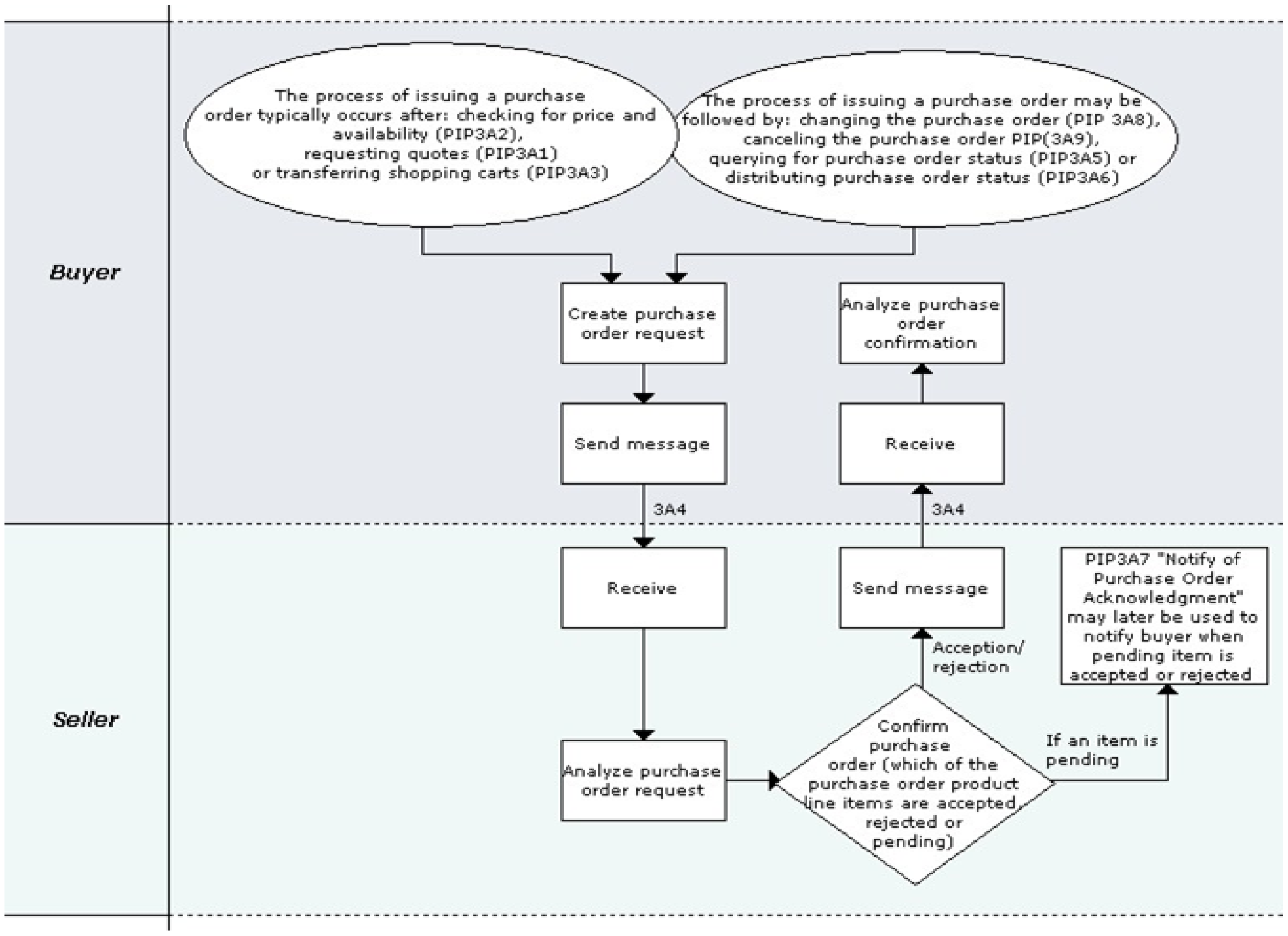

Buyer
Seller
The process of issuing a purchase order typically occurs after: checking for price and availability (PIP3A2), requesting quotes (PIP3A1) or transferring shopping carts (PIP3A3)
The process of issuing a purchase order may be followed by: changing the purchase order (PIP 3A8), canceling the purchase order PIP(3A9), querying for purchase order status (PIP3A5) or distributing purchase order status (PIP3A6)
Create purchase order request
Send message
3A4
Receive
Analyze purchase order request
Analyze purchase order confirmation
Receive
3A4
Send message
Acception/ rejection
Confirm purchase order (which of the purchase order product line items are accepted, rejected or pending)
If an item is pending
PIP3A7 "Notify of Purchase Order Acknowledgment" may later be used to notify buyer when pending item is accepted or rejected

Figure 29: Business Process Scope Diagram

Figure 30 illustrates the UML activity diagram for accomplishing Request Purchase Order business activity for PIP3A4.

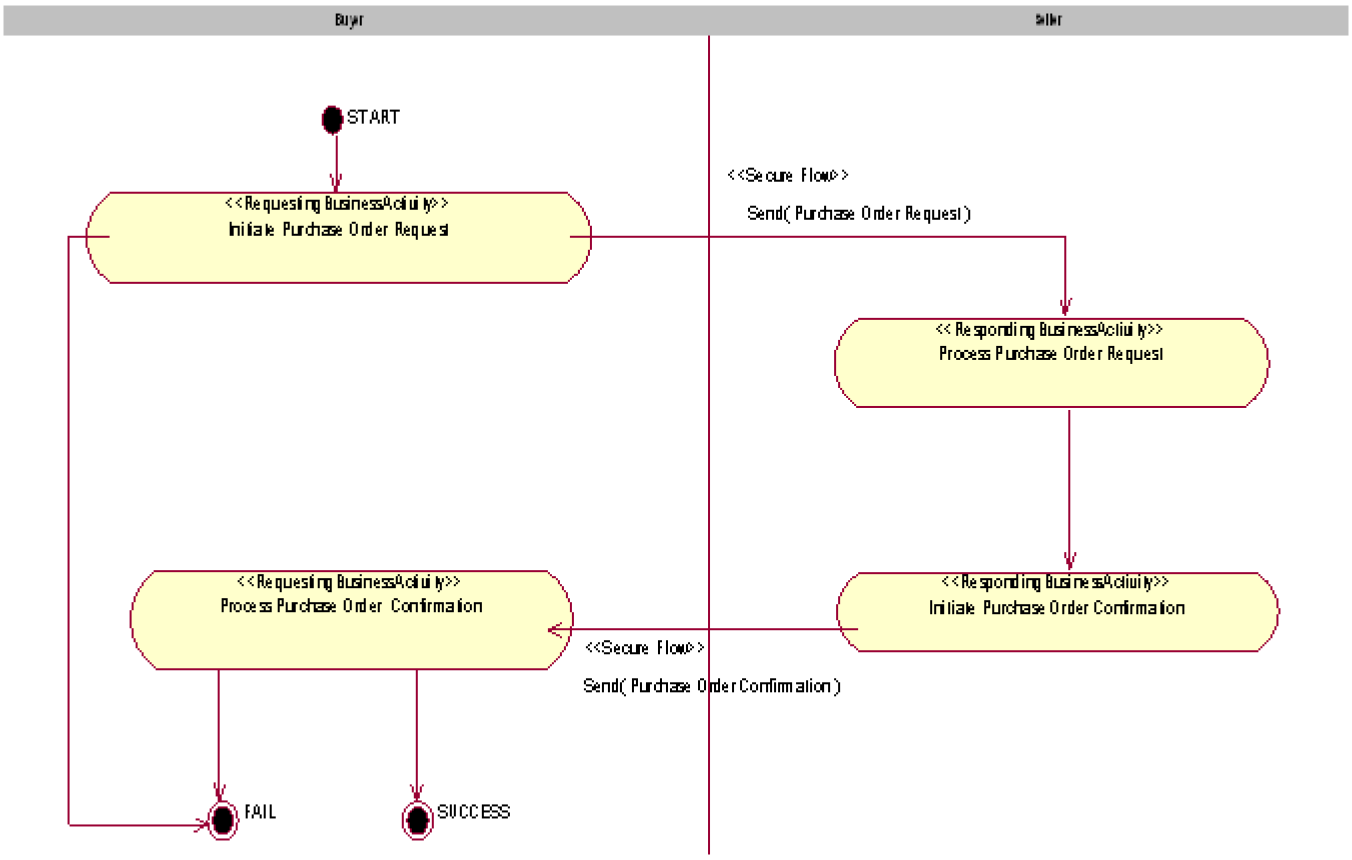

Buyer
Seller
START
<<Requesting BusinessActivity>>
Initiate Purchase Order Request
<<Secure Flow>>
Send(Purchase Order Request)
<<Responding BusinessActivity>>
Process Purchase Order Request
<<Requesting BusinessActivity>>
Process Purchase Order Confirmation
<<Responding BusinessActivity>>
Initiate Purchase Order Confirmation
<<Secure Flow>>
Send(Purchase Order Confirmation)
FAIL
SUCCESS

Figure 30: Activity Diagram of Request Purchase Order

According to the Table 7 the activity needs 24 hours to perform and 2 hours each acknowledgement message.

Business Activity Performance Controls							
Role Name	Activity Name	Acknowledgment of Receipt					
		Non-Repudiation Required?	Time to Acknowledge	Time to Perform	Retry Count	Is Authorization Required?	Non-Repudiation of Origin and Content?
Buyer	Initiate Purchase Order Request	Y	2 hrs	-	3	Y	Y
Seller	Initiate Purchase Order Confirmation	Y	2 hrs	24 hrs	3	Y	Y

Table 7: Business Activity Performance Controls

According to the method proposed above, the BPMN process has two pools Buyer and Seller. It provides 4 main tasks, two for each side. The buyer Initiates (creates) Purchase Order Request. The Seller processes the Order Request for 24 hours and finally sends Confirmation to the buyer. At the end buyer processes the Order Confirmation. Between the exchanges of the messages there is a need of receipt acknowledgement of messages. This action takes 2 hours for each acknowledgement. The Figure 31 demonstrates the BPMN model of PIP3A4.

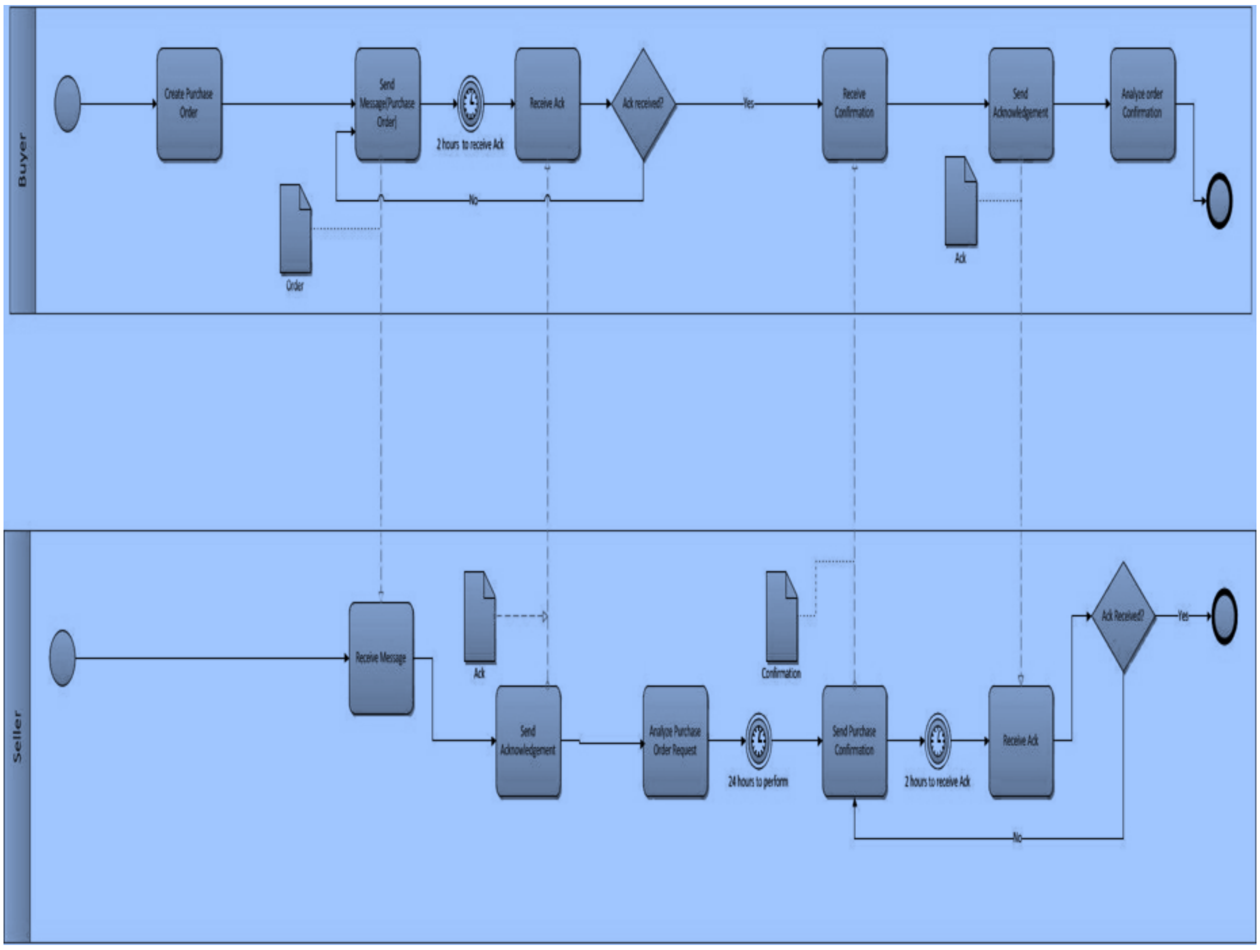


Figure 31: BPMN model of PIP3A4

As it is shown, messages are translated as data objects. Moreover, there are some supplementary tasks, which are decided in order to model the PIP as accurate as possible. The gateways are occurred; depend on the instructions given from the specification.

5.4. Summary

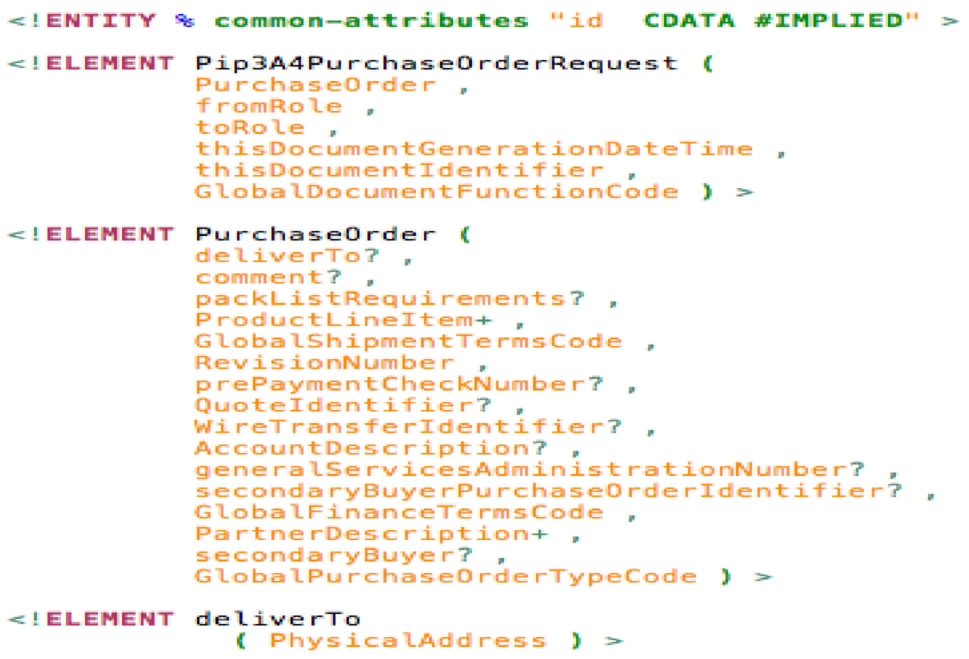
This chapter provided an empirical method to model PIPs through BPMN. It provided some instructions in order to succeed in transforming PIPs into BPMN Processes. It demonstrated the tool, which was used to model the PIPs. Finally, it exhibited an example of a BPMN model occurred from a PIP. The following section will introduce modeling RosettaNet PIPS implemented as web services.

6. RosettaNet PIPs as Web Services

This section introduces the methodology used in this for implementing RosettaNet PIPs as Web Services. It demonstrates business message structure of a PIP. It also provides information about the technologies and tools used in order to succeed in developing such services. In next section introduces the message structure of a PIP.

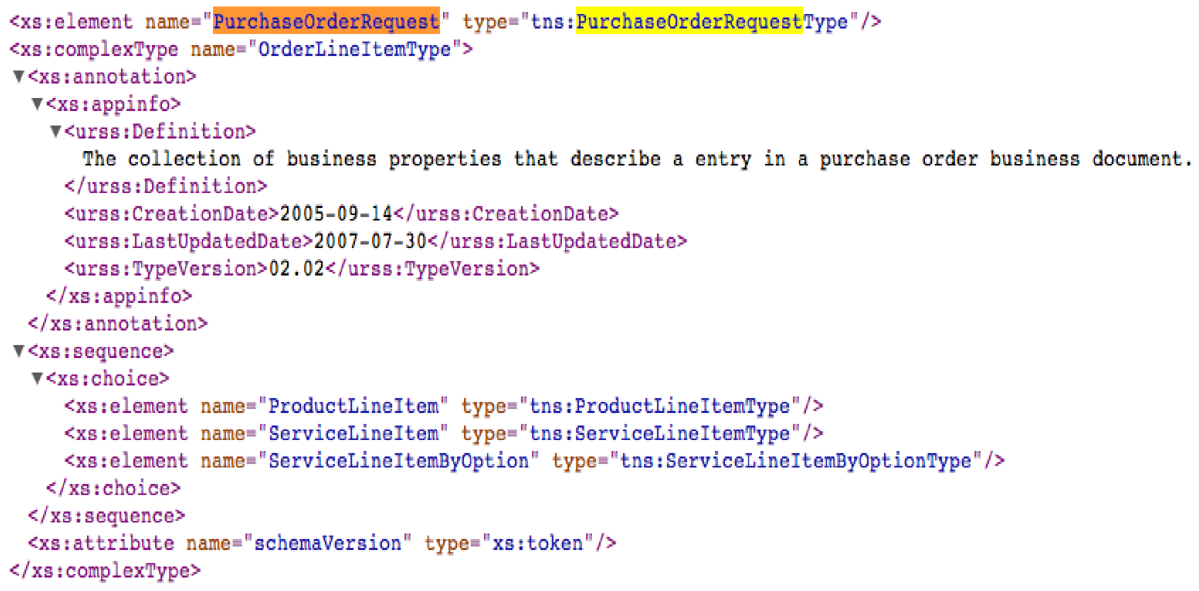
6.1. RosettaNet PIP message structure

The RosettaNet PIPs use the Extensible Markup Language (XML) to describe the data being exchanged between trading partners. There are two XML formats used by RosettaNet. PIPs are at the implementation level represented by DTD and XSD files (Figure 32 and Figure 33).



```
<!ENTITY % common-attributes "id  CDATA #IMPLIED" >

<!ELEMENT Pip3A4PurchaseOrderRequest (
          PurchaseOrder ,
          fromRole ,
          toRole ,
          thisDocumentGenerationDateTime ,
          thisDocumentIdentifier ,
          GlobalDocumentFunctionCode ) >

<!ELEMENT PurchaseOrder (
          deliverTo? ,
          comment? ,
          packListRequirements? ,
          ProductLineItem+ ,
          GlobalShipmentTermsCode ,
          RevisionNumber ,
          prePaymentCheckNumber? ,
          QuoteIdentifier? ,
          WireTransferIdentifier? ,
          AccountDescription? ,
          generalServicesAdministrationNumber? ,
          secondaryBuyerPurchaseOrderIdentifier? ,
          GlobalFinanceTermsCode ,
          PartnerDescription+ ,
          secondaryBuyer? ,
          GlobalPurchaseOrderTypeCode ) >

<!ELEMENT deliverTo
            ( PhysicalAddress ) >
```

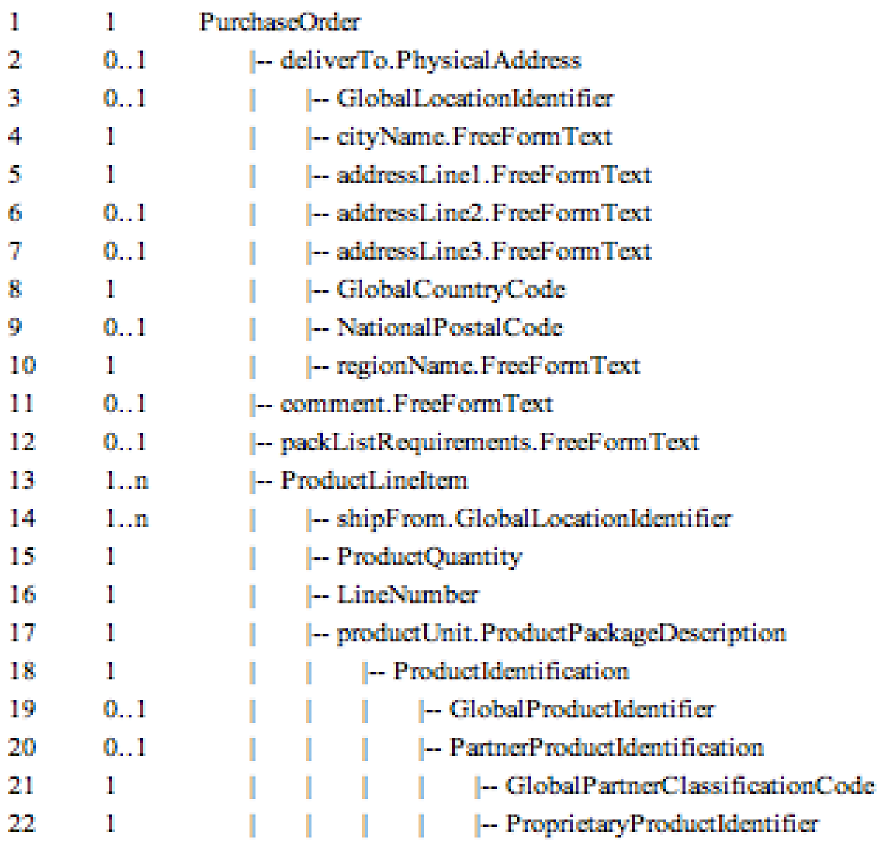
Figure 32: Part of DTD PIP3A4 PurchaseOrderRequest



```
<xs:element name="PurchaseOrderRequest" type="tns:PurchaseOrderRequestType"/>
<xs:complexType name="OrderLineItemType">
▼<xs:annotation>
  ▼<xs:appinfo>
    ▼<urss:Definition>
       The collection of business properties that describe a entry in a purchase order business document.
     </urss:Definition>
     <urss:CreationDate>2005-09-14</urss:CreationDate>
     <urss:LastUpdatedDate>2007-07-30</urss:LastUpdatedDate>
     <urss:TypeVersion>02.02</urss:TypeVersion>
   </xs:appinfo>
 </xs:annotation>
▼<xs:sequence>
  ▼<xs:choice>
     <xs:element name="ProductLineItem" type="tns:ProductLineItemType"/>
     <xs:element name="ServiceLineItem" type="tns:ServiceLineItemType"/>
     <xs:element name="ServiceLineItemByOption" type="tns:ServiceLineItemByOptionType"/>
   </xs:choice>
 </xs:sequence>
 <xs:attribute name="schemaVersion" type="xs:token"/>
</xs:complexType>
```

Figure 33: Part of XSD PIP3A4 PurchaseOrderRequest

Each RosettaNet PIP includes a separate human-readable file that describes all the data elements and the overall document structure. Both business and technical users to understand the total vocabulary required for the PIP use the Message Guidelines. This file is published as an HTM-formatted document that is viewable using a web browser. The Message Guidelines show how each data element is structured within one or more data blocks. This is achieved through an outlining approach. The most indented right text will be the actual data elements. Many lines in the message guideline may be names those reference data blocks, which represent groups of two or more data elements. These names do not hold actual business data. RosettaNet Message Guidelines specify the cardinality of each data element. Cardinality indicates how many values for single data elements are allowed and if the data element is required for PIP compliance. Mandatory data elements are referenced with a one “1” meaning only one data value is required; or as one to many “1..n” meaning at least one and perhaps more data values can be included. Optional data elements may be referenced with as zero or one “0..1” meaning up to one data value may be included or zero to many “0..n” meaning zero, one or many data values may be included. The Figure 34 shows the format of HTM format document.



```
1    1      PurchaseOrder
2    0..1     |-- deliverTo.PhysicalAddress
3    0..1     |     |-- GlobalLocationIdentifier
4    1        |     |-- cityName.FreeFormText
5    1        |     |-- addressLine1.FreeFormText
6    0..1     |     |-- addressLine2.FreeFormText
7    0..1     |     |-- addressLine3.FreeFormText
8    1        |     |-- GlobalCountryCode
9    0..1     |     |-- NationalPostalCode
10   1        |     |-- regionName.FreeFormText
11   0..1     |-- comment.FreeFormText
12   0..1     |-- packListRequirements.FreeFormText
13   1..n     |-- ProductLineItem
14   1..n     |     |-- shipFrom.GlobalLocationIdentifier
15   1        |     |-- ProductQuantity
16   1        |     |-- LineNumber
17   1        |     |-- productUnit.ProductPackageDescription
18   1        |     |     |-- ProductIdentification
19   0..1     |     |     |     |-- GlobalProductIdentifier
20   0..1     |     |     |     |-- PartnerProductIdentification
21   1        |     |     |     |     |-- GlobalPartnerClassificationCode
22   1        |     |     |     |     |-- ProprietaryProductIdentifier
```

Figure 34: Part of HTM document from PIP3A4

6.2. Implementation

As it was discussed above the messages of PIPs are described into XML documents (XSD or DTD). In real world, organizations that want to use PIPS decide together which PIPS they include in their transactions and they implement them according to their needs. This means that they do not include the whole information of each message because of its huge size but only the elements that they need.

The implementation of PIPS as Web Services is a complicated process. The language is used for the implementation is Java. The Figure 35 describes the whole procedure that must be followed. The XSD or DTD, which describe the data elements of the PIP message, should be turned into Java Classes. The Java Classes are used to create the proper PIP message. The PIP message is sent through the web service. Depending on the PIP, there are one or two web services. If the PIP is one-action, there is one web service that sends the message.

If the PIP is two-action, there are two web services, which send messages.

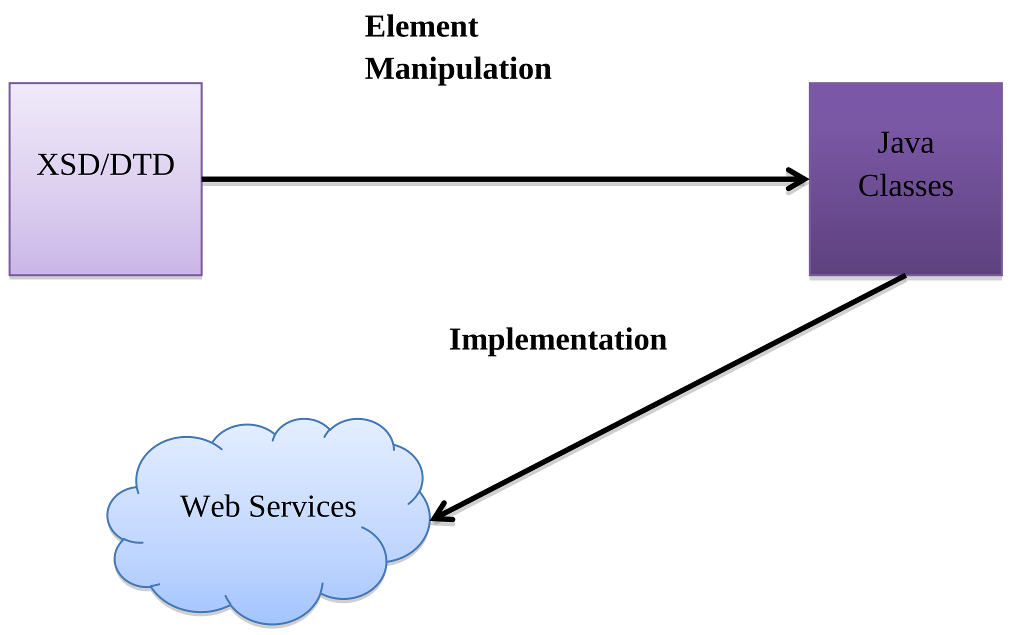


Figure 35: Architecture of PIP web services

6.3. Data Binding

XML schemas or DTD describes Rosetta Net messages. A XML Schema describes the structure of an XML document. The XSD could be transformed in Java Classes. There are two ways to accomplish such a target.

The one way is to create the related java classes manually according to XSD. This procedure is very lengthy, particularly in mapping the PIP XSD to java classes because of the huge size of data that is consisting.

The other way, which is used to this work, is the use of a binding compiler that creates schema-derived classes. Such a compiler is the Java Architecture for XML Binding (JAXB) [48], which this work is based on. JAXB allows mapping Java classes to XML representations. JAXB provides two main features: the ability to marshal Java objects into XML and the inverse, i.e. to un-marshal XML back into Java objects. In other words, JAXB allows storing and retrieving data in memory in any XML format, without the need to implement a specific set of XML loading and saving routines for the program's class structure. In Figure 36, the whole process of JAXB is described.

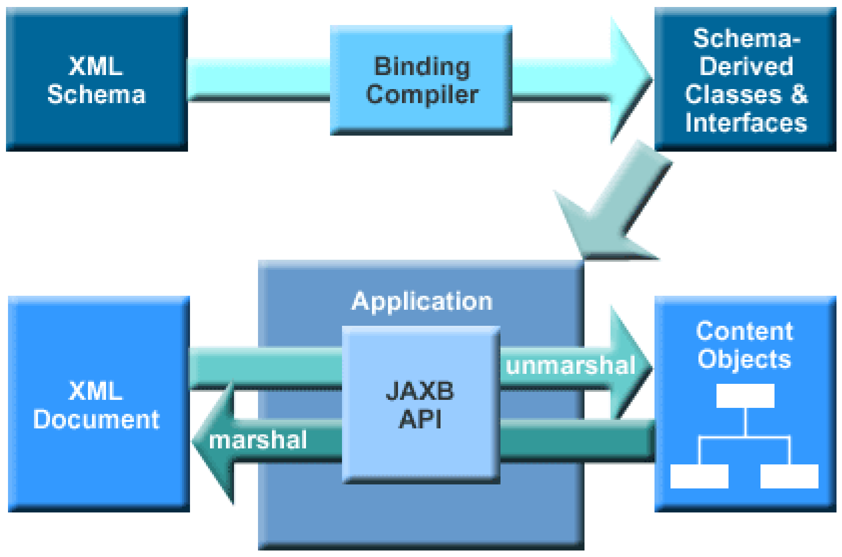


Figure 36: JAXB process

In this work, the XSD or DTD document is processed by JAXB and creates the derived Java Classes. These classes are manipulated in such way, in order to become the constructing elements for the Web Services. As there are some conflicts about the produced bindings' results, there are manual modifications. The produced classes are implemented and developed in order to satisfy the specifications and the guidelines, which are provided by PIP.

For Example, PIP3A4 describes a purchase order request. One of the DTDs is provided demonstrates elements that describe a purchase request. The DTD is processed through JAXB and provides a skeleton of derived java classes. The procedure is shown in Figure.

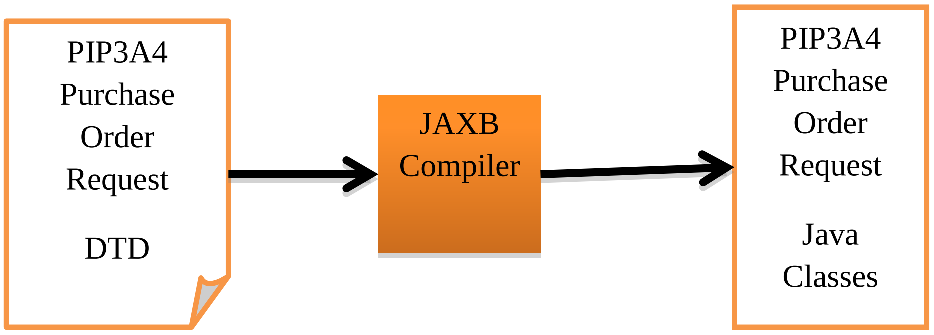


Figure 37: JAXB procedure in PIP3A4

An instance of an element that is included in the Purchase Order Request is the Purchase Order (see Figure 38).

```
<!ELEMENT PurchaseOrder (deliverTo?, comment?, packListRequirements?,
ProductLineItem+, GlobalShipmentTermsCode, RevisionNumber,
prePaymentCheckNumber?, QuoteIdentifier?, WireTransferIdentifier?,
AccountDescription?, generalServicesAdministrationNumber?,
secondaryBuyerPurchaseOrderIdentifier?, GlobalFinanceTermsCode,
PartnerDescription+, secondaryBuyer?, GlobalPurchaseOrderTypeCode)>
```

Figure 38: Purchase Order Element

When it is compiled through JAXB, it produces the class Purchase Order, which contains the elements are included the DTD part shown in Figure 38.A part of sample code is shown Figure 39. While Figure 40 shows the UML class Diagram of whole class.

```
public class PurchaseOrder {

protectedDeliverTodeliverTo;

protected Comment comment;

protectedPackListRequirementspackListRequirements;

@XmlElement(name = "ProductLineItem", required = true)

protected List<ProductLineItem>productLineItem;
```

Figure 39: Sample Java code of Purchase Order Class

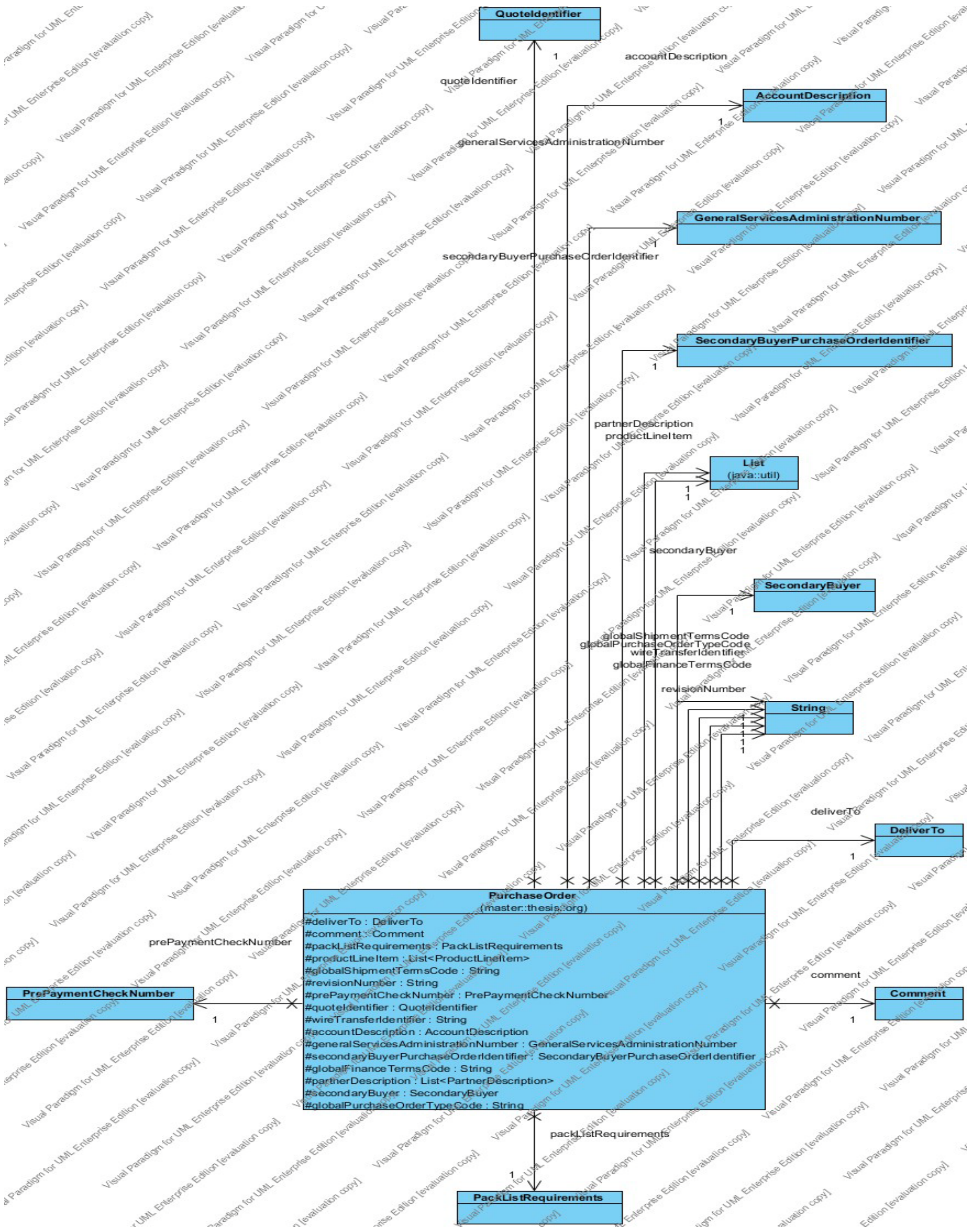

QuoteIdentifier
AccountDescription
GeneralServicesAdministrationNumber
SecondaryBuyerPurchaseOrderIdentifier
List
(java::util)
SecondaryBuyer
String
DeliverTo
Comment
PrePaymentCheckNumber
PackListRequirements
PurchaseOrder
(master::thesis::org)
#deliverTo : DeliverTo
#comment : Comment
#packListRequirements : PackListRequirements
#productLineItem : List<ProductLineItem>
#globalShipmentTermsCode : String
#revisionNumber : String
#prePaymentCheckNumber : PrePaymentCheckNumber
#quoteIdentifier : QuoteIdentifier
#wireTransferIdentifier : String
#accountDescription : AccountDescription
#generalServicesAdministrationNumber : GeneralServicesAdministrationNumber
#secondaryBuyerPurchaseOrderIdentifier : SecondaryBuyerPurchaseOrderIdentifier
#globalFinanceTermsCode : String
#partnerDescription : List<PartnerDescription>
#secondaryBuyer : SecondaryBuyer
#globalPurchaseOrderTypeCode : String
quoteIdentifier
accountDescription
generalServicesAdministrationNumber
secondaryBuyerPurchaseOrderIdentifier
partnerDescription
productLineItem
secondaryBuyer
globalShipmentTermsCode
globalPurchaseOrderTypeCode
wireTransferIdentifier
globalFinanceTermsCode
revisionNumber
deliverTo
comment
prePaymentCheckNumber
packListRequirements

Figure 40: Purchase Order UML class diagram

The DTD is transformed into this group of Java classes is approximately shown in the (because of its huge size every class is not demonstrated).

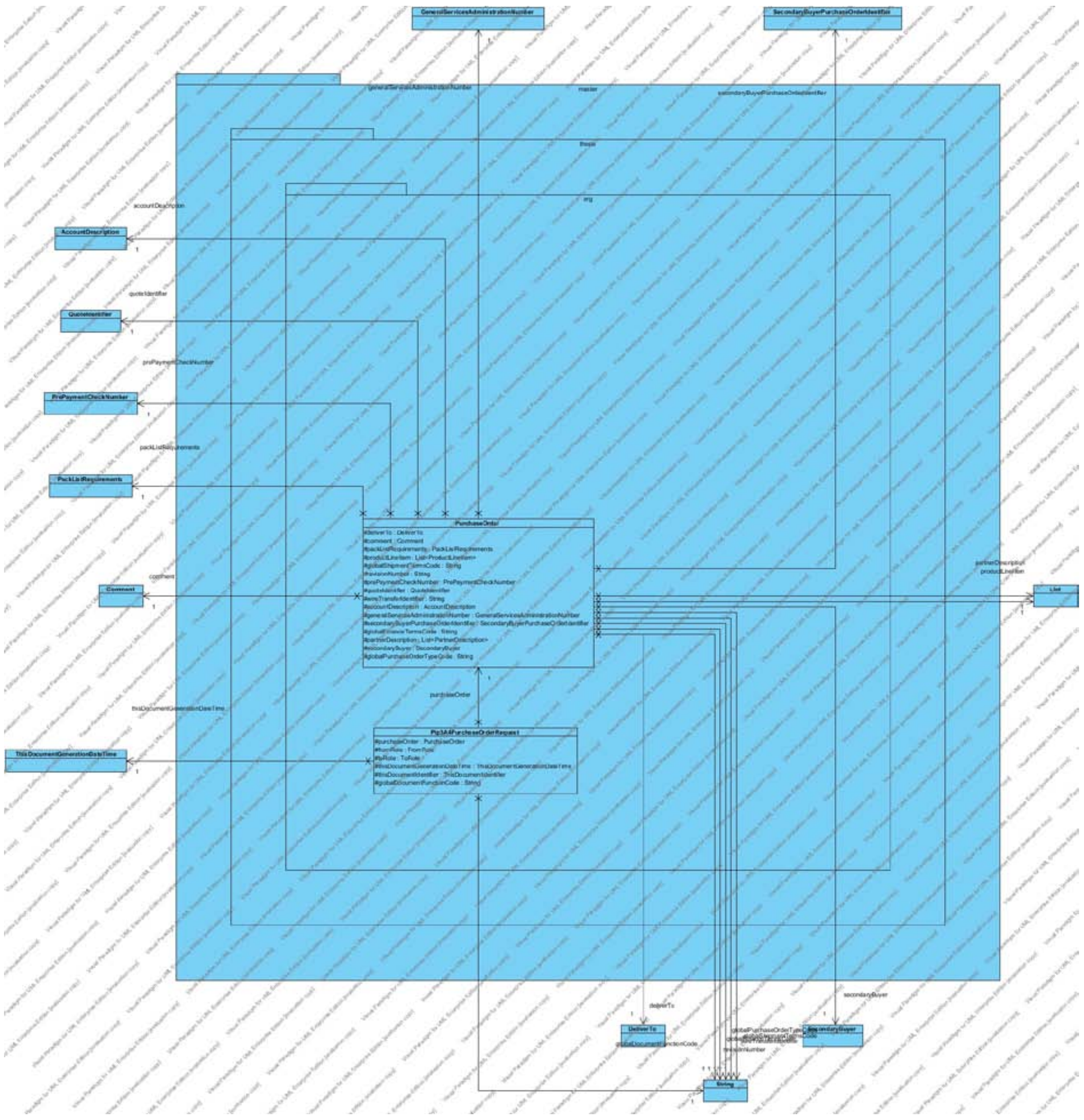


Figure 41: DTD to Java Classes (PIP3A4 purchase order request)

6.4. Web Services

As it was discussed before, PIPS are designed to describe relations and transactions between two business entities (organizations). Depending on whether one or two action messages exist in a PIP, a RosettaNet PIP can also be classified as a one-action PIP or a two-action PIP (Figure 42). A request-response business transaction pattern results in a two-action PIP, whereas a notification business transaction pattern results in a one-action PIP. If there is an one-action PIP, one web service is required to implement the transaction. One organization sends through the web service the required information to the other as the PIP specification introduces. If there is a two-action PIP, two web services are required to implement the transaction. The two organizations communicate and exchange information through the web services. One side requests something and the other side responses in order to answer if it could fulfill the request.

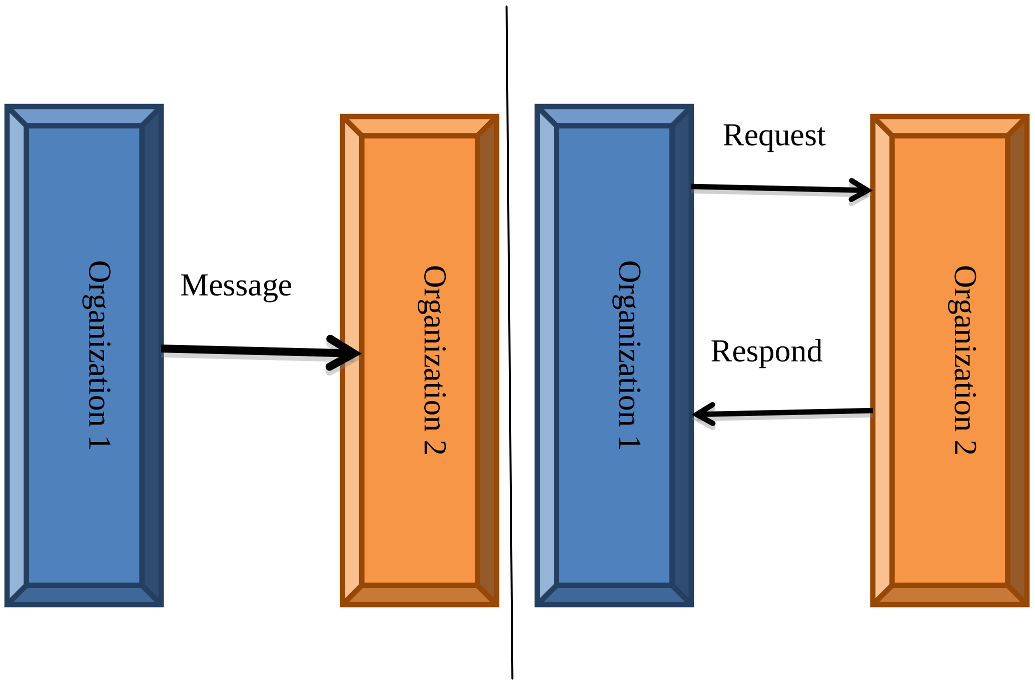


Figure 42: One-action PIP | Two-Action PIP

The methodology is followed, in order to implement such web services, is based on the bottom up approach. The manipulated derived Java classes generated by JAXB are used as the basic construct elements to create the web services. Its PIP message is constructed and sent through the web service.

The web services implementation is based on Java API for XML Web Services (JAX-WS) [49]. JAX-WS relies heavily on the use of annotations as it specified. The two annotations in the RetailerService class (see Figure): @WebService and @WebMethod. A valid endpoint implementation class must include a @WebService annotation. The annotation marks the class as a web service. The name property value in the @WebService annotation identifies a Web Service Description Language (WSDL) portType (in this case, "RetailerServicePortType"). The serviceName ("RetailerService") is a WSDL service. TargetNamespace specifies the XML namespace used for the WSDL (in this case targetNamespace = "http://retailerService/). All the properties are optional. The @WebMethod annotation exposes a method as a web service method. The operationName property value in the annotation of the RetailerServiceclass identifies a WSDL operation (in this case, CreatePurchaseOrder). Both properties are optional. If you don't specify them, the WSDL operation value defaults to method name, and the action value defaults to the targetNamespace of the service.

```
@WebService( name = "RetailerServicePortType", serviceName = "RetailerService",
targetNamespace = "http://retailerService/")

public class RetailerService {

        @WebMethod

        public void createPurchaseOrderRequest(

                @WebParam(name = "carInformationList")
CarInformationListcarInformationList)
```

Figure 43: Part of Web Service

The @WebParam annotation allows to specify the direction of the parameter, if the parameter will be placed in the SOAP header, and other properties of the generated wsdl:part.

On the other hand, the @WebResult annotation allows specifying the properties of the wsdl:part that is generated for the method's return value.

As the Web Service is implemented, it is deployed on Apache Tomcat [50]as a WAR file. The WAR file includes the WSDL file, which provides the operations and the elements of the web service. Figure 44provides part of WSDL targeting the PIP3A4 purchase order request. The RetailerService_schema1.xsd provides information about the elements of the WSDL.

```
?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.4-b01-. -->

<definitionstargetNamespace="http://retailerService/" name="RetailerService"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://retailerService/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<types><xsd:schema>

<xsd:import namespace="http://retailerService/" schemaLocation="RetailerService_schema1.xsd"/>

</xsd:schema></types>

<message name="createPurchaseOrderRequest">

<part name="parameters" element="tns:createPurchaseOrderRequest"/>

</message>

<message name="createPurchaseOrderRequestResponse">

<part name="parameters" element="tns:createPurchaseOrderRequestResponse"/>

</message>

……………………………………………………………………………………………………………………

<portType name="RetailerServicePortType">

<operation name="createPurchaseOrderRequest">

<input message="tns:createPurchaseOrderRequest"/>

<output message="tns:createPurchaseOrderRequestResponse"/>

</operation>

……………………………………………………………………………………………………………………

<service name="RetailerService">

<port name="RetailerServicePortTypePort" binding="tns:RetailerServicePortTypePortBinding">

<soap:address location="http://localhost:8080/RetailerService/RetailerService"/>

</port></service></definitions
```

Figure 44: WSDL describing PIP3A4 purchase request order

6.5. Summary

This chapter provided the implementation methodology of Web Services based on RosettaNet PIPs. It introduced the techniques and the tools used. It demonstrated information for JAXB bindings, implementation code, and the WSDL parts, and deployment instructions. The following section will show an example of using the actual implementation in a simple simulated world scenario based on an Automobile

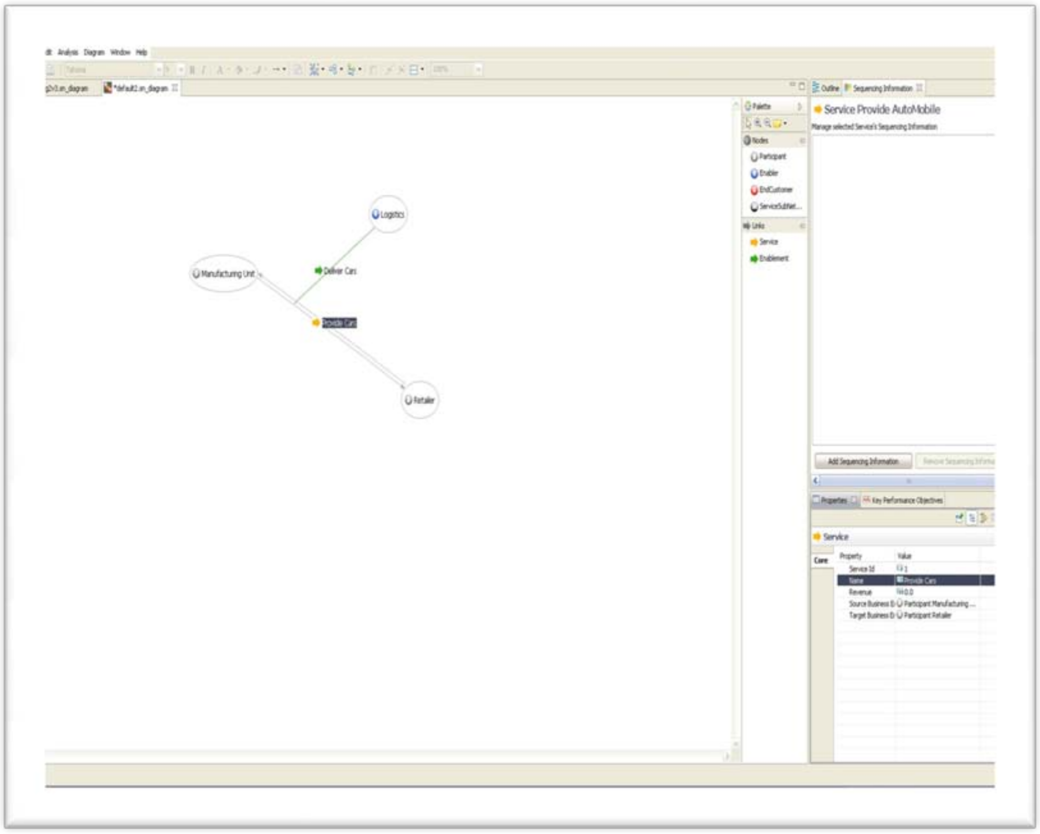
7. Automobile scenario based on RosettaNet PIPs services choreography.

This chapter describes the use of RosettaNet PIPs in a real-world Scenario. RosettaNet PIPs provide the suitable message exchange patterns in order to fulfill the transactions between business entities. The business entities activities are implemented as Web Services based on RosettaNet Pips. The whole process is divided into BPEL orchestrations that are communicate as choreography. Next section introduces the scenario is used.

7.1. Automotive case study

The business scenario to be used on this thesis is based on an automotive industry case study. It is focused on a complex and geographically distributed supply chain in the automotive sector and has been proposed by the companies 360Fresh and IBM in the S-Cube deliverable "Report on Common Pilot Cases" [51]. Automobile Incorporation (Auto Inc), located in South East Asia, is a local branch of a large enterprise in the automobile industry in Europe, comprising a regional headquarter in Singapore, a manufacturing factory in Vietnam, several regional distribution and logistics provider, and several warehouses located in different countries in South East Asia. Auto Inc sells automobile products to retail customers in the surrounding countries. The main business tasks of the manufacturing factory include importing and assembling automobile body parts from the EU headquarter supplier, importing and assembling other parts (like wheels, brakes, seats, etc.) from regional suppliers, painting, integrating accessories (e.g. air conditioner, CD player, etc), testing and releasing the final products. Other material and semi-finished products can be ordered from the regional suppliers in surrounding countries. Depending on the product specifications, the assembling, integrating and painting tasks use varying materials and products, and might be executed in disparate ways as well.

Different distribution logistics providers participate in the process to provide the transportation of finished products from the manufactory to the warehouses, and from the warehouses to the retail customers. The providers are selected according to the transportation routes and rules. With the intention of making the implementation feasible on the time span available, a simplified version of the aforementioned business scenario has been chosen. Therefore, only the "Order" business process, that is, the Retail Customer's ordering of new cars, including its three participants (AutoInc Manufacturing Unit, Retail Customer and Logistics Provider) must be considered. The simplified service network is shown at Figure 45.



Property	Value
Service Id	1
Name	Provide Cars
Revenue	0.0
Source Business E	Participant Manufacturing ...
Target Business E	Participant Retailer

Figure 45: Service Network Designed by Snap[52][53]

The Auto Manufacturing Unit is responsible for constructing new cars and providing cars the Retailers. The Retailers order cars from the Auto Manufacturing Unit. The logistics makes the shipments of the cars (enables the service is provided from Manufacturing Unit to Retailer).

7.2. Description of the Business Process' Orchestrations and Web Services

In order to allow the interactions between the businesses partners on the scenario described in a loosely coupled, reusable and composable manner, each of these business entities must expose its interfaces as Web Services. Such interfaces must be in accordance to the standards used to describe the business transactions in this project, the RosettaNet PIPS. This section describes briefly what these are, which of them were used on the model developed, how the orchestrations and choreographies described by them were modeled and, finally, identifies and characterizes the Web Services to be implemented.

The business scenario described in this document represents a typical supply chain interaction among partners. Therefore, it makes sense to apply RosettaNet PIPs to enrich the abstract business model created for such a service network and move towards standardization of the communication among the participants. Table 8 shows the PIPs selected as applicable to the model, while the following paragraphs describe which ones were chosen to be used on which situations and explain why they are applicable to such cases.

Interactions	PIP	Name
Retail Outlet → Manufacturing Unit	**PIP3A4**	**Request Purchase Order**
Manufacturing Unit→ Logistics Provider	**PIP3B12**	**Request Shipping Order**
Logistics Provider → Manufacturing Unit	**PIP3B13**	**Notify Shipping Order Confirmation**
Logistics Provider → Retailer Outlet	**PIP3C5**	**Notify Billing Statement**

Table 8: PIPs selected for the scenario

First of all, the Retail Outlet places purchase orders to the AutoInc Manufacturing Unit, just as standardized on PIP 3A4, requesting cars to be produced to supply its local stock. The Manufacturing Unit, then, receives the order and analyses it to see if what the client want is currently feasible. If not, it sends an order rejection confirmation back, which is received by the Retail Outlet and the process terminates. If the purchase order can be fulfilled, AutoInc Manufacturing Unit sends the order acceptance confirmation back and starts manufacturing the ordered cars. The Retail Outlet receives this confirmation and waits for the billing information to arrive.

After producing all the cars requested on the purchase order, the AutoInc Manufacturing Unit issues a shipping order to the Logistics Provider, detailing which cars must be transported from which stock warehouse to which customer and what the total price of the order was. This request of shipping orders is made according to PIP 3B12.

After receiving the shipping order, the Logistics Provider processes it and sends a confirmation back to the Manufacturing Unit, stating the details of the collection and delivery of the cars. This notification of the shipping order confirmation is done as put on PIP 3B13.

When the Manufacturing Unit receives the shipping order confirmation, it does the final preparation of the ordered cars for the shipment to the customer. Finally, the Logistics Provider collects the cars and starts the transportation process. In parallel, it adds the shipment price to the Manufacturing Unit price and notifies the billing to the Retail Outlet. This notification is done as identified on PIP 3C5.

7.3. Business Entities' Web Services

As aforementioned, the choreographies among the business partners are enabled by the use of Web Services. The relation of all the needed interfaces as well as a high-level description of each one is given on this section. The inputs and outputs of each service are also listed. Since the complete XML schemas for the messages to be exchanged on the PIPs are very detailed and long, only a summarized description is provided in this document.

7.3.1. Retail Outlet

ConfirmPurchaseOrder: this service is used by the business partners that want to send the confirmations of requested purchase orders to the Retail Outlet, one part of the PIP 3A4. On the described business scenario, the AutoInc Manufacturing Unit is the partner that uses this interface. The interface is described in the Table 9.

Input Message Structure (ConfirmPurchaseOrderRequest)	
Purchase Order Information	Id
	Status(Accepted/Rejected)
	Number of Items
	Final Manufactory Price
	Expected Shipping Date
Output Message Structure (ConfirmPurchaseOrderRespone)	
Customer Information	Company Name
	Company Address
	Phone Number
	Company Tax Number
Purchase Order Information	Id
	Acknowledge Flag

Table 9: ConfirmPurchaseOrder Interface

SendBillingStatement: this service is used by the business partners that want to send billing information for a given purchase order including shipping costs to the Retail Outlet, as on PIP 3C5. On the described business scenario, the Logistics Provider is the partner that uses this interface (Table 10).

Input Message Structure (SendBillingStatementRequest)	
Logistics Provider Information	**Company Name**
	Company Address
	Phone Number
	Company Tax Number
Purchase Order Information	**Id**
	Number of Items
	Final Manufactory Price
Shipping Order Information	**Id**
	Delivery Address
	Shipment Date
	Expected Delivery Date
	Delivery Agent
	Total Shipment Price
Billing Information	**Id**
	Total Price
	Payment Due Date
	Payment Details
Output Message Structure (SendBillingStatementResponse)	
Customer Information Company Name	**Customer Information Company Name**
	Company Address

Table 10: SendBilling Statement

7.3.2. Logistics Provider

CreateShippingOrder: the business partners that want to request a new shipping order to the Logistics Provider, as part of the PIP 3B12, use this service. On the described business scenario, the AutoInc Manufacturing Unit is the partner that uses this interface (Table 11).

Input Message Structure (CreateShippingOrderRequest)	
Customer Information	Company Name
	Company Address
	Phone Number
	Company Tax Number
Car Information	Chassis Number
	Type
	Color
	Location
Purchase Order Information	Id
	Number of Items
	Final Manufactory Price
Delivery Information	Expected Time to Delivery
	Must Meet TTD
	Delivery Address
Destination Information	Company Name
	Company Address
	Phone Number
	Company Tax Number
Output Message Structure (CreateShippingOrderResponse)	
Logistics Provider Information	Company Name
	Company Address
	Phone Number
	Company Tax Number
Shipping Order Information	Id
	Number of Items
	Delivery Address
	Expected Shipment Date
	Expected Delivery Date
	Total Shipment Price

Table 11: CreateShippingOrder Interface

7.3.3. AutoInc Manufacturing Unit

CreatePurchaseOrder: the business partners that want to request a new purchase order to the AutoInc Manufacturing Unit, as part of the PIP 3A4, use this service. On the described business scenario, the Retail Outlet is the partner that uses this interface (Table 12).

Input Message Structure (CreatePurchaseOrderRequest)	
Customer Information	Customer Information Company Name
	Company Address
	Phone Number
	Company Tax Number
Car Information	Type
	Color
	Accessories (list)
	Quantity Requested
Delivery Information	Expected Time to Deliver
	Cancel If Cannot Meet ET
	Delivery Address
Output Message Structure (CreatePurchaseOrderResponse)	
	Company Name
	Company Address
	Phone Number
	Company Tax Number
Purchase Order Information	Id
	Status (Pending)

Table 12: CreatePurchaseOrder Interface

ConfirmShippingOrder: the business partners that want to send the confirmations of requested shipping orders to the AutoInc Manufacturing Unit, as part of the PIP 3B12, use this service. On the described business scenario, the Logistics Provider is the partner that uses this interface (Table 13).

Input Message Structure (ConfirmShippingOrderRequest)	
Shipping Order Information	Id
	Number of Items
	Delivery Address
	Expected Shipping Date
	Expected Delivery Date
	Total Shipment Price
Output Message Structure (ConfirmShippingOrderResponse)	
Shipping Order Information	Id
	Acknowledge Flag

Table 13: ConfirmShippingOrder Interface

NotifyShippingOrderConfirmation: this service is used by the business partners that want to send confirmations of the shipment of items from a certain shipping order to the AutoInc Manufacturing Unit, as stated on PIP 3B13. On the described business scenario, the Logistics Provider is the partner that uses this interface (Table 14).

Input Message Structure (NotifyShippingOrderConfirmationRequest)	
Shipping Order Information	Id
	Number of Items
	Delivery Address
	Shipment Date
	Expected Delivery Date
	Total Shipment Price
Output Message Structure (NotifyShippingOrderConfirmationResponse)	
Shipping Order Information	Id
	Acknowledge Flag

Table 14: NotifyShippingOrderConfirmation Interface

In order to give a better overview of the defined Web Services and how the partners use them to interact, Figure 46has been designed.

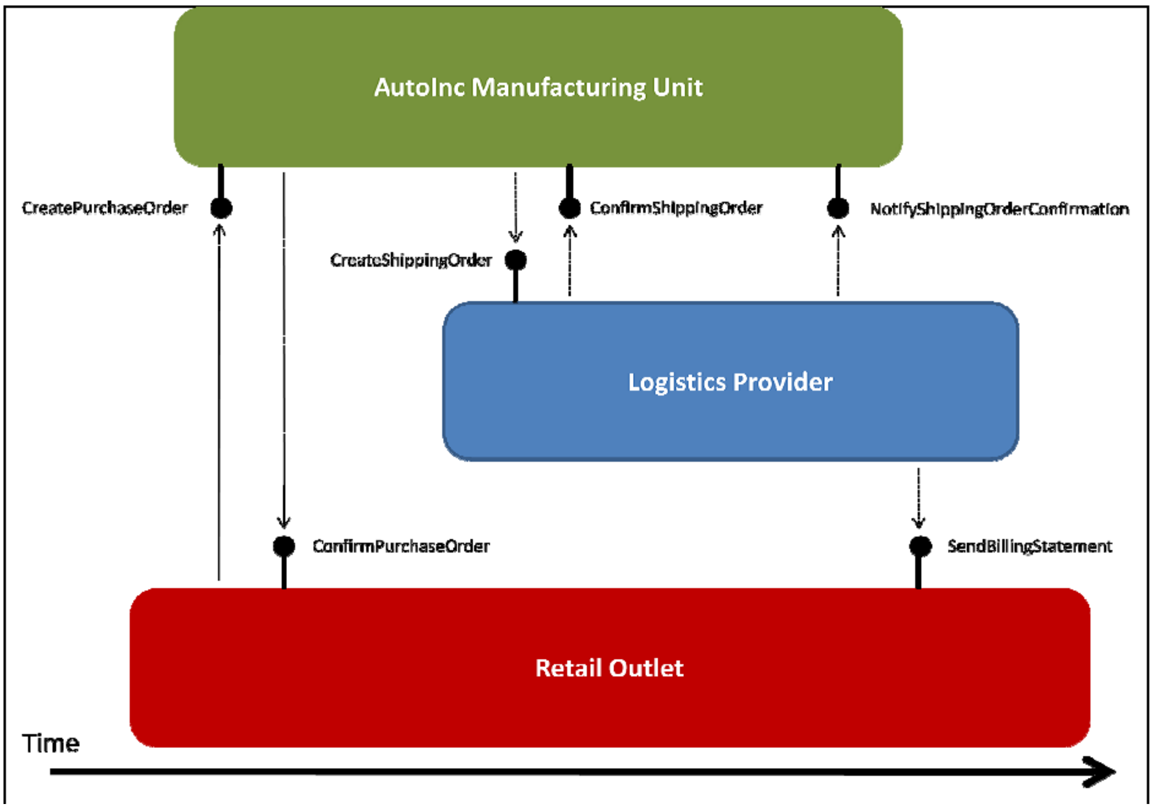


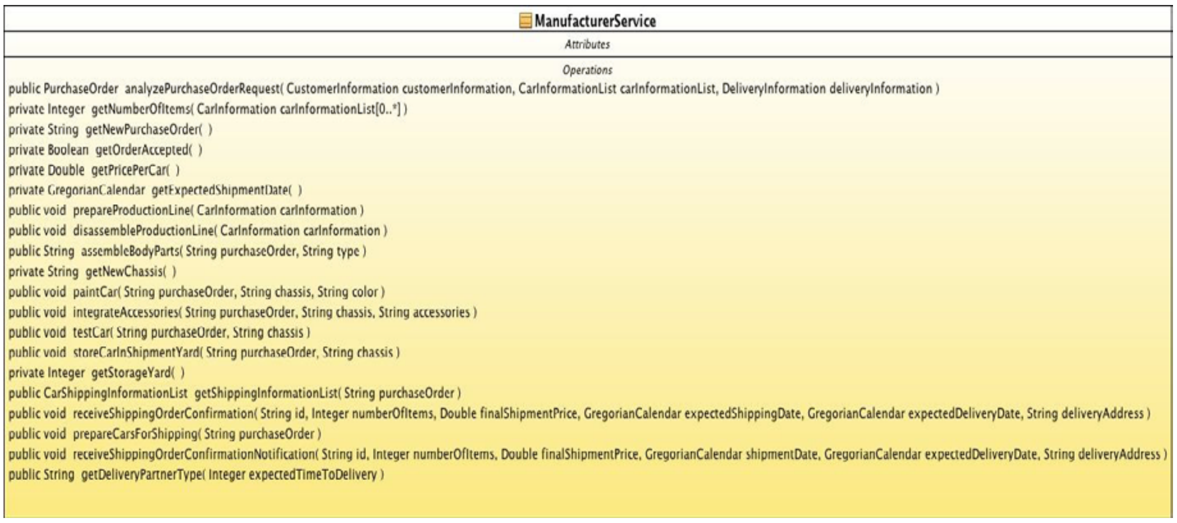
Figure 46: Overview of all interactions between the business partners and the related Web Services

7.4. Implementation of Business Entities Web Services

The choreographies among the business partners are enabled by the use of Web Services. There are three main web services that are implemented here – one each for the Manufacturing Unit, the Retail Outlet and the Logistics Provider respectively. The following describe the operations supported by each web service, and also the inputs and outputs of each service.

Manufacturing web service is responsible for the activities of the Manufacturing Unit, which involve three main external web service operations via Analyze Purchase Order, Receive Shipping Order Confirmation and Receive Shipping Order Confirmation Notification. However there are several other operations that invoke internal web services for processes like Prepare Production Line, Disassemble Production Line, Assemble Body Parts, and Paint Cars and so on.

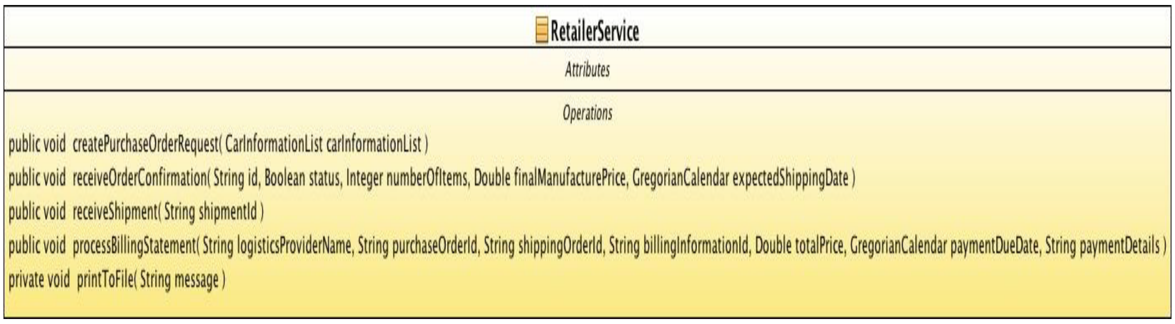
In the whole process, there is a necessity to maintain certain information somewhere in between partner interactions. In this case, for a given production order, the information about the cars that are produced are to be stored in a database. In order to provide this facility, an in-memory database has been implemented using the Singleton Design Pattern. The Chassis-Database is the data structure that has been created here to hold the information in the form of Hash Maps. The Interface of such Web Service is provided in the Figure.



```
ManufacturerService
Attributes
Operations
public PurchaseOrder  analyzePurchaseOrderRequest( CustomerInformation customerInformation, CarInformationList carInformationList, DeliveryInformation deliveryInformation )
private Integer  getNumberOfItems( CarInformation carInformationList[0..*] )
private String  getNewPurchaseOrder( )
private Boolean  getOrderAccepted( )
private Double  getPricePerCar( )
private GregorianCalendar  getExpectedShipmentDate( )
public void  prepareProductionLine( CarInformation carInformation )
public void  disassembleProductionLine( CarInformation carInformation )
public String  assembleBodyParts( String purchaseOrder, String type )
private String  getNewChassis( )
public void  paintCar( String purchaseOrder, String chassis, String color )
public void  integrateAccessories( String purchaseOrder, String chassis, String accessories )
public void  testCar( String purchaseOrder, String chassis )
public void  storeCarInShipmentYard( String purchaseOrder, String chassis )
private Integer  getStorageYard( )
public CarShippingInformationList  getShippingInformationList( String purchaseOrder )
public void  receiveShippingOrderConfirmation( String id, Integer numberOfItems, Double finalShipmentPrice, GregorianCalendar expectedShippingDate, GregorianCalendar expectedDeliveryDate, String deliveryAddress )
public void  prepareCarsForShipping( String purchaseOrder )
public void  receiveShippingOrderConfirmationNotification( String id, Integer numberOfItems, Double finalShipmentPrice, GregorianCalendar shipmentDate, GregorianCalendar expectedDeliveryDate, String deliveryAddress )
public String  getDeliveryPartnerType( Integer expectedTimeToDelivery )
```

Figure 47: Manufacturing methods

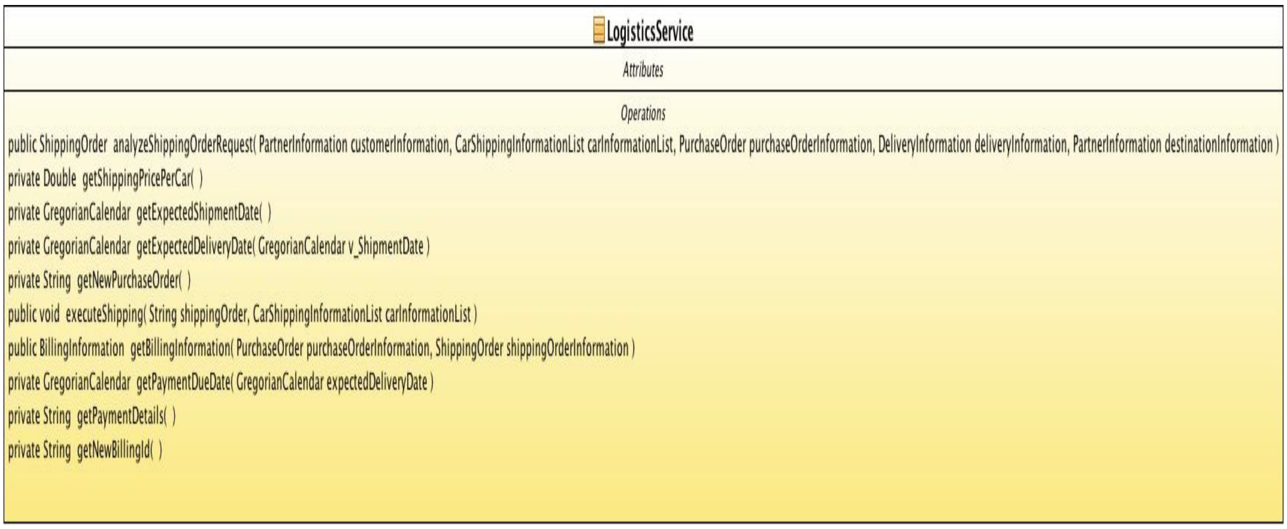
The Retail Outlet service methods are demonstrated in Figure.



```
RetailerService
Attributes
Operations
public void  createPurchaseOrderRequest( CarInformationList carInformationList )
public void  receiveOrderConfirmation( String id, Boolean status, Integer numberOfItems, Double finalManufacturePrice, GregorianCalendar expectedShippingDate )
public void  receiveShipment( String shipmentId )
public void  processBillingStatement( String logisticsProviderName, String purchaseOrderId, String shippingOrderId, String billingInformationId, Double totalPrice, GregorianCalendar paymentDueDate, String paymentDetails )
private void  printToFile( String message )
```

Figure 48: RetailerService methods

The Logistics service methods are shown in Figure 49.



```
LogisticsService
Attributes
Operations
public ShippingOrder  analyzeShippingOrderRequest( PartnerInformation customerInformation, CarShippingInformationList carInformationList, PurchaseOrder purchaseOrderInformation, DeliveryInformation deliveryInformation, PartnerInformation destinationInformation )
private Double  getShippingPricePerCar( )
private GregorianCalendar  getExpectedShipmentDate( )
private GregorianCalendar  getExpectedDeliveryDate( GregorianCalendar v_ShipmentDate )
private String  getNewPurchaseOrder( )
public void  executeShipping( String shippingOrder, CarShippingInformationList carInformationList )
public BillingInformation  getBillingInformation( PurchaseOrder purchaseOrderInformation, ShippingOrder shippingOrderInformation )
private GregorianCalendar  getPaymentDueDate( GregorianCalendar expectedDeliveryDate )
private String  getPaymentDetails( )
private String  getNewBillingId( )
```

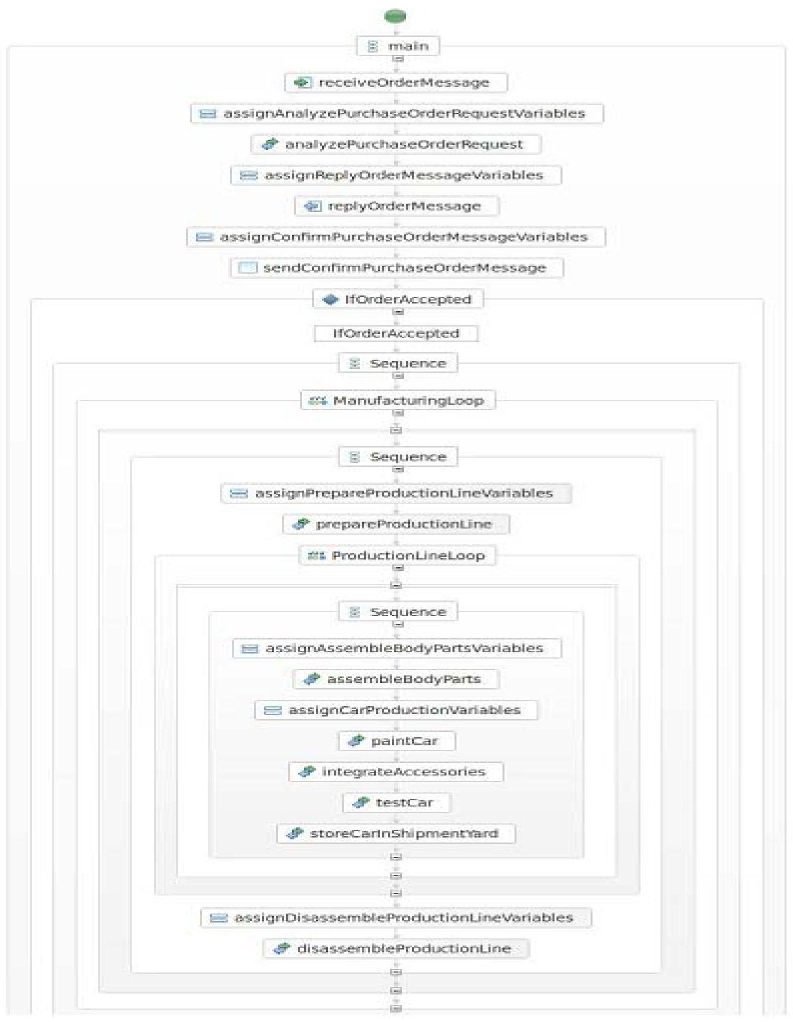
Figure 49: Logistics Service Methods

7.5. Service Orchestrations

The service orchestrations (see section4.3.1. Orchestration) that involve the several message exchanges between the Manufacturing Unit, the Logistics Provider and the Retail Outlet are implemented via the BPEL language. The service orchestrations are implemented as three different BPEL processes – one each for the Manufacturing Unit, the Logistic Provider and the Retail Outlet. The reason for having three separate BPEL process is because in a real world scenario, each business entity is completely unaware about the internal operations of its partner companies. For example in this case, when the Retail Outlet sends a Purchase Order Request to the Manufacturing Unit, the details about the processes used by the Manufacturing Unit to analyze the order remains a ‘black-box’ to the Retail Outlet. All the Retail Outlet knows is whether the Order has been accepted or rejected. As it is meant, the exchange of information through the entities as modeled and developed as Choreography of Web Services (see section 4.3.2. Choreography). In this case, the BPEL processes are developed using the BPEL Plug-in for the Eclipse IDE [54] and they were deployed in Apache ODE [55]. The detail about each BPEL process is explained in the following.

7.5.1. Manufacturing Unit Orchestration

This process begins with the receipt of the Order Information from the Retail Outlet. Once the manufacturing receives the Order from the Retail Outlet, the message is analyzed and a Reply message is sent to the Retail Outlet using the PIP 3A4 standards. If it is determined that the Order is accepted, then the production process begins for the required number of cars. This is represented by the Production Line Loop which runs as many times as the number of cars to be produced, and involves the processes: Assembly of body parts, Painting, Accessories integration, Testing and Storage of the car in the shipment yard. After the manufacture of all the cars is completed, a Shipping Order Request is sent to the Logistics Provider using the PIP 3B12 standards. Once the reply is received from the Logistics Provider confirming the Shipping Order, the cars are prepared for shipment. The entire BPEL process for this can be seen in Figure 50.


main
receiveOrderMessage
assignAnalyzePurchaseOrderRequestVariables
analyzePurchaseOrderRequest
assignReplyOrderMessageVariables
replyOrderMessage
assignConfirmPurchaseOrderMessageVariables
sendConfirmPurchaseOrderMessage
IfOrderAccepted
IfOrderAccepted
Sequence
ManufacturingLoop
Sequence
assignPrepareProductionLineVariables
prepareProductionLine
ProductionLineLoop
Sequence
assignAssembleBodyPartsVariables
assembleBodyParts
assignCarProductionVariables
paintCar
integrateAccessories
testCar
storeCarInShipmentYard
assignDisassembleProductionLineVariables
disassembleProductionLine

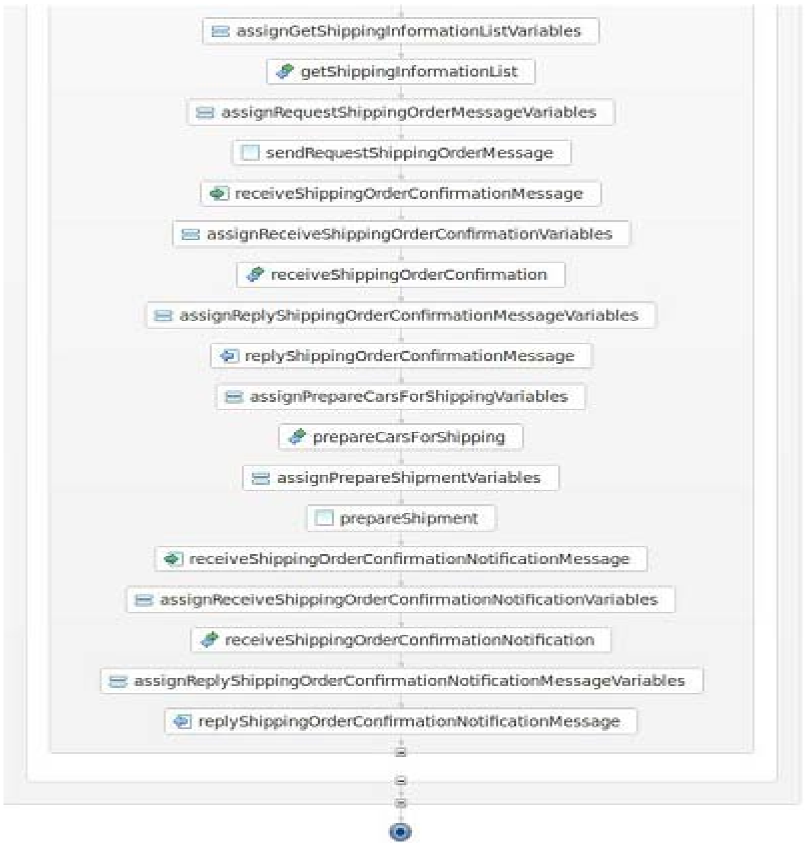

assignGetShippingInformationListVariables
getShippingInformationList
assignRequestShippingOrderMessageVariables
sendRequestShippingOrderMessage
receiveShippingOrderConfirmationMessage
assignReceiveShippingOrderConfirmationVariables
receiveShippingOrderConfirmation
assignReplyShippingOrderConfirmationMessageVariables
replyShippingOrderConfirmationMessage
assignPrepareCarsForShippingVariables
prepareCarsForShipping
assignPrepareShipmentVariables
prepareShipment
receiveShippingOrderConfirmationNotificationMessage
assignReceiveShippingOrderConfirmationNotificationVariables
receiveShippingOrderConfirmationNotification
assignReplyShippingOrderConfirmationNotificationMessageVariables
replyShippingOrderConfirmationNotificationMessage

Figure 50: Manufacturing Unit Orchestration

7.5.2. Logistics Provider Orchestration

The orchestration here starts with the Logistics Provider receiving the Shipping Order Request from the Manufacturing Unit. The Logistics Provider then accepts or rejects the order and then sends a notification to the Manufacturing Unit about the status of the order. If the order is confirmed, it also sends a notification of Shipping Order Confirmation as per the PIP 3B13 standard. Once the order is confirmed, the shipment is collected from the Manufacturing Unit and is to be delivered to the Retail Outlet. After this, a Notify of Billing Statement message is sent to the Retail Outlet according to the PIP 3C5.The entire BPEL process for this can be seen in Figure

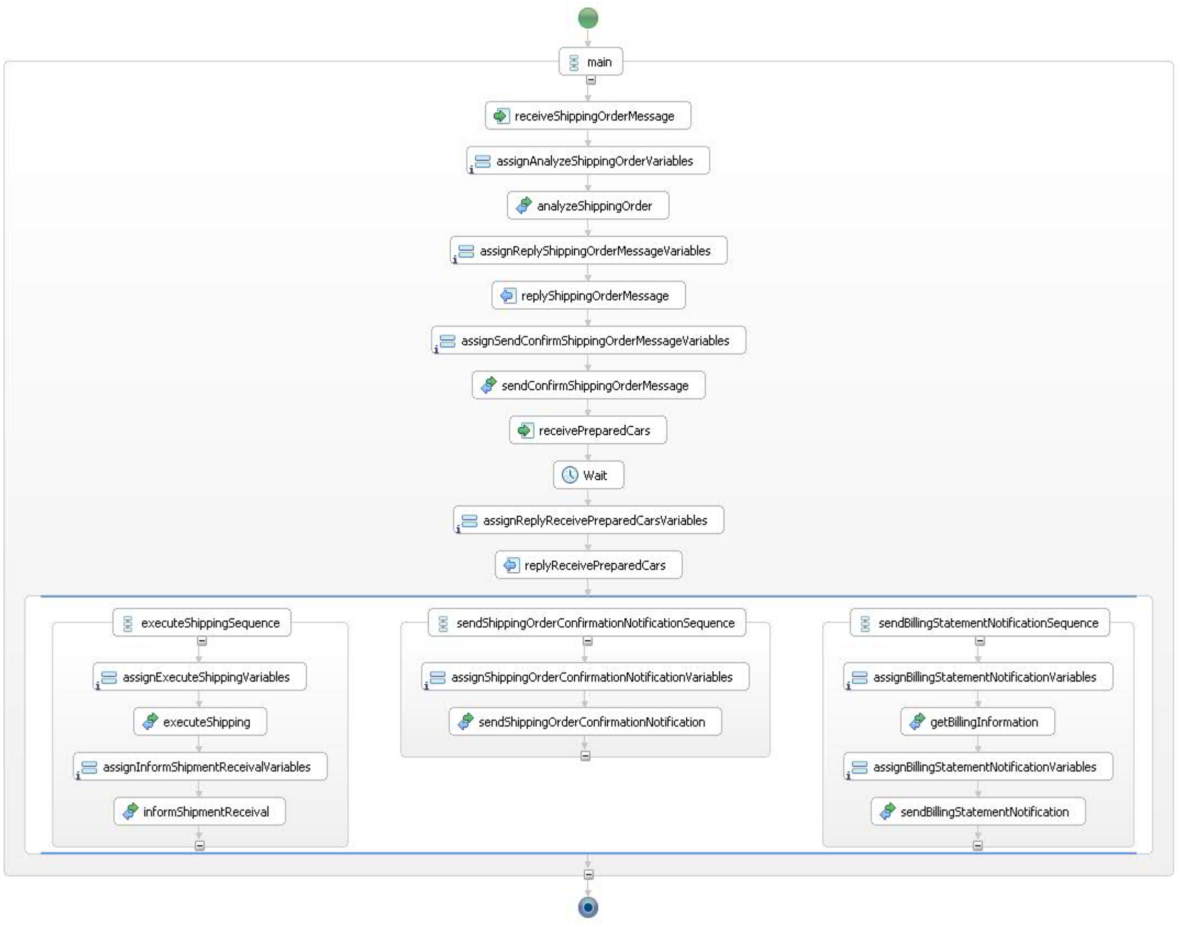

main
receiveShippingOrderMessage
assignAnalyzeShippingOrderVariables
analyzeShippingOrder
assignReplyShippingOrderMessageVariables
replyShippingOrderMessage
assignSendConfirmShippingOrderMessageVariables
sendConfirmShippingOrderMessage
receivePreparedCars
Wait
assignReplyReceivePreparedCarsVariables
replyReceivePreparedCars
executeShippingSequence
assignExecuteShippingVariables
executeShipping
assignInformShipmentReceivalVariables
informShipmentReceival
sendShippingOrderConfirmationNotificationSequence
assignShippingOrderConfirmationNotificationVariables
sendShippingOrderConfirmationNotification
sendBillingStatementNotificationSequence
assignBillingStatementNotificationVariables
getBillingInformation
assignBillingStatementNotificationVariables
sendBillingStatementNotification

Figure 51: Logistics Provider Orchestration

7.5.3. Retail Outlet Orchestration

The Retail Outlet first creates a Purchase Order Request, which is sent to the Manufacturing Unit using the PIP 3A4 message standards. It then receives the confirmation about whether the order has been accepted or rejected. If the order has been accepted, the Retail Outlet then receives the Billing Statement Notification from the Logistics Provider that is to deliver the cars. Finally when the Retail Outlet receives the shipment, the processing of the Billing Statement is carried out. This BPEL process can be seen in Figure.

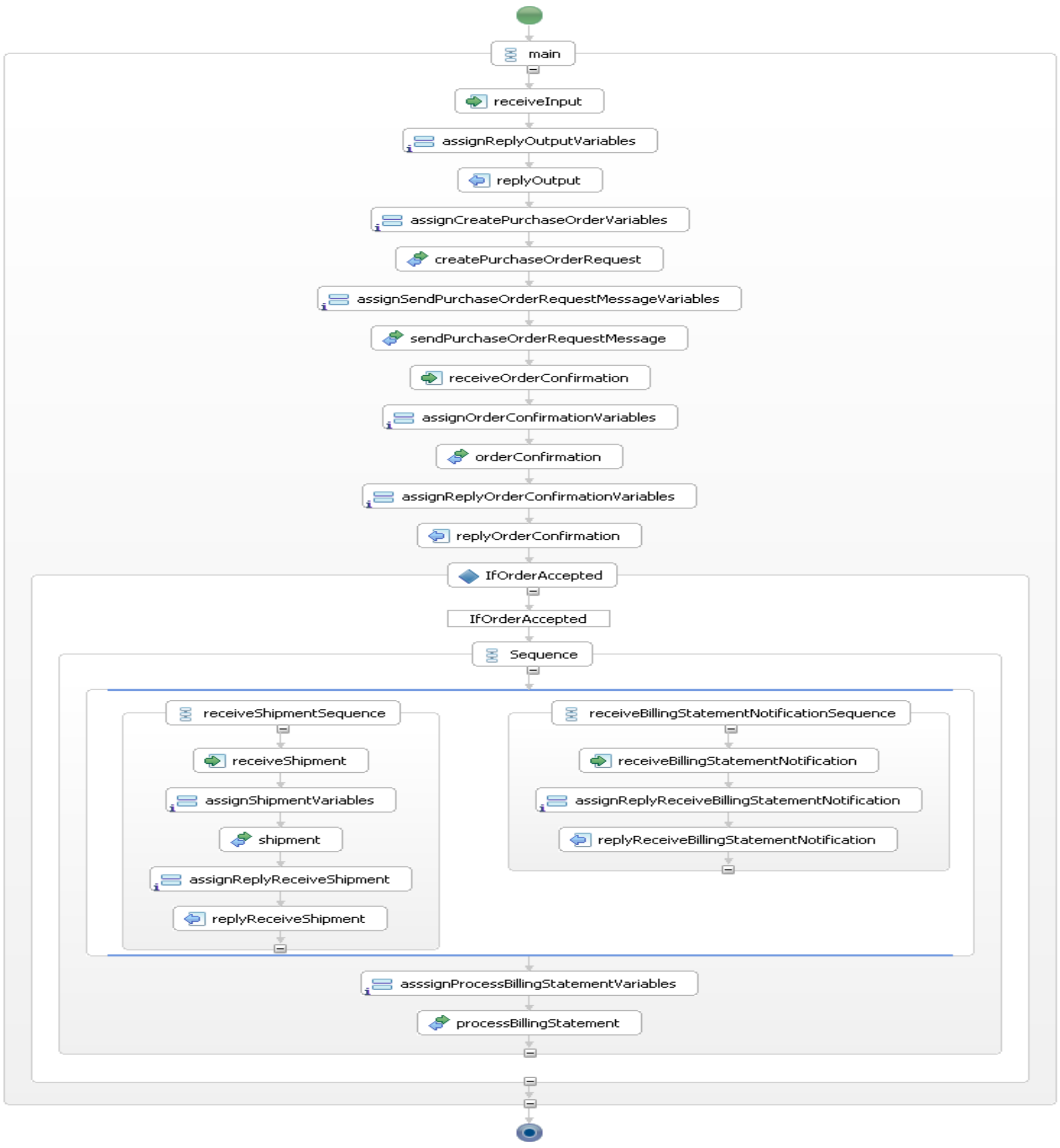

main
receiveInput
assignReplyOutputVariables
replyOutput
assignCreatePurchaseOrderVariables
createPurchaseOrderRequest
assignSendPurchaseOrderRequestMessageVariables
sendPurchaseOrderRequestMessage
receiveOrderConfirmation
assignOrderConfirmationVariables
orderConfirmation
assignReplyOrderConfirmationVariables
replyOrderConfirmation
IfOrderAccepted
IfOrderAccepted
Sequence
receiveShipmentSequence
receiveShipment
assignShipmentVariables
shipment
assignReplyReceiveShipment
replyReceiveShipment
receiveBillingStatementNotificationSequence
receiveBillingStatementNotification
assignReplyReceiveBillingStatementNotification
replyReceiveBillingStatementNotification
asssignProcessBillingStatementVariables
processBillingStatement

Figure 52: Retail Outlet Orchestration

7.6. Summary

This chapter described a simplified real world scenario about an automotive case study. It introduced a way to build transactions between business entities based on web services build on RosettaNet PIPS. An Order Request is implemented and demonstrated through PIP3A4 Request Purchase Order, PIP3B12 Request Shipping Order, PIP3B13 Notify Shipping Order Confirmation, and PIP3C5 Notify

Billing Statement. The whole scenario is built on three business entities, which exchange such messages through the implemented web services. Finally the procedure is executed as Web Services Choreography (exchange messages between three Orchestrations, one of each entity). Next chapter is the Conclusion and the Future Work of this thesis

8. Conclusions and Future Work

The need for connecting information systems of collaborating organizations is becoming increasingly common: advantages such as increased speed, efficiency, and reliability, can be gained by automating inter-organizational business processes. RosettaNet is an industry consortium that maintains the RosettaNet e-business framework, which specifies inter-organizational business processes for multiple industries. These process specifications include messages that are exchanged between organizations, and associated messaging choreography.

This master's thesis report presented an approach towards applying Service Oriented Architecture concepts in the design and implementation of RosettaNet PIPs based on Web services. It also presents RosettaNet PIPs under BPMN principles. The idea behind this work is to connect the world of RosettaNet framework and the world of Web Services. The main target is to provide an architecture and methodology to implement RosettaNet-Driven Business Interactions. For this reason, RosettaNet Standards are analyzed, modeled, and developed through the guidelines provided. The developed RosettaNet PIPs web services are used to simulate a scenario of automobile based on Choreography.

The services that constitute the RosettaNet PIPs have been implemented in the Java and used well-established open source libraries and frameworks like Java Architecture for XML Binding (JAXB) library implementation of derived Java Classes through the XSD/DTD documents provided by RosettaNet and Java API for XML Web Services (JAX-WS) for implementing and deploying the Web services of this work. The Apache ODE engine has been used as the runtime BPEL engine for the orchestrations and choreography of the implemented scenario. Apache Tomcat server is used as the deployment server of the services. Finally, the testing of the functionality of web services and BPEL choreography was done in SOAP-UI [56].

Additionally, this work utilizes standards like BPMN. The RosettaNet PIPS are modeled as business processes. An empirical mapping from PIPs specification to BMPN is provided. The modeling and designing in Eclipse BPMN designer.

However, the connection between RosettaNet PIPs and SOA is still a difficult subject. This work proposes a way to model and implement business interactions through Web Services. There some other works in the same spirit. These works are based on building RosettaNet Solution in SOA principles too. They demonstrate a methodology on how to simulate e-business dialogues based on abstract BPEL processes. They try to suggest mappings between RosettaNet DTDs (XSDs) elements and actions with the WSDL types and operations. They match PIP roles to Partners in WS-BPEL. Exception messages from RosettaNet RNIF are mapped to the exception handling mechanisms of BPEL4WS [57][58] [59].However, the approach of these methodologies is theoretical. There is not any implementation provided in order to make clear conclusions. Furthermore, there is not any analysis about the technologies used in order to implement the RosettaNet solution.

On the other hand, the solution is proposed in this work, it demonstrates a more technical approach to implement such a solution. It provides information about all technologies are used and shown a comprehensive example based on the implementation. It proposes roadmap for a distributed implementation of RosettaNet PIP compositions that does not need a reliable messaging infrastructure. The roadmap cares for the dynamic nature of business collaborations by reusing the implementation of communication protocols. Additionally, this work offers a modeling approach of RosettaNet as it provides an empirical methodology for transforming RosettaNet PIPs business processes into BPMN. The transformation is based on the guidelines of the RosettaNet Standards. To sum up, the solution proposes a more detailed methodology from the other solution. The other solutions stay on theoretical basis, whereas this work demonstrates a more technical approach. In addition, it offers a comprehensive solution which deals both in modeling and implementing RosettaNet Standards through the SOA principles.

Future work and more is needed in the interaction between the organizations as each organization acts in a different way. Organizations should be moved their main operations to the Web to take advantage of the potential of more automation, efficient business processes, and global visibility. Service Oriented Architecture is able to show this way.

Bibliography

1. **Dogac, Asuman.** Special Issue on Electronic Commerce. *ACM SIGMOD* . 1998, Vol. Rec 27(4).
2. **Brodie, M.** The B2B E-commerce revolution: convergence, chaos, and holistic computing. *Brinkkermper S, Lindencrona E, Solvberg A (eds) Information System engineering: State of the art and research themes.* London UK : Springer, 2000.
3. **Tornatzky, L.G. and M, Fleischer.***The process of technological innovation.* Lexingthon, New York : s.n., 1990.
4. **Porter, M.E. and Stern, S.***Innovation: location matters.* s.l. : MIT Sloan Management Review, 2001. pp. 28-36.
5. **Patterson, Kirk A., Grimm, Curtis M. and Corsi, Thomas M.***Adopting new technologies for supply chain management.* s.l. : Elsevier, 2003. pp. 95-121.
6. **W3C.** XML TECHNOLOGY. [Online] http://www.w3.org/standards/xml/.
7. **Rosettanet.***Rosettanet.* [Online] http://www.rosettanet.org/.
8. **W3C.** Document Type Definition. [Online] http://www.w3.org/TR/html4/sgml/dtd.html.
9. **W3C**. XML Schema. [Online] http://www.w3.org/XML/Schema.
10. **Badakhchani, Hussein.** Introduction to RosettaNet. [Online] September 11, 2004. http://www.genesis.uk.com/wordpress/?page_id=970.
11. **RosettaNet.** RosettaNet Implementation Framework: Core Speciification Release. July 13, 2001.
12. **RosettaNet**. RosettaNet - Standards - RosettaNet Standards. [Online] http://www.rosettanet.org/Standards/RosettaNetStandards/tabid/473/Default.aspx.
13. **P., Beynon-Davies.***E-Business.* Basingstoke : Palgrave, 2004. ISBN 1-4039-1348-X.
14. **Kotinurmi, Paavo.***Technical Look at RosettaNet.* Helsinki : SoberIT Helsinki University of Technology, 2002.
15. **Rosettanet.** PIPs. *Rosettanet.* [Online] http://www.rosettanet.org/Standards/RosettaNetStandards/PIPs/tabid/475/Default.aspx.
16. **RosettaNet.** RosettaNet: Overview: Clusters, Segments, and PIPs. [Online] 2010. http://www.rosettanet.org.
17. **Asuman, Dogac.***RosettaNet.* Ankara Turkey : Grenoble Ecole Management MEDFORIST Workshop RosettaNet, Middle East Technical University, 2003.
18. **RosettaNet.** PIP3A1 Request Quote specification V02.03. *RosettaNet.* [Online] 10 20, 2009. http://www.rosettanet.org/dnn_rose/DocumentLibrary/tabid/2979/EntryId/9550/DMXModule/624/Command/Core_Details/Default.aspx.
19. **OMG.** OMG Unified Modeling Language Specification, 1st ed.,. [Online] March 2003. http://www.omg.org/cgi-bin/doc?formal/03-03-01.
20. **UN/CEFACT.** UN/CEFACT's Modelling Methodology N090 Revision 10. [Online] November 2001. http://www.untmg.org.
21. **Christerson, Maguns and Johnston, Simon.***Business Modelling and E-commerce.* s.l. : Rational Software and Miller Freeman Inc, 1999.
22. **RosettaNet.***RosettaNet Business Dictionary (RNBD).* 2002.
23. **RosettaNet**. *RosettaNet Technical Dictionary.* 2004.
24. **D&B.** Frequently Asked Questions for D-U-N-S Number. [Online] D&B. http://fedgov.dnb.com/webform/displayFAQPage.do.
25. **GS1, UCC and EAN Internatinal now.***Global Trade Item Number (GTIN)* . May 2004.
26. **UNSPC.** UNSPSC FAQ's. [Online] http://www.unspsc.org/FAQs.asp#WhatistheUNSPC.
27. **vom Brocke, J.HKVJH and Rosemann, M.***Handbook on Business Process Management: Strategic Alignment, Governance, People and Culture.* Berlin : Springer, 2010. Vol 1.
28. **Kohlbacher, M.** The Effects of Process Orientation on Customer Satisfaction, Product Quality and Time-Based Performance. Paper presented at the 29th International Conference of the Strategic Management Society, Washington DC : s.n., 2009.
29. **Weske, Mathias.***Business Process Management: Concepts, Languages, Architectures.* s.l. : Springer, 2007. 3540735216, 9783540735212.
30. **NIH.** Business Process Management (BPM) Service Pattern. [Online] 2007. [Cited: June 22, 2011.]
31. **(BPMI), Business Process Management Initiative.** Business Process Modeling Notation BPMN. [Online] June 2010. http://www.bpmn.org.

32. **Erl, Thomas.***Service-Oriented Architecture (SOA): Concepts, Technology, and Design.* Indiana : Pearson Education Inc., 2005. ISBN-10 0-13-185858-0.
33. **Endrei, Mark, et al.***Patterns: ServiceOriented Architecture and Web Services* . s.l. : IBM, 2004. SG24-6303-00.
34. **W3C.** Web Services Description Language (WSDL). [Online] 2003. http://www.w3.org/TR/wsdl.
35. **W3C**. Simple Object Access Protocol (SOAP). [Online] 2003. http://www.w3.org/TR/SOAP/.
36. **W3C**. Universal Description, Discovery, and Integration (UDDI). [Online] 2003. http://www.uddi.org.
37. **OASIS.***Organization for the Advancement of Structured Information Standards.* [Online] 1993. http://www.oasis-open.org.
38. **W3C.***The World Wide Web Consortium (W3C).* [Online] 1994. http://www.w3.org.
39. *W3C:Web Service Architecture.* [Online] 2003. http://www.w3.org/TR/ws-arch /.
40. **Papazoglou, M.***Web Services: Principles and Technology.* s.l. : Pearson Prentice Hall, 2007. ISBN 978-0321-15555-9.
41. *Web services Interoperability Organization.* [Online] 2002. http://www.ws-i.org.
42. **Potts, Stephen and Kopack, Mike.***SAMS: Teach Yourself Web Services in 24 hours.* Indianapolis : SAMS, 2003. ISBN 0-672-32515-2.
43. *Web Services Addressing (WS-Addressing).* [Online] 2004. http://www.w3.org/Submission/ws-addressing/.
44. **Barros, A., M., Dumas and P., Oaks.** Standards for Web Service choreography and orchestration: Status and Perspectives. s.l. : In C. Bussler and A. Haller, editors, Business Process Management Workshops, 2005, Vol. 3812.
45. **Peltz, Chris.** Web Services Orchestration and Choreography. *Computer pp.46-52.* October, 2003, Vol. 36, no. 10.
46. **OASIS.***Web Services Business Process Execution Language Version 2.0.* [Online] http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html.
47. **Eclipse Foundation.***BPMN Modeler.* [Online] 2011. http://www.eclipse.org/bpmn/.
48. **Grebac, Martin.***JAXB.* [Online] http://jaxb.java.net/.
49. **java.net.***JAX-WS.* [Online] http://jax-ws.java.net/.
50. **Apache Software Foundation.***Apache Tomcat.* [Online] http://tomcat.apache.org.
51. **S-Cube.** Report on Common Pilot Cases. *S-cube.* [Online] http://www.s-cube-network.eu/.
52. **Petridis, P.***Towards a universal Service Network-centric framework to design, implement and monitor Services in complex Service Ecosystems: The Service Network Analysis Prediction Tool(SNAPT).* s.l. : Computer Science Department, University of Crete, 2010.
53. **Stratakis, Giorgos.***Analyzing Service Networks from different perspectives using the service network analysis & prediction tool(SNAPT).* s.l. : Computer Science Department, University of Crete, 2010.
54. **Eclipse Foundation.***BPEL Designer Project.* [Online] 2011. http://www.eclipse.org/bpel/.
55. **Apache Software Foundation.***Apache ODE.* [Online] http://ode.apache.org/.
56. **SmartBear Software.***SOAP-UI.* [Online] http://www.soapui.org/.
57. **Schönberger, Andreas and Wirtz, Guido.***Realising RosettaNet PIP Compositions as Web Service - A Case Study.* In Proceedings of CSREA EEE : s.n., 2006. pp. 141-147.
58. **Khalaf, Rania.***From RosettaNet PIPs to BPEL processes: A three level approach for business protocols.* 2007. pp. Pages 23-38. ISSN 0169-023X.
59. **IBM.***An Approach to Moving Industry Business Messaging Standards to Web Services.* 2004.