Thesis Submitted for the Degree of
Hovedfag/Master

A Model Driven Approach to domain standard specifications examplified by Finance Accounts receivable/ Accounts payable

By

BAHADAR KHAN

Department of Informatics

The Faculty of
Mathematics and Natural Sciences

University of Oslo
Oslo 2005

Preface

This thesis was written as a part of a master degree at the University of Oslo. The thesis work was conducted at SINTEF. The work has been carried out in the period November 2002 and April 2005.

This thesis might be interesting to anyone interested in Domain Standard Specification Language developed by using the MDA approach to software development. The Model Driven Architecture (MDA) allows to separate the system functionality specification from its implementation on any specific technology platform. I have suggested UML-SSS (it stands for UML for Standards Service Specifications) in thesis and applied it to Finance domain with example Account Receivable and Account Payable.

I would like to thank all who helped me especially, my supervisor professor Dr. Arne Jørgen Berre at SINTEF for encouraging me in this thesis, reading the text, giving advice, and providing timely and valuable feedback.

I am thankful to friends and family who have supported me and shown interest in my work: thank you for the motivation you have provided me! And special thanks to my wife Asima Tehseen who encouraged me in the work to complete this thesis.

Oslo, May 2005
Bahadar Khan

Contents

List of Figures

List of Tables

1 Introduction

In this chapter I will describe the developments of IT technology especially related to raise abstraction level and need of software based on accounting standards. The thesis has two focus areas.

A. **Domain standard Specifications Language**
B. **Finance AR/AP standard specification**

1.1 Domain Standard Specification Language

Domain standards in the IT industry, for technology areas such as communication and storage, and for domains such as Finance, Healthcare, Geographic information has so far typically been specified in concrete technologies such as EDIFACT representation, CORBA interfaces or Web service interfaces (WSDL) or SQL table structure. However, it seems that such implementation technologies are more varied, and are changing faster than the domain semantics.

It takes a lot of time to change one technology to another for a specific domain. This problem is crucial because all new technologies tend to be discarded and forgotten due to the continuous development of computer and information technologies. For example COBOL programs that were written in the 80's for the banking sector and some of them are still in use today, are now it is desirable to convert to some modern technology but it is a very expensive task. If we can find documentation of these programs with platform independent models it may be easy to convert these programs to new technologies.

We can see that without some smart techniques and methodology we should not be able to take advantage of all opportunities that the technological progress can offer us. It is not the first (and certainly not the last) time we meet this problem and it has been always solved by using the same approach: by raising the abstraction level in software development.

At the early age of computer industry the programs were written in machine codes. People soon realized that this was a very inefficient method of programming. Writing programs with '0's and '1's needed enormous efforts from programmers and development of large systems was impossible. Assembly languages added a level of abstraction to programming by introducing human readable commands. Later on 3GL languages added another level of abstraction. The commands became even closer to human languages by encapsulating processing logic. The major benefit of 3GL was portability. Another step in the

raising of abstraction in software development is the appearance of middleware and virtual machines. Middleware offers common services which are independent of operating systems and platforms.

We can see that the complexity of software increases as the technology evolves and to deal with this complexity new abstraction levels must be added to the development process. The development of Unified Modelling Language (UML) in recent years gives us an alternative to raise the abstraction level of software systems. It is used in many phases of software development process like: business modelling, requirements modelling, architecture modelling, database design modelling and many more. It is popular to build models because they represent a simplification of reality and help us to analyze complex systems that we cannot comprehend in its entirety.

Only models are not enough, we need models with some modelling technique like MDA, PIM and PSM for standards specifications of software systems to overcome the complexity of software system. There are many international organizations which are doing very good job of specifying standards for many software systems and part of systems in the world, but standards developed by these in the field of modeling, data exchange, messaging, services and processes are of special interest for financial sector. Some of leading name in this regard are Object Management Group (OMG), OASi, OASIS, UN/CEFACT and XBRL.

1.2 Domain Standard Example – Finance AR/AP

Accounting concepts have been stable for over 500 years but the majority of GL systems are still non-standard and difficult to make interoperable. Generally it is due to the lack of domain standards for accounting software. Accounting and financial sector was one of the first sectors which started using the first computer in 1943 and computer accounting became globally introduced in early fifties when computer revolution became global. Use of computers became *de facto* industry standard soon after the computer revolution and computer software systems brought many changes in financial markets globally.

Changes in financial markets happen very often and rapidly. Innovation of computer and information technology has many folded changes in management of financial accounting. These changes make new opportunities and challenges for both the financial sector and for software developers. Most of the financial institutions and companies demand new and better IT software systems, to compete with market rivals in these new and demanding situations. These institutions are turning to new technologies as a way to drive globalization, deregulate product offerings, integrate legacy systems and improve the overall quality of service. The ability to write such highly functional programs for the financial sector is not an easy task, and the price we must pay for that ability is the increasing complexity of the software development process. As the complexity of the system gets greater and greater, the task of building the software gets harder and harder. It is not unusual nowadays that the source code for some programs is

many hundred thousands of lines. It is obvious that such code is very difficult to comprehend and maintain. Because of this growing complexity, the financial industry needs software systems based on international standards that are sufficiently adaptable, interoperable, that can be reused later in development of other software, that can handle the massive amounts of information generated, and can deliver it to where it is needed when it is needed.

1.2.1 Standard Specification

We can not get involved with financial systems without hearing about standards, but what do we mean by standards?, what do they comprise?, who makes them?, and how do we choose which one to use? It is difficult to answer all these questions in this thesis, but I shall take a brief look at these questions.

Standards set out what are widely accepted as good principles, practices, or guidelines in a given area. The development and implementation of internationally accepted IT, economic, and financial standards can help to promote sound domestic financial systems and stable international financial systems. The development, adoption, and successful implementation of international standards yield both national and international benefits. But on the other hand standards alone are not an end in themselves but a means for promoting sound financial systems.

Standards are formed to serve the business community and not vice-versa. Often we need to implement two standards, one for domestic community, and other for international community. The important thing to know about IT standards is that there is more than one or, more accurately, more than one syntax upon which the messages are built. The following concepts have significant importance when we talk about domain standards specification i.e. data exchange, messaging, services and processes.

Standard specification of software systems is often region characterized i.e. one for EU, other for USA and Asia. The leading organizations, which are working for standards in the area of IT and financial systems are UN/CEFACT, ISO, ANSI, UNTDI, OASIS, OMG and many others.

It is difficult to compare these standard specifications or systems developed by different standardizing bodies because most of them are developed for special purposes with special demands and needs. But I look into some common properties of system models like visualization of a system, specifying the structure or behaviour of a system and the template that guides us in constructing a system and helps us to realize a standards specification

2 Background and Problem Statement

This chapter includes the background and problem statement for both focus areas of the thesis. First section includes the background and developments in modeling techniques. The second section includes issues and background of Accounts Receivable and Accounts Payable along with problem statement for AR/AP. At the end of chapter, a general documentation problem faced by software developer is discussed.

2.1 Domain Standard Specification Language

In system development the main goal of the activity is production of a running system. The most important assets for the running system are the developed or generated code that is compiled and executed. Technology-specific standards will have trouble, getting established, where platform incompatibility prevents achieving this critical mass. Sometimes the problem is even deeper than this. In some industries, architecturally excellent standards have been adopted in the formal sense but failed to gain hold because they were written for a platform that few companies were willing to support.

Traditionally, software development has been a series of mappings from the domain idea, to design models, and on to source code. These mappings tend to be slow and lead to errors and duplication of effort in problem solving, designing and coding. Model-based development of software systems is not new it has been used around since the 70's, but we have not had tools that would help to address the requirements of model based development

In a model based development a system is described from different aspects using different models. A model can be abstract or detailed it depends on the modeling language. OMG (Object Management Group) states some requirements for a model:

> "*A model is a representation of part of the function, structure and /or behavior of a system." [5]*
>
> Or
>
> "*A model has to be formal, which means that it has to have a well-defined syntax and semantics. The syntax may be graphical or textual."[5]*

Modeling is useful for a wide spectrum of domains and activities, some of which are unrelated to coding. At the highest level of abstraction, a business or domain model focuses not on software, but instead on the nature of the problem under consideration and at the next stage architecture modeling will focus on internal

structure of software. The success of software development lies in the domain standard specification language and methodology which ties these models together. A model-driven form of software development is becoming more and more effective for the advanced and progressive software.

2.1.1 Model Driven Architecture (MDA)

MDA is an effort of OMG to raise the level of abstraction for development of software. As mentioned in above section software developer have gradually lifted the abstraction level from 1s and 0s, to assembly language, to third generation languages, and now to yet more abstract modeling languages. Model Driven Architecture is a comprehensive approach to information systems engineering that systematically addresses the complete life cycle of automating business processes through software. MDA focuses on formalizing and standardizing the artifacts associated with designing, deploying, integrating, and evolving supporting software applications and therefore a good technique for developing domain standards.

The MDA is based on the idea of meta-modeling. It merges the different OMG standards having been developed and used separately so far into a common view by applying common meta models to them. However, it is not necessary to step in too deep into the meta worlds of modeling to understand the underlying concepts. The core of Model Driven Architecture is based on OMG's modeling standards:

- the Unified Modeling Language (UML),
- the Meta-Object Facility (MOF),
- and the Common Warehouse Meta-model (CWM).

The MDA defines an approach to system specifications that separates the specification of the system functionality from the specification of the platform specific implementation. This is done by specifying standards to model the system in a reusable way. This allows two main applications:

- A system can be defined platform independently and then can be realized on multiple platforms through auxiliary mapping standards.
- Different applications can be integrated by explicitly relating their models, even if they do not run on the same platform type.

Key to MDA is the importance of models in the software development process. Within MDA the software development process is driven by the activity of modeling our software system. MDA defines PIM, PSM, and code and how these are related to each.

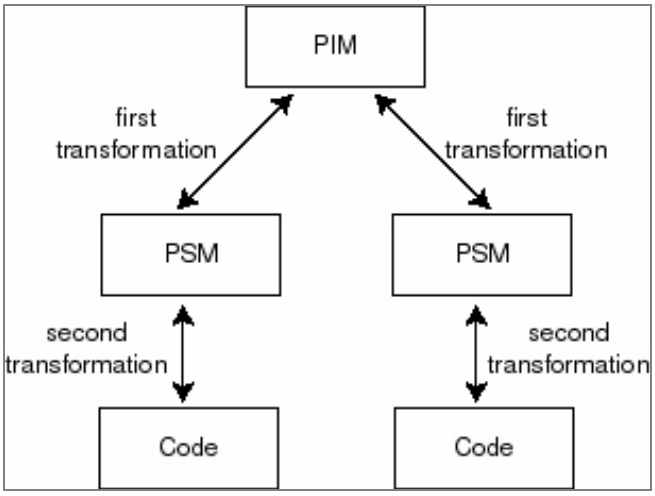

Figure 2.1 Model to model transformation in MDA

We build a PIM model in MDA with a high level of abstraction that is independent of any implementation technology; this is called a Platform Independent Model (PIM).

Next, the PIM is transformed into one or more Platform Specific Models (PSM). A PSM is tailored to specify our system in terms of the implementation constructs that are available in one specific implementation technology, for example a database model, an EJB model. A PIM is transformed automated into one or more PSM.

The final step is to transform a PSM to code. Because a PSM fits its technology very closely, this transformation is rather trivial. The complex step is the one in which a PIM is transformed to a PSM. Recent development in OMG has also added the CIM level, Computational Independent Model, - which describes the context for a system.

The above discussion means that MDA has shifted the focus of software developers from PSM and code to PIM. The PSM, that are needed are generated by transformation from PIM to PSM. We will work independently of the details and specifications of the target platforms, there is a lot of technical detail that we don't need to bother with. These technical details will be automatically added by the PIM to PSM transformation. This improves the productivity and will help to make domain standards for a long time.

2.2 Problem Statement for AR/AP

All businesses have external balances. These balances include accounts receivable from customers, accounts payable to suppliers and various financial liabilities and assets such as bank accounts and borrowings. The human resources spent on administering, communication, billing, reconciliation, and settlement of interparty balances in western countries is certainly above 10 million person years per year.

Commercial banking itself consists largely of mechanisms for correct and secure interparty balances. The lack of standardization in managing interparty balances also imposes logistical costs such as printing, postage, and driving to banks. It has been made good efforts to specify standards for e-commerce and eProcurement in recent years but mostly these specifications cover order to invoice process and do not specify standards for Account Receivable and Account Payable which always follow the invoice.

2.2.1 E-commerce

Internet has opened the door for almost all companies and many of them are doing their business through e-commerce. Purchase orders and invoices are exchanged electronically. AR/AP is closely related to purchase order and invoice because input like party name, amount, due date etc. come from invoice. Many companies experience a number of problems like data exchange, messaging and processing when they come to an electronic peer-to-peer collaboration. Therefore we need international standards to overcome these problems.

Data exchange is becoming a necessity in ERP because any data which is output of one process could be input for another process. Problems that occur at data exchange between independent systems are that when one company sends an invoice to the buyer, it is possible that different terms are being used in the models.

For example the supplier company uses the term itemName and the buyer uses the term productName which refer to the same semantic meaning. The problem with use of synonym terms occurs when integration of data is required. Similarly other problems like semantic incompatibility i.e. the same term may be chosen by two systems to denote completely different concept, data representation conflicts and attribute integrity constraint conflicts can also occur during integration of data. It means that two parties' assessment of both character and timing of recognition into payables, receivables, inventory, or other accounts will sometimes be inconsistent. In other words, the accounting processes will not be completely automated and they will be costly.

2.2.2 Account receivable and Account Payable

Accounts receivable is unpaid customer invoices, and any other money owed to seller by his customers. The sum of all customer accounts receivable is listed as a current asset on balance sheet. Seller should keep accounts receivable ledger for each customer. The accounts receivable ledger is a record of each customer's charges and payments.

When a customer purchases something, supplier will first record the sale in the sales and cash receipts journal. This journal will have accounts receivable debit and credit columns. Charge sales and payments on account are entered in these two columns, respectively. Then, each day, the credit sales recorded in the sales

and cash receipts journal is posted to the appropriate customer's accounts in the accounts receivable ledger. This allows us to know not only the total amount owed to us by all credit customers, but also the total amount owed by each customer.

Entries made in the sales and cash receipts journal are also totaled at the end of the month, and the results are posted to the accounts receivable account in General Ledger. This account is often called accounts receivable "control account." It means that after all posting is completed the total amount of customer balances in the accounts receivable ledger will be the same as the balance in the control account in the general ledger. If they aren't the same, we have made an error somewhere along the line.

For most businesses, statements should be sent once a month to all customers with an account balance. The statement should show the following:

- a beginning balance (the previous month's ending balance)
- all invoices charged during the month
- payments on account during the month
- any debit memos or credit memos
- an ending balance
- a due date

Which mean that invoice, payment, due date and balance or amount are key words in account receivable ledger.

Accounts payable is the unpaid bills of the business; the money buyer owe to his suppliers and other creditors. The sum of the amounts he owes to his suppliers is listed as a current liability on your balance sheet. Buyer should keep accounts payable ledger account for each supplier. Expenses from the cash disbursement journal are, at the end of each day, posted to the appropriate accounts payable ledger. The accounts payable ledger is a record of what he owes each vendor.

Accounts payable ledger helps buyer to control his expenditures and payables. If he maintains accurate payable ledgers, it will be easy for him to double check the bills you get from his suppliers. At the end of the month, reconcile accounts payable ledgers with the accounts payable control account. The control account is the total accounts payable balance from your general ledger. The beginning accounts payable total, plus purchases on account during the month, minus payments on account during the month, should equal the ending accounts payable total. Compare this amount to the sum of the individual accounts payable ledgers. This will help to discover any errors made in recording payables. Reconciliation might also help to catch any errors on vendor bills.

AR and AP systems, historically, have interoperated very closely with software applications involved in selling, purchasing, cash management, and inventory. AR/AP has two levels of integration i.e. External integration and Internal integration.

2.2.3 External integration

External integration must address the expectations for AR/AP in an Internet environment, in which the transaction creation, management and settlement cycle is increasingly automated. External balance in AR/AP ledger has some common properties. These properties are universal and inherent. The universal attributes of an external transaction entry in the subject's books mostly include identity of the *party* (e.g. customer or supplier), *amount of money*, *date and time* the transaction was concluded or executed, description of what was exchanged (e.g. string, document, document reference or XML message.), *due date* (expectations regarding date of settlement), and settlement method (expectation regarding bank, settlement agent or method)

2.2.4 Internal integration

Along with the external integration AR/AP must have internal integration within the software environment of the enterprise in which the efficiency of applications for selling, purchasing and other operations are not compromised by the fact that receivables collection, payables settlement and other balance sheet operations are performed centrally within a single AR/AP system. In a successful integration, users of these applications have complete and timely views of the state of payables and receivables settlement which are essential to operation. Conversely, the AR/AP system has complete and timely knowledge of the payables, receivables and other balance sheet actions executed by users on various operating applications.

The internal integration of an AR/AP with the other information or accounting systems is therefore be a large and difficult task, especially in the absence of standard specification of data exchange, messages and interfaces. The lack of standardization in managing external party balances imposes costs beyond software or IT costs, to include rigidities in people's activities and roles, rigidities in organizational structure, inability to take advantage of new vertical and horizontal business solutions, and loss of access to markets both in sales and sourcing especially when business is being done through E-commerce.

2.2.5 Resolution of business differences

It is also relevant that at the moment of consummating a transaction, the amounts and consideration are sometimes ambiguous. It is inherent in the operation of many markets that these invalid or open contracts are created and ultimately must

be adjusted or canceled after the fact, within AR/AP systems. We need least-common-denominator interfaces or usage models which facilitate the finding, correction and resolution of business differences between parties to transactions.

2.2.6 Levels of aggregation

Different parties have different levels of aggregation. For example, some parties have historically maintained AR/AP records as Customer or Supplier accounts containing only Statement totals, or containing only Invoice totals, while maintaining large numbers of line items or details in sub-systems not accessible to the AR/AP system. As a result, automation of reconciliation with these companies at the detail level is a problem. Numerous side effects arise in these situations such as credit/debit memos at inappropriate levels of aggregation.

2.3 The Maintenance and Documentation Problem

Another general problem with large software programs is maintenance and documentation. Software developers mostly concentrate on code writing and feel their main task is code writing. Document writing during development process costs time and slow down the development process. The availability of documentation supports the task of those that come later. So, developer feels like doing something for the sake of prosperity, not for your own sake. It is more difficult to keep up to date documentation. Developers make changes in source code under and after the development process. It becomes very difficult for new comer to maintain such systems.

The main task of developer is to develop a system that can be changed and maintained afterwards. Despite the feelings of many developers, writing documentation is one of their essential tasks. Some programming languages support to produce documentation from code i.e. Java and Eiffel, but it is low-level documentation, the higher level documentation still needs to be maintaining by hand. It is difficult to make standards when developers don’t give priority to documentation and only rely on source code. The documentation at higher level of abstraction is an absolute must due to the given complexity of the systems that are built.

2.4 Hypothesis

"It is possible to create a UML profile, suitable for the specification of domain standard services - that may be implemented on various platforms." i.e. a UML Standard Service Specification profile.

3 Requirements

The section 3.1 presents the requirements for domain standard specification language. The solution for these requirements for domain standard specifications is presented in chapter 5. The requirements for finance AR/AP are given in section 3.2. The solution for these requirements is presented in chapter 6.

3.1 Requirements for Domain Standards Specification Language

Table 3.1 shows the requirements for domain standard specifications and further description of important requirements is given below. Requirements have also been adapted from the ISO TC211 19103 [18] and CEN/TC287 requirements to a Conceptual Schema language.

Requirement	Description
R1	Formal Language
R2	Specification of CIM and PIM and PSM models
R2a	Structure
R2b	Behaviour (Process/Services)
R2c	Constraints
R3	Ease of use – graphical notation.
R4	Platform independent data types.
R5	Computer processability
R6	Non functional aspects like performance, QoS, error handling, security issues, usability, reliability, availability, adaptability, supportability.
R7	IT standard and continuous development.
R8	Support from existing working tools.
R9	Methodology and process description.

Table 3.1 Requirements for domain standard specification language

3.1.1 R 1 - Formal Language

The modeling and description language shall be formal and applicable in describing data, services and process – and constraints at the PIM level. The language shall be independent of implementation and PSM level. (i.e. support for MDA)

Ideally, the language should adhere to some international standard like ISO 100% principle for conceptual modelling, and be able to describe all necessary static and dynamic concepts (structure and behaviour) and related constraints. Formal - means well-defined semantics, and both a lexical and a graphical computer-processible syntax would be necessary.

3.1.2 R2 - CIM, PIM and PSM

The modeling language shall support the MDA approach, and should be able to express the three types of models, Computation Independent Model, Platform Independent Model and Platform Specific Model.

3.1.3 R2a - Structure

The requirements for structural descriptions include requirements for description of structure in terms of entities/features/objects and their properties and relationships/associations. It is a need to be able to describe aggregation and ordering and to be able to specialize/generalize entities/features/objects.

3.1.4 R2b - Behaviour

The requirement for description of behaviour comes in particular for operations/behaviour of entity types, multiple inheritance. In addition it has been said that a description of semantics preferably should be done through algebraic (mathematical) specifications . It also required support for the specification of interfaces for the description of services. In addition it has been required that the description of semantics of behaviour should be supported by some kind of assertions, pre/post-conditions or predicate calculus.

3.1.5 R 2c - Constraints

Both structural and behavioural constraints should be explicitly expressible.

3.1.6 R 3 - Ease of use - graphical notation

The description language shall either include or be easily linked to a graphic notation. The graphic notation can be a subset of the lexical description language.

Ease of use also means that descriptions in the language should be easy to understand, formulate and change.

3.1.7 R4. Data Types

Data types should be independent of any target platform. It needs to define a set of platform independent data types for PIM modeling. Data types can categorized into two main kinds, basic types and composite types.

- **Basic types**
  The requirement for basic types is that it must be small set that represent the basic needs for identifying types for model properties.

- **Composite types or derived types**
  Composite types consist of one or more basic types.

3.1.8 R5- Computer processability

The description language shall be interpretable and processable by computers. Consequently, checking and consistency of data descriptions, i.e. conceptual schemas, expressed by the language can be made by software. The processing possibility could also be a basis for executable specifications.

3.1.9 R6. Non functional aspects

This requirement is self explained but security issues are precise below.
Security Issues
For most applications, the use of the network depends on the assured security level and functions. These functions should be as transparent as possible to the user and involve a minimum of effort, and at the same time, should provide an agreed level of security. Security aspects that need consideration for individuals are:

- **Confidentiality:** The user must be assured that the services provided will not expose the data kept or transported to any party, who is not authorised to see it.
- **Availability**: Constant availability of the services may be crucial to the end user. For this reason, the operator must guarantee the agreed availability, and take all necessary steps to maintain it.
- **Consistency, integrity**: The network provider must guarantee that the data kept or transported is not changed in any way, in order to preserve the integrity of the information content.
- **Authentication, access control**: When exchanging information between end users and systems it may be necessary to supplement the data exchange with a procedure to verify the identity of the user and/or the system, and to allow/deny access. This involves an authentication procedure that can take place at two different levels:

• at network level (address exclusion range, closed user groups mechanisms);
• at application or operating system level (access by user identification, accompanied by some token and/or certificate).

- **Non-repudiation**: For some types of information exchange, it may be of significance (formally, legally, or commercially) that neither the sender nor the receiver can repudiate the fact that the information was sent and received.

3.1.10 R7- IT "Standard" and continuous development

The description language should be an official, preferably international accepted standard. It is NOT a requirement that the language is a current international standard, however, it should be ongoing and continuous development around the language, - and this is best ensured if there is some strategy for standardization around the language.

3.1.11 R8 - Supporting software

Software products shall be available to support the usage of the description language in respect of the requirements listed above. Various kind of tools-support is useful, such as syntax/semantic checkers, various translators, etc.

3.1.12 R9 - Methodology and process description

The methodology should be simple and practicable. A methodology should define the process that we use to gather requirements, analyze them, and design an application that meets them in every way.

3.2 Requirements for AR/AP service – A finance example

I have developed the following requirements specification for an AR/AP example, in the context of work done in the Object Management Group and problems described in section 2.2. Proposals are solicited for the definition of interfaces for a universal, AR/AP ledger which meets two top-level, conceptual requirements of external interfaces and internal interfaces. I will identify the external interfaces, relationships and semantics that are required for accounting and business application interoperability with AR/AP systems. Some of these requirements have also been adopted from Revised Submission in response to OMG's Finance DTF RFP for an AR/AP Facility [1].

The key concepts of an AR/AP Ledger are defined as follows:

- AR/AP Ledger – A superset of the General Ledger including all of its interfaces, but having further extensions necessary to provide a

permanent repository of transactions executed with respect to external parties, and to achieve the other goals of General Ledger.

- Transaction – a balanced set of two or more entries (**debits** and **credits**) to a general ledger or AR/AP ledger.
- AR/AP entry – a discrete amount, together with its associated reciprocal party identifier, transaction date, description, expected settlement date and method, and account code.
- Posting – The act of committing an individual transaction consisting of a balanced set of two or more entries (debits and credits) to a general ledger or AR/AP ledger.
- Account – An attribute of a transaction entry (row), which classifies that entry with any valid value in the Chart of Accounts list. The values in the chart of accounts may be statutory classifications for tax or financial reporting, but are usually short or mnemonic values which support additional purposes in workflow, transaction validation, reporting, etc.

It define how other applications like General Ledger, Purchasing, Invoicing, and other similar applications could interface and interoperate with the AR/AP. Briefly it should support:

- The interfaces required to support interoperability of AR/AP applications with independently developed GL, sales/purchasing, and AR/AP systems.
- How to create, read, update and delete transactions and entries in the AR/AP ledger.

3.2.1 Requirements table for AR/AP

The requirements for AR/AP are shown in table given below and described in next subsections.

Requirement	UML-SSS example
Req.1	MDA specification
Req.2	Interfaces and behavior
Req.3	Views of balances
Req.4	Classic Double Entry Accounting (CDEA)
Req.5	Party roles
Req.6	Group-by Queries
Req.7	Non functional aspect like security
Req.8	Relationship to existing OMG specification

Table 3.2 Requirements for AR/AP

3.2.2 Req.1 MDA specification

It shall provide UML models along with MDA technique and describing CIM, a platform-independent UML model of AR/AP (PIM), a platform-specific model

(PSM) based on the UML profile EJB, and platform-specific models (PSM) for other technologies.

3.2.3 Req.2 – Interfaces and behavior

Example shall provide a sufficient level of description of interfaces and behaviors to allow for independently developed accounting applications (including legacy) to interoperate using submitted AR/AP interfaces.

3.2.4 Req.3 – Views of balances

Example shall provide views of the balances and details of AR/AP transactions as they existed at any specific point in time.

3.2.5 Req.4 - Classic Double Entry Accounting (CDEA)

We shall incorporate classic double entry accounting (CDEA) as the basic semantics of representing transactions. CDEA is the system of recording transactions in two or more offsetting debits and credits, which add up to zero, with each row having date/time and account classifications necessary for statutory GAAP and tax reporting (generally accepted accounting principles).

3.2.6 Req.5 – Party roles

AR/AP shall support party roles, identifiers or structures which unambiguously support the distinction between AR and AP items for the same party not having right of offset (netting), but which are not bound to particular roles (or names of roles) such as Customer or Supplier.

3.2.7 Req.6 – Group-by Queries

AR/AP shall support interfaces that enable roll-up. For purposes of this requirement, roll-up is defined as the summarizing of multiple rows of AR/AP into aggregates along at least two dimensions (i.e. group-by queries). These dimensions will include summaries by party ranges (customer or supplier), by date ranges, and by party ranges by date ranges as a minimum.

3.2.8 Req.7 – Non functional aspect like security

Security aspects that need consideration for are confidentiality, availability, consistency, and access control.

3.2.9 Req.8 - Relationship to Existing OMG Specifications

This is optional requirement. We may reuse or depend upon the following existing OMG technologies or we shall discuss relationships to these OMG specifications in our specification.

- General Ledger Facility
- AR/AP
- Currency Facility
- Event Service
- Transaction Service
- Party Management Facility

4 Evaluation of existing Standard specification languages

In this chapter I will evaluate some existing standard specification languages. The existing candidates chosen are described first, and at the end of section they are evaluated for the requirements described in chapter 3.

There are many international institutions and groups which are working on standard specifications of different software systems. They are using different languages, techniques and methods. I will look into some existing approaches used by different internal organizations or groups for domain standard specifications which are directly or indirectly relevant to my work. And I will analyse these existing specification in the light of requirements mentioned in section 3.1. The choice of existing candidates for standard specification is made on the basis of notations e.g. language and methodology they use in standard specification.

Name	Notation	Method
OMG	IDL	Adhoc
OASIS	XML	BCM
UN/CEFACT	XML	UMM
XBRL	XML	Adhoc

Table 4.1Existing standard specification candidates

Table 4.1 shows the existing candidates for evaluation with language for notation and methodology. The existing candidates can be divided into two groups according to notations or languages i.e. one group who uses IDL, and other who uses XML. OMG uses the IDL for notations and have not any specific methodology in this case but it is adhoc. Second group consists of OASIS, UN/CEFACT and XBRL which use XML as notation language and have different methodologies for specification of domain standard. I decided to go into details of two finance related standards and one methodology or process related standard specification chosen from above mentioned candidates i.e.

- CORBA IDL (General Ledger specification)
- XBRL XML (W3C XML) (General Ledger Schema)
- UN/CEFACT UMM

4.1 CORBA IDL (General Ledger Facilities and AR/AP)

CORBA IDL is the product of Object Management Group (OMG). OMG has established many widely used standards like IDL, UML and MOF. They have used these languages to standardize a number of software like General Ledger (GL), AR/AP and many others. I have chosen GL example to evaluate IDL.

General Ledger
The incomes and expenditure of a company or person who is doing business will be registered in a book/system called General Ledger. General Ledger is a collection of accounts and their associated postings. GL interoperates with many other systems like salary, Accounts Receivable and Accounts Payable, Inventory, Sales and Purchase Order Processing systems, therefore the developer have to care about theses integration. Management of a general ledger is a fundamental responsibility of all individuals and companies through accounting systems. The general ledger facility of OMG is open source specifications to manage the general ledger interoperate problems.

The OMG General Ledger Facility is platform specific based on CORBA IDL. The main idea behind the development of GL was the platform independent ability of middleware CORBA. It defines the interface, and their semantics, that are required to enable interoperability between General Ledger systems and accounting applications, as well as other distributed objects and applications for accounting purposes. It uses Core Components and Business Information Entities (BIEs), from ebXML for data types and has some UML models and XML schemas. But it does not fulfill the requirements of MDA CIM, PIM and PSM. It is a PSM based on CORBA/IDL.

OMG AR/AP Facility

AR/AP is another example from OMG where IDL is used to specify standards. The first submission was platform specific based on IDL. Revised submission of OMG's AR/AP facility is PIM based but it is again based on first Platform Specific Model submission. Data types are used from GL and ebXML core component types in the UN/CEFACT. IDL data types used in first specification was converted into platform independent types which can also be used in ebXML and are user-defined types.

The AR/AP facility is a set of 12 interfaces using OMG/ IDL to support interoperable with other ledger/systems. These interfaces provide basic requirements for interpretability with different clients for example to identify payment transaction not already posted an e-banking application will use the ArapTransactionRetrieval interface and similarly use the ArapTransactionLifecycle interface to post the new payment transaction as shown in figure 4.1.

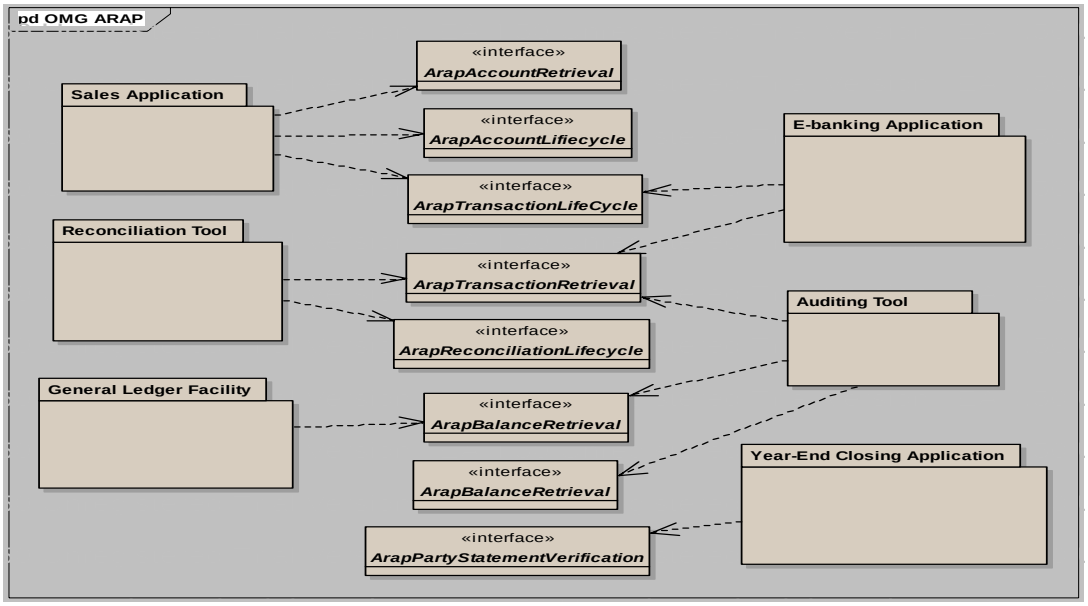

Figure 4.1 OMG AR/AP interfaces

The packages used in above figure are described in table number 4.2.

Name	Description
Sales/Purchase Application	A sales application, in order to register the sales transaction associated with an invoice will use the ArapAccountRetrieval interface to verify whether accounts already exist for the customer and the products involved, make appropriate calls to ArapAccountLifecycle to create any necessary accounts, and finally the ArapTransactionLifecycle interface to post the sales transaction to the AR/AP ledger.
E-banking Application	An e-banking application will use the ArapTransactionRetrieval interface to identify payment transactions not already posted, and then the ArapTransactionLifecycle interface to post the new payment transactions
Reconciliation	A reconciliation tool will use ArapTransactionRetrieval to retrieve payment and debt entries, and ArapReconciliationLifecycle as the accountant reconciles the payments with the debts
Auditing Tool	An auditor would use an application that calls ArapTransactionRetrieval and ArapBalanceRetrieval to verify transaction details and the resulting balances, and ArapReconciliationRetrieval to verify the matching of debts against payments.
Year-End Closing Application	An accountant preparing for year-end closing has to identify any discrepancies between the company and its business partners regarding debts and payments. For this purpose, a supporting application will use the ArapPartyStatementVerification interface to extract statements to send to each business partner, and to verify the statements received from the business partners
General Ledger Facility	An OMG General Ledger could use the ArapBalanceRetrieval interface in order to update the balance of its control accounts “Accounts Payable” and “Accounts Receivable”.

Table 4.2 AR/AP Packages

AR/AP fulfils some requirements, but it does not fully matches the requirements described in chapter 3 especially CIM and massages in a real sense. But it can be a good start for further work in this field.

4.2 XBRL XML (W3C XML) General Ledger Schema

eXtensible Business Reporting Language (XBRL) is a member of the family of languages based on XML, or Extensible Markup Language, which is a standard for the electronic exchange of data between businesses and on the internet. Under XML, identifying tags are applied to items of data so that they can be processed efficiently by computer software. XBRL utilizes the World Wide Web consortium (W3C)

After the success of reporting program, XBRL wants to grip the chance of using the power of XML for General Ledger. They know the limitations and inefficiencies of data interchange standards before XML, which was designed to be inflexible and limited to trading partners. Recent changes happened in IT and financial sector require extensible, flexible, multi-national solution that can exchange the data required by internal finance, accountants, and creditors that can be used in XBRL financial reporting. XBRL GL is an effort to bring to gather US and European accounting systems. It contains the information necessary to drill down from XBRL for financial reporting. It also provides details for audit and budget planning. XBRL GL will serve to store or convey information beyond that expected by either US or European users.

XBRL GL will help a company in its upgrade from a low-end accounting product to a mid-range accounting product. Small companies will get data easily data from out side. The receiving end will get data, and will review the data and understand what it has been given. In a consolidation across different accounting techniques, system will not miss data elements because automated tools collect the necessary information.

XBRL can export and import data without undue mapping. It does not lose information because it doesn’t need human entry. Other errors like rekeying are also limited.

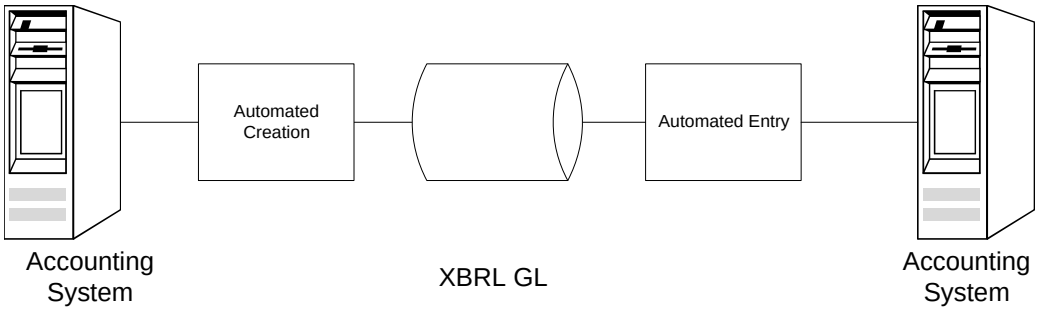

Figure 4.2 XBRL general Ledger Process [16]

It doesn't need to change formatting for report writing. The main advantage of XBRL GL is that it uses mapping which is easy to maintain.

XBRL and Standard Specification

XBRL is a subset of XML, the common language of data exchange on Internet which provides a widely embraced open standard technology for data exchange and transformation. It does not create new standards in this field but uses standards developed for XML. It provides the users with a standard format in which to prepare reports that can be subsequently presented in a variety of ways and information can be exchanged between different software applications.

XBRL is an open framework that provides for concurrent development of XBRL specifications in other countries and jurisdictions. The IASB has developed its taxonomy to accomplish this purpose. The countries like Australia, New Zealand, UK, Germany and Spain are taking this international framework to create their own taxonomies. Harmonization between IASB and national XBRL taxonomies will create greater interoperability of financial statement data for faster and better analysis.

Microsoft has a long-standing commitment to help, develop and promote the widespread adoption of fundamental Internet standards. The company sees XBRL as not only the future standard for financial reporting, but also a logical business choice. As such, Microsoft is a charter member of the XBRL Consortium, an international consortium of more than 140 of the world's largest accounting, technology, government and financial services bodies devoted to developing and promoting the adoption of XBRL as a standard.

XBRL is not a set of accounting standards. Accounting standards are the domain of the existing Generally Accepted Accounting Principles (GAAP) and regulatory standards bodies. But XBRL should support international accounting standards and languages other than the American dialect of English. XBRL is a platform on which reporting standards content will reside and be represented.

As mentioned above XBRL is based on XML, let us explain first ebXML before moving toward next candidate in our discussion.

4.3 Electronic Business using extensible Markup Language (ebXML)

It is not directly related to my work but it makes possible the idea of using World Wide Web and Internet for exchanging business messages. Internet opened a new window for business developers which is expending very fast. By taking advantage of the Internet and other available networks, ebXML opens up business to many more potential trading partners, in more places in the world than before. It provides a single framework for exchanging business data anywhere in the world that has access to these networks. There for I decided to include in existing standard specification and evaluate.

ebXML is designed to extend the benefits of **e-business** to much wider aspect. It provides a common way to electronically exchange business data expressed in XML from one computer to other. ebXML seeks to develop “A single set of internationally agreed upon technical specification that consists of common XML semantics and related documents structures to facilitate global trade.’’ (ebXML by Alan Kotok & David R.R. Webber)

ebXML uses UML to overcome hurdles of hardware platform, operating systems, software packages and network services. XML itself provides the ability to utilize any written language using a technique called Unicode double byte encoding systems. On other hand ebXML cuts across industry and business function boundaries, it is accessible to all trading parts in any kind of business or line of business.

ebXML System Overview

Overview of the interaction of two companies is shown in figure below. It shows a high-level use case scenario for two *Trading Partners*, first configuring and then engaging in a simple business transaction and interchange.

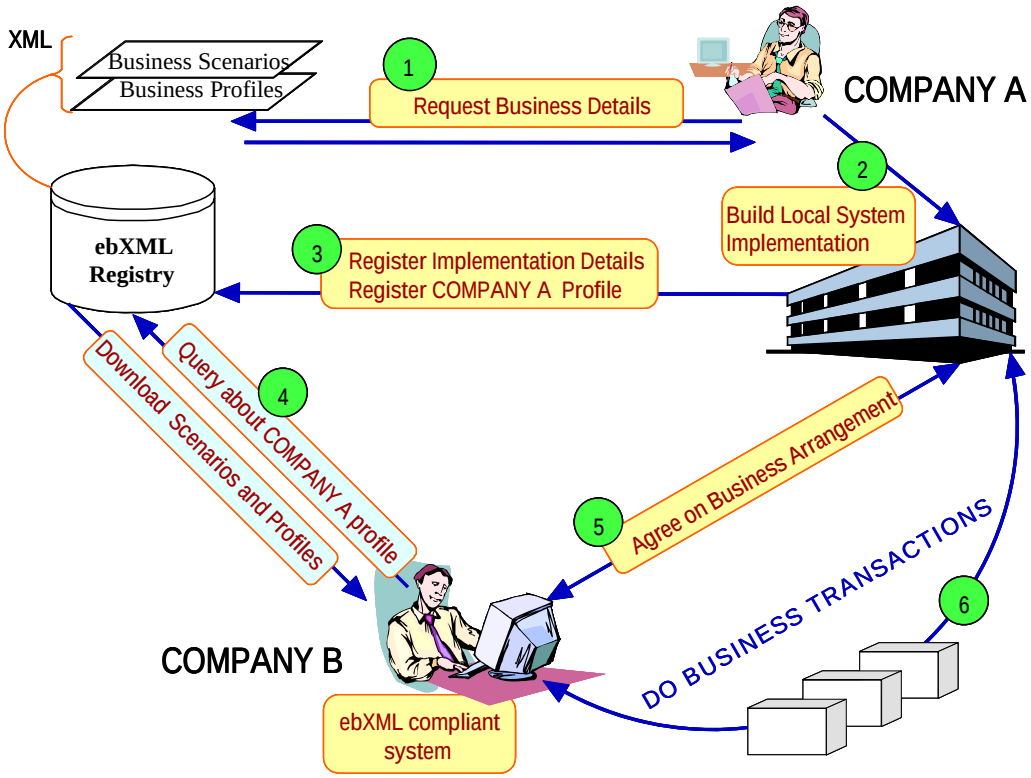

Figure 4.3 A high level overview of the interaction of two companies conducting eBusiness using ebXML [11]

Company A has become aware of an ebXML *Registry* that is accessible on the Internet. Company A, after reviewing the contents of the ebXML *Registry*, decides to build and deploy its own ebXML compliant application. Custom software development is not a necessary prerequisite for ebXML participation. ebXML compliant applications and components may also be commercially available as shrink-wrapped solutions.

Company A then submits its own *Business Profile* information (including implementation details and reference links) to the ebXML *Registry*. The business profile submitted to the ebXML *Registry* describes the company's ebXML capabilities and constraints, as well as its supported business scenarios. These business scenarios are *XML* versions of the *Business Processes* and associated information bundles (e.g. a sales tax calculation) in which the company is able to engage. After receiving verification that the format and usage of a business scenario is correct, an acknowledgment is sent to Company A.

Company B discovers the business scenarios supported by Company A in the ebXML *Registry*. Company B sends a request to Company A stating that they would like to engage in a business scenario using ebXML. Company B acquires an ebXML compliant shrink-wrapped application.

As ebXML gained rapidly attention internationally and they continued to work for more flexible and technology-neutral business architecture where it was possible. UN/CEFACT Steering Group (CSG) created the e-Business Transition Working

Group (eBTWG) in July 2001 for the purpose of continuing the UN/CEFACT's role in pioneering the development of XML standards for electronic business. The group was formed to build on the success of the earlier ebXML Joint Initiative between UN/CEFACT and OASIS, which delivered its first set of specifications in May 2001. This group has written a working draft for Electronic Business Architecture(UEB) where they have defined Core Components. Using Core Components as part of the ebXML framework will help to ensure that two trading partners using different syntaxes (e.g. XML and EDIFACT) are using business semantics in the same way on condition that both syntaxes have been based on the same Core Components. This enables clean mapping between disparate message definitions across syntaxes, industry and regional boundaries.

Core Components (CC)

A Core Component is a Design Phase artifact. A Core Component that represents a singular business concept with a unique business semantic definition. A Basic Core Component is constructed by using a Core Component Type. Basic Core Components are used in developing Aggregate Core Components. The Core Component may be modified or constrained for use within a particular Business Collaboration instance. During the Design Phase only, a Business Context may be used to constrain the Core Component to a specific adaptation of the Business Collaboration. Once constrained or modified by a Business Context, it is called a Business Information Entity (BIE).

Business Information Entity

A piece of business data or a group of pieces of business data with a unique business semantic definition is called BIE. A Business Information Entity can be either a Basic Business Information Entity (BBIE) or an Aggregate Business Information Entity (ABIE).

A BIE may be further modified prior to multiple Trading Partners designing a Trading Partner Agreement. The Business Context is finalized when a specific Business Process is bound to a Trading Partner Agreement; for example, it would not be possible to know the geo-political Context Driver until the geographical information is available from both Trading Partner Profiles. Once a Trading Partner Agreement has been finalized, it is not possible to modify or constrain the Business Message Payloads any further, without changing the Trading Partner Agreement

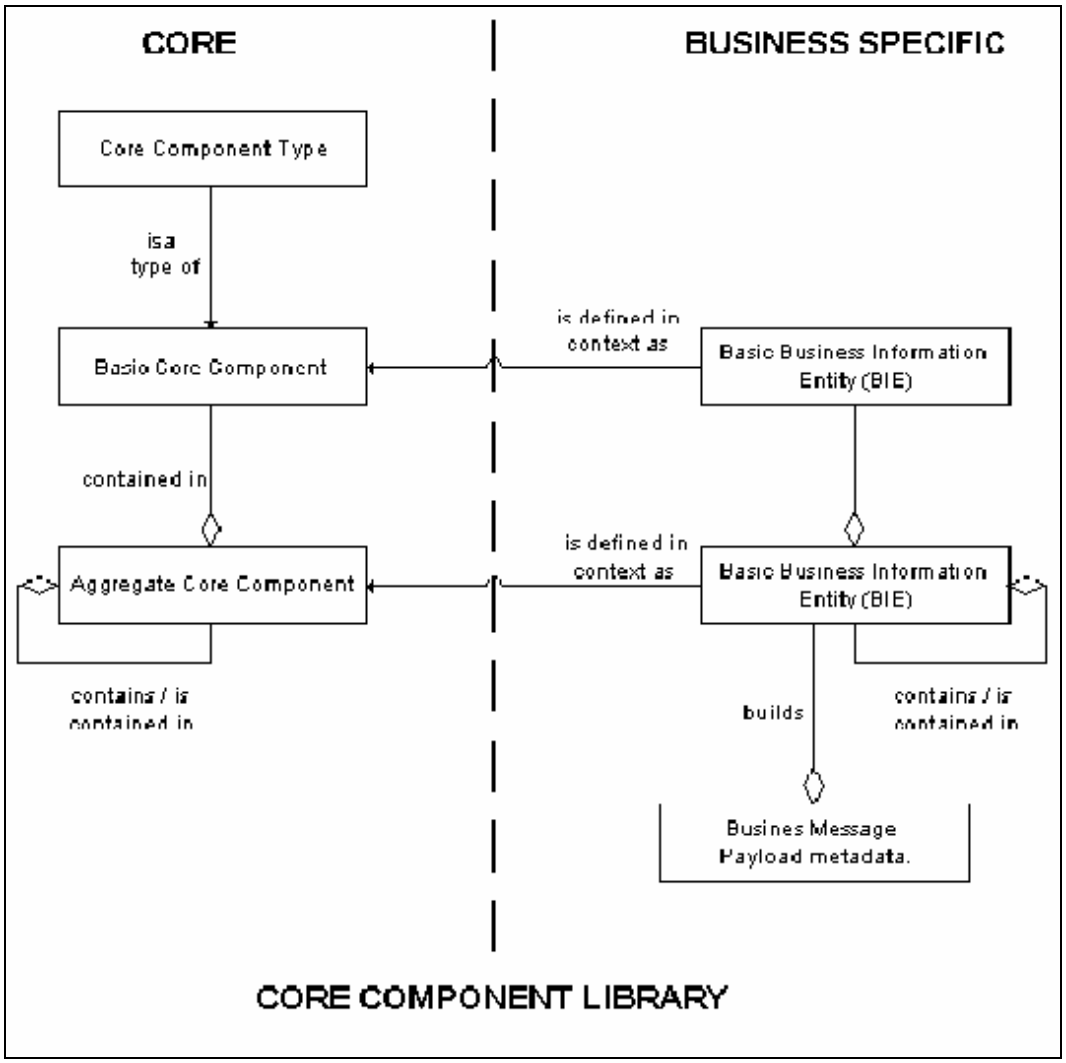

Figure 4.4 Core Component Library [11]

When it will interface with other components, Core Component or Business Information Entity may be referenced indirectly or directly from a Business Collaboration. This may be done via an intermediary document (Assembly Document) that describes how to construct a Business Message Payload during the Design Phase. The Assembly Document may specify a single, or group of Core Components, or Business Information Entities (required or optional) as part of a Business Message Payload instance.

4.3.1 Business Process Specification Schema (BPSS)

BPSS is developed by ebXML, the main goal of it is to provide the bridge between e-business process modeling and specification of e-business software components. It is available in both UML version and XML version. The UML version of ebXML is merely a UML class diagram. The XML version of the *ebXML Business Process Specification Schema* provides the specification for XML based instances of ebXML Business Process Specifications, and as a target for production rules from other representations. Both a DTD and a W3C Schema is provided.

The architecture of the ebXML Business Process Specification Schema consists of the following functional components:

- UML version of the Business Process Specification Schema.
- XML version of the Business Process Specification Schema.
- Production Rules defining the mapping from the UML version of the Business Process Specification Schema to the XML version.
- Business Signal Definitions.

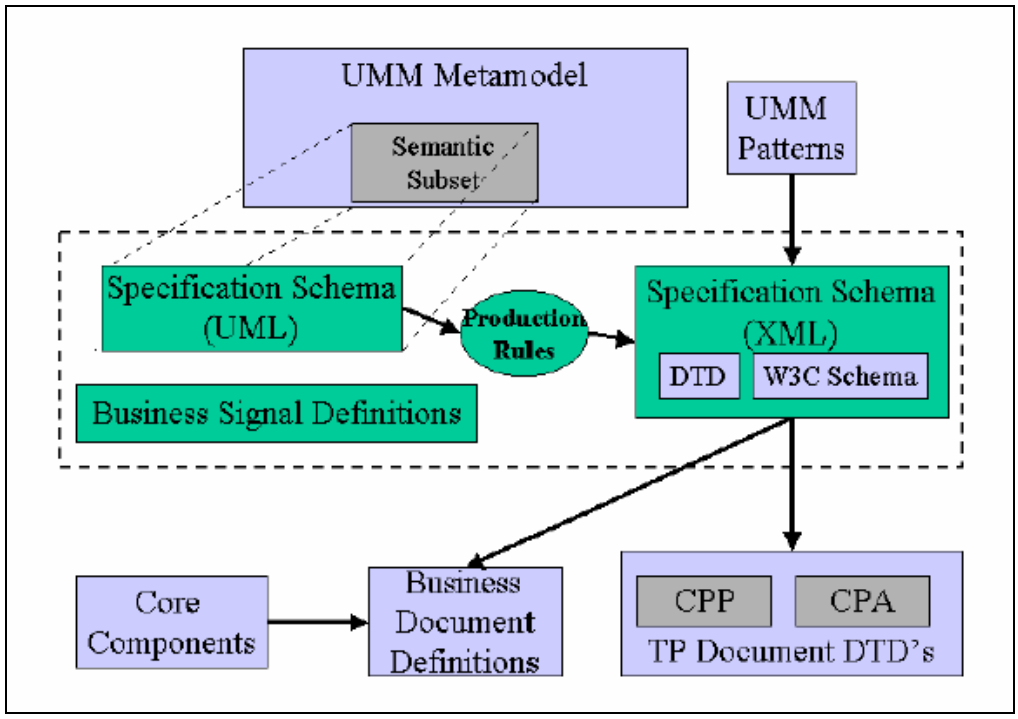

Figure 4.5 Relationship of ebXML Business Process Specification Schema to UMM, CPP/CPA and Core Components [12]

These components together allow to specify the run time aspects of a business process model.

4.3.2 ebXML and Standard Specification

UN/CEFACT and OASIS, the two leading organizations in the field of standardization sponsor electronic Business using Extensible Markup Language. ebXML uses existing standards for e-commerce and XML. It is a "suite" of specifications that includes an "open" architectural framework for global B2B interchanges. The ebXML framework is based upon non-proprietary technology that encourages interoperability.

ebXML is composed of three infrastructure components and several other efforts such as ones focused on document creation, business process definition, etc. The

infrastructure components are orthogonal in design. They may be used together or separately in implementing an infrastructure. ebXML infrastructure components which are approved by OASIS AND UN/CEFACT include:

- *Collaborative Protocol Profile (CPP*): It defines XML data structures which describe what each trading partner supports, the components necessary to conduct electronic commerce, such as data communications, security, processes, document types, telephone contacts, etc.
- *Registry and repository*: It defines the access interfaces, security and information storage format for any information that needs to be widely, yet securely shared among trading partners or potential trading partners.
- Messaging: It defines the means to move data between trading partners in a secure, reliable manner.

Standard specifications developed by OASIS in the form of ebXML Core Component are widely used for data types and messaging. It is very helpful to establishing new standard specifications especially PIM data types.

4.4 OASIS

Organization for the Advancement of Structured Information Standards (OASIS) is international consortium that drives the development, convergence, and adoption of e-business standards. The consortium produces Web services standards along with standards for security, e-business, and standardization efforts in the public sector and for application-specific markets. Two standards developed by OASIS, one for e-commerce i.e. UBL and other for methodology i.e. BCM are related to both parts of my thesis.

4.4.1 UBL 1.0

Universal Business Language is designed to provide a universally understood and recognized commercial syntax for legally binding business documents generated during order-to-invoice business process. UBL operate within a standard business framework such as ISO 15000 (ebXML) to provide a complete, standards-based infrastructure that can extend the benefits of existing EDI systems to businesses of all sizes.

UBL consists of schemas which are modular, reusable, and extensible in XML-aware ways. It is an implementation of ebXML Core Components Technical Specification 2.01, the UBL Library is based on a conceptual model of information components known as Business Information Entities (BIEs). These components are assembled into specific document models such as Order and Invoice. These document assembly models are then transformed in accordance with UBL Naming and Design Rules into W3C XSD schema syntax. This approach facilitates the creation of UBL-based document types beyond those specified in this 1.0 release.

UBL order-to-invoice business process include the UML class diagrams of the document components on which the schemas are based and UML class diagrams describing all the document assemblies. It has also Spreadsheet models defining the document assemblies and descriptions of two example implementations and formatting specifications for UBL basic business document types.

UBL provides the basic information which is very important for Account Receivable and Account Payable. AR/AP could be based on invoice package of UBL.

4.4.2 Business Centric Methodology (BCM)

BCM is a methodology developed by OASIS for business integration. The Business-Centric Methodology (BCM) is a complementary approach to current architectures and methods for constructing business-oriented services. The main features of the BCM are:

- Involves a layered approach for strategically managing artifacts and constraints while achieving semantic interoperability.
- Enables precise communication between business users and technical experts as well as between Enterprise applications and their respective business partner systems
- Addresses integration problems through pragmatic and semantic interoperability mechanisms
- Exploits the dynamic nature of common mechanisms

BCM layers and information architecture is illustrated in diagram given below.

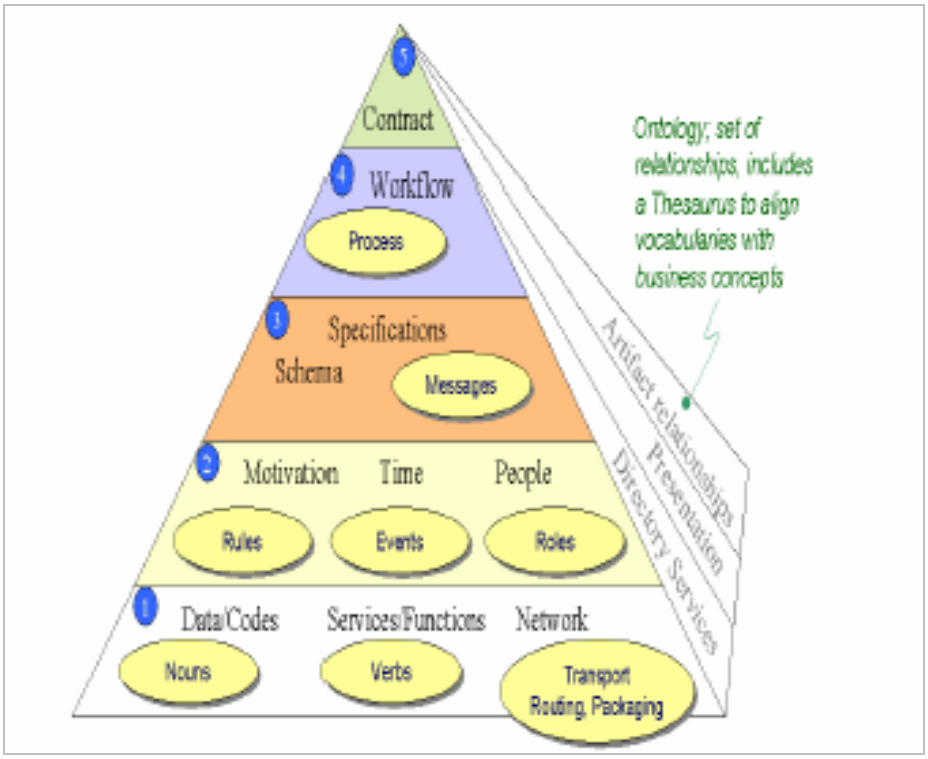

Figure 4.6 BCM processes [15]

4.5 UN/CEFACT UMM

United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) has developed the Modeling Methodology. The UMM [14] is an incremental business process and information model construction methodology that provides levels of specification granularity suitable for communicating the model to business practitioners, business application integrators, and network application solution providers. The main objectives of UMM are given below.

- Have a comprehensive business process and business information meta-model as well as a comprehensive process analysis methodology.
- Retains business acumen that is reusable over generations of implemented technology
- Provides a methodology and supporting components to capture business process knowledge, independent of the underlying implemented technology
- Helps discover and define a set of reusable process and information descriptions. Patterns help enforce consistent, reproducible results from the UMM-MM across business domains and their business domain experts and analysts
- Implements processes that help insure predictable results from a software project
    - Facilitates the specification of reusable/reproducible process models, in objects and interface-specific object

behaviour descriptions that are technology and protocol insensitive.
    - Focuses on technology and protocol independent steps of a software engineering process.
- Is an extension of UML
    - Is a UML profile used to describe the UMM components to specify the business do- main specific stereotyping that supports a complete business process and information definition to describe and analyze individual business processes.
- Structures the Business Operational View (BOV) of the Open-edi Reference Model into layers of “views”.

The UMM can be employed by business analysts to define external and internal Business Collaboration Frameworks. The UMM can be used to define the Business Collaboration Framework implemented between two or more parties. The UMM can be employed from the top-down or bottom-up or using both approaches simultaneously. The end result of an integrated use of the UMM would be a defined Business Collaboration Framework.

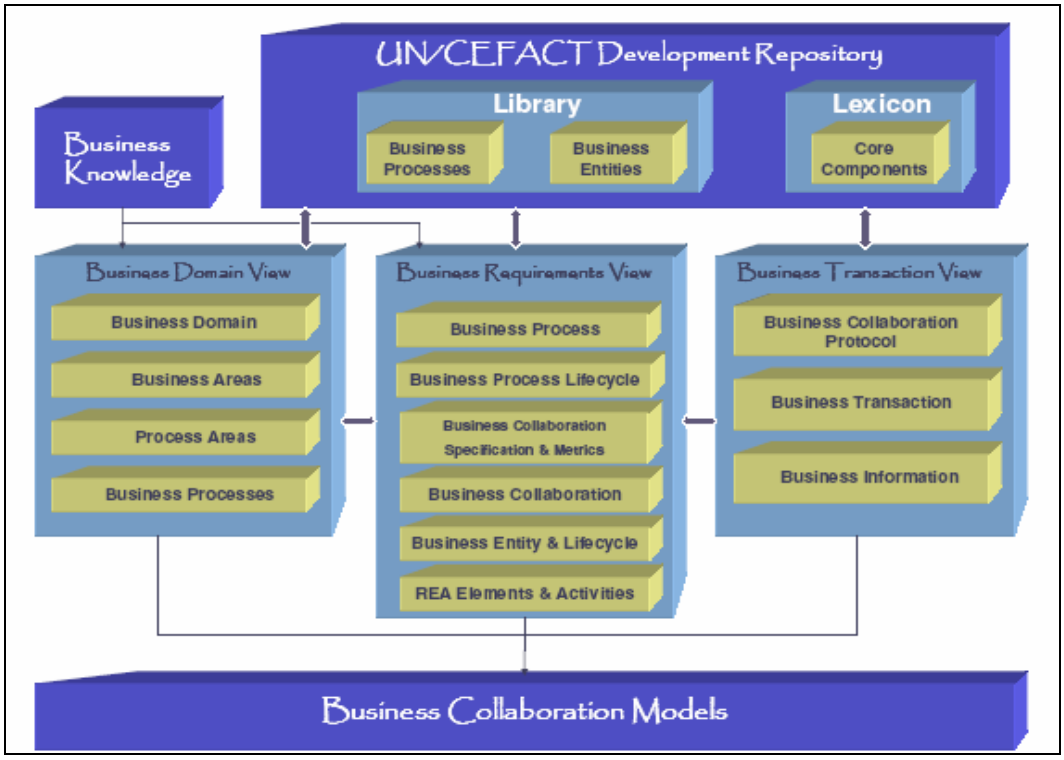

Figure 4.7 Overview of UMM Worksheets and Models [14]

Modeling using the UMM

Business modelling in UMM has three major UMM views, i.e. Business Domain View (BDV), Business Requirements View (BRV) and Business Transaction View (BTV). Procedures within each of these views describe how to populate the worksheets. The worksheets help to collect and organize the information needed

to produce the minimum UMM models for that work area. A high-level overview of these worksheets and models is shown in Figure 4.7.

Data Types in UMM
Data types in UMM include primitive pre-defined types and user-definable types. Pre-defined types include numbers, string and time. User-definable types include enumerations. An enumeration is a user-defined data type whose instances are a set of user- specified named enumeration literals. The literals have a relative order but no algebra is defined on them. UML avoids specifying the syntax for constructing type expressions because they are so language-dependent.

4.6 Evaluation of existing language candidates

Table number 4.3 shows that there is no single language that meets all requirements. The closest to a solution is UML-UMM therefore we propose to use UML as standards specification language with some improvements in UMM in the next chapter. The main problem with IDL and XML is that they are platform specific and do not fulfill the requirement of model-driven technique with CIM and PIM. UML-UMM has a structural way of presenting the models but it does not meet all requirements mentioned in section 3.1.

Language Requirement		IDL	XBRL XML-XSchema	UML-UMM
R1	Formal Language	+/-	+/-	+/-
R2	Specification of CIM and PIM and PSM models	-	-	+/-
R2a	Structure	-	-	+
R2b	Behaviour (Process/Services)	+	+	+
R2c	Constraints	-	-/+	+/-
R3	Ease of use – graphical notation.	+	-	+
R4	Platform independent data types.	-	+	+
R5	Computer processability	+	+	+/-
R6	Non functional aspects like performance, QoS, error handling, security issues, usability, reliability, availability, adaptability, supportability.	+/-	+	-
R7	IT standard and continuous development.	+	+	+
R8	Support from existing working tools.	+	+	-/+
R9	Methodology and process description.	-	-	+

Table 4.3 Evaluation of language candidate

5 UML-SSS: “UML method for Standards Service Specifications”

We propose UML-SSS as an improvement to the UML UMM approach, by having a simplified methodology following MDA ideas. In addition to UMM and MDA we have used COMET [9] method to develop UML-SSS. The core of UML-SSS is: modeling language UML, COMET Methodology and MDA. UML-SSS stands for UML for Standards Service Specifications.

5.1 Modeling Language and Notations

As software becomes more and more complex, the need for modeling increases similarly. We must have a modeling language for making models otherwise the purpose of modeling and raising abstraction level will not be fruitful. I will use UML as modeling language, based on the conclusions from chapter 4.

5.1.1 Unified Modeling Language (UML)

The Unified Modelling Language (UML) can be used for the modelling of software architectures. According to OMG definition of UML is:

> “UML is a language used for specifying, visualizing, constructing, and documenting artefacts of software systems, as well as for business modelling and other non software systems” [2].

UML has been excessively used in modelling Object Oriented Systems. The notation it provides mostly supports the specification of a system’s artefacts at each phase of the system’s lifecycle. Thus, UML is able to document and describe a system’s requirements using e.g. use-case diagrams, a system’s structure and behaviour using e.g. class diagrams, sequence diagrams, state and activity diagrams, etc. or even a system’s physical deployment using deployment diagrams. The diagrams can be divided into two groups:

Structural diagrams

- **Class diagram**: This shows the relationships between a set of classes. This is the most common diagram in UML, and is a structural view of how classes are related.
- **Collaboration diagram**: This shows the structural organization of objects that communicate with each other.

- **Component diagram**: This shows the relationship between components. A component is a physical manifestation of software, and may be a library, executable, DLL, and so on. Components are built and combined to form applications.
- **Deployment diagram**: This shows the relationship between physical entities like a CPU, printer, or workstation. It can also show where components are deployed.

Behavioural diagrams

- **Use Case Diagram**: This captures functional requirements. It shows the relationships between actors and use cases. Actors are entities external to the system, such as users and other systems. A use case is an end-to-end sequence of actions (including variants) that result in an observable and useful result.
- **Sequence Diagrams**: This is an interaction diagram that shows the communication between objects in a time-ordered fashion.
- **State Diagram**: This contains a finite state machine that describes the behaviour of a class. State machines are an excellent way to describe event-driven behaviour.
- **Activity Diagram**: This shows the flow of control through activities in a system.

Furthermore, UML is capable of producing code out of the system's specification. UML is becoming more and more important in software architecture and development. Another strong reason in the favour of UML is that we can model just about any type of application, running on any type and combination of hardware, operating system, programming language, and network, in UML. Its flexibility lets us model distributed applications that use just about any middleware on the market. Built upon the MOF metamodel which defines *class* and *operation* as fundamental concepts, it's a natural fit for object-oriented languages and environments such as C++, Java, and the recent C#.

5.1.2 COMET 2.4

COMET [9] is a methodology for constructing software systems by using components. COMET methodology was developed in COMBINE project. It means Component and Model based development Methodology and COMbine Methodology. It is en object-oriented analyses and design methodology. It follows the object-oriented paradigm for modelling where software systems are viewed and modelled as a set of collaborating objects. During analysis, design and implementation it encourages the use of business objects. Briefly we can describe COMET as model-focused, UML based approach aimed at supporting the process

of developing and maintaining products and product families. It is object-oriented, component based analysis and design methodology.

COMET is a stepwise description that gives practical guidelines of how to analyse, design, and implement software systems that is based on business objects. The focus is on systems running in a distributed, component based environment. It encourages the belief that systems based on self contained components with sound system architecture will be easier to maintain. It also encourages the reusability with reference model concept which provides a mean of standardising concepts, models, and patterns within a domain.

5.2 Model Overview in UML-SSS

We propose two main domain models i.e. Business Domain and System Domain where system domain will be derived from business domain. UML-SSS consists of following four models:

- *Business Model (CIM)*
- *Requirements Mode*
- *Architecture Model (PIM)*
- *Platform Specific Model* (PSM)

Requirement Model may be optional. These models contain work products that provide the viewpoint specifications for the component-based system or the individual component that is being developed. My focus will be on business model in Business Domain and architecture model and Platform specific model in System Domain. The model overview is shown in figure 5.1 which also shows the relationship between the model word and the real word.

The modelling activity starts with Business model and requirements model, both these models are business domain of system. These models could be mapped to architecture model with help of QVT, MOF transformations or architecture model is derived manually from business model. All models are further described in next subsections.

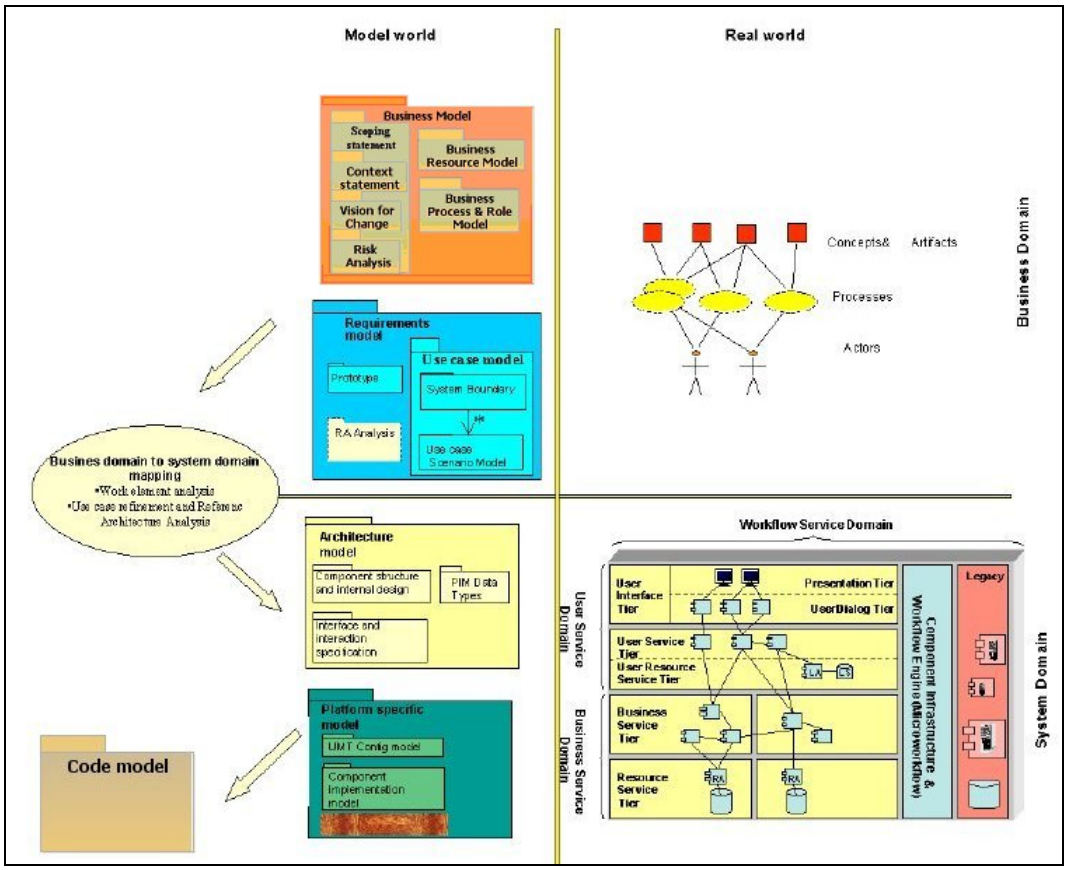

Figure 5.1 UML-SSS model overview

5.2.1 Business Model (CIM)

Business Model corresponds to CIM in general. The business model contains human actors and technical systems, some of which are software systems. Models on the business level involving a software system depict the intended use of this software system by the actors. In other words, the collection of business models that involve a software system gives the functional requirements for that system. Models on the business level are therefore of paramount importance in software development to ensure that business needs are catered for by the software systems. However, if these models are not related to the software system level they loose much of their value. The software systems in the business models should be decomposed into a configuration of interacting software components so that the software system requirements are met by this configuration. The figure below illustrates the structuring concepts of the business model. Business Model consists of Scoping Statement, Business Resource Model and Business Process & Role Model.

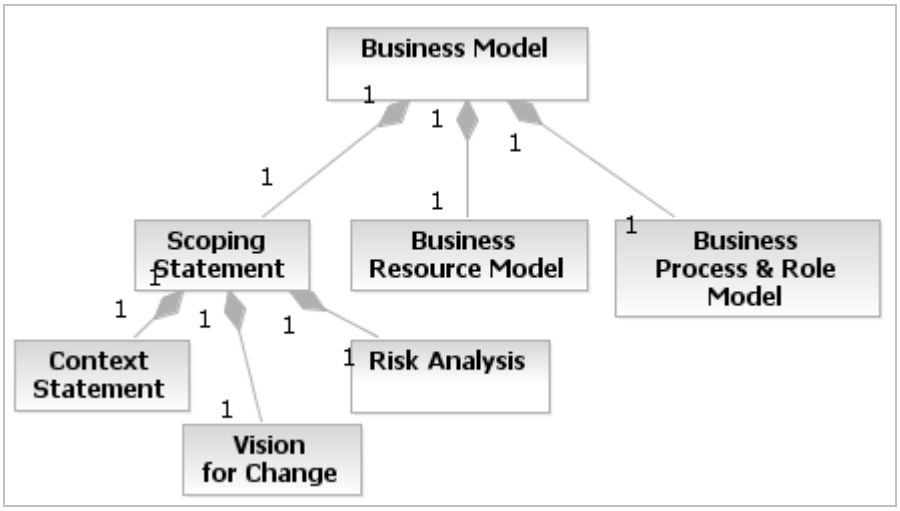

Figure 5.2 Business model structuring concept

Scoping Statement
It will be used to denote mostly those pars of business model that are not presented in UML. It consists of Context Statement, Vision for change and Risk Analysis. The main purpose of Context Statement is to define the scope of business model and position in its context. This informal representation takes the form of a domain picture aiming to give an overall understanding of the domain. The risk analysis describes marketing factors that might influence the product, good or bad, and things that are required that are not described in the business vision and product description work product like non functional requirement. In a vision for change we'll explain what to improve, the motivation, a description or indication of what the improvements might be and a gap analysis.

Business Process and Role Model
Business Process and Role Modelling is a behavioural model that defines, in terms of roles and steps, the business processes of the domain which are relevant to the Product, and which will enable the goals to be met, and gives a full definition of the roles in the business, including those fulfilled by the Application Components which are to be developed. The main objective of the Business Process Model is to identify and detail all the business processes supported by the Product to the extent necessary to detail the roles of the Product.

This model may be at a number of levels of detail, from a high level description of the business processes down to detailed specifications for the business services that each IT element in the Product will provide. It includes a full definition of the roles in the business, focusing on those fulfilled by the system or component to be developed. Figure 5.3 is an example of business process and role model.

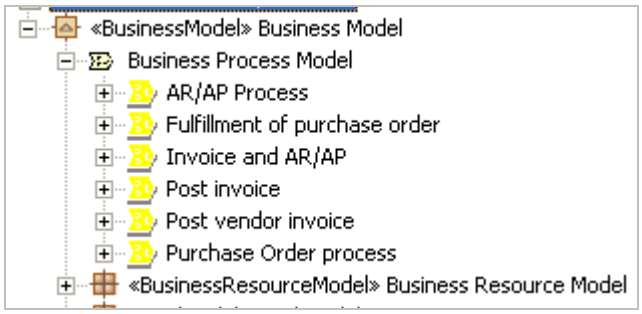

Figure 5.3 Business Process & Role Model

Business Resource Model

The Business Resource Model is an information model that identifies and defines the main things (and concepts) of the domain that are relevant to the Product, these being, in general, those things that do things in the business (including the Product itself), and those things that have things done to or with them, and details the relationships between these concepts.

5.2.2 Requirements Model

Requirement Model will be optional model in UML-SSS. We can specify requirement by using this model instead of writing them in plain text. The requirements model identifies the system requirements. These include functional requirements, non-functional requirements (quality of service) and constraints. Non-functional requirements are statements concerning performance, availability, security, reliability and so forth. There are also other requirements specifying constraints that will have impact on the system to be developed, for instance available resources, special customer preferences, company strategy etc. The following models are important in requirement modelling.

- System Boundary
- Use case Model
- Use case Scenario Model

5.2.3 Architecture Model (PIM)

The Architecture Model is the core model for development of a software, it describes the overall architecture of the system and its partitioning into components in terms of collaborations of components and subsystems, component structures, component interactions, and component interfaces and protocols. It describes two aspects of the component collaboration, namely the static (structure) and dynamic (behaviour). The structural model describes the components, their dependencies, and their interfaces; the dynamic model describes the component interactions and protocols. The result is to define components in terms of what interfaces they provide, what interfaces they use, and how these interfaces should be used

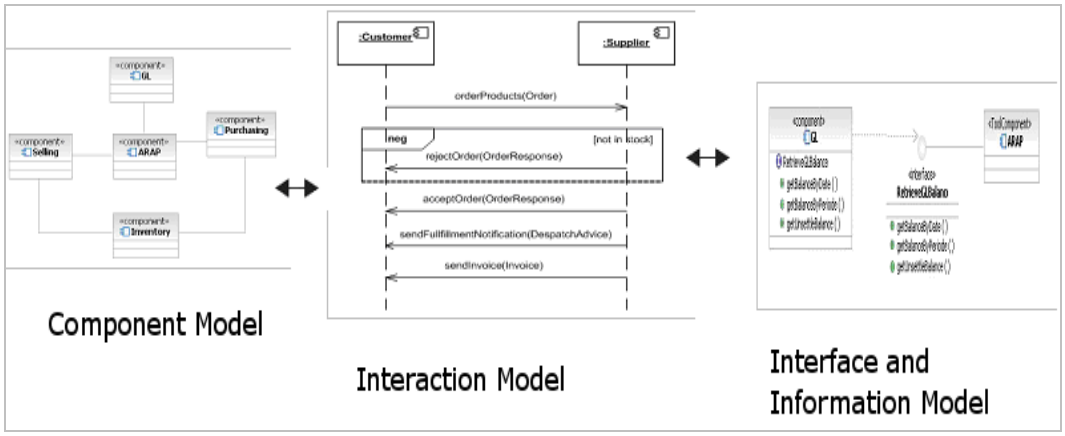

Figure 5.4 Architecture model work products

Figure 5.4 shows the models (work products) that are part of an architecture model specification. The main models used in architecture model are as follows:

- **Component structure model** describes the high-level components and their interdependencies.
- **Component interaction model** describes the interactions between the high-level components.
- **Interface model describes** the details of the component interfaces, i.e. their operations and detailed behaviour.
- **Information model describes** the information structures that are conveyed through component interfaces. The information model may be included in the Interface Model if convenient.

5.2.4 Data types

Data types are specified in Platform Specific Modeling. We use the basic data types provided by PSM and generate or construct other one which we need. But when we talk about PIM we have to think about platform independent data types. PIM data types are needed to keep the PIM to platform independent. COMET support the platform independent data types which we require in business resource model or architecture model. We can choose data types from three international standards which are:

- OMG's IDL which is ISO standard
- Core Component types of ebXML.
- The third option is the ISO/IEC 10404 "Language Independent datatypes" which is also a set of data types independent of any target implementation.

If we divide data types into two main categories like basic type or primitive type and composite types or derived types then data types from ISO/IEC 10404 will be used as basic data types and composite data types will be derived from Core Component data types.

The example of Core Component types is show in figure below. I have used *String, Boolean, Integer and UnlimitedNatural* as basic types.

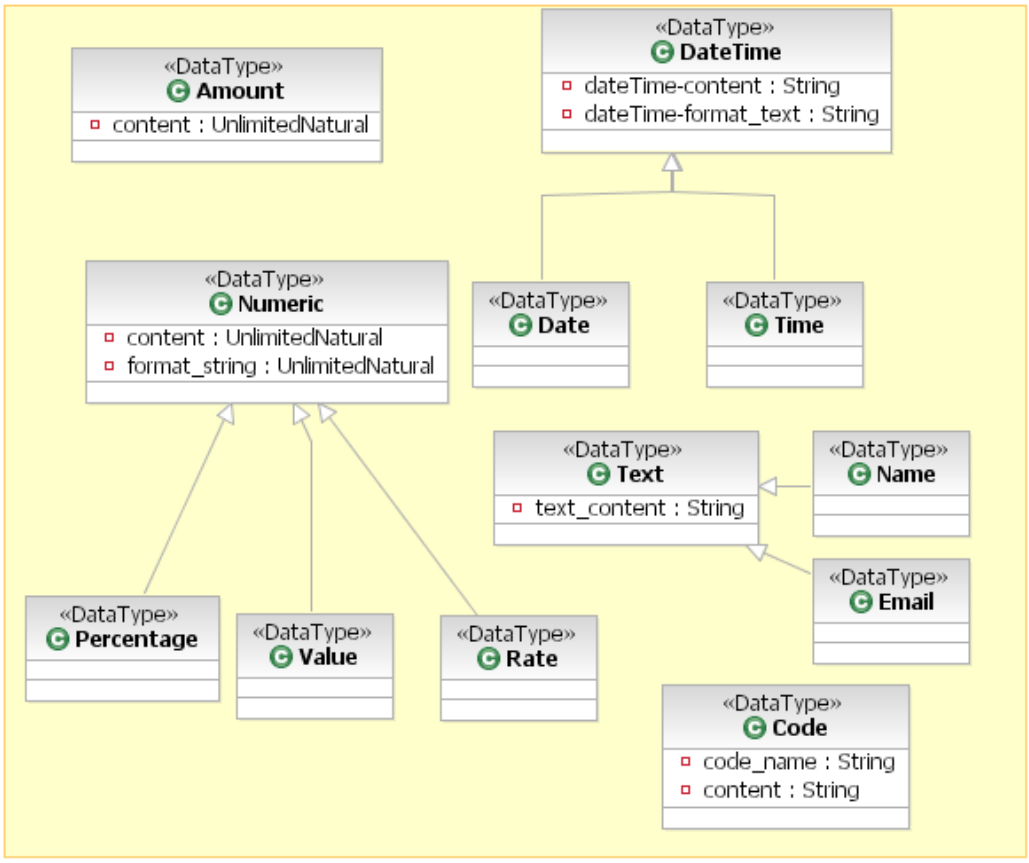

Figure 5.5 PIM data types

The decision of PIM data types is also related to UML because it is assumed language for PIM modelling. Therefore a set of chosen base types can be used as types for UML attributes, parameters, and return values for operations. The representation of data types in UML will not be difficult. The simple base types can be used directly. Enumerates (enumerations) and state must be defined as UML data types. Named data types can be defined as UML data types. (How is their base defined?). Subtypes and extended types like “range” can be used directly as types in attribute/operation declarations or as part of a type definition. The aggregate data types Set, Bag, Array, Sequence, and Table can be used directly as types in attribute/operation declarations or as part of a type definition.

5.2.5 Platform-Specific Model

The Platform-specific Model contains the following work products:

- The Platform Profile Model (explicit PSM), which specifies the system in alignment to the actual technology profile for the specific platform.

- The Component Implementation Model, which describes the implementation of the component specifications in a given programming language, and the deployment properties/configurations for the target computing platform (hardware, operating system, etc.) in which the system is to run.

In addition two other work products may be used i.e. UMT Configuration Model (Implicit PSM in a code generator tool) and the Deployment Model should describe the deployment properties/configurations for the target computing platform (hardware, operating system, etc.) in which the Product is to run.

5.3 Methodology and Process

A methodology formally defines the process that we use to gather requirements, analyze them, and design an application that meets them in every way. There are many methodologies, each differing in some way or ways from the others. I will primarily use COMET methodic with MDA concept and modeling language UML in my proposal. It will be iterative and incremental process.

Iterative and incremental
UML-SSS stands for a combination of iterative and incremental development, where an initial statement of requirements is developed in the beginning. Then the system is designed and implemented in increments, one or a few use-cases at a time. Each increment should result in a usable system.

5.4 Applying the MDA framework

The first step when creating an MDA-based application is to create a Platform-Independent application Model (PIM). In the MDA, a model is defined to be a representation of a part of the function, structure and/or behaviour of a system; i.e., the definition is usable in the Modeling and Simulation domain quite well. The PIM will be expressed in UML in terms of the appropriate core model. The core models are available in form of UML Profiles of which a number already are well along their way to be standardized by the OMG.

The next step is that if the model shall run as an application, then we have to convert this model from general application to a Platform Specific Model (PSM). The PSM is derived from the PIM using standardized transformation rules. While the PIM defines the necessary functionality, the PSM specifies how this functionality is realized on a special platform. The focus of MDA is on PIM and PSM but UML-SSS starts its modeling process with business model i.e. CIM. Therefore I will start from Business Model as described in UML-SSS and transform it into PIM. This transformation from CIM to PIM is not supported by working tools mentioned in next subsection it is manual process which guides to PIM.

But next two transformations i.e. from PIM to PMS and then code generation from PSM are supported by working tool at some extent. The figure 5.6 shows the process of transformations.

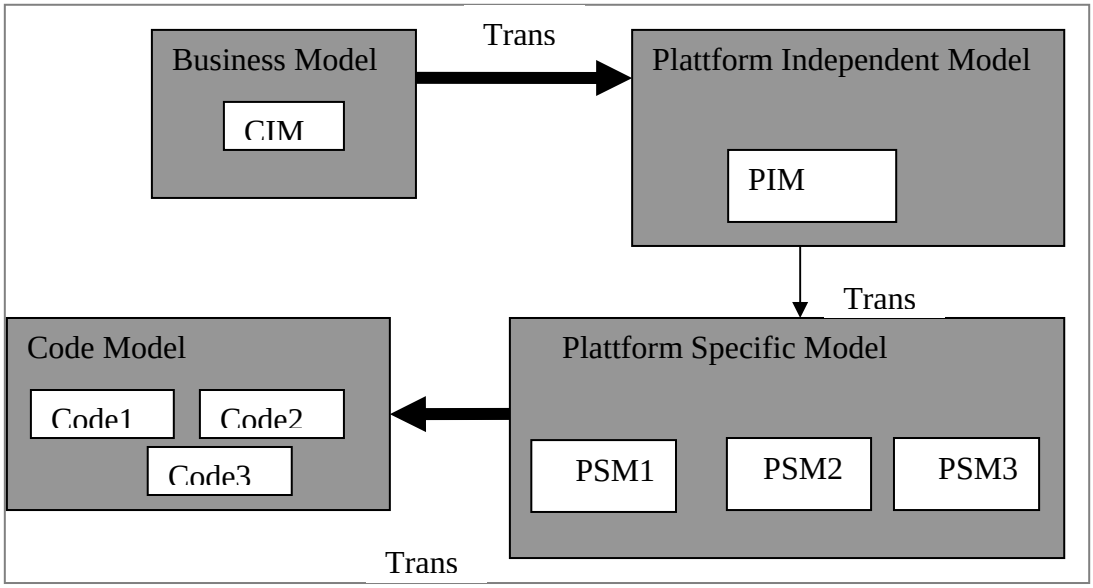

Figure 5.6 Applying MDA to UML-SSS

The PIM to PSM Transformations
To start the MDA process we need to build a platform-independent model (PIM) that comprises the all models described in UML-SSS, I have used UML to develop PIM. Number of transformations is equal to number of PSMs, in this case there are three PSMs, and therefore we need three transformations.

The PSM to Code Model Transformation

I will generate code for each PSM therefore the number of transformation will be equal to number of PSMs.

5.5 Working tool

Mostly UML-based tools implement a particular methodology; it is often not be practical to pick a tool and then try to use it with a methodology that it wasn't built for. But, some methodologies have been implemented on multiple tools so this is not strictly a one-choice environment.

I will use the IBM Rational Software Modeller/Architect connected with a Model Transformation tool (UMT or similar) to generate (in principle) WSDL/XML and EJB/Java interfaces or CORBA IDL from the same models). I will use OptimalJ too for generating PSM from PIM.

6 UML-SSS Application to Finance AR/AP

In this chapter we will apply the UML-SSS as described in chapter 5 to Finance AR/AP. We will take a typical example some purchase and sale to explain AR/AP interaction with other systems and to describe the business context for the AR/AP application using UML-SSS methodology. To understand the whole domain we will start from order-to-invoice process and then this broad domain will be broken down into different processes. As my focus is on accounts receivable and account payable therefore it will be explained or modeled in detail and other processes are taken to understand the business domain.

6.1 Computation Independent (Business) Model (CIM)

CIM is enterprise view point of the AR/AP system or component for business owners, developers, managers and users. CIM models focus on a system's environment and requirements rather than on its structure. CIM models are referred to as AR/AP system domain models. As mentioned in the chapter 5 the Business Model consists of a *Scoping Statement, the Business Process & Role Model and the Business Resource Model.*

6.1.1 Scoping Statements

Scoping Statements in UML-SSS, consisting of the following:

- The context statement
- Vision for change
- Risk analysis

6.1.1.1 The context statement

The context statement defines the scope and positions of AR/AP model in its context. There are four important actors some are involved in AR/AP scenario i.e. buyer, seller, General Ledger and Banks (Buyer bank and seller bank), these are shown in figure below and described in table 6.1.

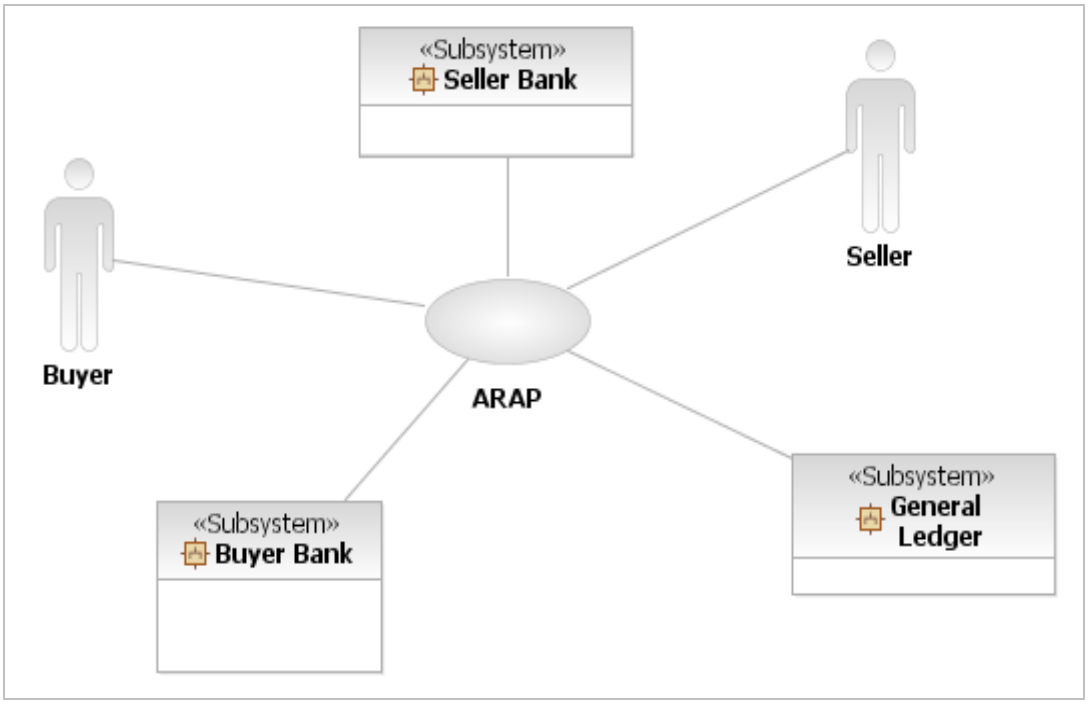

Figure 6.1 Buying and selling context diagram for AR/AP

Stakeholder	Description
Buyer	Buyer sends purchase order, receives goods and invoice and pay it through its bank. (Invoice is registered in AP).
Seller	Seller responds PO, sends goods through shipment, sends shipment invoice and sales invoice through Sales system and receives payment from buyer in its bank account.
General Ledger	GL is the main financial book where all accounting entries are registered. In this case posting to relevant AR/AP accounts in GL.
Buyer Bank	Receives payment order from Buyer and pay it to seller bank, which sends information to the Seller.
Seller Bank	Receives payment from buyer's bank and credit the seller account.

Table 6.1 Description of stakeholder in AR/AP context diagram

Order-to-invoice process with AR/AP

The activity diagram for order-to-invoice process shown below has five major steps. It shows the general processes involved in purchasing and selling. I will focus on last two steps i.e. Invoice and AR/AP, Payment and AR/AP. Theses two processes are important for AR/AP ledger. I will describe generally the first two steps but last two steps with more detail in the next subsections.

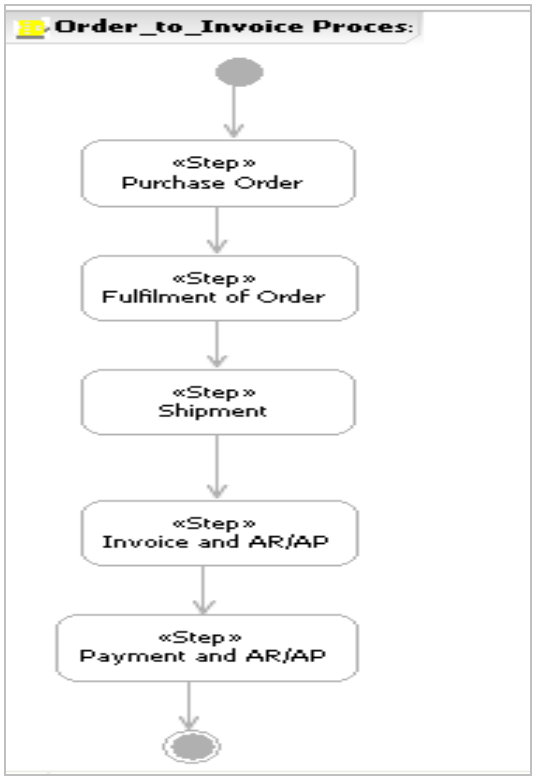

Figure 6.2 Order-to-Invoice processes

The description of each step show in Order-to-Invoice process is given below in table 6.2.

Process	Description
Purchase Order	Purchase order is send by buyer to Seller with description of goods or services which he wishes to buy. It is a promise to buy. It may support revision or cancellation of the Purchase Order.
Fulfilment of Order	Fulfilment of Order is a process where Seller receives purchase order, checks the stock and sends shipment invoice to shipping company. He may send fulfilment information i.e. Sales Order to Buyer.
Shipment	Shipment process covers all shipping details.
Invoice and AR/AP	After fulfilment of purchase order and shipping, seller sends invoice to Buyer and it is posted to Account Receivable in accounting ledger. On the other side Buyer receives the invoice and posts it to Account Payable in accounting ledger.
Payment and AR/AP	This process is initiated by Buyer. He pays the invoice through his bank and entry is made in AP ledger. Buyer bank sends information to seller bank and Seller receives the payment in his bank account and entry is made in AR ledger.

Table 6.2 Order-to-Invoice processes

6.1.1.2 Vision for change

As mentioned in chapter 1&2 software system are becoming more and more complex and complicated. We have to raise the level of abstraction to understand and control the system development in future. We can save time by reusing the standards specified components for AR/AP. It will be highly automated tool to support the posting, retrieval and reconciling to account receivable and account payable directly at the end of order-to-invoice process.

6.1.1.3 Risk Analysis

The risks and non functional aspect involved in the AR/AP specification is listed below. The software developer, who will implement the AR/AP should take care of these issues.

Issue	Description
Security	The AR/AP system will contain very business sensitive information about the customer and must implement a very strict access control mechanism. Only authorized customer or supplier should have access to system.
Compatibility	AR/AP must be compatible with old versions and other internal and external systems.
Technology	The PIM and PSM based models in UML.
Muti-user and transactions	The system must support multiple users in a distributed organization to work in parallel and synchronize their schedules into one common global schedule.
Performance	The system must support an efficient communication protocol between the clients and the server.

Table 6.3 Risk Analysis for AR/AP

6.1.2 The Business Process & Role Model

This model is the further detail of general purchase and sale process illustrated in the above subsection. Order-to-Invoice Process is a very large process. It consists of five steps i.e. Purchase Order, Fulfilment of Order, Shipment, Invoice and AR/AP, and Payment and AR/AP. I will not go into details of all these steps here, but Purchase Order and Fulfillment of Purchase Order steps will be explained generally in activity diagrams below, Invoice and AR/AP step is described in more details and it is further divided into many sub steps and explained with different activity diagrams in the following subsections.

6.1.2.1 The purchase order process:

This step has many sub steps and two main documents which are explained in table given below. When buyer wants to buy some thing from seller it sends purchase order to the buyer. He initiates the process the Complete-PO, which creates document for purchase i.e. Purchase Order with all necessary information which buyer has registered in the system. The next step Send PO sends this document to the Seller. The Seller has a step Receive PO to handle the Purchase Order from the Buyer, which receives the document and pass control to Process PO. The Seller checks the ordered goods in warehouse and responds to the buyer request. The seller can respond the buyer in three ways: It can reject the order if goods are not available at warehouse, or accept the order, or accept the order with some changes, these steps are handled in Reject order, Accept order and Accept with changes respectively and PO response is send to the buyer.

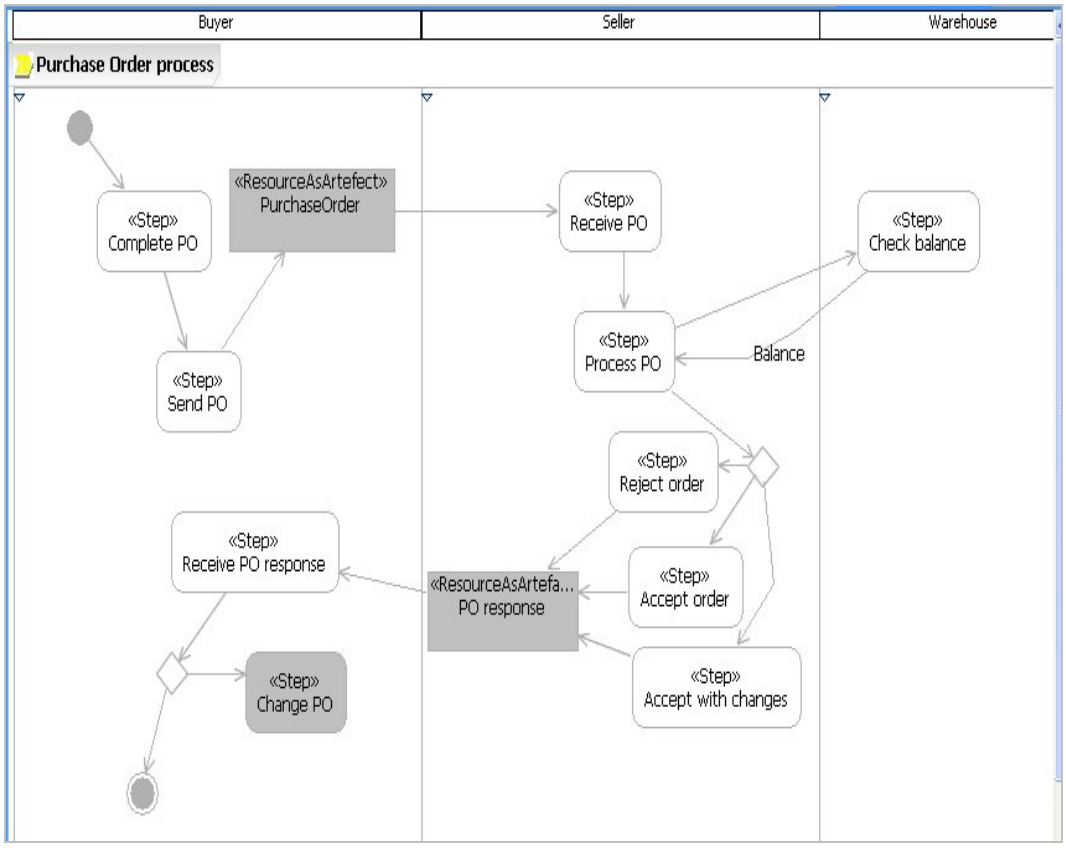

Figure 6.3 purchase order process

When the Buyer receives the PO response, he has two options, if it is a acceptance of order, he waits for delivery of goods. In other case he may change or cancel the purchase order it depends on trading agreement this should be done in Change PO step which is not described here.

Name	Description
PurchaseOrder	It is a document send by Buyer to Seller to inform him that he wishes to purchase goods or services.
PO response	It is a document send by Seller to Buyer in the response of PurchaseOrder. It has information weather the order is accepted, rejected or accepted with changes.
Change PO	Change PO is a further step to change or cancel the order. It has the same steps as mentioned in purchase order.

Table 6.4 Documents sent and received in purchase order process and Change PO.

6.1.2.2 Change PO

This step is similar to the Purchase Order process, it repeats approximately all the steps listed in section 6.1.2.1 above.

6.1.2.3 Fulfilment of Purchase Order

This step is initiated by the Seller as illustrated in diagram 6.4 for the fulfilment of purchase order. The goods are despatched to Buyer or messages is send to Transport Company for transportation of goods to the Buyer. The Transport Company issues transport advice and send it to the Seller who further sends it to the Buyer.

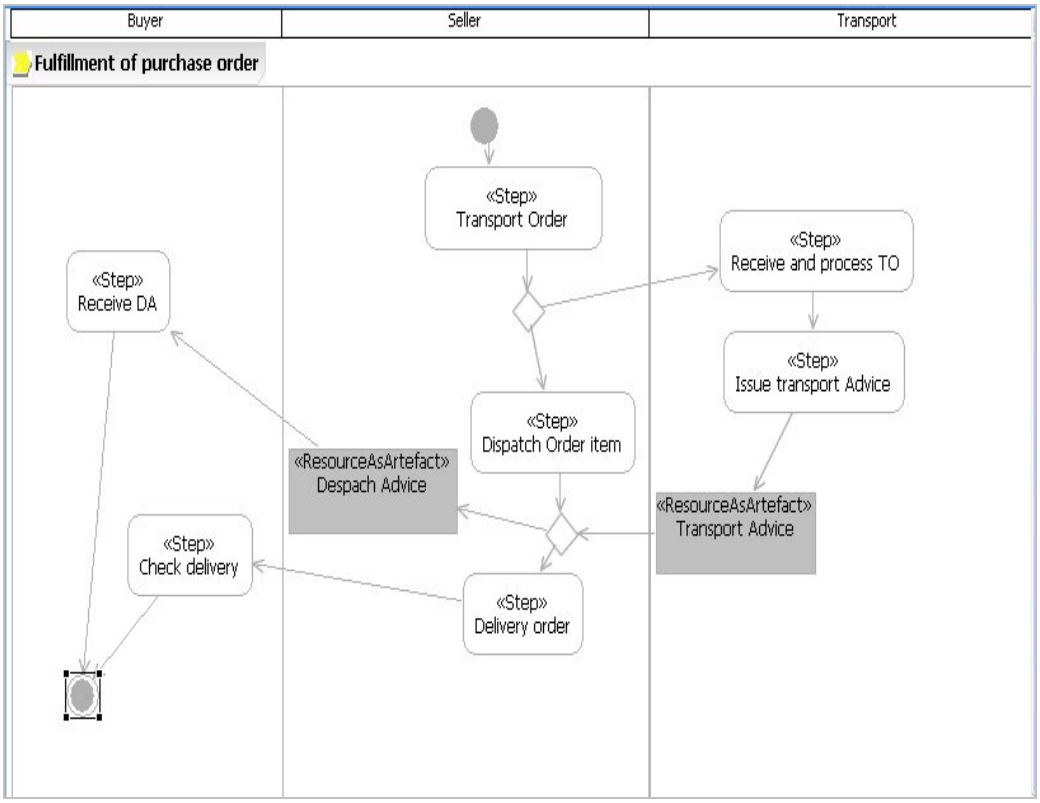

Figure 6.4 Fulfilment of purchase order

Buyer first receives Despatch Advice from the Seller and ordered goods later on. He checks the goods delivered against the advice. The two documents in this process are explained in the table given below.

Name	Description
Dispatch Advice	It is a document sends by the Seller to Buyer to inform him that goods or services ordered have been despatched or delivered to him through transport company.
Transport Advice	Transport company issues advice for the fulfilment of order received from Seller and confirms that good are despatched to buyer.

Table 6.5 Documents sent and received in Fulfilment of purchase order.

6.1.2.4 Invoice and AR/AP

The accounting process starts with sending of invoice by supplier to buyer. Invoice is the request for payment by the seller to the buyer, for the payment of goods or services delivered. Activity diagram for Invoice and AR/AP shown below has many steps involved in invoice to posting to AR/AP ledger, and it has two documents i.e. Invoice and Invoice response. The three steps which are marked light grey will be further divided into sub steps and are explained in the table given below.

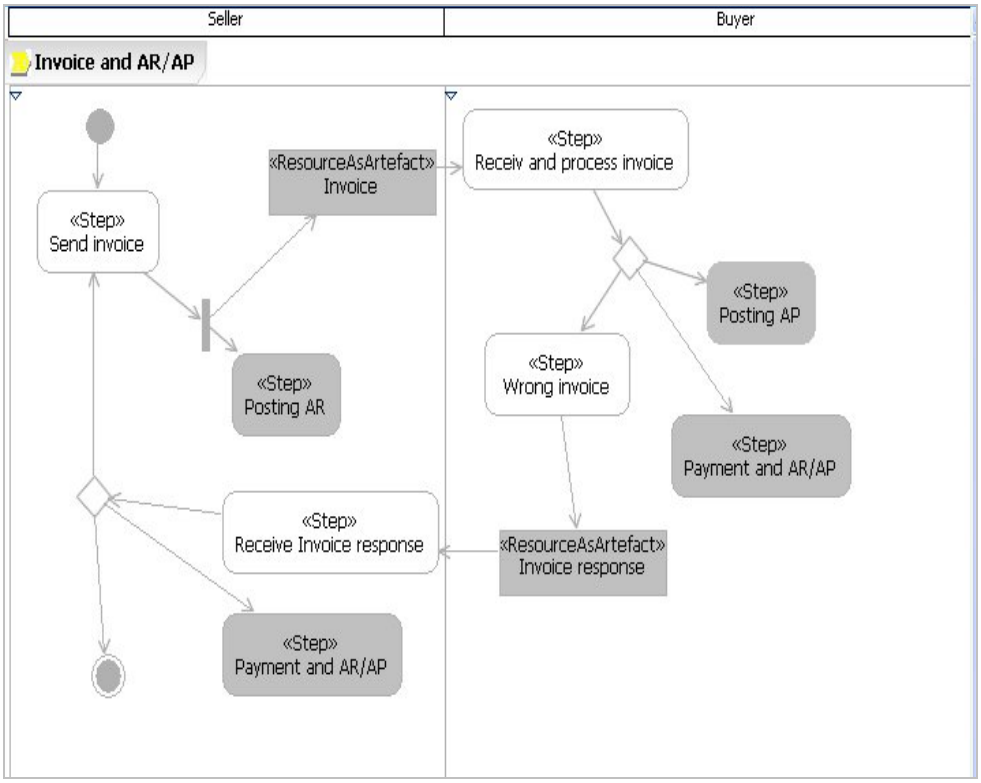

Figure 6.5 Invoice and AR/AP

Invoice and AR/AP is mostly initiated by seller, with the step “Send invoice”. When invoice is sent to buyer, it is also posted to account receivable in the seller’s ledger, by debiting respective buyer party’s account and crediting tax and revenue accounts. When buyer receives the invoice he also posts it in his ledger by crediting the account payable and debiting other relevant accounts.

Name	Description
Invoice	It is sent by the seller to the buyer to request for payment for goods or services.
Invoice Response	Response is send by the buyer to the seller in response to an invoice to inform him of discrepancies in the invoicing process.
Posting to AR	In this step I will explain the process of posting to account receivable in next sub section.
Posting to AP	In this step I will explain the process of posting to account payable in next sub section.
Payment and AR/AP	This is the last step in Order-to-Invoice process with accounting details.

Table 6.6 Documents sent and received in invoice and AR/AP process

If invoice does not match with goods delivered or services provided, the buyer can send invoice response to the seller. He can send new credit or debit invoice, by going through the same processes as mentioned above.

6.1.2.5 Activity diagram for Account Receivable and Account Payable

The main purpose of an Accounts Receivable and Accounts Payable is to keep track of money owed to seller by his customers and the money buyer owe to his suppliers and other creditors. It means that Account Receivable is the value of the issued invoices or shipments that have not yet been paid and the Accounts Payable are the value in shipments or invoices received that have not been paid. In order to keep track of these unsettled payments, an AR/AP facility must support

- Posting of outgoing and incoming invoices, usually originating in separate sales and purchasing systems.
- The registration of outgoing and incoming payments, usually received through a bank or other external settlement agency.
- Matching of the payments against the invoices.
- Identification of business differences with customers and suppliers which have lead to discrepancies between debts and payments.
- Posting of adjustments or corrections that arise as a result of resolving differences with debtors and creditors (customers and suppliers).

Thus, the main classes/objects in AR/AP will be Invoice, Account, Party Accounts, Product Account, Payment, Payment terms, Bank information and Tax. The activity diagram 6.6 is a graphical illustration of all points mentioned above.

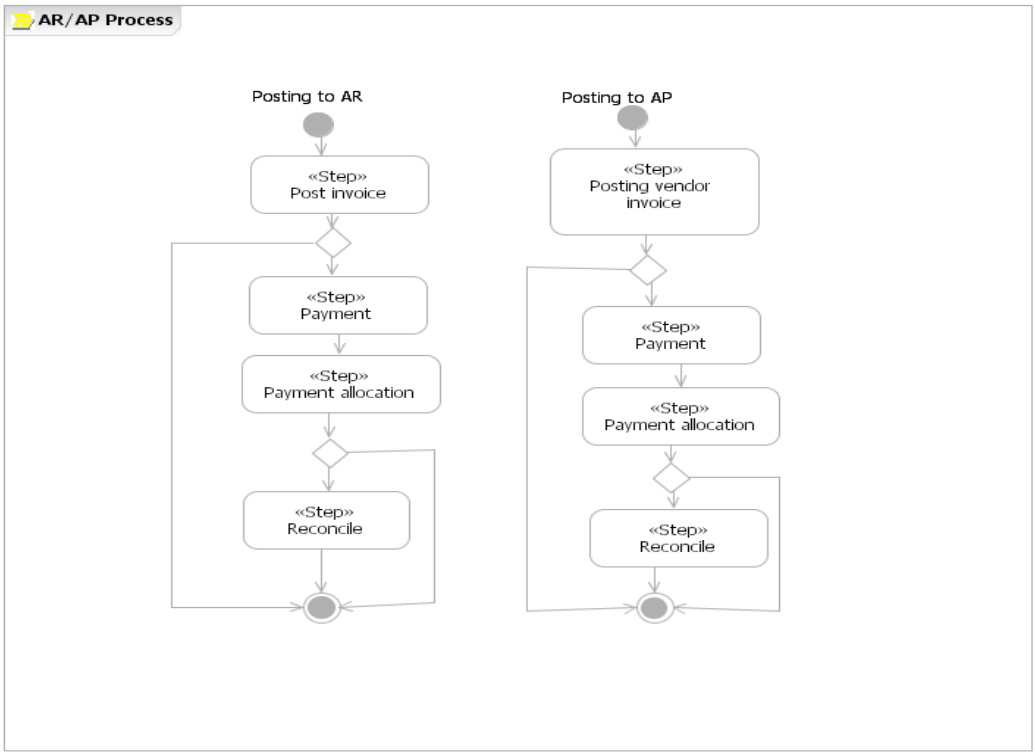

Figure 6.6 AR/AP processes

I will not make activity diagrams for all theses issues, but I concentrate on posting only, other steps can be done similarly. The AR/AP process activity diagram shows the steps from invoice to payment and reconciliation, and each step is explained in table nr 6.7.

Name	Description
Post Invoice	This step starts posting to accounts receivable by debiting AR and crediting trade revenue and tax account.
Payment	If payment is for a single invoice it can be allocated directly to that invoice both in buyer and seller's ledgers.
Payment Allocation	If payment is for multiple invoices or is a partial payment then it should be processed in payment allocation.
Reconcile	It reconciles the invoices paid.
Posting vendor invoice	This step starts posting to accounts payable by crediting AP and debiting other relevant accounts.

Table 6.7 Description of AR/AP processes

The firs two steps i.e. post invoice and post vendor invoice are explained with activity diagrams in next subsections but the other steps mentioned in diagram 6.6 are not explained with activity diagrams. They can be explained similarly when we implement AR/AP system. They are dropped here for simplicity.

6.1.2.6 Post Invoice

The activity diagram for post invoice shows the process of invoice registration to Accounts Receivable and General Ledger. Account Receivable ledger initiates the process after receiving message from Seller entity as explained above.

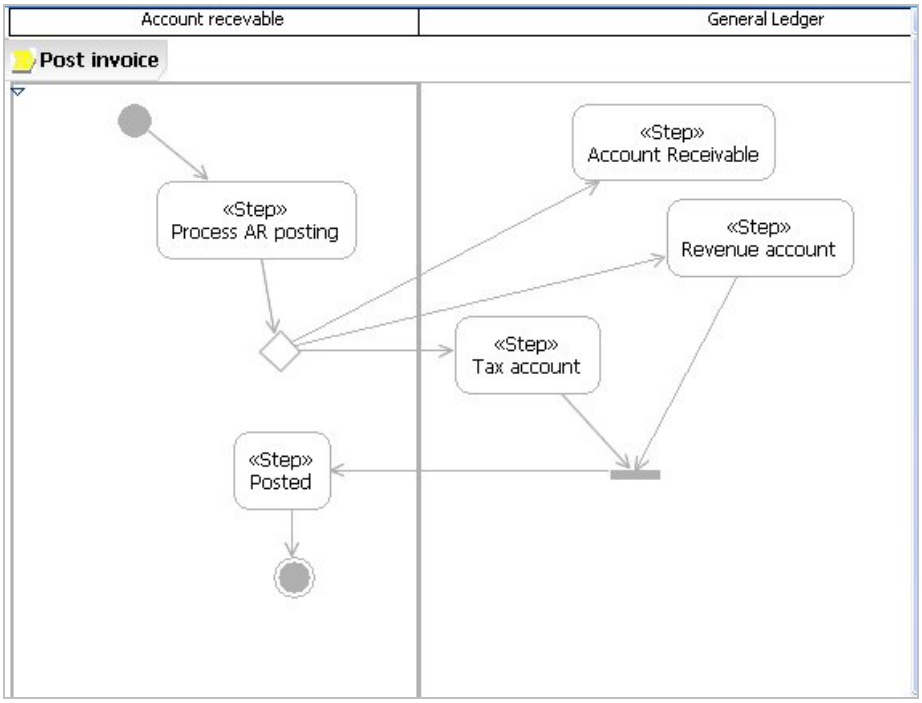

Figure 6.7 Activity diagram post invoice

The invoice is posted for each party in account receivable ledger and the same balance is sent to General Ledger for posting to respective other accounts. This whole process will be posted in one transaction. This step has five sub steps some are explained in table under.

Process	Description
Process AR posting	It initiates the posting of invoice to AR for each party separately.
Account Receivable	A/C in general ledger where to registered outstanding balance of AR in GL.
Tax account	A/C in general ledger where to registered outstanding balance of tax payable to authorities in GL.
Revenue account	A/C in general ledger where to registered outstanding balance of income by sale of goods or services in GL.
Posted	It shows the entries posted to AR.

Table 6.8 Description of post invoice processes

Similarly activity diagram for accounts Payable will be explained in next sub section, as posting vendor invoice.

6.1.2.7 Posting vendor invoice

This step is similar to posting invoice described in above subsection; the activity diagram for post vendor invoice shows the process of registration of invoice to Accounts Payable and General Ledger. Account Payable entity initiates the process after receiving message from Buyer entity as explained above.

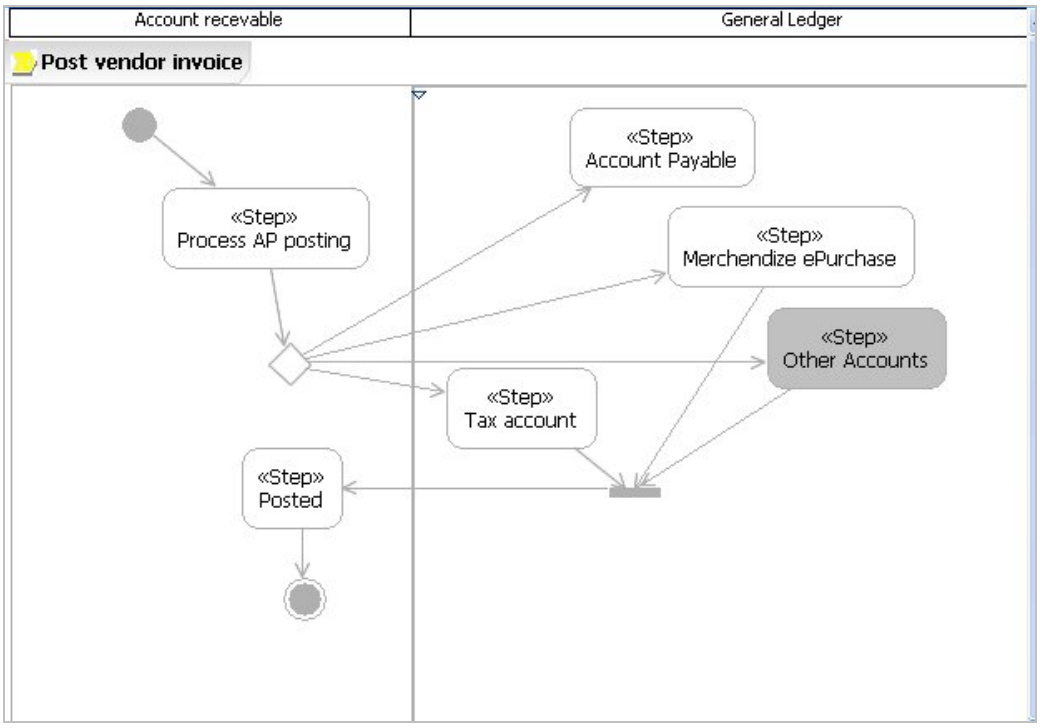

Figure 6.8 Activity diagram post vendor invoice

The vendor invoice is posted for each party in account payable ledger and the same balance is sent to General Ledger for posting to respective other accounts, this posting is also done in a single transaction. This step has five sub steps some are explained in the table given below.

Process	Description
Process AP posting	It initiates the posting of invoice to AP for each party separately.
Account Payable	A/C in general ledger where to registered outstanding balance of AP in GL.
Merchandize ePurchase	A/C in general ledger where to registered outstanding balance of purchase of goods or services by in GL.
Tax account	A/C in general ledger where to registered outstanding balance of tax payable to authorities in GL.
Other Accounts	Other account s like discount, trade discount etc.
Posted	It shows the entries posted to AP.

Table 6.9 Post vendor invoice

6.1.3 The Business Resource Model

The Business Resource Model identifies and defines the main concepts of the domain that are relevant to AR/AP. Information resources are modelled using classes and class diagrams as described in UML-SSS. The AR/AP Business resource diagram shows a simple resource model for AR/AP. I have not listed all classes here because my focus is on the methodology for specification of standards. This document is not a complete standard specification of AR/AP but it shows how to apply UML-SSS to specify a domain standard. Figure 7.9 shows important classes in Business Resource Model.

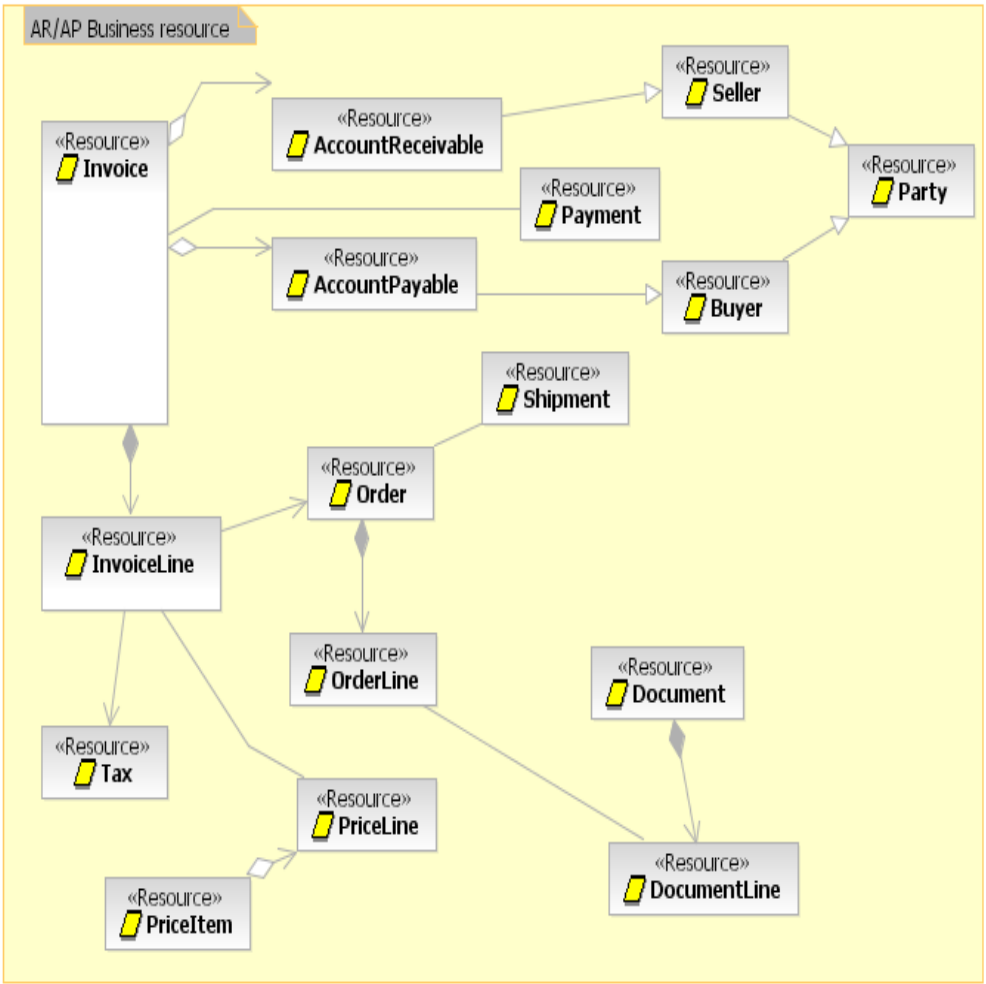

Figure 6.9 Business Resource Model for AR/AP

All the classes in Business Resource Model are describes precisely in table given below. The main classes in business resource model are Party, Invoice, Purchase Order, Shipment, Tax, Item and Document; these classes may contain further aggregation of different classes relevant to each main class.

Name	Description
Party	This class contains general information to identify party.
Seller	It is a seller entity.
Buyer	It is a buyer entity.
AccountReceivable	Suppliers party for payment issues.
AccountPayable	Buyer party for payment issues.
Invoice	General Class for invoice
InvoiceLine	This class inherits from invoice and contains information that allows identifying to which order and order line the invoice line refers.
Order	Purchase order class for identification of order information.
OrderLine	Class for description of purchase order at line level, each order has one or more lines.
Document	Document is a main container for information sent from a sender to a recipient i.e. from buyer to seller or vice versa.
DocumentLine	This is the further description of document at line level, each document contain one or more lines
Tax	Class to mention tax information. It may be further divided into more classes under implementation.
PriceLine	Detail of price at line level for different item.
PriceItem	Class to represent price of items.
Shipment	General class to identify shipment party, may have own packages under implementation.
Payment	Payment Information of invoices.

Table 6.10 Classes in Business Resource Model

The Business Resource Model for internal AR/AP components may have classes like Account, PartyAccount, ProductAccount, Entry and Transaction. Theses classes may be used with or with out AR/AP prefix.

6.2 Architecture Model (PIM)

UML-SSS architecture model consists of following four models, component structure model, component interaction model, interface model and information model. I have chosen two ERP systems to interoperate with each other in buying and selling scenarios. Compiere and SAP are used for this purpose. Compiere is an open source ERP and is available for experiment. I have used Compiere to understand the whole process from order to invoice and payment to accounting entries in respective ledgers. SAP is proposed to use at other end, but I have not used it directly.

6.2.1 Component structure model

It describes the high-level components and their interdependencies.General context diagram for communication between different systems involved in AR/AP is given below. There are six systems or subsystems which are involved in AR/AP interoperation. The two ERP's are the main buyer and seller systems where one let us Compiere sends purchase order to SAP and the SAP respond it and sends message to Transport Company who has his own subsystem to handle transport orders. After the transport information SAP sends invoice to Compiere. It may happen that some ERP does not have AR/AP in this case it will interoperate with external AR/AP to fulfill the accounting entries after creation of invoice.

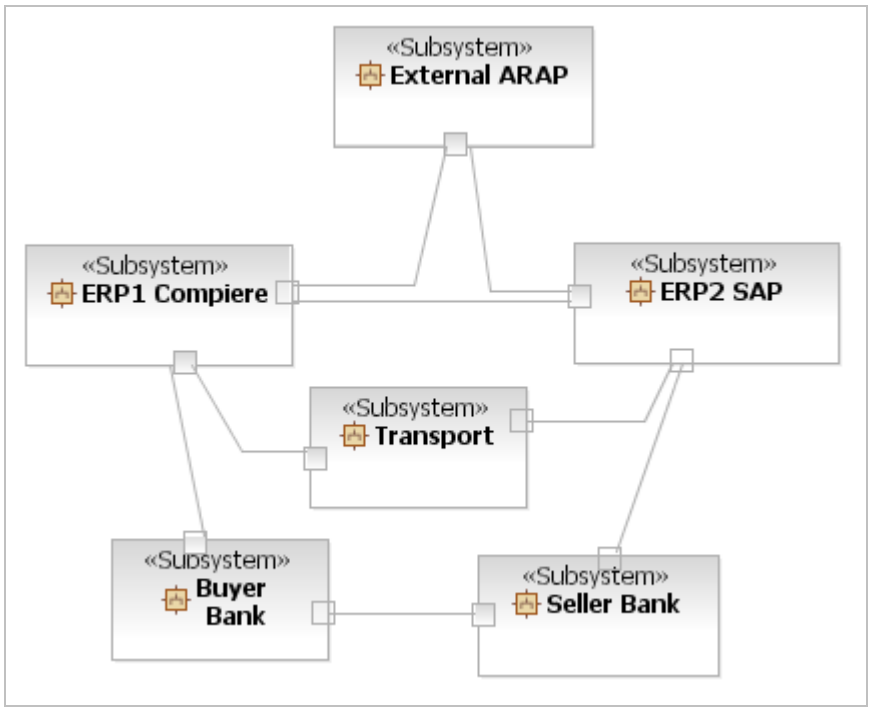

Figure 6.10 AR/AP Component Interaction

The Buyer and Seller banks are involved when payment of an invoice is made by the buyer i.e. SAP communicate with its bank for payment to Compiere. Buyer bank sends information to seller bank which further sends message to seller system. The communication is proposed through ports, where each system provides ports for interoperation.

Further we will go into details of different parts of ERP1 i.e. Compiere, especially those parts which are concerned with AR/AP.

6.2.1.1 Details of ERP1

The important components in an ERP1 are GL, AR/AP, Selling, Purchasing and Inventory components. It may have modules like Payroll, System Management, Cash management and many others but these are not related to AR/AP.

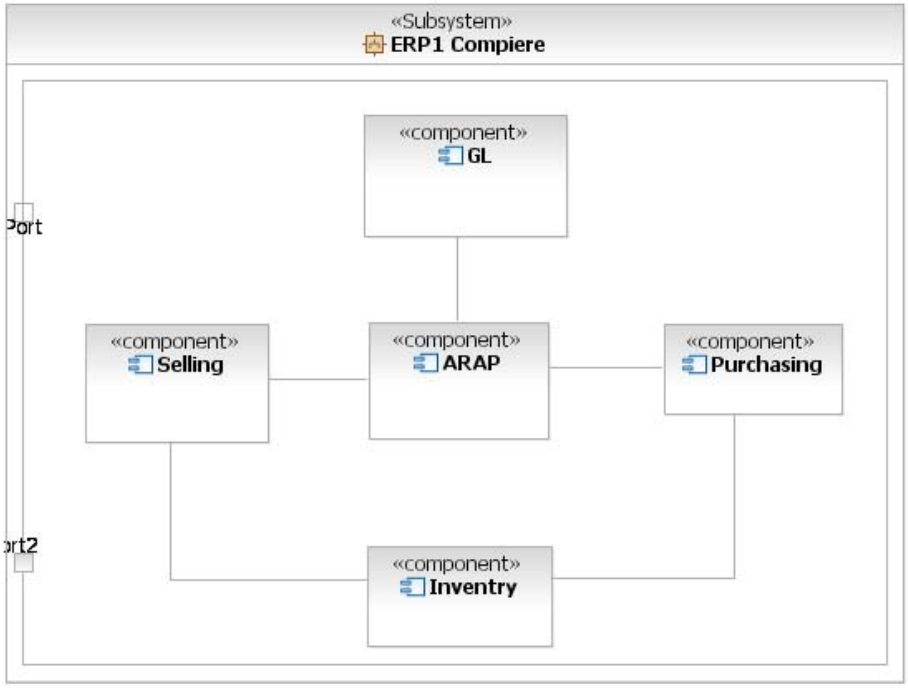

Figure 6.11 Compiere component overview

Figure 6.11 shows the internal interaction of components and external interaction with other systems through two ports. AR/AP interacts with minimum three components Selling, Purchasing and General Ledger. AR/AP interaction will be further specified with required interfaces and provided interfaces in next sub sections but first we see the internal structure of AR/AP.

6.2.1.2 AR/AP Component

Component diagram for AR/AP shows the communication of AR/AP with other components within ERP and outside ERP along with important internal structure of AR/AP. The explanation of internal parts of AR/AP is given below.

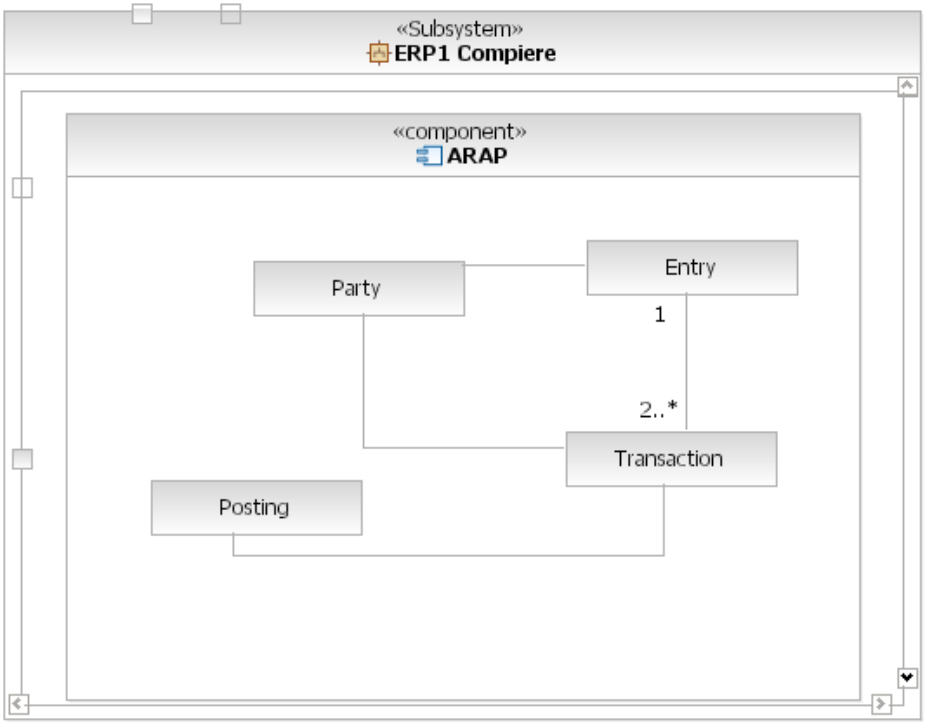

Figure 6.12 AR/AP components

Entry is a discrete amount, together with its associated reciprocal party, transaction date, description, expected settlement date and method, and XBRL type or account code.

Transaction is a balanced set of two or more entries (debits and credits) to a general ledger or AR/AP ledger. Account is an attribute of a transaction entry (row), which classifies that entry with any valid value in the Chart of Accounts list. The values in the chart of accounts may be statutory classifications for tax or financial reporting, but are usually short or mnemonic values which support additional purposes in workflow, transaction validation, reporting, etc.

Posting is the act of committing an individual transaction consisting of a balanced set of two or more entries (debits and credits) to a general ledger or AR/AP ledger.

Party is an identifier to specify the parties involved in the given transaction.

6.2.2 Component interaction model

Interaction model describes the interactions between the high-level components. Buying and selling process often start with quotations, buyer sends request for quotation to Seller Company and it sends quotation to Buyer Company which can be binding or not it depends on contract between involving partners. The next step in purchasing and selling is Purchase Order send by Buyer Company to Seller

Company and Sales Order vice versa. The following sequence diagram is a simplified process based on the activity diagrams written in section 6.1.

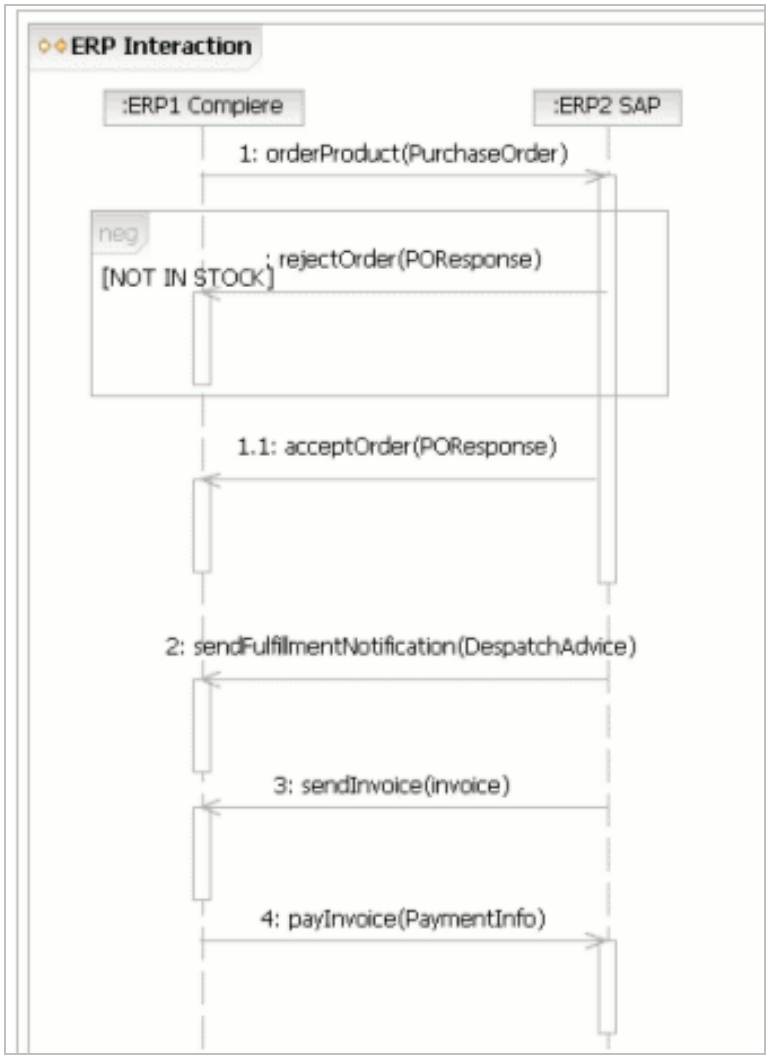

Figure 6.13 Sequence diagram Component interaction

In the next step we go into further detail with transportation of goods or services which may go through the Transport Company. It involves three objects i.e. Buyer Company, Seller Company and Transport Company.

Here I have used notations from UML 2.0 some “ref” and “alt” to cover maximum information in one sequence diagram. After purchase order, sales order and transportation Seller Company sends invoice to Buyer. Posting of invoice will affect the accounts receivable in Seller Company’s AR Ledger and similarly AP Ledger of Buying Company it is explained in sequence diagram ARAP-posting. Last step is payment which involves three objects i.e. Seller and Buyer Bank(s) in addition to Buying and Selling Companies it can be explained in a separate sequence diagram Payment Invoice. Payment will reverse the entries posted

during invoice in AR/AP. The details of posting to AR/AP will be described in sequence diagram “ARAP-posting” which is decomposition of both ERP systems but for simplicity I have decomposed only one of them.

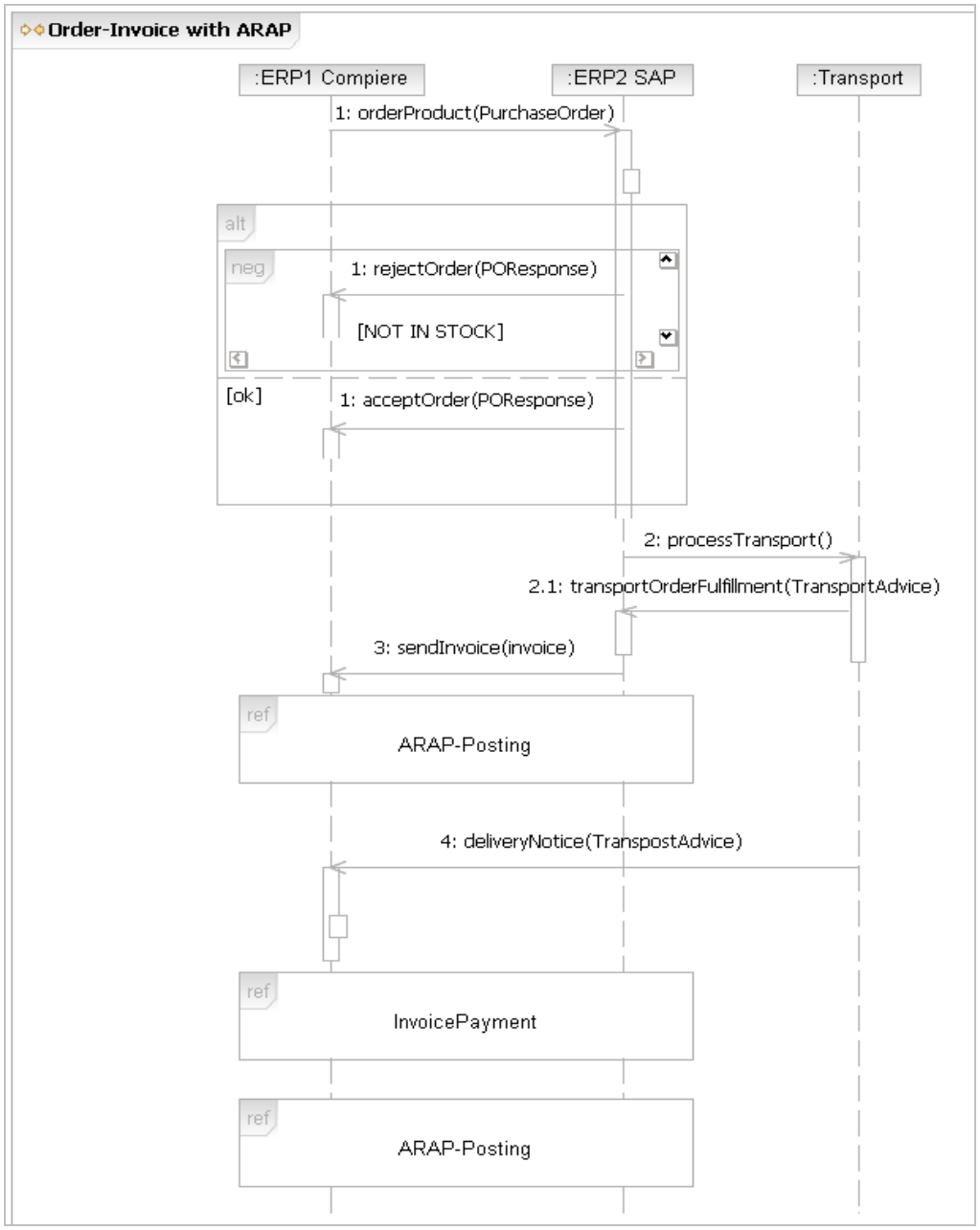

Figure 6.14 Sequence diagram for Order-to-Invoice with AR/AP

Here are the details of invoice posting to AR. Seller component sends message to ARAP component for posting to AR. ARAP get the account receivable for respective party and other relevant accounts and oppdateAccount updates balances in all relevant accounts.

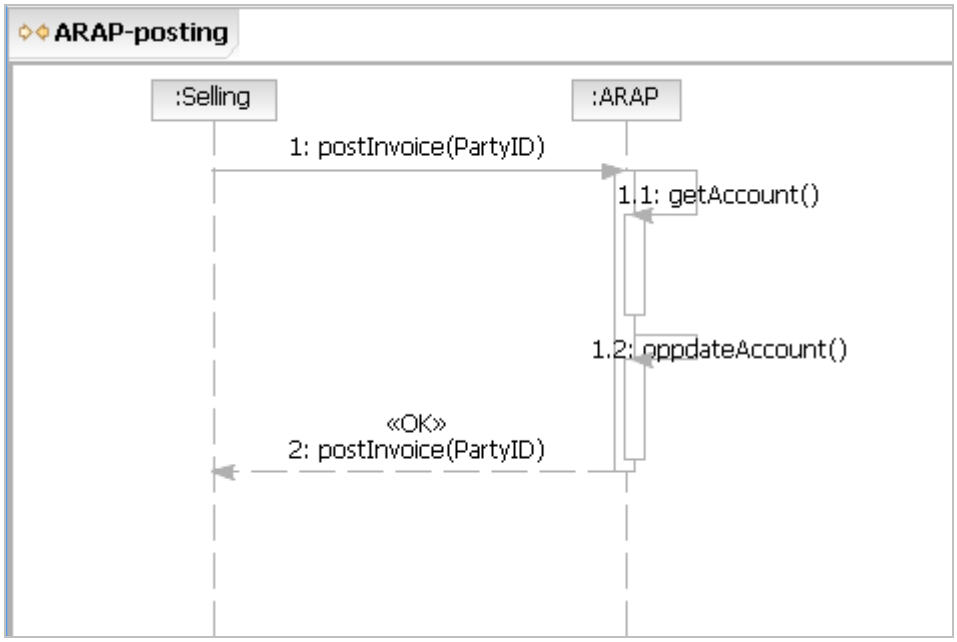

Figure 6.15 Sequence diagram ARAP posting

6.2.3 Interface model

Interface model describes the details of the component interfaces, i.e. their operations and detailed behaviour. My focus is on Accounts receivable and Accounts Payable, therefore I will discuss only interfaces required and provided by AR/AP to other components like Selling and Purchasing, and General Ledger. But I will not illustrate all interfaces here I mention some examples of interfaces in AR/AP. Three Interfaces provided to Selling component are shown in figure 6.16 i.e. RetrieveAccount, PostToARAP and NewAccount.

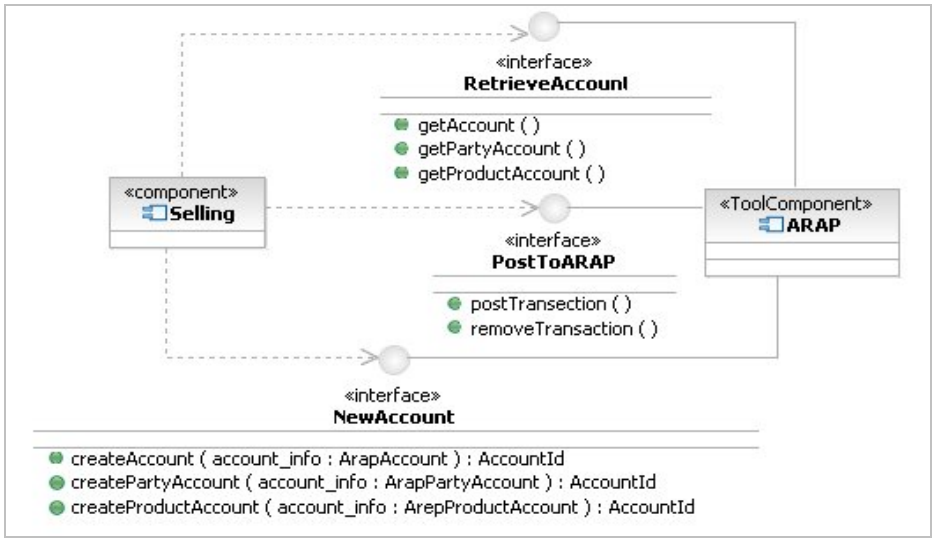

Figure 6.16 Interfaces provided by ARAP to Selling component

When Selling component sends messages to AR/AP component for posting to AR for particular Party and Product account, interface RetrieveAccount checks the respective accounts in AR/AP ledger. If accounts already exist, RetrieveAccount call the PostToARAP to complete the transaction, otherwise it calls to NewAccount for creating new account and then call to PostToARAP interface for completion of transaction. The description of interfaces is given in table given below.

Interface	Description
RetrieveAccount	The RetrieveAccount interface supports to retrieve the chart of accounts, including customer, supplier, and product accounts, in the ledger for the current company.
NewAccount	The NewAccount service manages the accounts in the ledger, facilitating the customization of the chart of account selected when the ledger was created, including the creation, modification, and deletion of accounts for customers, suppliers, and products.
PostToARAP	The PostToARAP interface is used for entering transactions into the AR/AP facility.

Table 6.11 Interfaces between ARAP and Selling component

Similarly another example of interfaces is the interfaces between GL and AR/AP. It is shown in diagram number 6.17 given below.

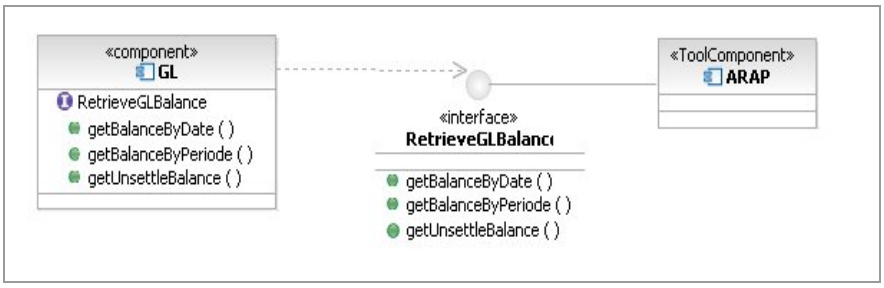

Figure 6.17 Interfaces between GL and ARAP

6.2.4 Information Model

The information model describes the information structures that are conveyed through component interfaces. The information model may be included in the Interface Model but I have chosen it some separate model. The Component Information Model should contain a set of UML class diagrams representing the types/classes with attributes and relationships. The Component Information Model is platform independent, i.e. we use the platform-independent data types mentioned in section 5 and omit detailed technology details.

Let us see a general class diagram for AR/AP in a wider perspective of Order-to-Invoice scenario. It shows the interaction between different objects. These classes can further be divided into subsystem or component by grouping them according to functionality. I will not use this diagram to generate PSM but instead use only class Account with Arap prefix, and data types required for account entities.

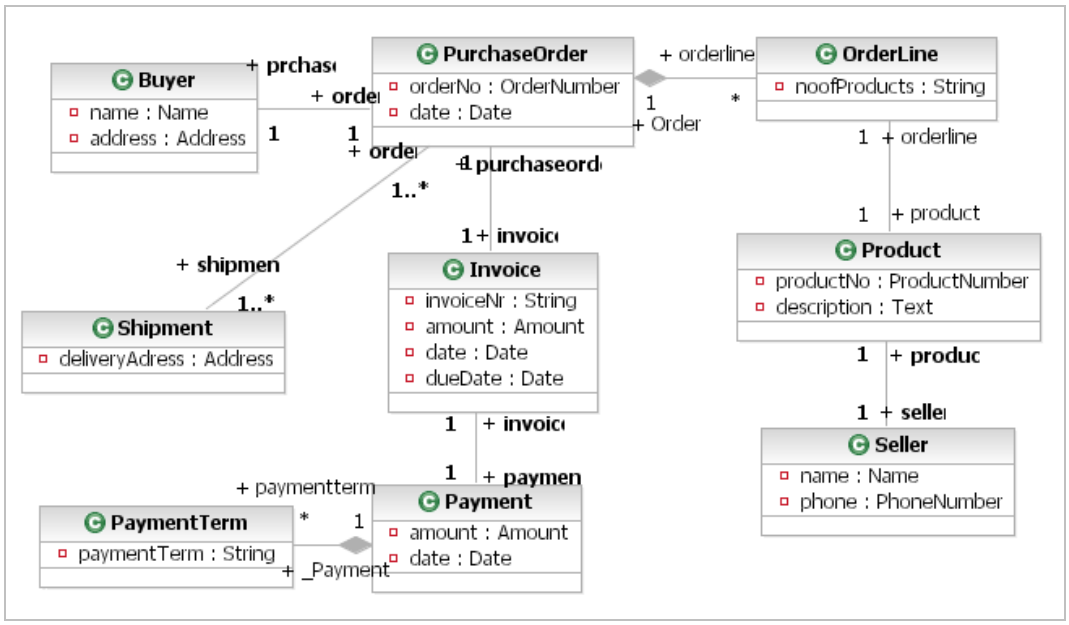

Figure 6.18 AR/AP class diagram

Data Types - As mentioned in chapter 5 I have decided to use data types from existing standards like Core Component from ebXML and data types used in old AR/AP submission. It means that I will use some basic types and derived types. Figure 6.19 shows some important derived data types which I will use in AR/AP account classes; it can be shown in UML class diagram with stereotype.

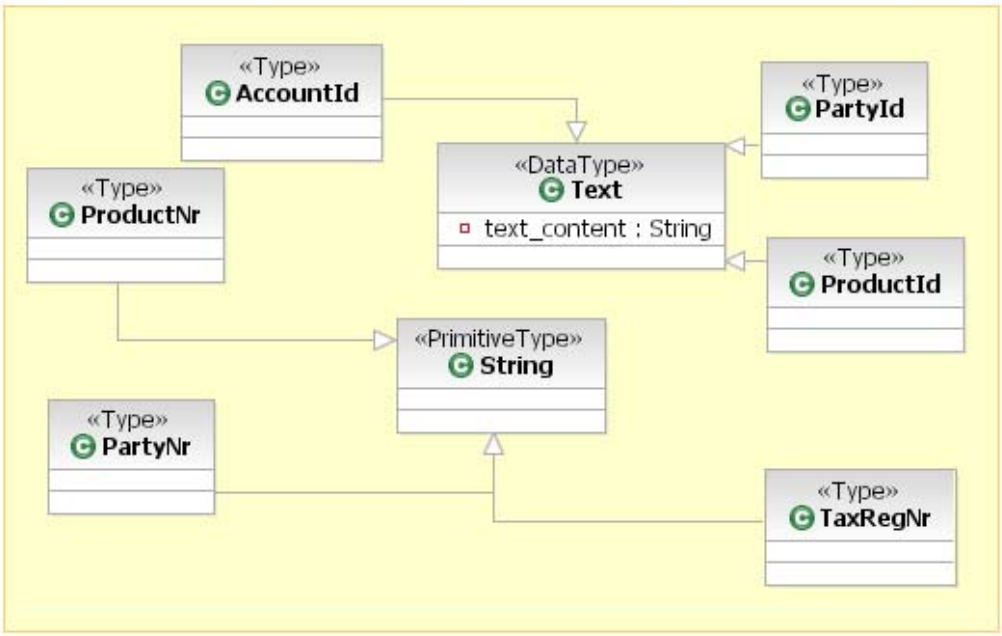

Figure 6.19 AR/AP Data types

Table given below explains the semantic of main Data Type in above class diagram.

Type Name	Description
PartyId	It is a legal person which can be a customer or a supplier. A PartyId is tied to customer and supplier accounts and is used to look up information in an external Party service.
ProductId	ProductId identifies a product, i.e. something which can be sold to a customer or bought from a supplier.
PartyNr	It is unique identifier of party.
ProductNr	It is unique identifier of product
TaxRegNr	Tax registration number of involving parties.

Table 6.12 Description of AR/AP Data Types

Accounts are also core entities of an ARAP. We define a set of different accounts to be used for different purposes. We use enumeration class for this purpose. The AccountType enumeration defines the different account types to be allowed in the ARAP Facility.

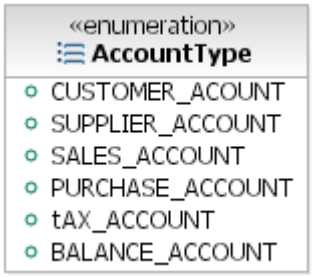

Figure 6.20 Enumeration data type

The main accounts used in AR/AP are illustrated in diagram 6.21, where the ArapAccount type is used to represent all of the account types except CUSTOMER_ACCOUNT, SUPPLIER_ACCOUNT and PRODUCT_ACCOUNT. A party is related to the ArapParyAccount through the party_Id attribute.

The ArapPartyAccount type is a logical subclass of ArapAccount, used to represent accounts that relate to a party, SUPPLIER_ACCOUNT and CUSTOMER_ACCOUNT.

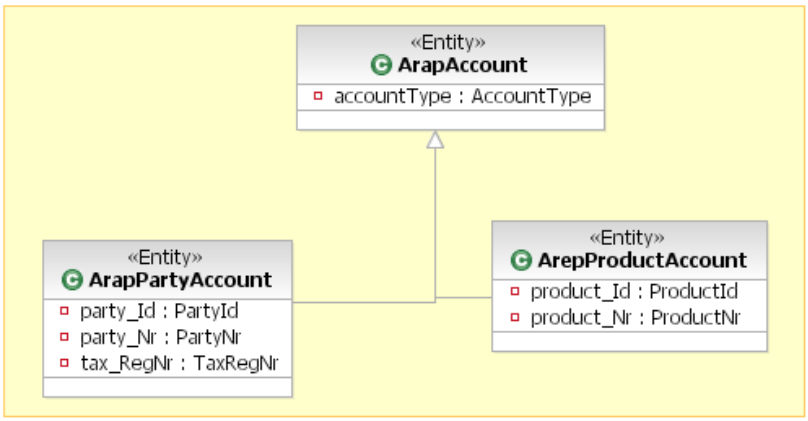

Figure 6.21 AR/AP Accounts

The ArapProductAccount type is a logical subclass of ArapAccount, used to represent products in the ledger, hence covering the PRODUCT_ACCOUNT account type. A product is related to the ArapProductAccount through the product_id attribute.

7 Platform specific Modeling

This chapter contains explanation of two mappings or transformations, first is PIM to PSM and second is PSM to Code. PIM to PSM mapping is 1xPIM = n x PSM. It means that we derive many PSMs from the same PIM. I have made different platform specific models from the same platform independent models. I have transformed only some part of PIM into PSM for simplicity and they are shown in this chapter, other parts of AR/AP can be similarly transformed. The focus is on the methodology or process involved in PIM to PSM transformation. I have used two working tools for transformation purpose i.e. OptimalJ and Rational Software Modeler.

7.1 PIM to PSM Mapping

In PIM to PSM mapping I have used diagrams 6.16 and 6.21. The Platform Specific Models generated are EJB, WEB (presentation) and Web Services. PIM to PSM mapping varies from tool to tool; some tools with integrated transformation rule-sets might generate more specialized code than others. If one uses MOF, QVT or ATL for model to model transformation then one can get more control on transformation rules. It is clear that two different tools might produce different PSM mappings even for the same platform. It is thus not sufficient to standardize only the PIM; also the resulting PSMs with code/interfaces need to be standardized.

7.1.1 EJB PSM

Diagram 7.1 shows different components generated from interfaces NewAccount, PostToArap and RetreiveAccount and AR/AP accounts classes mentioned above. Let us make a EJB platform specific model from diagrams 6.16 and 6.21 shown in PIM model, this PSM for EJB is generated by OptimalJ. OptimalJ generates all components needed by EJB from given domain or PIM model. The components generated are EJBSessionComponent, EJBEntityComponent and EJBBusinessLogicConstraintHandlerComponent.

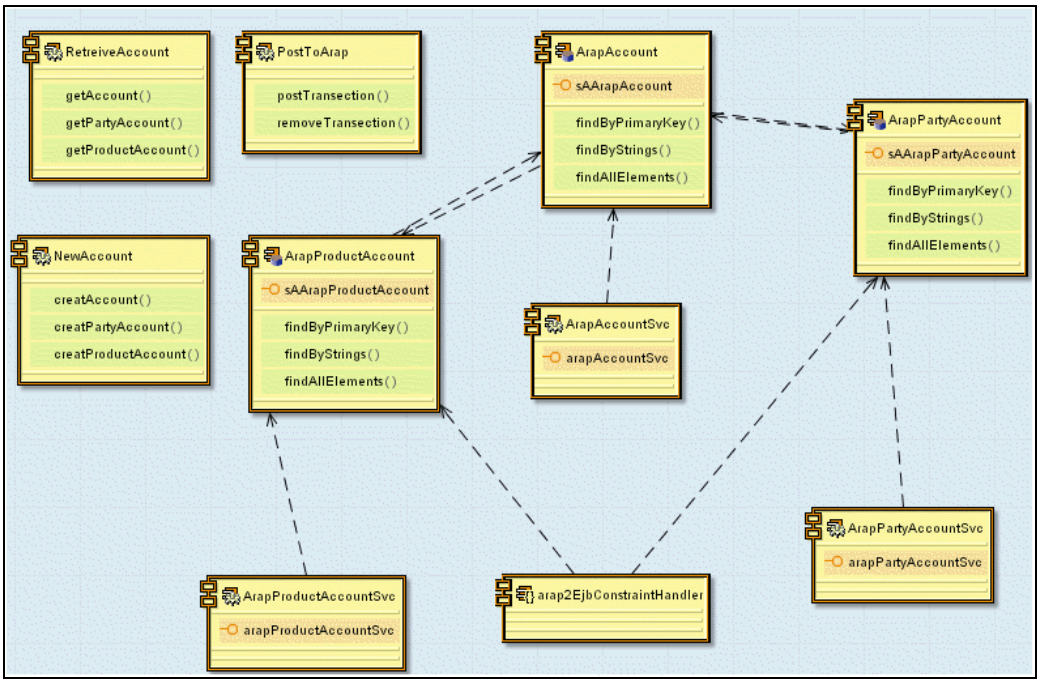

Figure 7.1 EJB Model from OpmtimalJ

OpmtimalJ puts some find methods and serving attributes to components generated for all classes. It provides the opportunity to add the business method before generating codes.

7.1.2 WEB PSM

It is java or EJB based presentation of web model generated by OptimalJ from domain model (PIM). Figure 7.2 illustrates the WebComponents with different WebFlow methods.

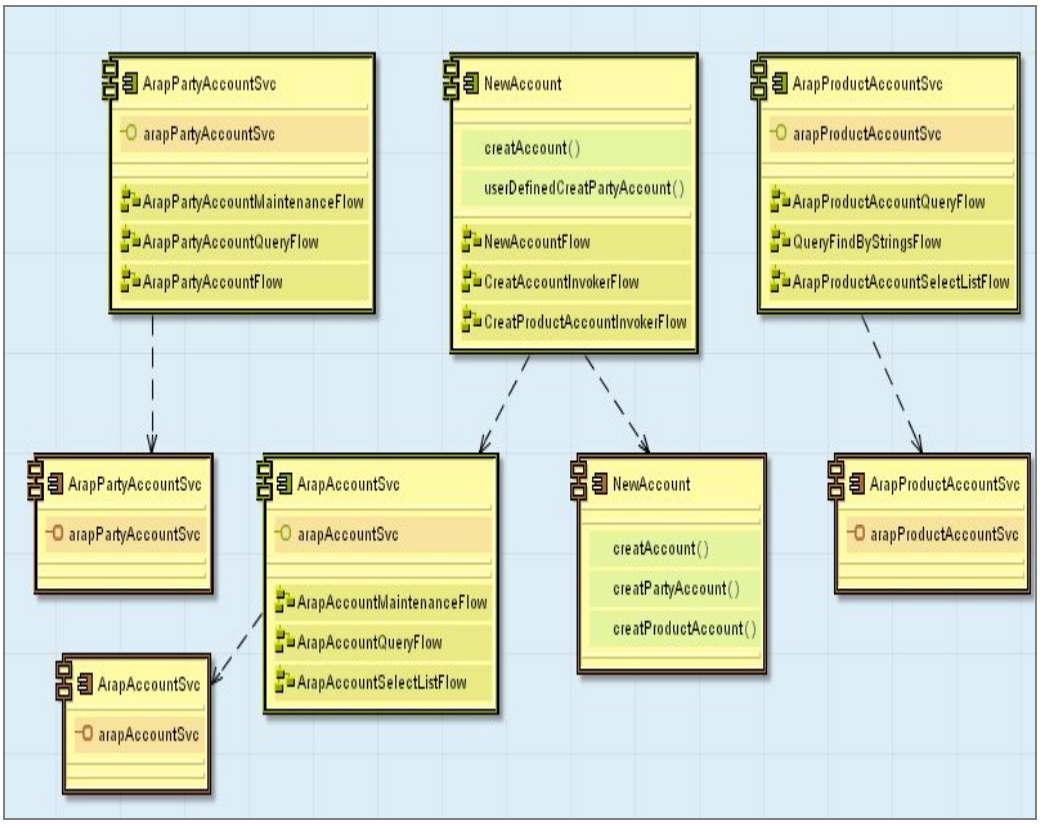

Figure 7.2 WEB (presentation) PSM from OptimalJ

The diagram 7.3 shows the detail of all components and attribute for Web PSM. It shows the WebComponents with web action, web pages, web serving attribute and web data types. The web data types are converted from PIM data types.

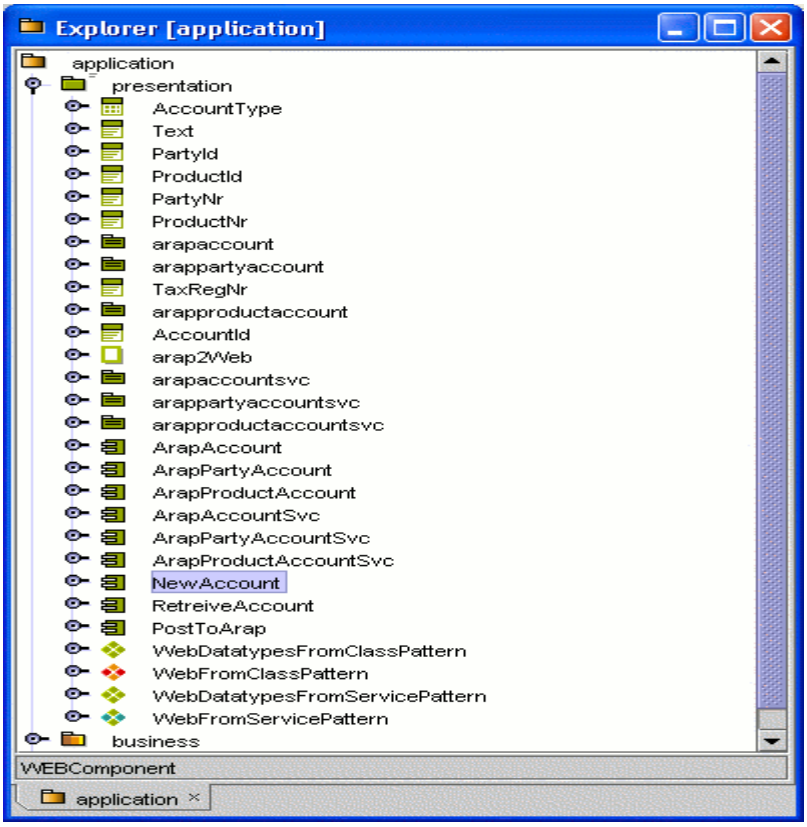

Figure 7.3 All components of WEB (presentation) PSM

7.1.3 Web services PSM

The platform specific model for web services is generated with the help of two tools i.e. OptimalJ and Rational Software Modeler. The diagram 7.4 shows the Web service model generated with help of OptimalJ where each EJBSessionComponent has a WebServiceComponent, it shows the components for the interface NewAccount. It is not a PIM to PSM mapping but it is generated from EJB model, it means that it is a PSM to PMS mapping, OptimalJ does not support direct PIM to PSM mapping for the Web Services.

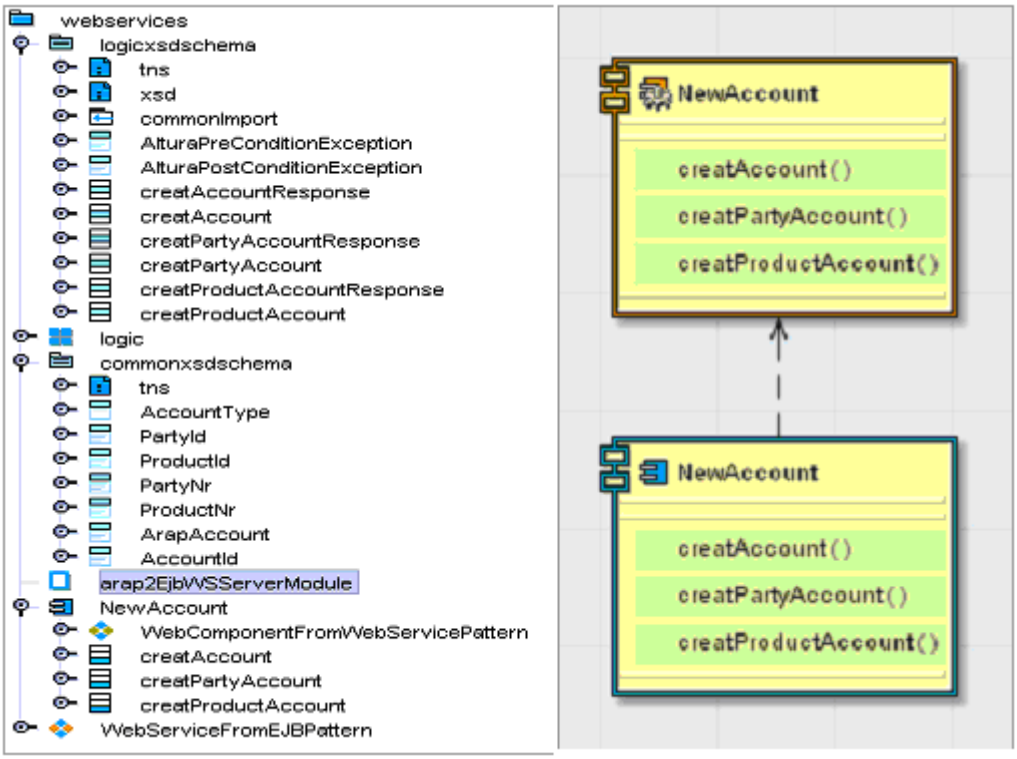

Figure 7.4 Web Services PSM from OptimalJ

Web service platform specific model generated from Rational Software Modeler with the help of ATL transformation is given below. In ATL we can transform model to model, package to package, interface to interface and operation to operation etc. We take a UML model as input and transform it into UML model some output in model to model transformation, similarly other transformations are made.

```
rule Model2Model {
      from srcMdl : UML2!Model
      to trgMdl : WSUML2!Model (
            name <- 'PlatformModel',
            ownedMember <- srcMdl.ownedMember
      )
}
```

Figure 7.5 Model to Model transformation with ATL

If we apply this ATL transformation to interfaces shown in figure 6.16, the Web Service Platform will look like the diagram 7.6. Each interface gets an implementation class with similar methods. There are no major differences between two Web Services PSM, both tools has their own notation, OptimalJ has web service component for each interface and RSM has implementation class for each interface.

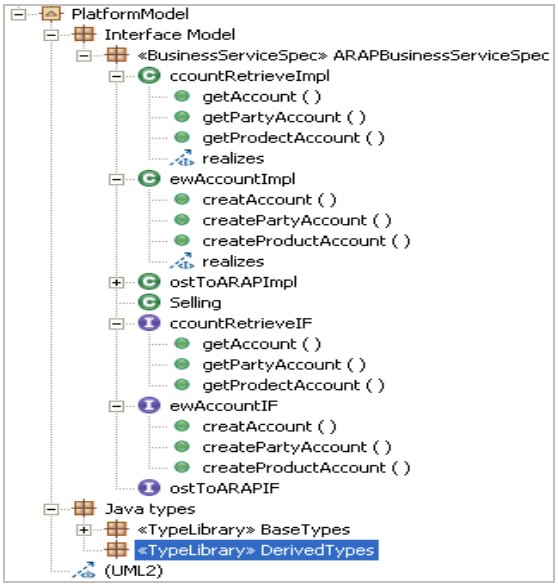

Figure 7.6 Web Service PSM from RSM

7.2 Mapping PSM to Code

When we have a PSM, code generation of interfaces is not a large task. Mostly all working tools which support PIM and PSM generate code from the PSM model. I have generated java codes for EJB, Web Presentation and Web Services.

7.2.1 EJB Code

The java codes are generated for EJB from the OptimalJ. As mentioned earlier OptimalJ generates some tool specific Java information, the diagram given below shows the Java codes for different classes in PSM. Codes for EJB remote and home interfaces are also generated.

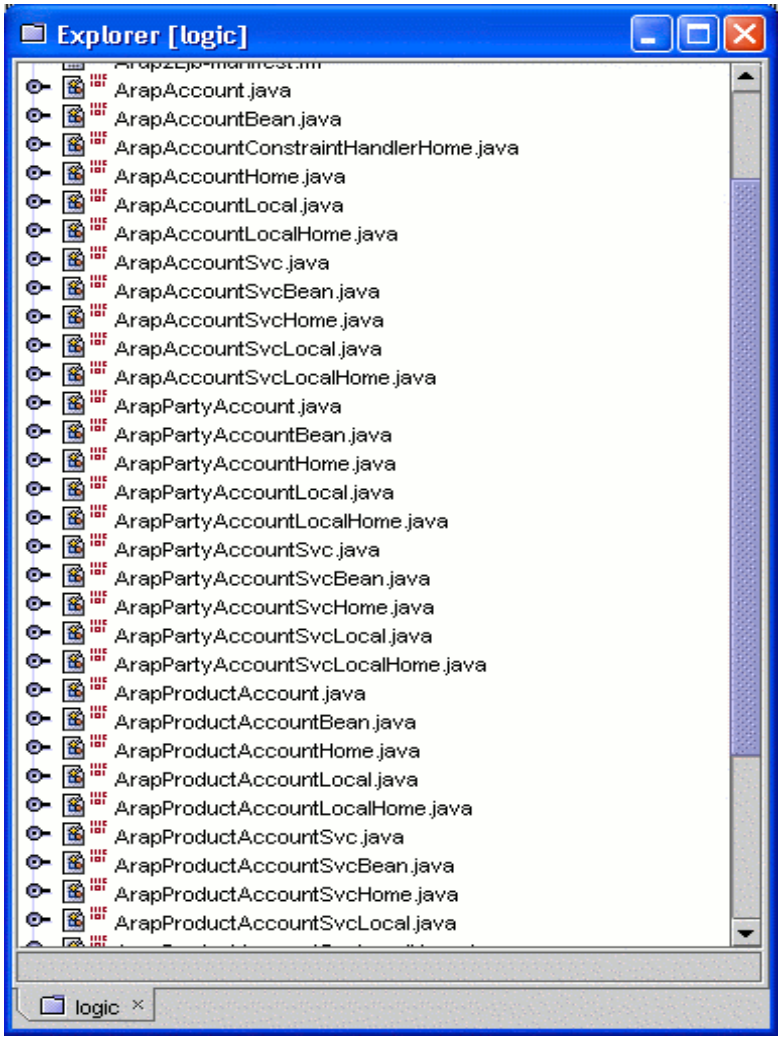

Figure 7.7 Java codes for EJB.

If we see the code for interface NewAccount, it will be en extension to EJBObject, The code generated by OptimalJ for a Session Bean Remote Interface are shown in figure 7.8.

```
package arap2.application.business.logic;
import java.rmi.*;
import javax.ejb.*;
import java.util.Collection;
import java.util.Map;
import com.compuware.alturadev.ejb.*;
import com.compuware.alturadev.application.*;
import arap2.application.business.common.*;
import arap2.application.business.common.*;

/**
* Session Bean Remote Interface
*/
public interface NewAccount extends EJBObject {

   public arap2.application.business.common.AccountId creatAccount(ArapAccountKey account_info)
      throws com.compuware.alturadev.application.AlturaPreConditionException,
com.compuware.alturadev.application.AlturaPostConditionException, RemoteException;
   public arap2.application.business.common.AccountId creatPartyAccount(ArapAccountKey
account_info)
      throws com.compuware.alturadev.application.AlturaPreConditionException,
com.compuware.alturadev.application.AlturaPostConditionException, RemoteException;
   public arap2.application.business.common.AccountId creatProductAccount(ArapAccountKey
account_info)
      throws com.compuware.alturadev.application.AlturaPreConditionException,
com.compuware.alturadev.application.AlturaPostConditionException, RemoteException;
}
```

Figure 7.8 Java code for interface NewAccount

The Code generated from Rational Software Modeller with help of ATL transformation for same interface is very simple and easy to understand. The difference of auto mapping and manual mapping for generation of code is enormous.

```
package org.sintef.no;
import java.util.Collection;
import java.io.Serializable;

/**
 *  Generated class NewAccount
 *  @author Bahadar Khan
 */
public interface NewAccount {
      /*
     * Operations
     */
    public void creatAccount ();
      public void createPartyAccount ();
      public void createProductAccount ();

} // End of interface NewAccount
```

Figure 7.9 Code generated with ATL transformation RSM.

7.2.2 Web Code

The web codes are similar to EJB code. They are also java codes for web interfaces and classes given in PSM for Web. The figure 8.8 shows the code for some classes in Web PSM.

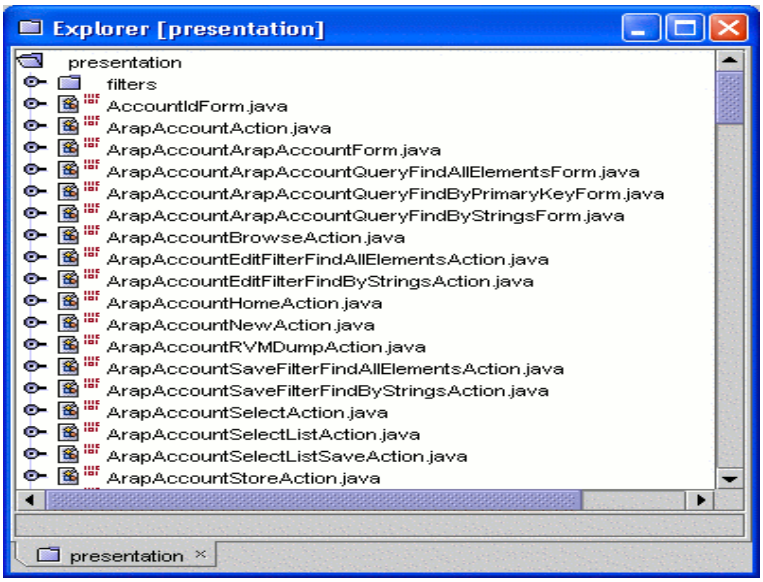

Figure 7.10 Java code for Web PSM

7.2.3 Web Service code

The process of code generation is more or less trivial after the PIM to PSM. I have generated codes with help of both work tools. The code generated by OptimalJ are mostly java codes or in other words generated from EJB. The Rational Software Modeler does not generate code automatically, but we can generate the code with help of ATL and MOF transformation.

I have used the ATL transformation for the generation of Web Service codes. An example of transformation of interface to code is given below, we transform UML package into interface package and for each interface the massages, data types, port types and binding services are transformed.

```
uml.Package::interfacePackages () {
  if (self.getStereotype() = business_service_stereotype){
    self.ownedMember->forEach(i:uml.Interface) {
      i.wsdlMessages()
      i.wsdlPortType()
      i.wsdlBindings()
      i.wsdlService()

    }
```

The codes generated in XML format for three methods in interface NewAccount are given below.

```
<wsdl:message name="creatAccount">
   </wsdl:message>

   <wsdl:message name="createPartyAccount">
   </wsdl:message>

   <wsdl:message name="createProductAccount">
   </wsdl:message>
```

7.3 Summary

The experiment in this chapter has shown that there are many ways of getting from a PIM model to different PSM models and corresponding code/interfaces.

Different transformation tools need first to be able to understand the UML-SSS models, and then to have a rule set to transform this to a PSM or directly to code/interfaces. Since these rules can be different for different tools, even for the same PSM, it will be necessary to standardize not only the PIM, but in addition also the corresponding PSMs for chosen platforms.

8 Analysis and Evaluation

In this chapter UML-SSS will be analyzed and evaluated comparatively with the existing domain standard specification languages IDL and UML-UMM. The second part of thesis i.e. Finance AR/AP will be evaluated according to requirements presented in section 3.2.

8.1 Evaluation of UML-SSS

UML-SSS is a UML based domain standard specification language and will thus have several aspects where it is similar to UML-UMM. The evaluation uses the requirements presented in section 3.1 as a framework.

Requirement		**UML-SSS**
R1	Formal Language	+/-
R2	Specification of CIM and PIM and PSM models	+
R2a	Structure	+
R2b	Behaviour (Process/Services)	+
R2c	Constraints	+/-
R3	Ease of use – graphical notation.	+
R4	Platform independent data types.	+
R5	Computer processability.	+
R6	Non functional aspects like performance, QoS, error handling, security issues, usability, reliability, availability, adaptability, supportability.	+/-
R7	IT standard and continuous development.	+
R8	Support from existing working tools.	+/-
R9	Methodology and process description.	+

Table 8.1 Evaluation of UML-SSS

The evaluation of UML-SSS is summarized in table 8.1. UML-SSS introduced a structured way of modeling domain using a set of models, where the MDA approach helps in keeping the PIM models consistent. It also helps to understand the several viewpoints of domain with different model. It means that the

requirement **R2** is fully met. Requirement **R2c** is not fully met because of limitations of UML in constraint. UML 2.0 has a support for constraints within models, but we can still get problems with tools and they are not fully implemented.

Requirements **R3, R4** and **R5** are fully met. UML is becoming de facto standard in modeling and is not difficult to use. Many working tool has support for the graphical notations provided by UML.

Requirement **R6** is partially met. We mentioned some of the non functional requirements in Business Model under Scoping Statement as risk analysis but implementation viewpoint of this requirement is not mentioned in PIM.

Requirement **R7** and **R9** are fully met, but **R8 only** partially met. We have various tools which supports MDA techniques and transformation from PIM to PSM. But it is difficult to define and find support for transformation from CIM to PIM. Existing tools will have pre-existing constraints to what a PIM model should look like, and they might have chosen different transformation rules and philosophies for the transformations to PSM and code/interfaces.

We can make PIM to PSM transformation our selves with the help of QVT or ATL and MOF. Similarly we can make transformation from model to model at any level with help of MOF. It seems that the best approach for standardization purposes is to also standardize the corresponding PSMs with their associated code/interfaces that have been generated. Also the transformation rules that have been used should be described. Ideally it should be enough to standardize the PIM and the transformation rules, but in practice there are so many details involved, that we recommend also to standardize the corresponding PSM code/interfaces.

8.2 Evaluation of Example, UML-SSS applied to Finance AR/AP ref. Requirements

The evaluation of AR/AP example is based on the requirements stated in chapter 6 and the UML method for Standards Service Specifications is applied to these requirements for Accounts Receivable and Accounts Payable. It is not a complete specification of AR/AP but en example to test implementation of domain standard specification language. Table 8.2 shows the summary of requirements and their implementation with UML-SSS.

The requirement **Req.1** met in full. The example is full implementation of MDA technique with Business Model (CIM) and Platform Independent Model and Platform Specific Model. Architecture Model illustrated in subsection 6.2 is PIM for AR/AP. PSM is explained in chapter 7 and has EJB, Web and Web Services platforms models.

Req.2 met in full. I have not explained and specified each of the operations for each of interfaces comprising the complete AR/AP facility, including preconditions, post conditions, and exception conditions. But I have mentioned

only three interfaces to explain the methodology; similarly other interfaces needed in AR/AP could be explained.

Req.3 met also in full. Similar to Req.2 some interfaces are mentioned in example and other could be specified at implementation level.

Req.4 Classic Double Entry Accounting (CDEA) as the basic semantics of representing transactions is met in full.

Req. 5 met in full. The AR/AP requires separate accounts for the customer and supplier roles of each party; information about debts and payments are recorded onto these separate customer and supplier accounts.

AR/AP Requirement		**UML-SSS example**
Req.1	MDA specification	+
Req.2	Interfaces and behavior	+
Req.3	Views of balances	+
Req.4	Classic Double Entry Accounting (CDEA)	+
Req.5	Party roles	+
Req.6	Group-by Queries	+/-
Req.7	Non functional aspect like security	+/-
Req.8	Relationship to existing OMG specification	+/-

Table 8.2 Evaluation of Finance AR/AP example

Req. 6 partially met. The RetreivelGLBalance interface does not contains all operations that support retrieval of aggregated information along the required dimensions.

Req.7 and **Req.8** met in partially. We stated the nonfunctional aspect like security in risk analyses but guide lines or solutions to avoid these problems is not given in respective sections. Req.8 was optional. I have used GL and AR/AP facilities of OMG in this example but not others standards mentioned in this requirement.

9 Conclusion and further work

In the introductory section I showed that current software projects tend to be very big and complicated and that we need a new approach to software development to deal with this complexity. Many people believe that modeling is useful to raise the abstraction level and to overcome the complexity of software systems. In particular for domain standards, where the same standard interfaces might be realised in different ways on different underlying platforms. The higher abstraction level will here help us to isolate the platform independent aspects of a standard, from the platform specific aspects. The MDA is the new step in the evolution of the software development and initiative to develop a standard software development methodology where models play the central role.

The MDA defines an approach to system specifications that separates the specification of the system functionality from the specification of the platform specific implementation. This is done by specifying standards to model the system in a reusable way. This allows two main applications:

- A system can be defined platform independently and then can be realized on multiple platforms through auxiliary mapping standards.
- Different applications can be integrated by explicitly relating their models, even if they do not run on the same platform type.

The MDA is a good candidate for domain standard specifications language. It focuses on platform independent models to enhance the portability and interoperability. We can get a very good result by applying the MDA technique to COMET and UML UMM for domain standard specifications.

9.1 Requirement

This thesis presents two sets of requirements i.e. one for Domain Standard Specification Language and other for applying it to Finance AR/AP.

Domain Standard Specification Language requirements focus on formal modeling language with CIM, PIM and PSM where it emphasizes on structure, behaviour and constraints. This further include requirement like platform independent data types and computer processability. The other requirement covers the notation, non functional aspects, working tools, methodology and non stop development of It standards.

Finance AR/AP requirements introduce additional requirements for accounts receivable and account payable. The focus of these requirements is on interoperability of AR/AP with other systems or components like general ledger,

bank and eCommerce. Other requirements for this section are classic double entry accounting, non functional relationship to existing OMG solution (optional) and internal queries.

9.2 Evaluation of Existing Standard Specification Languages

I investigated some IDL, XML and UML as domain standard specification languages with practical examples. The examples are taken from finance point of view and it covers the both part of thesis.

IDL Interface Definition Language is a platform specific language. It is not possible to make models in IDL for a specific domain. It strongly supports the platform specific development in CORBA environment. IDL is used to specify standard for OMG General Ledger Facility and AR/AP. It has strong support for interoperability, behaviour and computer processability.

XML Extensible Markup Language is widely used for exchange of data and model to model transformation. I chose the XBRL General Ledger to analyse the benefits and shortcoming of it. We could not make graphical models with XML, but we can use it for their transformation.

UML Unified Modeling Language is evaluated with UN/CEFACT) Modeling Methodology and it is best for modelling platform independent models. I have used UML for standard service specification.

9.3 UML-SSS

UML-SSS is presented as domain standard specification language in this thesis. It is implemented as UML profile with UMM and COMET methodology. Mostly all models are taken from COMET and implemented with MDA technique. But it suggests the start of software development from CIM not directly from PIM as most MDA articles advocate. The evaluation of UML-SSS is given in table 8.1.

Data Types - Platform independent data types are taken from existing international standards like Core Components and UN/CEFACT UMM. A small set of basic data types like String, Integer and Boolean are used directly and others are derived from these as classes with stereotypes.

9.4 UML-SSS application to Finance AR/AP

UML-SSS is applied to specify standards for account receivable and account payable. The evaluation of is work is summarized in table 8.2. It does not cover all aspects of AR/AP but there are only important parts of AR/AP which are mentioned here. The main purpose of this example was to judge the UML-SSS practically. It shows that UML-SSS is easy to apply not only to Finance but to other domains like Health, Geographic and etc.

9.5 Future Work

UML-SSS does not fulfill the requirements fully. Several aspects can be the base for future enhancements.

- Constraints can be applied at the platform independent model level and implemented in a tool with help of some plug-ins.
- Nonfunctional aspect can be evaluated and implemented at the platform independent model level, - for example CORAS security profile can be used for this purpose.
- Standard transformation rules from the PIM level to various PSMs for different platforms can be defined using QVT and MOF.

Similarly **AR/AP** example has several aspects which need enhancements in future.

- UML-SSS should be applied to specify completely the AR/AP standard specification as it is on the agenda of OMG Finance Task Force.
- Interfaces required for req.6 can be made and explained according to methodology of UML-SSS.
- Nonfunctional aspect like security can be improved after the improvement of UML-SSS in this field.

Terminology

ANSI

The American National Standards Institute (ANSI) promotes the use of U.S. standards internationally, advocates U.S. policy and technical positions in international and regional standards organizations, and encourages the adoption of international standards as national standards where they meet the needs of the user community.

Balance sheet

Balance Sheet is an itemized statement that lists the total assets and the total liabilities of a given business to portray its net worth at a given moment of time. The amounts shown on a balance sheet are generally the historic cost of items and not their current values.

BCM

Business-Centric Methodology (BCM) is developed by OASIS a specification which will provide business managers with a set of clearly defined methods with which to acquire agile and interoperable e-business information systems within communities of interests.

BPEL

Business Centric Methodology (BCM) methods are a product of the OASIS Business Centric Methodology Technical Committee. For more information see [15]

COMET

COmponent and Model based development METhodology (COMET) is a methodology for constructing software systems by using components. COMET methodology was developed in COMBINE project.

CORBA/IDL

The Common Object Request Broker Architecture (CORBA) and Interface Definition Language of OMG.

Credit

A credit is one of the two values in a double-entry accounting system entry. At least one component of every accounting transaction (journal entry) is a credit amount. For every credit there is an equal and offsetting debit. Credits increase liabilities and equity and decrease assets. For this reason, you will see credits entered on the right-hand side (the liability and equity side of the accounting equation) of a two-column journal or ledger.

CORAS
The EU-funded CORAS project (IST-2000-25031) developed a tool-supported methodology for model-based risk analysis of security-critical systems.

Current Liability

Current Liabilities are liabilities to be paid within one year of the balance sheet date.

Debit
A debit is one of two values in an accounting entry. For every debit there is an equal and offsetting credit. At least one component of every accounting transaction (journal entry) is a debit amount. Debits increase assets and decrease liabilities and equity. For this reason, you will see debits entered on the left-hand side (the asset side of the accounting equation) of a two-column journal or ledger.

Domain
Domain is *what* a piece of software is about. At one level of detail it may be banking transactions or, at another level, it may be user interface controls. As we separate the implementation from the specification, the domain becomes the central focus of the specification. Product development is a effort to identify and separate the many domains involved in solving a problem, describe the domains in models and languages, and integrate them into a product

EDIFACT
EDIFACT stands for Electronic Data Interchange for Administration, Commerce and Transport. Along with ANSI X12, EDIFACT was one of the first information standards created for e-business transactions.

EJB

EJB (Enterprise Java Beans) The J2EE middle tier infrastructure designed to support business components.

eProcurement
eProcurement is the term to describe the use of electronic methods in every stage of the purchasing process from identification of requirement through to payment, and potentially to contract management i.e. for transactional processes. Tools include marketplaces using techniques such as eCatalogues and punch-out.
ERP
ERP stands for Enterprise Resource Planning and is the software to support entire business processes. ERP systems shift the focus from functions to processes. ERP Software Solutions typically consists of modules such as Marketing and Sales, Finance, Accounting, Field Service, Production, Inventory Control, Procurement, Distribution and Human Resources. ERP applications are the nerve center for many information systems and business processes. Changes to ERP systems have far-reaching implications throughout the enterprise and up and down the supply chain.
General Ledger
General Ledger is the record of all account entries.

IASB

The International Accounting Standards Board (IASB) Foundation is the parent entity of the International Accounting Standards Board, an independent accounting standard-setter based in London, UK.

ISO

International Organization for Standardization (ISO) is a network of the national standards institutes of 150 countries, on the basis of one member per country, with a Central Secretariat in Geneva, Switzerland, that coordinates the system.

Journal

A journal is a book or page where accounting entries are made. Journals are sometimes referred to as books of original entry. The chronological, day-to-day transactions of a business are recorded in sales, cash receipts, and cash disbursement journals. A general journal is used to enter period end adjusting and closing entries and other special transactions not entered in the other journals.

MDA

MDA (Model Driven Architecture) An approach to IT system specification that separates the specification of functionality from the specification of the implementation of that functionality on a specific technology platform.

MOF

The Meta-Object Facility (MOF) is an abstract language and a framework for specifying, constructing, and managing technology neutral metamodels. It is OMG standard

OASIS

Organization for the Advancement of Structured Information Standards (OASIS) is a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards.

OMG

Object Management Group (OMG) was established in 1989 and is the world's largest software consortium with an international membership of vendors, developers, and end users. Its mission is to help computer users solve enterprise integration problems by supplying open, vendor-neutral portability, interoperability and reusability specifications based on Model Driven Architecture. OMG has established numerous widely used standards such as OMG CORBA, UML, MOF and General Ledger (GL) to name a few significant ones. GL is related to my work; let us know what OMG GL is.

Posting

To post is to summarize all journal entries and transfer them to the general ledger accounts. This is done at the end of an accounting period.

Process
A business process defines how an organization achieves its purpose, and is designed to add value. It is composed of atomic steps at the lowest level, which are related to each other by workflow rules. A process is assigned to an organization role to enable workflow and security management." [13]

QVT
Query/View/Transformation (QVT) OMG standard.

UML
UML (Unified Modeling Language) an OMG standard language for specifying the structure and behaviour of systems. The standard defines an abstract syntax and a graphical concrete syntax.

UMM
UN/CEFACT Modeling Methodology (UMM). United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) has developed the UMM

UN/CEFACT
United Nation Center for Trade Facilitation and Electronic Business is the international standard for electronic data interchange

WSDL
Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information

XBRL
eXtensible Business Reporting Language (XBRL) is a member of the family of languages based on XML, which is a standard for the electronic exchange of data between businesses and on the internet.

XML
XML (Extensible Markup Language) An industry standard that enables the definition, transmission, validation, and interpretation of data between applications and between organizations.

References:

[1] *Revised Submission in response to OMG's Finance DTF RFP for an AR/AP Facility. OMG DTC Document finance/02-03-01, from http://www.omg.org*

[2] OMG "Unified Modelling Language Specification", Version 1.4, September 2001, available from www.omg.org.

[3] Bill Meadows, Lisa Seaburg, *Universal Business Language 1.0*

http://docs.oasis-open.org/ubl/cd-UBL-1.0/, published 15 September 2004.

[4] Anneke Kleppe, Jos Warmer, Wim Bast: MDA Explained, The model Driven Architecture: Practice and Promise, 2003. Addison-Wesley. ISBN: 0-321-19442 X.

[5] Geir . Ottersen UMLB2B: Model-based development of a B2B Internet interaction System April 2003 University of Oslo.

[6] OMG, Model Driven Architecture (MDA). 2001, from www.omg.org.

[7] (2005) Compuware website: http://www.compuware.com/products/optimalj/

[8] (2005) MDA Journals from website of Business Process Trends http://www.bptrends.com/

[9] Arne-Jørgen Berre, Brian Elvesæter, Jan Øyvind Aagedal, Jon Oldevik, Arnor Solberg, Bjørn Nordmoen; Component and Model-based development Methodology Adapted from COMET I and COMBINE

http://www.uio.no/studier/emner/matnat/ifi/INF5120/v04/undervisningsmateriale/COMET_Method_v2-4.pdf., 05 April 2004.

[10] David S. Frankel. Model Driven Architecture. Wiley Publishing, 2003.

[11] (2003) Whitepapers from Website of ebXML; http://www.ebXML.org

[12] ebXML Business Process Specification Schema, version 1,01, May 2001.

Available on web side; http://www.ebxml.org/specs/ebBPSS.pdf

[13] Chris Marshall, Enterprise Modeling with UML, Addison Wesley 1999.

[14] UN/CEFACT Modeling Methodology (UMM) User Guide

CEFACT/TMG/N093 (V20030922).

[15] *Business Centric Methodology (BCM) Creating practical tools for business integration.* http://www.oasis-open.org/committees/download.php/5931/BCM%20Executive%20Brochure.pdf *April 2005.*

[16] *Luther Hampton, David Vun Kannon: Extensible Business Reporting Language (XBRL) Specification* http://www.xbrl.org *December 2003.*

[17] *(2005) Todd F. Boyle, General Ledgerism (s t u f f t h a t c o u n t s -- s e m a n t i c s o f g e n e r a l l e d g e r s) http://legerism.net*

[18] *Dr. Arne-Jørgen Berre, ISO/TC 211 WG1 – WI 3 Conceptual Schema Language, 1996-12-15.*

[19] *Carol Costa, C. Wesley Addison; Alpha Teach Yourself Accounting, Alpha 2001.*

[20] *Peter Eeles, Kelli Housten, Wojtek Kozaczynski; Building J2EE Application with the Rational Unified Process, Addison- Wesley 2002.*

[21] *Sergei Savenko, Combined PIM-PSM, Master thesis, 19th April 2004.*

[22] *(2005) Website Modellering med objekter, INF 5120 UIO;* http://www.uio.no/studier/emner/matnat/ifi/INF5120/

[23] *(2004) Website VentureLine at;* http://www.ventureline.com/glossary.asp

[24] *(2005) Article on eProcurment from website;* http://www.ogc.gov.uk/embedded_object.asp?docid=1002131

[25] *(2005) Website of IDABC(Interoperable Delivery of European eGovernment Services to public Administrations, Businesses and Citizens);* http://europa.eu.int/idabc/

[26] *OMG General Ledger Facility ISO RM-ODP Computational Viewpoint*

Revision 4.1 December 21st, 1998 (OMG DTC Document finance/99-03-05)

[27] *(2001) Website, XBRL for General Ledger (XBRL GL);* http://www.xbrl.org

[28] *(2005) Website Compiere;* http://www.compiere.com/