Model-based System Development
www.ifi.uio.no/inf5120

Part III

MDE for SOA

(COMET-S with BMM, BPMN and UPMS)

Notes for Course material

“Model Based System Development”

INF5120 – Spring 2008

<table>
<thead>
<tr>
<th>Classification</th>
<th>Notes for course participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Responsible</td>
<td>Arne-Jørgen Berre, SINTEF and University of Oslo</td>
</tr>
<tr>
<td>Authors</td>
<td>Arne-Jørgen Berre, Brian Elvesæter</td>
</tr>
<tr>
<td>Contributors:</td>
<td>Projects INTEROP, ATHENA, SWING, SHAPE, COIN</td>
</tr>
<tr>
<td>Task</td>
<td>INF5120 Course notes</td>
</tr>
<tr>
<td>Status :</td>
<td>Version 1.00</td>
</tr>
<tr>
<td>Date :</td>
<td>May 4th, 2008</td>
</tr>
</tbody>
</table>
Table of Contents

Part III  1

Notes for Course material ........................................................................................................................................ 1

“Model Based System Development” .................................................................................................................. 1

INF5120 – Spring 2008 ........................................................................................................................................ 1

Executive Summary .............................................................................................................................................. 4

I  COMET-S - Overview ........................................................................................................................................ 5

I.1 The different models in COMET-S .......................................................................................................................... 5

II  Relationship to OMG MDA – Model Driven Architecture ............................................................................ 6

III  COMET - Business model .................................................................................................................................. 7

III.1 COMET Business model overview .......................................................................................................................... 7

III.1.1 COMET Business model overview .................................................................................................................... 7

III.1.2 Modelling framework for the business model ........................................................................................................... 8

III.1.3 The context statement - BMM ................................................................................................................................. 8

III.1.4 The vision statement - BMM ....................................................................................................................................... 9

III.1.5 Risk analysis - BMM ................................................................................................................................................. 9

III.1.6 The goal model - BMM ............................................................................................................................................. 10

III.1.7 The community model – BMM Stakeholder .............................................................................................................. 10

III.2 BMM – Business Motivation Metamodel ........................................................................................................ 10

III.2.1 What is the Business Motivation Model? .................................................................................................................. 11

III.2.2 Other Elements of a Full Business Model .................................................................................................................. 11

III.2.3 Business Rules in the Business Motivation Model .................................................................................................... 12

III.2.4 Methodologies and the Business Motivation Model ................................................................................................. 12

III.2.5 2. The Core Elements of the Business Motivation Model ....................................................................................... 12

III.2.6 BMM notation - The End Concepts .......................................................................................................................... 14

III.2.7 2.3 BMM notation - The Means Concepts .................................................................................................................. 17

III.2.8 2.3 BMM notation - Influencers and Assessments ....................................................................................................... 18

III.2.9 The business process and roles model - BPMN .......................................................................................................... 21

III.3 BPMN - The Business Process Modeling Notation .......................................................................................... 21

III.3.1 Modelling Objectives .............................................................................................................................................. 22

III.3.2 Methods and techniques .......................................................................................................................................... 22

III.3.3 The business resource model – UML Class model and OWL .................................................................................. 25

III.3.4 Work analysis refinement model (WARM) ................................................................................................................. 26

IV  COMET - Requirements model ............................................................................................................................ 29

IV.1 Introduction – COMET Requirements model ...................................................................................................... 29

IV.1.1 Use Case Model ...................................................................................................................................................... 30

IV.1.2 Prototype ................................................................................................................................................................. 32

IV.1.3 Non-functional requirements ................................................................................................................................ 32

IV.1.4 Reference Architecture Analysis ............................................................................................................................. 32

IV.2 COMET - Requirements metamodel .................................................................................................................... 34

IV.2.1 Introduction ............................................................................................................................................................ 34

IV.2.2 Model Structure ........................................................................................................................................................ 34

IV.3 COMET - Requirements model to service architecture model transformation .................................................. 35

IV.3.1 Requirements model to architecture model transformation ......................................................................................... 35

IV.3.2 Service information model – Class Diagrams ........................................................................................................... 37

V  COMET-S - Service architecture model ................................................................................................................ 39

V.1.1 Introduction ............................................................................................................................................................. 39

An architectural and business focused approach to SOA .......................................................................................... 39
Executive Summary

This document is part I in a series of four part of lecture notes for the course INF5120, Model Based System Development, for the spring of 2008, as follows.

Part I – MDE – Model Driven Engineering
Part II – SOA - Service Oriented Architectures
Part III – MDE for SOA, (COMET-S with BMM, BPMN and UPMS)
Part IV – MDI – Model Driven Interoperability

Part I focuses on *Model Driven Engineering* – with an introduction to principles of metamodelling and related standards and technologies, in particular related to MDA and Eclipse EMF. The relationship between UML profiles and Domain Specific Languages (DSL) is introduced, as well as an overview of various model transformation technologies including model-to-model with ATL and model-to-text with MOFScript. It is shown how method engineering can be supported by the OMG SPEM standard and the Eclipse EPF framework.

Part II focuses on *SOA - Service Oriented Architectures* – with a basis in concepts for service oriented computing, with a special emphasis on technologies for web services with XML, WSDL and BPEL. The basis technologies for the semantic web is also introduced with RDF and OWL, and semantic web services with OWL-S and WSMO. A last section presents agent-oriented computing with multi agent systems (MAS) and a platform independent model for agents (PIM4Agents).

Part III focuses on *MDE for SOA - Model Driven Engineering for Service Oriented Architectures* – and applies the principles of model driven engineering to service oriented architectures. The starting point for this approach is the COMET methodology used in previous INF5120 courses, this year enhanced to become COMET-S for Services through the use of new standard UML profiles and metamodels. The Business model uses in particular BMM (Business Motivation Metamodel, BPMN (Business Process Modeling Notation). The Requirements model supports mappings from use cases to services definitions. The service architecture model uses the new UPMS (UML Profile and Metamodel for Services). The platform specific model will vary depending on the target platform. The course has been using the JEE platform as reflected in the Oblig exercises in the course.

Part IV focuses on *MDI - Model Driven Interoperability* – and illustrates how a model driven approach can be applied to the problem domain of interoperability through the use of horizontal mappings and transformations. The approach to this is illustrated with the AIF (ATHENA Interoperability Framework) and the AIM (ATHENA Interoperability Methodology) and a set of articles on MDI.

In addition there will be an example Part V – Buyer/Seller Purchase Order example for part III (MDE for SOA) that will be provided as an illustration of the use of COMET-S in practice.
I COMET-S - Overview

I.1 The different models in COMET-S

The starting point for the COMET-S methodology (COMET for Services) is the existing COMET methodology as documented at www.modelbased.net/comet.

COMET-S update this methodology with newly available metamodels and UML profiles from the ongoing OMG standardization activities, in particular with BMM (Business Motivation metamodel), BPMN (Business Process Modeling Notation) and UPMS (UML Profile and Metamodel for Services).

The following figure gives an overview of the four main modeling areas in COMET-S, Business model, Requirements model, Architecture model and Platform specific model.

Figur 1 COMET-S with use of BMM, BPMN and UPMS
II  Relationship to OMG MDA – Model Driven Architecture

The figure below shows a principal structure of relationships between models from the OMG CIM to PIM to PSM levels. Each level can have a set of interrelated models.

![Diagram showing relationships between OMG CIM, PIM, and PSM levels]

Figur 2  Possible areas for CIM to PIM to PSM and code mappings

In the context of COMET-S we currently focus on a subset of these:

![Diagram showing focused areas for COMET-S]

Figur 3  COMET-S focus on Service, Process and Information areas

Note how requirements modeling takes a role in the mappings between CIM business models and PIM architecture models, and the use of model-to-model (ATL) and model-to-text (MOFScript) is applied for the mappings between PIM architecture models and PSM and code platform models.
III COMET - Business model

III.1 COMET Business model overview

III.1.1 COMET Business model overview

The **business model** is used to express the part played by the **product** (system or component) being developed in the context of the business that will fund its development (or purchase it) and use it. It also provides elements that link extremely well to the **architecture model**.

The business model includes goals, business processes, steps within business processes, roles and resources. The scope (or domain) of the model is any part of the world defined as interesting for a company, organisation or others, and which has some impact on the required behaviour or other characteristic of the product. The business model might be broadly or narrowly scoped, e.g. describing the entire business of a company or describing the immediate environment and context of a product under consideration. Thus, the business model is recursive in the sense that some business models might detail aspects of another more broadly scoped/higher level business model. The set of work products of a business model is outlined below.

A **business model** consists of the work products listed below.

- A set of **scoping statements**, consisting of the following:
  - The **context statement**, which defines the scope and positions this business model in its context.
  - **Vision for change**, which describes what to improve, the motivation (i.e. what is wrong with the current situation), a description or indication of what the improvements might be and a gap analysis.
  - **Risk analysis**, which identifies the business and technical risks related to a development project for the proposed Product.
- A **goal model**, which describes the business goals that will be met by implementing and then using the Product.
- A **community model**, in which is prepared through a combination of three modelling activities:
  - Business process and role modelling, which prepares 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. (with BPMN)
  - Business resource modelling which prepares an information model that identifies and defines the concepts of the domain that are relevant to the Product, including the actors that do things, and the artifacts that have things done to or with them. (with Class modeling)
  - Work analysis refinement modelling (WARM) which further refines the resource model and the process model by categorising resources by the kind of role they play (e.g. human user, tool, business service, etc), and by identifying, for every step in the process model, the responsibility in the system for performing it. The
WARM activity also allows identification of work elements, which are used by the architecture modeller (or mapping tool) to derive its first-cut service model. Note that the WARM is not a separate model as such, merely refinement of a model from an earlier stage in the process.

It is most important to note that the above structure is only a structure. It should not be thought of as defining the sequence of activities for developing these work products. Thus, for example, although the Context Statement and Vision for Change would obviously be started and well developed before significant development of the other elements takes place, they are not cast in stone at the beginning of the project, and may be revised at any time subsequently. Similarly the Goal model may be revised as the Business Process and Resource models are developed. Finally, and most commonly, the Business Process model and the Resource model are almost invariably developed in parallel.

III.1.2 Modelling framework for the business model

III.1.3 The context statement - BMM

III.1.3.1 Modelling objectives

The purpose of the Context Statement is to define the scope of a business model and to position it in its context. This may be done by identifying relationships to some other business models or by developing and getting agreement to a more informal representation of the target business being modelled in its environment. This informal representation takes the form of a domain picture aiming to give an overall understanding of the domain. It focuses on describing
stakeholders and their relationships and identifies stakeholders concerns. It typically covers key value-chains and information flows. The emphasis is on the process of building it, agreeing it with subject matter experts and thereby understanding the context. Thus, it may not be maintained in subsequent phases of the development of the Product.

III.1.3.2 Methods and techniques

The first step in developing any business model is to identify and record the names of all people who will have an interest in having the Product developed or in its use, the business stakeholders, together with the nature of their interest. This list may change over time but should always include representatives of all having an interest in the Product in any way. The following are examples of business stakeholders who should be involved:

- people who will authorise funding for development of the Product;
- people who are responsible for design and maintenance of the business processes to be supported by the Product;
- people who will use the Product;
- people who will be responsible for the acceptance of the Product;
- people who will be responsible for managing operation of the Product

The approach taken to developing a Context Statement depends on the existence or not of some higher level business model of which the new model forms a part. Whatever the approach, however, the process consists of fact finding, discussion and agreement with all business stakeholders, using one or more small workshops or interviews followed by one or more feed-back sessions at which draft models are presented and walked through.

III.1.4 The vision statement - BMM

This work product is a statement of what to improve, the motivation (i.e. what is wrong with the current situation), a description or indication of what the improvements might be and a gap analysis (a clear understanding of difference between the current and desired situations).

III.1.4.1 Modelling objectives

This work product adds detail to the scope given in the Context Statement, so that Business Stakeholders can make informed judgements about approving the (appropriate phase of) development of the Product.

III.1.4.2 Methods and techniques

This document would be produced by a series of meetings, and the preparation and circulation drafts until agreement on its content is achieved by the Business Stakeholders identified at the start of the Project.

III.1.5 Risk analysis - BMM

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. An estimate of the anticipated return of investment (ROI) is also part of this work product.

III.1.5.1 Modelling objectives

The objective of the Risk Analysis is to identify all risks to the project and ensure that suitable contingency plans are in place.
III.1.5.2 Methods and techniques
The approach to developing the risk analysis is to examine all possible ways that the programme, from development through to the Product’s in service life, might go wrong, and to identify the level of risk involved (probability of a bad thing happening), the cost that would be incurred if it happened, and contingency plans to reduce that cost to acceptable levels. All types of risk should be considered, including commercial risk (failure to give the intended return on investment), business risk (impact on the business of failure of the Product, either before or after deployment), programme risk (failure to deliver on time), development risk (Product development is more difficult or costly than expected) and support risk (high cost of user support or maintenance because of Product fragility).

III.1.6 The goal model - BMM

III.1.6.1 Modelling objectives
The purpose of the Goal Model is to agree with the Business Stakeholders the business goals that will be met by implementing and then using the Product, so that a set of required high level business processes can be identified for further analysis in the Business Process Model. Once produced and agreed, the goal model serves as a reference, throughout the Product's life, that ensures that a full assessment may be made of all the business implications of any proposed changes to the Product which have any impact on the business processes it supports.

III.1.6.2 Methods and techniques
The goal model describes a loose hierarchy of goals of the business within the particular area of concern, starting with the goals of a Business Stakeholder in developing or buying the Product, and leading to the detailed business goals met by the Product or its users when using it.

III.1.7 The community model – BMM Stakeholder

The Community Model is a container for that part of the Business model that details the business processes and business resources that are relevant to the Product. It is called "Community Model" because the concept of Community (which represents a collection of resources working together, in one or more processes, to achieve one or more goals) is the key to performing recursive, parallel, and decomposition of both behaviour and structure, essential for successful business process modelling. Thus the model's primary structure is of a set of communities, each of which is itself structured in terms of what it is (resources) and what it does (processes). The first step in preparing the Community Model, and the contained Business Process and Resource models, is to create a package to contain the Community model, and, within it, a further package, stereotyped as <<Community>>, to represent the top level community. Within this <<Community>> package, create two further packages to contain respectively, the Resources visible at this top level, and the enabling behaviours that support the goals defined in the Goal model.

III.2 BMM – Business Motivation Metamodel

The Business Motivation Model provides a scheme or structure for developing, communicating, and managing business plans in an organized manner. Specifically, the Business Motivation Model does all of the following:

- It identifies factors that motivate the establishing of business plans.
It identifies and defines the elements of business plans.
It indicates how all these factors and elements inter-relate.

Among these elements are ones that provide governance for and guidance to the business — Business Policies and Business Rules.

III.2.1 What is the Business Motivation Model?

There are two major areas of the Business Motivation Model.

- The first is the Ends and Means of business plans. Among the Ends are things the enterprise wishes to achieve — for example, Goals and Objectives. Among the Means are things the enterprise will employ to achieve those Ends — for example, Strategies, Tactics, Business Policies, and Business Rules.
- The second is the Influencers that shape the elements of the business plans, and the Assessments made about the impacts of such Influencers on Ends and Means (i.e., Strengths, Weaknesses, Opportunities, and Threats).

The Ends, Means, and Influencers are related to each other in order to answer the following two fundamental questions:

1. **What is needed to achieve what the enterprise wishes to achieve?**
   This question is answered by laying out the particular elements of the business plans — in other words, the Means necessary to achieve the desired Ends.

2. **Why does each element of the business plan exist?**
   This question is answered by identifying the particular Ends that each of the Means serves, and the Influencers that underlie the choices made in this regard. This is what is meant by motivation.

All elements of the Business Motivation Model are developed from a business perspective. The basic idea is to develop a business model for the elements of the business plans before system design or technical development is begun. In this manner, the business plans can become the foundation for such activity, connecting system solutions firmly to their business intent.

III.2.2 Other Elements of a Full Business Model

The Business Motivation Model is not a full business model. For example, the elements of business plans do not prescribe in detail any of the following, each of which is an essential part of a full business model.

- **Business Processes.** Business plans include Courses of Action — what the enterprise has to do to achieve its Ends — transformed into Business Processes that encompass activities, sequencing, dependencies, interactions, triggering by business events, etc. Business Process specification is outside the scope of business plans. However the Business Motivation Model does include a placeholder for Business Process, to provide for integration with emerging Business Process standards.

- **Workflows.** The basis of workflow is assignment of responsibilities for Business Processes to roles in the organization. Design of workflow is outside the scope of business plans. However, the
business plans may include Strategies or Tactics that configure organization structure to achieve effective workflow.

- **Business Vocabulary.** Also needed is full specification of the terms and facts needed to support the business. However, the business plans do provide a business basis for such development — namely, the concepts and vocabulary used in the elements of the business plans, particularly in its Business Rules. Refer to Appendix E, Section E.2 for comments about additional elements of a full business model.

### III.2.3 Business Rules in the Business Motivation Model

Business Rules play an important role in development of business plans. For example, they can serve the following purposes:

- Make business tactics sufficiently well developed to guide the actual performance of work.
- Provide fallback positions when some element of the business plans fails.
- Resolve conflicts when the Ends the business seeks are in conflict with one another. Because of this key role — which is often make-or-break for the very success of business plans — developing the motivation for Business Rules from the business perspective is fundamentally important.

### III.2.4 Methodologies and the Business Motivation Model

It is important to note that the Business Motivation Model is not in any sense a methodology. Indeed, it is entirely neutral with respect to methodology or particular approach, with only several general exceptions as follows:

- The requirements development process should be business-driven.
- Organized business plans should be a fundamental deliverable in any such process.
- Business Rules and Business Processes are key elements of such business plans.

One way to think of the Business Motivation Model is as a blueprint purposely designed to support a range of methodological approaches. Implementation of the Model would result in the elements of business plans being stored and related to other information about the enterprise, no matter what methodology was used for discovering and defining them. In the design for any such implementation, each concept of the Business Motivation Model (i.e., each concept listed in the Concepts Catalog) should be assigned two attributes — 'name' and 'description' — to be included in the implementation. User enterprises and repository vendors could, of course, choose to include additional attributes.

### III.2.5 2. The Core Elements of the Business Motivation Model

The main elements of the business plans are its Ends and Means.

#### 2.1 BMM meta-model

Core BMM model can be illustrated below:
Figur 5  Core Concepts of the Business Motivation Model
III.2.6 BMM notation - The End Concepts

An End is something the business seeks to accomplish. The important thing to remember about an End is that it does not include any indication of how it will be achieved. In describing Ends, it is useful to document who defined the End and at what point in time, so that an audit trail exists for future reference. This, of course, cannot always be mandated.

Categories of End

End concepts can be arranged in a hierarchy, as shown in Figure 2.

Figur 6 Ecore model for the Business Motivation model

Figur 7 The Hierarchy of 'End' Concepts
An End may be either a Vision or some Desired Result (a Goal or an Objective). The essence is that these are kinds of things, in varying detail, that the enterprise is trying to accomplish. Vision is an overall image of what the organization wants to be or become. It usually encompasses the entire organization and is long-term in its perspective. Desired Results, on the other hand, are the more specific Goals and Objectives that the enterprise, or some part of it, intends to achieve.

**Vision**
A Vision describes the future state of the enterprise, without regard to how it is to be achieved. A Vision is the ultimate, possibly unattainable, state the enterprise would like to achieve. A Vision is often compound, rather than focused toward one particular aspect of the business problem. A Goal, in contrast, should generally be attainable and should be more specifically oriented to a single aspect of the business problem.
A Vision is supported or made operative by Missions. It is amplified by Goals.

**Desired Result**
A Desired Result is an End that is a state or target that the enterprise intends to maintain or sustain.
A Desired Result is supported by Courses of Action.

**Categories of Desired Result**
Desired Result includes the following concepts:
- Goal
In an Objective, a Goal tends to be longer term, qualitative (rather than quantitative), general (rather than specific), and ongoing. Compared to a Goal, an Objective tends to be short term, quantitative (rather than qualitative), specific (rather than general), and not continuing beyond its timeframe (which may be cyclical).

Objectives differ from Goals in that Objectives should always be time-targeted and measurable. Goals, in contrast, are not specific in these ways.

Desired Results are supported by Courses of Action, which can be either Strategies or Tactics. Generally, Goals are supported by Strategies, and Objectives are achieved by Tactics. In many enterprises there is a continuum from major Strategies that impact the whole of the business to minor Tactics with limited, local effects. The dividing line between 'minor Strategy' and 'major Tactic' is blurred. Also, over time, some Courses of Action may evolve from Tactic to Strategy, and some Strategies may devolve into Tactics. Some enterprises do make a hard distinction between Strategies and Tactics; these enterprises may choose to pair Strategies only with Goals, and Tactics only with Objectives.

Other enterprises use other bases for distinguishing Strategies and Tactics. For example, some enterprises distinguish between Strategy and Tactic based on planning horizon. In this case, Strategies are put into place to support the long-term Goals — i.e., a planning horizon that is typically several years or more — while Tactics are the Courses of Action implemented to deal with the shorter planning horizon of a year or less (the current operational plans). Still other enterprises distinguish Strategy (a Course of Action that is for "the gaining of a specific advantage") from Tactic, which is a Course of Action that is for "the deployment of specific resources to gain that advantage."

**Goal**

A Goal is a statement about a state or condition of the enterprise to be brought about or sustained through appropriate Means. A Goal *amplifies* a Vision — that is, it indicates what must be satisfied on a continuing basis to effectively attain the Vision.

A Goal should be narrow — focused enough that it can be *quantified by* Objectives. A Vision, in contrast, is too broad or grand for it to be specifically measured directly by Objectives. However, determining whether a statement is a Vision or a Goal is often impossible without in-depth knowledge of the context and intent of the business planners.
**Objective**

An Objective is a statement of an attainable, time-targeted, and measurable target that the enterprise seeks to meet in order to achieve its Goals.

- **Attainable.** It is self-evident that Objectives should be attainable. If they are not, the business plans are unrealistic and will likely fail.
- **Time-targeted.** All Objectives should be time-targeted. This means that either an absolute timeframe (e.g., "by January 1, 2007") or relative timeframe (e.g., "within two years") should be included in each Objective. This timeframe indicates when the Objective is to be met.
- **Measurable.** Objectives should be measurable. This means they must include some explicit criteria for determining whether the Objective is being met in practice. This criterion may be fairly exacting (for example, "on-time 95% of the time"). At the very minimum, the criteria must provide a basis for making a "yes or no" determination (e.g., "up and running"). Such criteria may be the basis for certain Business Rules, created specifically to compute or derive the relevant evaluation.

### III.2.7 2.3 BMM notation - The Means Concepts

A Means represents any device, capability, regime, technique, restriction, agency, instrument, or method that may be called upon, activated, or enforced to achieve Ends. Remember that a Means does not indicate either the steps (business processes and workflow) necessary to exploit it, nor responsibility for such tasks, but rather only the capabilities that can be exploited to achieve the desired Ends.

In describing Means, it is useful to document who *established* the Means and at what point in time, so that an audit trail exists for future reference. This practice, of course, cannot always be mandated.

**Categories of Means**

Means concepts can be arranged in a hierarchy, as shown in Figure 7.

---

![Figur 11 The Hierarchy of 'Means' Concepts](image-url)
A Means may be either a Mission, a Course of Action (a Strategy or Tactic), or a Directive (Business Policy or Business Rule). Mission, like its counterpart Vision, indicates a correspondingly long-term approach — one that is focused on achieving the Vision. Like Vision, Mission is not very specific; it is something broadly stated, in terms of the overall functioning of the enterprise. Apart from the basic Mission of the enterprise, the Means of the Business Motivation Model have been carefully organized into Courses of Action and Directives. In some respects, Courses of Action are the more basic of the two. In and of themselves, however, Courses of Action tend to be rather blunt instruments. They require Directives to have any real chance of success.

In contrast to Courses of Action, Directives cannot really stand on their own. They exist to give the Courses of Action a fine edge — in other words, to ensure that the Courses of Action will be applied intelligently and within the boundaries of what is acceptable or optimal for the enterprise. In short, Directives represent encoded (i.e., written down) knowledge that ensures the highest possible chances of success for the Courses of Action. A Directive always has to do with governance or guidance. A Course of Action, in contrast, identifies an active approach in moving toward the Ends. A Course of Action is always action-dominated (action-oriented).

**Mission**

A Mission indicates the ongoing operational activity of the enterprise. The Mission describes what the business is or will be doing on a day-to-day basis.

A Mission makes a Vision operative — that is, it indicates the ongoing activity that makes the Vision a reality. A Mission is planned by means of Strategies.

![Figur 12 Mission](image)

A Mission statement should consist of the following three items only and should be formed using brief statements of them.

**III.2.8 2.3 BMM notation - Influencers and Assessments**

To fully understand the elements of business plans, it is necessary to identify the Influencers that shape them. This helps communicate the full intent of the Means and Ends by explaining the context in which they were formulated. Influencers, however, are neutral — they are more or less simply just 'there' until someone makes an Assessment about how they are likely to impact some End or Means. Influencers should always be stated in a neutral, factual manner. Therefore, Influencers should be devoid of qualitative words. The presence of qualitative words indicates a statement about an Influencer — in other words, an Assessment.
This section describes the concepts that define these aspects of the Business Motivation Model. Understanding these elements supports intelligent revision of the business plans; possibly well after the business plans and related application system(s) have been implemented.

III.2.8.1 2.3.1 Influencers on the Ends and Means

**Influencer**

An Influencer can be anything that has the capability to 'produce an effect without apparent exertion of tangible force or direct exercise of command, and often without deliberate effort or intent.' The Influencers specifically of concern to business plans are those that can impact the enterprise in its employment of Means or achievement of its Ends. This impact has influence that is judged in Assessments.

**Categories of Influencer**

An enterprise can define whatever Influencer Categories it requires. Enterprises that do not have a preferred set of categories may choose to use the default set provided here, modifying and extending it as required.

- Two broad Influencer Categories: External Influencer and Internal Influencer,
- A set of general categories.

![Concepts Reflecting Influencers](image)

**Figur 13 Concepts Reflecting Influencers**

As the names imply, the distinction between the categories is based on whether the Influencer falls outside the organization's boundary (and hence outside its direct control) or inside its boundary.

**Assessment**

An Assessment is *a judgment about* some Influencer that affects the organization's ability to employ its Means or achieve its Ends. In other words, an Assessment expresses a logical connection (i.e., fact type) between Influencers and the Ends and/or Means of the business plans. In this way, an Assessment indicates which Influencers are relevant to which Ends and/or Means.

Different people or groups of people might make different Assessments of the very same Influencers on the very same Ends and Means — and perhaps even the same person or group of people at different points in time. It is very important to indicate which person or group of people
makes which Assessment of an Influencer at which point in time, so that an audit trail exists for future reference. This practice, of course, cannot always be mandated.

An Assessment can provide impetus for Directives that govern Courses of Action and/or support the achievement of Ends. In other words, the Directive is motivated by the Assessment. Assessments can also use other assessments. Typical use of this association is connecting an assessment to other assessments referenced while arriving at the judgment. These associations can then be used for many purposes, including:

- Decision support for making new assessments — identifying the full set of concerns taken into account in earlier assessments
- Compliance audit — justifying decisions made in reacting to change

### III.2.8.2 Categories of Assessment – Risk analysis - SWOT

The Business Motivation Model supports a general categorization structure for Assessment. SWOT — Strength, Weakness, Opportunity, Threat — is a frequently-used set of categories for Assessment. There are other approaches, but if an enterprise does not have another preferred set, SWOT is a sound default. SWOT is used as the illustrative approach for discussion and examples in the rest of this document.

When applying SWOT to making assessments, Assessments are categorized as follows:

- **Strength**
- **Weakness**
- **Opportunity**
- **Threat**

**Strength**
This category of Assessment indicates some advantage or area of excellence within the enterprise that can impact its employment of Means or achievement of Ends.

**Weakness**
This category of Assessment indicates some area of inadequacy within the enterprise that can impact its employment of Means or achievement of Ends.

**Opportunity**
Organization's employment of Means or achievement of Ends.

**Threat**
This category of Assessment indicates that some Influencer can have an unfavorable impact on the organization's employment of Means or achievement of Ends.
III.2.9 The business process and roles model - BPMN

The Business Process and Roles model (generally called the Business Process or Process model) defines

- the business processes of the domain which are relevant to the Product, and which will enable the goals to be met, and:
- the roles of the resources that perform those processes.

This model may be at a number of levels of detail, from a high level description of the business processes down to the WARM, which is a set of 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.

- The Business Process and Roles model is generally prepared at the same time as the associated Business Resource model.
- To build a Business Process model the Business Modelling Conceptual Model (or metamodel) is used.

Each business process is defined in terms of its steps, and each step performed by a resource at a higher level of detail may then be treated as a process performed by a community (of resources) and further analysed at a lower level of detail. Eventually, using the WARM concepts of Human Step, Tool Step and Immediate Step, “leaf” steps will be defined, and assigned to specific elements of the Product to be developed. The time to stop analysing is when, by analysing the processes, the Product’s role in them is defined, and when there are no more questions about the business to be answered.

III.2.9.1.1 Derive initial process model from the goal model

The Business Process Model is derived directly from the Goal Model. Goals may be thought of as high level statements of the things that have to happen in a business, each expressed as an outcome, but in a way that leaves unspecified how that outcome is to be made to occur. As goal analysis proceeds, the analyst becomes aware that he or she is increasingly describing not so much a desired outcome as the means by which a higher level outcome is to be achieved, namely an enabling behaviour. The distinction between goal and behaviour to achieve a goal is inevitably blurred and to some extent subjective. The key thing to remember is that, whereas a goal is some desired state of affairs, a behaviour is a specification for something that happens, the end result of which is the desired state of affairs.

Thus, the first step in creating the Business Process Model will be the identification of the enabling behaviours that have to happen for each goal to be achieved. Initially this is done through a brain-storming process and production of an unstructured list of enabling behaviours for each goal. This list is then consolidated into a single set of enabling behaviours that, together, support all goals. This is the starting point for the Business Process Model which may then be entered into the tool using the Business Process Modelling Profile.

III.3 BPMN - The Business Process Modeling Notation
Business process management (BPM) is a method of efficiently aligning an organization with the wants and needs of clients. It is a holistic management approach that promotes business effectiveness and efficiency while striving for innovation, flexibility and integration with technology. As organizations strive for attainment of their objectives, BPM attempts to continuously improve processes - the process to define, measure and improve your processes – a 'process optimization' process.

A great reality utilisation of BPM is Business process management notation (BPMN). BPMN provides a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation.

BPMN could define the notation and semantics of a Business Process Diagram (BPD) and represents the amalgamation of best practices within the business modeling community. The intent of BPMN is to standardize a business process modeling notation in the face of many different modeling notations and viewpoints.

This model may be at a number of levels of detail, from a high level description of the business processes down to the WARM, which is a set of 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.

Each business process is defined in terms of its steps, and each step performed by a resource at a higher level of detail may then be treated as a process performed by a community (of resources) and further analysed at a lower level of detail. Eventually, using the WARM concepts of Human Step, Tool Step and Immediate Step, "leaf" steps will be defined, and assigned to specific elements of the Product to be developed. The time to stop analysing is when, by analysing the processes, the Product's role in them is defined, and when there are no more questions about the business to be answered.

**III.3.1 Modelling Objectives**
The 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 (and its components, i.e. Application Components, Business Service Components and Tool Components).

**III.3.2 Methods and techniques**
The Business Process Model is derived through a set of activities that encompass brain-storming sessions, structured workshops, interviews and feed-back sessions, and detailed modelling using Business Process Diagram (BPM) which derives from BPMN. BPM would be implemented by BPMN Eclipse plug-in.

**III.3.2.1 BPMN Metamodel**
The Eclipse BPMN plug-in now we are using is made by EMF & GMF. Each element in BPMN all have the corresponding metamodel element. Figure 1 shows part of the BPMN metamodel.
The modeling in BPMN is made by simple diagrams with a small set of graphical elements. It should make it easy for business users as well as developers to understand the flow and the process. The four basic categories of elements are as follows:

- **Flow Objects**: Events, Activities, Gateways
- **Connecting Objects**: Sequence Flow, Message Flow, Association
- **Swimlanes**: Pool, Lane
- **Artifacts**: Data Objects, Group, Annotation

### III.3.2.2 BPMN Notation

BPMN has its specific notations. Table 1 shows the detail information about BPMN notations.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="http://org.eclipse/swf/score/util/ExtendedMetadata" alt="Start" /> <img src="http://org.eclipse/swf/score/util/ExtendedMetadata" alt="Intermediate" /> <img src="http://org.eclipse/swf/score/util/ExtendedMetadata" alt="End" /></td>
<td>An Event is represented with a circle and is something that happens. It could be Start, Intermediate or End. This element is a trigger or a result.</td>
</tr>
<tr>
<td><img src="http://org.eclipse/swf/score/util/ExtendedMetadata" alt="Task" /> <img src="http://org.eclipse/swf/score/util/ExtendedMetadata" alt="Process" /> <img src="http://org.eclipse/swf/score/util/ExtendedMetadata" alt="Activity" /></td>
<td>An Activity is represented with a rounded-corner rectangle and shows us the kind of work which must be done. It could be a task or a sub-process. A sub-process also has a plus sign in the bottom line of the rectangle.</td>
</tr>
</tbody>
</table>
| ![Gateway](http://org.eclipse/swf/score/util/ExtendedMetadata) ![Decision](http://org.eclipse/swf/score/util/ExtendedMetadata) ![Merge](http://org.eclipse/swf/score/util/ExtendedMetadata) | A Gateway is represented with a diamond shape and will determine different decisions. It will also determine forking, merging and joining of
A Sequence Flow is represented with a solid line and arrowhead and shows in which order the activities will be performed. A diagonal slash across the line close to the origin indicates a default choice of a decision.

A Message Flow is represented with a dashed line and an open arrowhead. It tells us what messages flow between two process participants.

An Association is represented with a dotted line and a line arrowhead. It is used to associate an Artifact, data or text to a Flow Object.

A Pool is represented with a big rectangle which contains many Flow Objects, Connecting Objects and Artifacts.

A Lane is represented as a sub-part of the pool. The lanes organize the Flow Objects, Connecting Objects and Artifacts more precisely.

Data Objects show the reader which data is required or produced in an activity.

A Group is represented with a rounded-corner rectangle and dashed lines. The Group is used to group different activities but does not affect the flow in the diagram.

An Annotation is used to give the reader of the model/diagram an understandable impression.

Table 1 BPMN notation

### III.3.2.3 BPMN Description

Inter-operation of business processes at the human level, rather than the software engine level, can be solved with standardization of the Business Process Modeling Notation (BPMN). BPMN provides a Business Process Diagram (BPD), which is a Diagram designed for use by the people who design and manage business processes. BPMN also provides a formal mapping to an execution language of BPM Systems (BPEL4WS). Thus, BPMN would provide a standard visualization mechanism for business processes defined in an execution optimized business process language.

BPMN provides businesses with the capability of understanding their internal business procedures in a graphical notation and will give organizations the ability to communicate these procedures in a standard manner. Currently, there are scores of process modeling tools and methodologies. Given that individuals will move from one company to another and that companies will merge and
diverge, it is likely that business analysts are required to understand multiple representations of business processes—potentially different representations of the same process as it moves through its lifecycle of development, implementation, execution, monitoring, and analysis. Therefore, a standard graphical notation will facilitate the understanding of the performance collaborations and business transactions within and between the organizations. This will ensure that businesses will understand themselves and participants in their business and will enable organizations to adjust to new internal and B2B business circumstances quickly. BPMN follows the tradition of flowcharting notations for readability and flexibility. In addition, the BPMN execution semantics is fully formalized. The OMG is using the experience of the business process notations that have preceded BPMN to create the next generation notation that combines readability, flexibility, and expandability.

BPMN will also advance the capabilities of traditional business process notations by inherently handling B2B business process concepts, such as public and private processes and choreographies, as well as advanced modeling concepts, such as exception handling, transactions, and compensation.

III.3.3 The business resource model – UML Class model and OWL

III.3.3.1 Modelling objectives

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.

The Business Resource Model can be specified as an information model using UML class diagrams, it can be linked to an ontology model, for instance expressed in OWL (or a UML profile for OWL) using semantic annotation, or be specified directly as an OWL model.

A Product comprises a complete and consistent set of models for the Product, namely Business Model, Requirements Model, Architecture Model and one or more Platform Specific Model together with the corresponding Executables

III.3.3.2 Methods and techniques

- The Business Resource Model is generally prepared at the same time as the associated Business Process & Roles model, and methods and techniques used are similar, i.e. activities that include brainstorming sessions, structured workshops, interviews and feedback sessions, and detailed modeling using a UML tool.

- The Resources that are of concern to the business will mostly, but not entirely, be discovered from consideration of the things that have to happen in the business (as contained in the Business Process Model). All resources either make those things happen, or are involved in those things that happen, in that, what happens, happens to them.

- In addition, reference should be made to, and consistency maintained with, the Context Statement, since that is a representation at a high level of all the main things of interest to the Product.

- The set of business concepts are then refined into a class model, at the level of detail being considered which defines the interesting attributes of and relationships between all the resources (both actors and artefacts) identified in the Business Process Model.
The conceptual model used is the same as for the Business Process model, namely the Business Modelling Conceptual Model, with additional freedom to add any relevant associations, attributes and state machines, as provided by standard UML.

Packaging of this class model is a modelling choice, but dependencies can be reduced by confining resources to the communities with which they are involved. Thus, each «Community» package would contain two packages, one with the relevant Business Process Model, the other with the relevant Business Resource Model.

III.3.3.3 Deliverables and notation

The Business Resource Model takes the form of a UML class model as part of the PIM for the Product. In cases where it is useful it may be accompanied by state machine models for any of the resources modelled.

III.3.4 Work analysis refinement model (WARM)

III.3.4.1 Modelling objectives

As its name suggests, the WARM is a refinement of the basic Business model which concentrates on "Work Analysis", i.e., which kinds of resources do which kind of work. Specifically, it concentrates on those resources that will form part of the Product being developed, and the behaviour (expressed as Steps in Processes) that they will be required to exhibit.

III.3.4.2 Methods and techniques

The WARM is not a separate model, or even a separately identifiable element in a model. It is a final refinement of both the Business Process & Roles model and the Business Resource model, (both, themselves, distributed amongst a variety of Community models), in which the Steps in the Business Process model are refined and assigned to specific refined Actor Resources. It is from this part of the model that the role, in the business, of a particular IT element, can be ascertained.

The WARM concepts that specialise Actor Resource are illustrated in the figure below:
Figur 16  WARM Concepts – Resource

Resources are refined to identify the IT elements that fulfil roles in the business, as follows:

- A Product in the WARM is an actor resource that fulfils a role in the business and maps to the executable part of a Product.
- A Business Service is a Resource with a role in the business, and represents an IT element that maps to a Business Service Component in the corresponding Architecture model.
- A Workflow is an actor resource that maps to a Workflow Definition in a corresponding Architecture model which contains the business logic of a process and determines the sequencing of steps within any instance of that process.
- A Human/Tool is an actor resource with a role in the business that is fulfilled by a human working with an IT element that maps to a Tool Component in the corresponding Architecture model.
- An Other System is an actor resource that represents an external automatic system that fulfils a role in the Business Process Model.

In addition, so that the model can be complete, we identify:

- A Human (user) is an actor resource that represents a human that fulfils a role in the Business Process Model.
- The System represents the complete IT facility that supports the business including all Products and the Infrastructure.

The WARM concepts that specialise the business process modelling concept of Step are illustrated below:
Figur 17  WARM Concepts – Step

In WARM refinement of the Business Process model, the kinds of step performed by resources in the model are further categorised as follows:

- A Human Step is a step performed by a human with no involvement of the Product being modelled.
- An Extended Step is a Step in which the intermediate states are of interest to the business, and may have to be remembered. This may be either because there are business reasons for such interest, or because other factors, including technology, require that the business be concerned. An extended step is a candidate for choreography by a Workflow.
- A Tool Step is a step performed by a human user interacting with a tool that is part of the Product. The human user will use some form of interactive device (e.g. a GUI) to interact with the Product. A Tool Step is a candidate for realisation by a Tool Component.
- An Immediate Step is a Step that is required to complete as soon as possible, and whose intermediate states are of no concern to the business. It is performed autonomously, with no intervention from a human. An Immediate Step may be mapped to an Operation on a Business Service Component (Process) in the Architecture Model. Each such operation on a Business Service Component in the Architecture will run in an ACID transaction, thereby either completing or leaving state as it was. The Name of the step is the verb or verb phrase that describes the process (for example, "Modify Customer record"). If a resource is specified as part of the name or description (for example "Modify Customer using Info from Sales Person") then the model should identify the relevant Resources.
IV COMET - Requirements model

IV.1 Introduction – COMET Requirements model

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.

Use cases and scenario descriptions are used to capture the functional requirements. These use cases and scenario descriptions are also used to structure some of the non-functional requirements and constraints. Figure 1 shows the work products of the requirements model.

Figure 1: Requirements model work products

The Requirements Model contains the following work products:

- **The Use Case Model.** The Use Case Model describes the Product in terms of actors, use cases and scenario descriptions. The Use Case Model consist of two main parts:
  - **The System Boundary Model,** which identifies the Product under consideration (area of concern) and describes the Product boundaries as well as the main services offered by the Product.
  - **The Use Case Scenario model,** which includes more detailed descriptions of the Product resulting from further analysis using the common use case detailing technique, by diving into a use case discovering new use cases and actors. Use case scenarios are also described in this work product.

- **Prototype.** One or more prototypes may be produced during development of the Product, in order to test understanding of the requirement, or particular aspects of the design (e.g. HCI).

- **Non-functional requirements.** This work product identifies constraints, general requirements and other kinds of non-functional requirements not fitting into the previous work products of the requirements model.
  - Non-functional requirements can be made part of the use case model as these kinds of requirements are associated with use cases according to the use case template. General non-functional requirements that apply for the whole system...
are associated with the system boundary which is also included in the use case model.

- **BCE model:** This work product provides the link between the Requirements model and the Architecture mode and is the output of a technique used to derive the system domain models (architecture model and platforms specific model) from the business domain models (the Business model and the Requirements model).

**IV.1.1 Use Case Model**

The Use Case Model consists of the System Boundary Model and the Use Case Scenario model.

**IV.1.1.1 System Boundary Model**

The System Boundary Model describes the system boundaries, the actors and their responsibilities, and the services offered by the system. Interactions between the system and its environment are identified and might also be detailed by modelling stimuli and responses using UML sequence diagrams.

**IV.1.1.1 Goals**

The System Boundary Model should:

- Identify and describe system boundaries, main services and actors.
- Assure a common understanding of the system and its purpose.
- Identify interactions between the system and its environment.

**IV.1.1.2 Methods and techniques**

The starting point for the development of the System Boundary Model is provided by the Business Model work products, the Context Statement, Vision for Change, Business Process and Role Model and Business Resource Model. Other important sources are the business stakeholders, who should be involved when determining the system boundary model.

Defining the boundaries of the system means finding out what is inside the system and what is outside the system (the outside is what the system interfaces with). The boundaries of the system are determined by the roles of the system in the Business Model. In the System Boundary Model, the boundaries of the system are defined in terms of actors and use cases. The actors are outside the system and represent the roles in the Business Model that interact with the system roles (roles inside the system boundary); the use cases are inside and represent the behaviour of the system roles.

**IV.1.1.3 Example**

Figure 2 shows an example of parts of a system boundary for a meeting room booking example, which is described in a UML use case diagram. The actors represent initiating system actors or external systems used by the meeting reservation system (e.g. the Email information system).
IV.1.1.2 Use Case Scenario Model

The Use Case Scenario Model digs into the identified use cases and describes these in detail. The COMET-S method describes a use case template that is used as a vehicle for this detailing.

IV.1.1.2.1 Methods and techniques

The major input to determining the Use Case Scenario model is System Boundary Model, but the Business Model work products and early versions of the Architecture Model are also valuable input.

The use case template is the baseline for developing the Use Case Scenario Model. Each use case identified in the System Boundary Model is detailed using the use case template. The use case template consists of the following parts:

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use case identification</td>
<td>Use case no. xx &lt;name&gt;. Each use case are identified for later reference</td>
</tr>
<tr>
<td>The UML use case diagram</td>
<td>UML use case diagram for the actual use case. The diagram should include extend and include relationships if there is any.</td>
</tr>
<tr>
<td>Goals</td>
<td>Description of the goal(s) for the use case, derived from the Goal Model in the Business Model. Preferred notation is prose text.</td>
</tr>
<tr>
<td>Actors/Roles</td>
<td>Description of the roles involved in the use case and their responsibilities. Preferred notation is prose text.</td>
</tr>
<tr>
<td>Description of scenarios (formal)</td>
<td>Description of the use case scenarios. This includes the primary scenario and related secondary scenarios. The scenarios are described using UML Activity diagram and/or UML sequence diagram.</td>
</tr>
<tr>
<td>Pre and post</td>
<td>Description of the pre and post conditions for the use case. The post condition will often</td>
</tr>
</tbody>
</table>
conditions correspond to the goals. Preferred notation is prose text. UML state diagram might also be used to describe pre and post conditions as well as intermediate states of a use case. OCL might be also be used to specify pre and post conditions.

| Non-functional requirements | Description of non-functional requirements for this use case. These are described using text in the UML documentation field for the element in question. |

Table 1: Use case template

The description part of the template describes scenarios. A scenario is an elaboration of a use case. Use cases are statements of system services. Scenarios add more detail and describe factors that may result in behavioural variations of a given use case. A Scenario describes the behaviour of the system in a particular situation. The use cases and the set of all scenarios together constitute the complete description of the services of a system. Scenarios are constructed by taking the use case and identifying possible different outcomes and different conditions that may result in different sets of scenarios.

Non-functional requirements are the collection of system requirements that are not directly related to what the system should do.

**IV.1.2 Prototype**

The Prototypes are part of the requirements model. Prototyping is very important to be able to reduce technical risks and to ensure end user involvement. Prototypes are commonly used to emphasise the UI, test critical use cases and test risky parts of the architecture.

It is common to build some prototypes during the elaboration phase to try out some ideas or to determine if a high risk feature is possible. As a result of the prototyping effort, you may decide that a certain feature is not feasible and remove it from the project. If that feature is key to the project it might even be decided to abandon the project.

**IV.1.3 Non-functional requirements**

This work product describes non-functional requirements that do not fit into the use case template. For instance, some non-functional requirements may be general for the whole system to be developed. These are described separately instead of being described with every use case. The distribution aspect of the system as well as interoperation with legacy systems and different software and hardware components might also be better described separate from the use cases.

**IV.1.4 Reference Architecture Analysis**

The Reference architecture analysis is the first step in modelling the architecture, representing an intermediate step between Business Domain Models (Business Model and Requirements Model) and the architecture design. The key source for the analysis is the Use Case Model. Thus, The Reference Architecture Analysis is developed using a use case driven approach and provides the basis for the initial development of the Architecture Model.

**IV.1.4.1 Goal**

The Reference Architecture Analysis should provide the link between analysis and design, in particular the link between the Use Case Model and the Architecture Model.
IV.1.4.2 Methods and techniques

Recall the Reference Architecture with associated tiers and component types.

The first step of this analysis, (typically executed as part of developing the first increment of the Architecture Model you analyse the System Boundary part of the Use Case Model with regards to the Reference Architecture. The System Boundary corresponds to the boundaries of the component of concern for the actual project. E.g., in projects concerning building or replacing an Application Component the System Boundary represents the Application Component of concern, where the actors is what's outside the Application component and the Use Cases is what's provided by the application component.

After the identification of this "main" component the next step is to analyze the System Boundary Model with respect to its set of use cases and actors. The aim of this task is to divide the System Boundary Model into a set of subsystems. Each subsystem will cover a collection of the System Boundary use cases. Thus, a subsystem contains a subset of the use cases of the System Boundary Model. Remark that the subsystems are non-exclusive, implying that a use case might be part of more than one subsystem. The result of this task is a functional decomposition of the main component. It is visualized using by group the use cases of the System Boundary use case diagram adding system boundaries representing the subsystems (example See figure below).

![Subsystem grouping example](image)

**Figur 18 Subsystem grouping example**

The actors/roles are important when it comes to obtaining the most appropriate decomposition as they group related use cases. Thus, the subsystem grouping are related to the actors/roles, and the subsystem design should strive to have as few relationships between actors and subsystems as possible. Related use cases and especially use cases that typically are executed in sequence should be provided by one subsystem. Recall that the subsystems might be overlapping when it comes to the use cases.
IV.2 COMET - Requirements metamodel

IV.2.1 Introduction

The requirements are used to capture system requirements that respond to goals from the business model. Use cases are used to formalise the requirements. Currently, the requirements metamodel is an informal tool that helps the user to structure a specification, and maintain traceability between business goals, requirements, and use cases.

IV.2.2 Model Structure

A requirements model is structured as shown below.

![Diagram](image)

**Figur 19 Structuring Concepts – Structure**

The System Vision contains a high-level textual description, which captures the overall ideas of the product. The System Boundary model is a high-level use case model, which captures the roles and responsibilities of the system. The BCE (Boundary, Control, Entity) Analysis Model is a technique (and an artefact) that details use cases into more detailed analysis diagrams, represented by UML classes/interfaces. The Use Case scenarios describe the scenarios of each system use case, with text and UML interaction/sequence diagrams. Other requirements are described in terms of formal/informal text, each of these requirements are associated with one or more use cases.
IV.3 COMET - Requirements model to service architecture model transformation

IV.3.1 Requirements model to architecture model transformation

We see a possibility of transforming the requirements model to a first draft of the architecture model, for example by generating a UserService for each actor (which we see is quite common the case). However, the transformation is a one-way transformation from requirement model to architecture model. It will constrain the models unnecessarily, especially the requirement model, to maintain a round trip transformation between the requirements model and the architecture model.

The following transformations are possible:

1. Map each actor to a UserService and create an Interface as a provided interface of the actor. For each of the use cases that the actor relates to add a corresponding operation to the Interface. Naming conventions: Use the Actor name to name the UserService and the same for the interface prefixed with an “I”. The operations should be named according to the use cases.

2. Map each <<Manage>> use case to a BusinessService providing CRUD operations as well as find and collection operations for the Resource(s) that are related to the <<Manage>> use case. Naming conventions: use the resource name prefixed with “Manage”

Possible way to handle extends and includes relationships:

1. Include -> reusable UserService with interface providing the “include” service or an operation in the interface of the extended UserService.

2. Extends-> Operation in the interface of the extended UserService

IV.3.1.1 Transformation specification

MOF have defined a metamodel for model transformations below we have specified the transformation rules for the Actor to UserService mapping based on this scheme. Figure 1 describes a simple transformation from a UML use case model to an architecture model with UserService components (classifiers) with interfaces.
Figur 20 Example – from use case models to architecture

The transformation map ‘Actor-2-UserServiceMap’ has one type of source object, an Actor. This signifies that the extent of actors in the source model is input for this transformation (one at a time). Source and target objects are modelled in terms of associations to types from the source and target metamodels, respectively. Source and target associations can be named, which signify named references to the source/target objects. A target association signifies the creation of one instance of the target type. From the example, the target ‘UserService’ signifies a creation of a UserService object. The references to these objects have global scope and can be used in, for example, FeatureMaps in the specification.

The classifier maps are either part of the transformation map or other classifier maps, defined through aggregate associations. An association defines a transformation path that implicitly (or explicitly, through OCL statements) carries source objects. When no constraint is specified, the sources of the owner become sources for the contained element (classifier map).

Figure 2 depicts a detailing of the transformation, which includes simple feature maps. A feature map is modelled as an attribute and may contain assignments or action specifications.

An assignment typically maps basic properties of the source, to basic properties of the target, e.g. their names. An action specification is a more complex operation that e.g. adds a realisation dependency to a target object.

Figur 21 Example – feature map

OCL constraints are used to constrain the input domain for a transformation class, based on the context of the owner. Typically, these define selection criteria for the set of objects to be handled by the receiving transformation class. OCL constraints are assigned to the associations between transformation classes. In the example in Figure 3, the OCL constraint on the association selects all use cases an actor communicates with.
In addition to transforming simple properties within FeatureMaps, more complex operations are needed to transform relational features, like adding an operation to an interface. Two different methods are used to support this. The first one is implicitly performed, by adding properties by referring relations from the metamodel, as in Figure 4.

![Figure 23 Referring to metamodel relations](image)

The second way is to specify feature maps that add a relationship to the target object, as shown in Figure 5.

![Figure 24 Adding properties in a FeatureMap](image)

The feature map accesses the properties of the target object and imposes an addition of a feature, e.g. a new realisation, as in the example. The add operation is assumed to be handled by the target object.

### IV.3.2 Service information model – Class Diagrams

The Service Information Model is presented as a set of UML class diagrams describing the information model for the associated component that is visible through the operations of each of the identified interfaces.

#### IV.3.2.1 Goals

The Service Information Model contains the definition of the information that is passed through the operations of each of the identified interfaces. The interface may be realised by one or more components.
IV.3.2.2 Methods and techniques

When a Requirements Model with a Use Case Model exists, this is a good starting point for identifying types in the Service Information Model. Actors/objects from the sequence diagrams are good candidates for being information types. Also, things passed and returned as input and result can be candidates for objects. Conceptually, it is useful to think of what is communicated (objects and values), who are communicating (Services with interfaces), and how are they communicating (through operations).

The Service Information Model should describe the structure of object types/classes that are part of an interface/feature. These object types are the types of the objects passed/received through the interface. Only non-basic types are described (i.e. types other than Integer, Boolean, etc.). These complex types (entity-types/data-types/features) are modelled using UML class diagrams with relevant operations, attributes and associations. To increase readability, it is often useful to incorporate the Service interface in the diagrams describing its contents. The Service Information Model is a conceptual model representing the things accessible from an interface - it does not suggest any realisation of these.

A rule of thumb for identifying the information types is to look for the nouns and subjects within the area of concern. Some common categories of these are:

- Actors (that are represented in the system)
- Person
- Organisation
- Group
- System/subsystem
- Resources (that are communicated or used by the system)
- Information (e.g. a reservation)
- Products (of the system)

The details of Service interfaces in the Service Information Model often represent concepts, entities or services known in the business. The knowledge of domain experts will often be useful when describing this model.

Given the Service Information Model, boundary conditions for each information object should be investigated. In addition, constraints specific to individual object types will be included in this model.

To verify the correctness, content and completeness of an information model, it is useful to create instance diagrams. The modelling of instance diagrams often reveal the need for remodelling, correcting and/or further detailing of the information model. Especially, the need for constraining the information model.

The constraints are specified using OCL (Object Constraining Language). OCL is part of UML standard. The class model contains the parts of the structure model that require specification of constraints on the state at specific points in time. The constraints in OCL should be accompanied by textual explanations describing the intentions of the formal OCL constraints in natural language.

The Service Information Model should contain a set of UML class diagrams representing the types/classes with attributes and relationships. The information detailed here represents information offered by services of a product. Constraints are written in OCL.

The Service Information Model should be platform independent, i.e. it should use the standard platform-independent data types and omit detailed technology details.
V COMET-S - Service architecture model

V.1.1 Introduction
The emerging UPMS (UML Profile(SOA-Pro) and Metamodel for Services (PIM4SOA)) standard from the OMG (Object Management Group) will provide a basis for platform independent specifications. An open source implementation of the UPMS standard is in development by the European SHAPE IST project. Input to the standardisation work has been provided also by earlier European projects like ATHENA, INTEROP, SWING and SODIUM.

The UPMS (UML Profile and Metamodel) standard from OMG aims to address the need for describing the interactions among the participants in a service oriented architecture independent of technological realizations. The UPMS model should also bridge the gap between the business analysts and the IT developers and support mapping and alignment between enterprise and IT models.

The primary purpose of an SOA is to provide an effective means of connecting consumers with needs to providers with appropriately matching capabilities. It is these participants and their connections that make up the services architecture.

Service Oriented Architecture (SOA) is a way of organizing and understanding organizations, communities and systems to maximize agility, scale and interoperability. The SOA approach is simple – people, organizations and systems provide services to each other. These services allow us to get something done without doing it ourselves or even without knowing how to do it - this provides us with efficiency and agility. Services also enable us to offer our capabilities to others in exchange for some value – thus establishing a community, process or marketplace. The SOA paradigm works equally well for integrating existing capabilities as well as creating and integrating new capabilities.

A service is a capability offered through a well-defined interface and available to a community (which may be the general public) using a standard communications infrastructure and communication protocols. SOA is an architectural paradigm for defining how people, organizations and systems provide and use services to achieve results. SOA-Pro as described in this specification provides a standard way to architect and model SOA solutions using the Unified Modeling Language®, UML®. The profile uses the built-in extension mechanisms of UML to define SOA concepts in terms of existing UML concepts. SOA-Pro can be used with current “off the shelf” UML tools but some tools may offer enhanced, SOA specific, support.

An architectural and business focused approach to SOA
SOA has been associated with a variety of approaches and technologies. The view expressed in this specification is that SOA is foremost an approach to systems architecture, where architecture is a way to understand and specify how things can best work together to meet a set of goals and objectives. The architectures described with SOA may be business architectures, mission architectures, community architectures or information technology systems architectures – all can be equally service oriented. The SOA approach to architecture helps with separating the concerns of what needs to get done from how it gets done, where it gets done or who or what does it. Some other views of SOA & “Web Services” are very technology focused and deal with the “bits and bytes” of distributed computing. These technology concerns are important and embraced, but are not the focus of SOA as expressed here.
SOA-Pro embraces and exploits technology as a means to an end but is not limited to technology architecture. In fact, the highest leverage of employing SOA comes from understanding a
community, process or enterprise as a set of interrelated services and then supporting that *service oriented enterprise* with *service enabled systems*. SOA-Pro enables business oriented and systems oriented services architectures to mutually support the enterprise mission.

SOA-Pro depends on Model Driven Architecture® (MDA®)\(^1\) to help map business and systems architectures, the design of the enterprise, to the technologies that support SOA, like web services and CORBA®. Using MDA helps our architectures to outlive the technology of the day and support the evolution of our enterprises over the long term. MDA helps with *separating the concerns* of the business or systems *architecture* from the *implementation and technology*.

Due to the business and architectural focus of SOA-Pro it is important to understand that terms like *service* can have both a business and technology interpretation. We can understand how a service of our enterprise, perhaps to sell books, is related to a component of our web infrastructure – perhaps a web service to process book orders. Concepts like “Service Contract”, “Participant” and “Service Realization” which you will see in this specification are intended for, and work well for, both business and systems architectures using the SOA paradigm.

**Top down and bottom-up SOA**

SOA-Pro can be used for basic “context independent services”. Such as the common Web-Service examples like “Get stock quote” or “get time”. Basic services focus on the specification of a single service without regard for its context. Since a basic service is context independent it can be simpler and more appropriate for “bottom up” definition of services.

SOA-Pro can also be used “in the large” where we are enabling an organization or community to work more effectively using an inter-related set of services. Such services are executed in the context of this enterprise, process or community and so depend on the services architecture of that community. A SOA-Pro services architecture shows how multiple participants work together, providing and using basic service to enable a business goal or process.

---

\(^1\) “Model Driven Architecture”, “MDA”, “CORBA”, “Unified Modeling Language”, “UML” and “OMG” are registered trademarks of the Object Management Group, Inc.
Diagram Services provides an overview of the SOA-Pro metamodel. Package Services defines a new set of capabilities that can be merged with UML2 to extend it with service modeling capabilities. Packages Partitions and Policies extend Services with some additional capabilities. The various integration packages extend Services with integrations with other OMG specifications. These integrations are done using package merge to weave in possibly optional service modeling aspects and capabilities.

V.1.3 Key Concepts of Basic Services

A key concept is, of course, the service. A service is a capability offered by one entity or entities to others using well-defined "terms and conditions" and interfaces. Those entities may be people, organizations or systems – we call these participants. Participants offer services on Ports with the <<Service>> stereotype. The service port is the interaction point where consumers of the service go to use that service.

The service port has a type which describes how to use that service, that type may be either a UML interface (for very simple services) or a ServiceInterface. In either case the type of the service port specifies, directly or indirectly, everything that is needed to interact with that service – it is the contract between the providers and users of that service.

The diagram to the right shows a participant, "Productions" that has <<service>> port named "scheduling". The type of the service port is the UML interface "Scheduling" that has two operations, "requestProductionScheduling" and "sendShippingSchedule". The interface defines how a consumer of the Scheduling service must interact where as the service port defined that "Productions" has the responsibility to offer that service – which could, for example, populate a UDDI repository. Note that a participant may also offer other services on other service ports.
V.1.4 Service Interfaces

Where a basic service is more complex than can be expressed in a UML Interface, a <<Service Interface>> can be defined as shown below.

Like a UML interface, a <<Service Interface>> can be the type of a service port. The service interface has the additional features that it can specify a bi-directional service – where both the provider and consumer have responsibilities to send and receive messages and events. The <<Service Interface>> is defined from the perspective of the service provider using three primary sections: The interfaces, the Service Interface class and the Behavior.

- **The Interfaces** are standard UML interfaces that are provided or used by the service interface. The interface that is provided specifies the messages that will be received by the provider (and correspondingly sent by the consumer). The interface that is <<used>> by the...
service interface defines messages or events that will be received by the consumer (and correspondingly sent by the provider).

- **The Service Interface and enclosed parts** specify the roles that will be played by the parties involved with the service. The role that is typed by the provided interface will be played by the service provider. The role that is typed by the <<used>> interface will be played by the consumer.

- **The behavior** specifies the interactions between the provider and consumer – the contract of interaction, without specifying how either party implements their role. Any UML behavior specification can be used, but interaction diagrams are the most common.

V.1.4.1 Participants and Service Ports

*Participants*. As represented in Figure 3, are types of actors defined both by the roles they play in services architectures and the services they provide and use. Participants can represent software components, organizations, actors or individuals. Services are provided and used through having *service ports* that have a type compatible with the Service Interface, or UML Interface they must provide and use.

These interaction points use the UML concept of a “port”. A port is the interaction point where a classifier interacts with other classifiers (see Figure 3). A port typed by a *service interface*, is a *service port*.

![Service Interface and Ports Diagram]

*Figure 6*

A *service port* is the point of interaction on a *participant* where a service is actually provided or consumed. On a service *provider* this can be thought of as the “offer” of the service (based on the *service interface*) and on the *consumer*. In summary, the *service port* is the point of interaction for engaging participants in a service via its *service interfaces*.

V.1.4.2 The Service Requisition

Just as we want to define the services provided by a participant using a service port, we want to define what services a participant consumes. Using a service is also defined using a port.
stereotyped as a <<Requisition>>. The type of a requisition port is also a Service Interface or UML Interface, as it is with a service port. The requisition port is the conjugate of a service port in that it defined the use of a service. As we will see below, his will allow us to connect service providers and consumers in a component.

![Diagram](image)

The OrderProcessing participant example, above, shows that OrderProcessing provides the “OrderProcessor” service using the “OrderProcessor” service interface defines on the “accept order” service port and uses the “ShippingRequest” service defined as a <<Requisition>> of the ShippingRequest interface. Note that this requisition is the conjugate of the service which is offered by a shipper, as defined below.

![Diagram](image)

By using service and requisition ports SOA-Pro can define both the service responsibilities and requirements of participants at the business or technical level.

**V.1.5 Key Concepts of Service Architectures**

One of the key benefits of SOA is the ability to enable a community or organization to work together more cohesively using services. This requires an understanding of how people, organizations and systems work together, or collaborate, for some purpose. We enable this collaboration by creating a services architecture model. The services architecture puts a set of services in context and shows how they work together for a community or organization.

Since we want to model how these people, organizations and systems collaborate without worrying, for now, about what they are, we talk about the roles these participants play in services architectures. Roles define the purpose of an entity in a particular context while the Participant classifier (or type) specifies a capability that fills the role in the context of a specific services architecture. Within a services architecture participant roles provide and consume any number of services. The purpose of the services architecture is to specify the SOA requirements of some organization, community or process to provide mutual value. The participants specified in a services architecture provide and consume services to achieve that value. The services architecture may also have a business process to define the tasks and orchestration of providing that value. The services architecture is a high-level view of how services work together for a purpose. The same services and participants may be used in many such architectures, providing reuse.

A services architecture has components at three levels of granularity: The Community Architecture is a “top level” view of how independent participants work together for some purpose. Within a participant, where there is some concept of “management”, there is a Participant Architecture which shows how sub-participants and external collaborators work together and would often be accompanied by a business process. Both the community architecture and participant architecture may be composed from other services architectures and service contracts.

Participants. As represented in Figure 7, are types of actors defined both by the roles they play in services architectures (the participant role) and the “contract” requirements of entities playing those roles. Each participant type may “play a role” in any number of services architectures but must fulfill the requirements of each. Requirements are satisfied by the participant having service ports that have a type compatible with the services they must provide and use.
So the services architecture (or SOA) is a network of participant roles providing and consuming services to fulfill a purpose. The services architecture defines the requirements for the types of participants and service realizations that fulfill those roles. The services architecture may be expressed as a Community Architecture (where the participants are independent entities) or a Participant Architecture under the control of a particular participant’s business process.

A Community Architecture (see Figure 8) is defined using a UML “Collaboration”. The purpose of a collaboration is to show how kinds of entities work together for some purpose. Collaborations are based on the concepts of roles. A role defines how entities are involved in that collaboration (how and why they collaborate) without depending on what kind of entity is involved (e.g. a person, organization or system). We say that an entity “plays a role” in a collaboration. In a Community Architecture, the Participant Roles are filled by Participants and Service Realizations. They define the Service Ports required of the entities that fill these roles.

A key part of a service is the Service Contract, see Figure 9. The service contract defines the terms, conditions, interfaces and choreography that interacting participants must agree to (directly or indirectly) for the service to be enacted – the service contract is the full specification of a service which includes all the information, choreography and any other “terms and conditions” of the service. The service contract is binding on both the provider and consumer of that service. The basis of the service contract is also a UML collaboration that is focused on the interactions involved in providing a service. A participant plays a role in the larger scope of a services architecture and also plays a role as the provider or user of service contracts.
There is also a Service Interface for each of the involved parties in a service contract. An important part of the Service Contract is the choreography. The Choreography is a specification of what is transmitted and when it is transmitted between parties to enact a service. The choreography specifies exchanges between the parties – the data, assets and obligations that go between the service provider, consumer and any other service interface.
The service contract choreography is a UML “Behavior” such as may be shown on an interaction diagram or activity diagram that is owned by the service contract. The choreography defines what must go between the contract roles—service Providers, Consumers and Service Interfaces—when, and how each party is playing their role in that service without regard for who is participating. The service contract separates the concerns of how all parties agree to provide or use the service from how any party implements their role in that service.

The requirements for entities playing the roles of service providers, consumers or service interfaces are expressed in the type of the role, as a UML class. The UML class is “Stereotyped” as a service interface and will provide a UML interface that has all of the operations or signal receptions needed for the role it types—these will be every obligation, asset or piece of data that the entity can receive as part of that service contract. Likewise, the UML class will “use” the UML interfaces of all service interfaces that it is connected to in the services contract—the used interfaces then represent the information, obligations and assets that are sent and received by a participant. Providing and using corresponding UML interfaces in this way “connects the dots” between the service contract
and the requirements for any participant playing a role in that service as provider or consumer.

Note that some “SOA Smart” UML tools might add functionality to help “connect the dots” between service contracts, services architectures and the supporting UML classes.

It should also be noted here that it is the expectation of SOA-Pro that services may have communications going both ways – from provider to consumer and consumer to provider and that these communications may be long-lived and asynchronous. The simpler concept of a request-response “function call” or “Operation” is a degenerate case of a service, and can be expressed easily by just using a UML interface.

Participants can engage in a variety of contracts. What connects participants to particular service contract is the use of a role in the context of a services architecture. Each time a service contract is used in a services architecture it implies that there must be a compliant service port on a participant – this is where the participant actually offers or uses the service.

One of the exciting capabilities of SOA is that it can work “in the large” where independent entities are interacting across the internet to internal departments and processes. This suggests that there is a way to “decompose” a community architecture and see how services can be implemented by using still other services. A participant can be further described by its internal architecture, a participant architecture – that architecture can also use internal or external services, participants as well as business processes and other forms of implementation. Our concern here is to show how the “internal structure” of a service participant is described using other services. This is done by defining a participant architecture.

The Participant Architecture is a class that is expected to have its external contract defined in some other portion of the model and its internal composition described using Service Contracts. The Participant Architecture shows how other participants and can be the “parts” of the composite and provide and use services to help implement the composite participants responsibilities. The participant architecture corresponds to a UML classifier with “internal structure” using parts and part connections to define a composite organization or a composite I.T. component. SOA-Pro supports decomposing services into finer grain services to any level.

The specification of a SOA is presented as a UML model and those models are generally considered to be static, however any of SOA-Pro constructs could just as well be constructed dynamically in response to changing conditions. The semantics of SOA-Pro are independent of the “design time” or “runtime” decision. For example, a new or specialized service contract could be negotiated “on the fly” and immediately used between the specific participants. The ability of technology infrastructures to support such dynamic behavior is just emerging, but SOA-Pro can support it as it evolves.

The interaction points use the UML concept of a “port”. A port is the interaction point where a classifier interacts with other classifiers (see Figure 5). A port typed by a service interface, provider or consumer is a service port.

![Figure 1 – Example Participant with Service Port](image-url)
A service port is the point of interaction on a participant or service realization where a service is actually provided or consumed. On a service provider this can be thought of as the “offer” of the service (based on the service contract) and on the consumer of a service this can be thought of as the “request” or “requisition” for that service (based on the service contract). The services architecture helps connect these offers and requests for a purpose. In summary, the service port is the point of interaction for engaging participants in a service and is bound to the service contract via its provider, consumer or service interface.

One of the exciting capabilities of SOA is that it can work “in the large” where independent entities are interacting across the internet to internal departments and processes. This suggests that there is a way to “decompose” a services architecture and see how services can be implemented by using still other services. A participant or service realization can be further described by its internal architecture – that architecture can also use internal or external services, participants and service realizations as well as business processes and other forms of implementation. Our concern here is to show how show the “internal structure” of a service participant is described using other services.

The UML collaboration serves as a basis for a realization by a set of components, that can be described using UML2.0 composite structure models. The services architecture is a specification component whose parts may be typed by candidate services or participants. This specification must be realized by some participant or agent.

In the next steps, we provide a realization for the specification and each of its parts. We must find, build, buy, etc. a participant capable of realizing each part of our services architecture.

Figure 2  Services organized into a services architecture
Each part of the specification must be realized by a part with the same name and compatible type in the realizing participant. This is shown as a composite structure, where the services provided and required are shown through the <<service>> and <<requisition>> stereotypes on ports, or alternatively using the conjugate type sign ~ such as shown for the InvoicingService. Each of the ports will have attached provided and required interfaces with operations and signals with signatures and input, output arguments. These elements can again be attached to ontology models for semantic annotations, and thus provide a full specification of the interfaces.

**Conclusion and Future work**

The UPMS metamodel and profile is currently being finalised by the UPMS merge team, with a timeline planned for the end of 2008. Further work will also include the integration with related OMG standards, such as BPMN 2.0 for Business process modelling, BMM for Business Motivation Modeling, ODM for Ontology modelling and IMM for Information Model Modeling.

**V.2 COMET - Service reference architecture**

The basis of the service reference architecture is a generic system architecture, or reference architecture. The reference architecture defines a set of logical tiers, each of which consists of a set of components (Figure 1).
The 4+2 tier reference architecture is separated into a local single user-space, called the user service domain, and a shared transactional business-space called the business service domain. As Figure 1 indicates, the user service domain consists potentially of four local tiers and the business service domain consists of two shared tiers. The 4+2 tiers are as follows:

- **The user interface tier** provides presentation and user dialog logic. Sometimes, it is useful to make the presentation and user dialog separation explicitly, in particular to support reuse of user dialog on multiple platforms with different graphical capabilities, e.g. Web, PDA and Mobile phones.

- **The user service tier** provides the user’s model, which may include user session logic and user-side representations of processes and information. It is an abstraction for a set of business services, making the business service provision (and the communication mechanisms) transparent to the user interface tier.

- **The user resource service tier** provides local persistence services. It is only present if persistence capabilities are required by components in the user service tier, e.g. to support disconnected or off-line operations, or to provide smart caching mechanisms. Access to the local storage is provided by local adapters, which typically are APIs to file storage or light-weight single-user databases.

- **The business service tier** provides components that represent business functionality and pervasive functionality (vertical vs. horizontal services). This tier provides enterprise-level services, and is responsible for protecting the integrity of enterprise resources at the business logic level. Components in this tier can be process-oriented, entity-oriented or workflow-oriented. For performance reasons, entity-oriented components are typically not exposed outside of this tier.

- **The resource service tier** provides global persistence services, typically in the form of databases. Resource adapters (e.g. JDBC or ODBC drivers) provide access, search and update services to databases and its data stored in a database management system (DBMS) like Oracle or Sybase.

V.2.1.1 Component Types

The figure below shows the component types of the architecture.
Figure 2: Component types
We have two abstract component types, tier components and composite components.
VI COMET-S - Platform specific model

VI.1.1 Introduction

As the name indicates this model defines the result of mapping the service model to an implementation on a particular infrastructure. Figure 1 illustrates the work products in the platform specific model.

![Platform specific model](image)

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 developed:

- UMT Configuration Model (Implicit PSM in a code generator tool)
- 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.

VI.1.2 Platform profile model – JEE Metamodel

VI.1.2.1 Goals

The purpose of this model is to specify the system in alignment to the actual technology profile for the specific platform (e.g. based on the UML profile for EJB).

In the example used in the OBLIG part of INF5120 we have been using JEE, and this is also illustrated further in the part V notes on the Buyer/Seller Purchase order example.
VI.2 Bibliography – Model Driven Development


