# Office of the
# Government Chief Information Officer

# AN INTRODUCTION TO
# OBJECT ORIENTED METHODOLOGY (OOM)

## [G52a]

Version : 1.5

**Oct 2009**

## COPYRIGHT NOTICE

# TABLE OF CONTENTS

# 1. PURPOSE

The objectives of this document are:

- to introduce the Object Oriented Methodology (OOM) and its benefits;
- to give an overview of the process stages in OOM;
- to introduce the major techniques in OOM.

# 2. SCOPE

The intended readers of this document are IT managers, end users and those who would like to have an overview of the OOM.

This document covers a brief overview of the OOM, its benefits, the processes and some of the major techniques in OOM.

# 3. REFERENCES

## 3.1 STANDARDS

Nil.

## 3.2 OTHER REFERENCES

Nil.

# 4. DEFINITIONS AND CONVENTIONS

## 4.1 DEFINITIONS

Nil.

## 4.2 CONVENTIONS

Nil.

# 5. INTRODUCTION

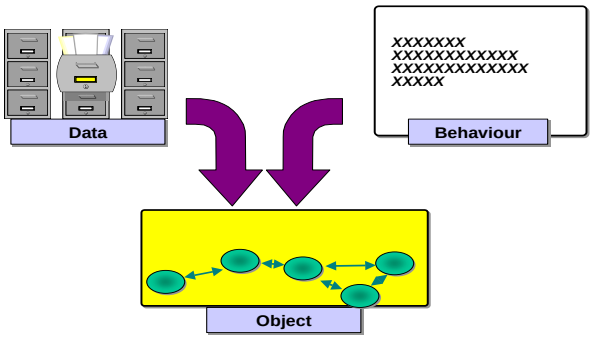
## 5.1 OVERVIEW OF OOM

Object Oriented Methodology (OOM) is a system development approach encouraging and facilitating re-use of software components. With this methodology, a computer system can be developed on a component basis which enables the effective re-use of existing components and facilitates the sharing of its components by other systems. By the adoption of OOM, higher productivity, lower maintenance cost and better quality can be achieved.

The ultimate objective of OOM is *application assembly* - the construction of new business solutions from existing components. The components are combined in different ways to meet the new requirements specified by the user community. Only completely new functionality will have to be built to complete the solution.

Software components can be assembled to form applications.

OOM applies a single object model that evolves from the analysis and design stage and carries all the way down to the programming level. An object contains both the data and the functions that operate upon that data. An object can only be accessed via the functions it makes publicly available, so that all details of its implementation are hidden from all other objects. This strong encapsulation provides the basis for the improvements in traceability, quality, maintainability and extensibility that are key features of well-designed Object Oriented systems.



## 5.2 HISTORY OF OOM

The use of OOM for analysing and designing systems began to mature towards 1990 with the launch of methodologies from the three industry-leading methodologists : Ivar Jacobson, Grady Booch and James Rumbaugh.

In 1989, the Object Management Group (OMG) was founded. The mission of OMG is to establish industry guidelines, detailed object management specifications and common frameworks for application development.

One of the best-known specifications maintained by OMG is the Unified Modeling Language (UML). The UML is a language for specifying, visualizing, constructing, and documenting the deliverables of software systems, as well as for business modelling and other non-software systems.

# 6. BENEFITS OF OOM

The following benefits can be achieved by adopting OOM:

- Improve productivity

  Application development is facilitated by the reuse of existing components which can greatly improve the productivity and facilitate rapid delivery.

- Deliver high quality system

  The quality of the system can be improved as the system is built up in a component manner with the use of existing components which are well-tested and well-proven.

- Lower maintenance cost

  The associated property of traceability of OOM can help to ensure the impact of change is localised and the problem area can be easily traced. As a result, the maintenance cost can be reduced.

- Facilitate reuse

  With this approach, a computer system can be developed on a component basis that enables the effective re-use of existing components. Opportunities for the reuse are facilitated by the accumulation and proper management of an inventory of reusable components either developed internally or acquired externally.

- Manage complexity

  The use of OOM eases the process in managing complexity. By the breaking down of a complex solution into different components and with each component encapsulated (e.g. treated as a black box) from others, complex development can be better managed.

# 7. CONSIDERATIONS FOR ADOPTING OOM

Given the characteristics of the methodology, OOM best suits IT projects which exhibit any or all of the following characteristics:

- Projects of medium to large scale

  The component-based development approach of OOM manages large-scaled applications by breaking down the complex solution into components. Furthermore, the modelling techniques of OOM are best used to model medium to large scaled applications, which usually involve complex business logic.

- Departments/project teams with a planned series of developments within a similar business area

  One of the major benefits of adopting OOM is reuse. Application development is facilitated by the reuse of existing components which can greatly improve the productivity and facilitate rapid delivery. In order to obtain the full benefits of reuse, the department/project team is preferably one which has planned series of developments within a similar business area so that the benefits of reuse can be realised during the implementation of the projects.

- Projects with an inventory of suitable components available either in-house or in the market

  The OOM facilitates the construction of new business solutions from existing components. With OOM, reuse of the available components in-house and/or off-the-shelf commercial packages/components may be achieved more readily. The components are combined in different ways to meet the new requirements specified by the user community.

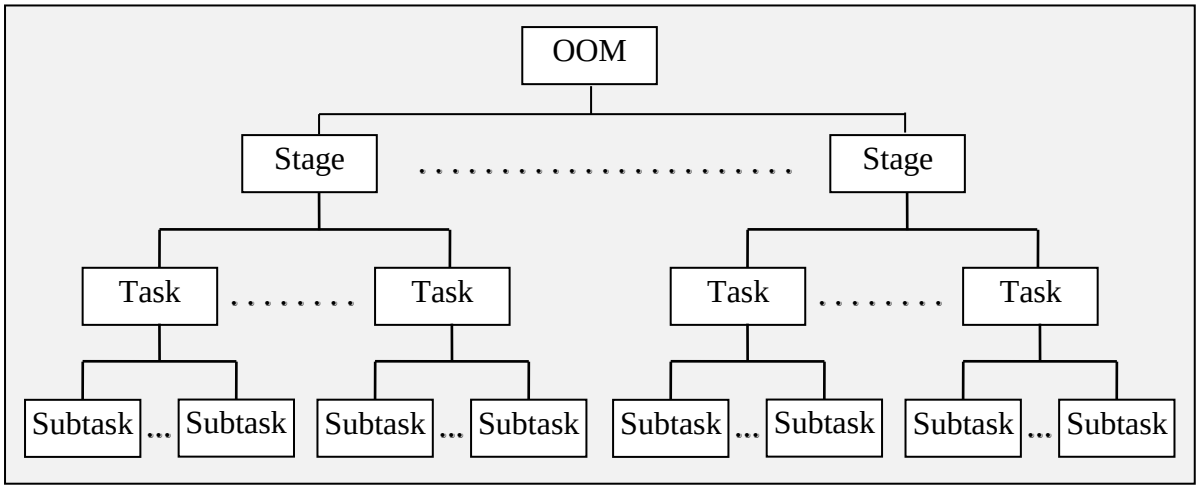
- Projects with development/implementation environment which provide adequate support for object technology

  The supporting features of the development/implementation environment, such as OO programming languages and appropriate development tools, are very important for realising the benefits of OOM. Without appropriate support in the development/implementation environment, the OO design may not be able to be implemented seamlessly and so affecting the benefits which can be achieved. While most of the latest development/implementation environments have already employed object technology, this factor should not be neglected totally.

# 8. THE OOM PROCESS

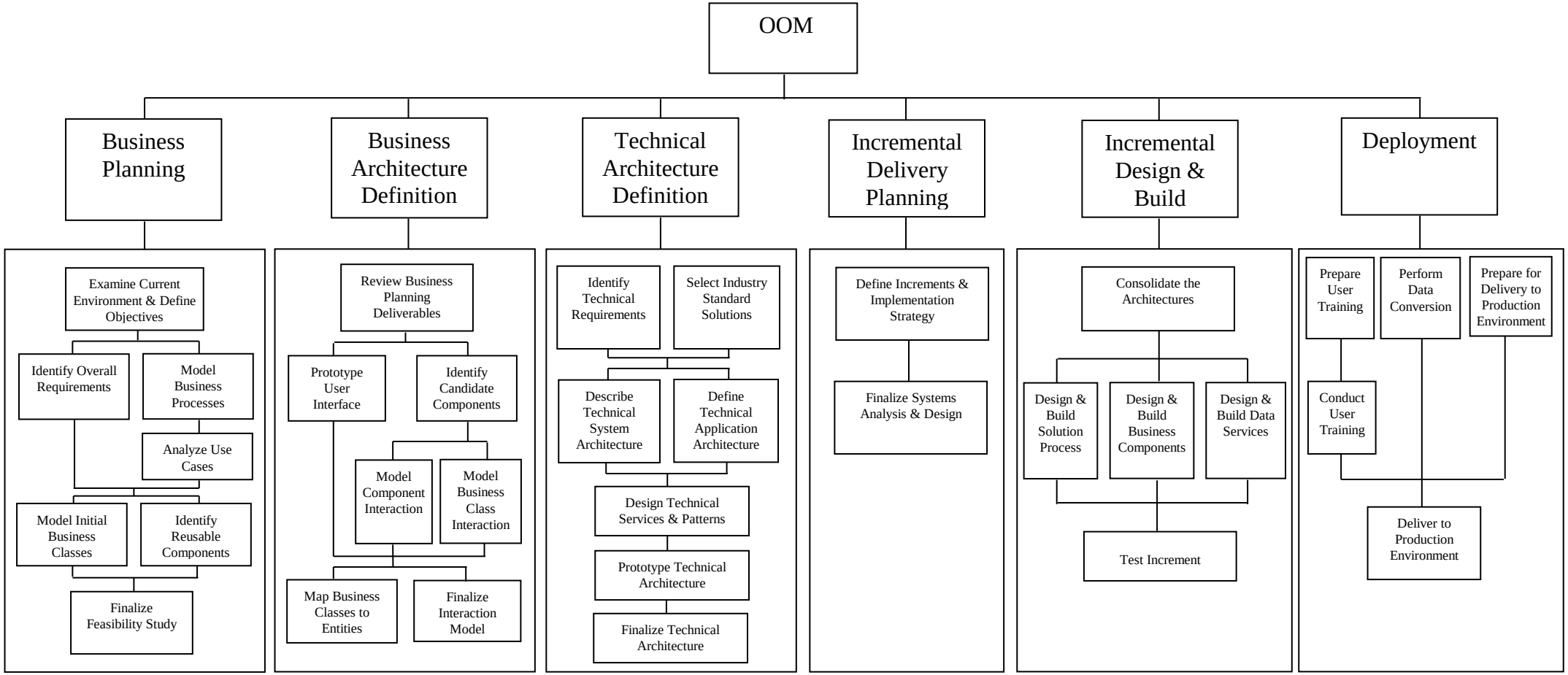
## 8.1 STRUCTURE OF OOM

The structure of OOM is divided into Stages. Each Stage consists of a number of tasks and each task is further decomposed into sub-tasks. The following diagram depicts clearly the structure of OOM.



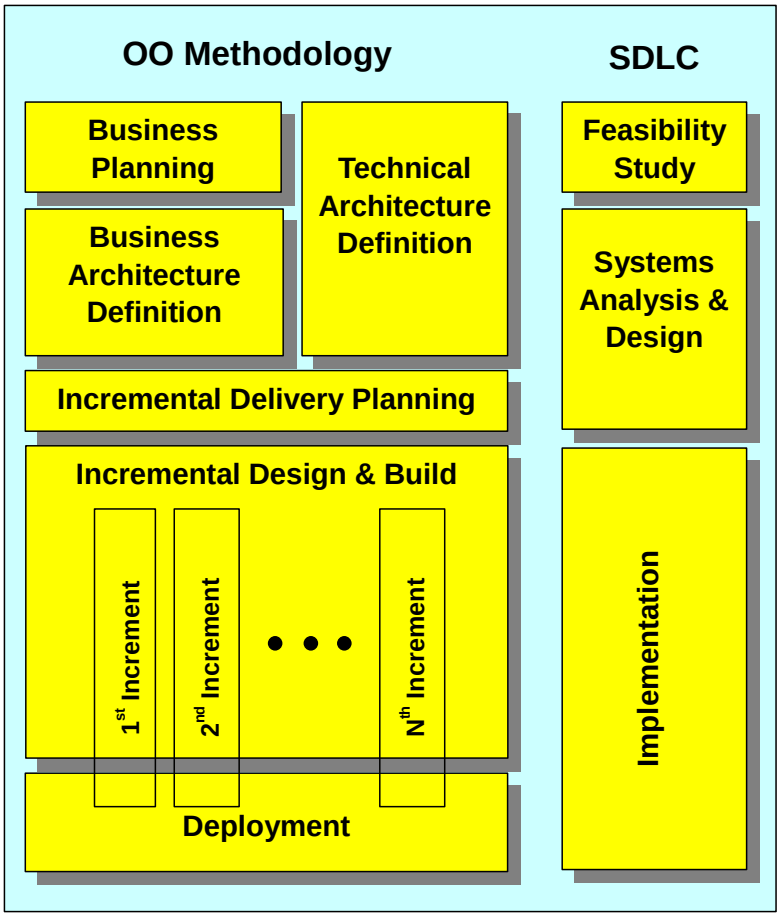
## 8.2 OVERVIEW OF OOM PROCESS

The following diagram shows clearly the Stages and Tasks in OOM :-



## 8.3 OOM AND SDLC

The mapping between OOM stages and SDLC (System Development Life Cycle) is depicted in the following diagram:



## 8.4 BUSINESS PLANNING

*Objectives*

The objectives of the Business Planning stage are:

- To identify the context and scope of study;
- To define the requirements that support the scope;
- To assess the reusability of existing components;
- To assess the feasibility of the project.

*Overview*

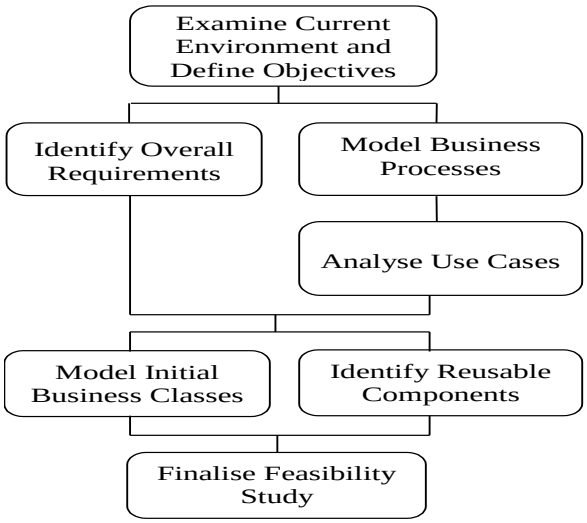
The Business Planning stage is usually conducted through a series of meetings and workshops with the business management and business users. These meetings initiate the development process by establishing a mutual understanding of the objectives, scope, user requirements and assess the feasibility of the development project.

The overall requirements including both functional and non-functional requirements are documented in the Requirements Catalogue. In parallel, a Business Process Model (BPM) is developed jointly with the users. This model describes the to-be business processes within the scope of study. Meetings and workshops with the users that perform the business processes are held to discuss, understand and document the business events and processes involved.

The processes in the BPM are then analysed to identify the detailed functional requirements of the IT system. The areas to be automated will be expressed by the Use Case Model by which the scope of the system are defined and agreed. Based on the agreed functionality, an initial Class Model will be developed. The Class Model defines both the data as well as the operations required for the manipulation of the data.

In order to minimise delivery effort, opportunity on reusing existing software components available in-house or in the market will be assessed. After that, a high level assessment on the costs and benefits and the feasibility of the project will be made.

The tasks of Business Planning stage are shown in the diagram below:




## 8.4.1 Examine Current Environment and Define Objectives

This task involves the examination of findings in previous studies, the understanding of current systems and business missions. Project team members and users should then be able to establish the objectives of the project.

## 8.4.2 Identify Overall Requirements

This task identifies the overall requirements including both functional and non-functional requirements which are documented in the Requirements Catalogue. The functional requirements should be documented in a high level way as the detailed requirements will be expressed by Use Cases in later tasks.

## 8.4.3 Model Business Processes

This task models the proposed business process flow. User workshops are conducted to discuss, understand and document the events and processes involved in terms of Process Hierarchy Model and Process Thread Model. Subsequent systems design and development should be focus on supporting the business processes.

### 8.4.4 Analyse Use Cases

This task captures the detailed functional requirements of the system in terms of Use Cases. Each Use Case describes a way in which the system can be used that will provide value to the user. It documents the interaction between the computer system and the actor. An actor can be a person, time or another computer systems.

### 8.4.5 Model Initial Business Classes

This task models the basic business classes. In general, these classes will lead to related entities in the data model. Therefore, the initial business class model can serve as a basis for database sizing.

### 8.4.6 Identify Reusable Components

This task is carried out in parallel with the previous task. It aims to identify services that are available from existing components such that delivery effort can be minimised.

### 8.4.7 Finalize Feasibility Study

This task is not required if a Combined FS/SA&D Phase approach is taken. It finalizes the Feasibility Study with cost/benefit analysis, impact analysis and estimation on subsequent implementation effort, etc.

## 8.5 BUSINESS ARCHITECTURE DEFINITION

*Objectives*

The objectives of the Business Architecture Definition stage are:

- To define the architectural elements and components;
- To identify component services requirement;
- To identify dependency between components;
- To understand how the requirements will be implemented in the component based infrastructure.

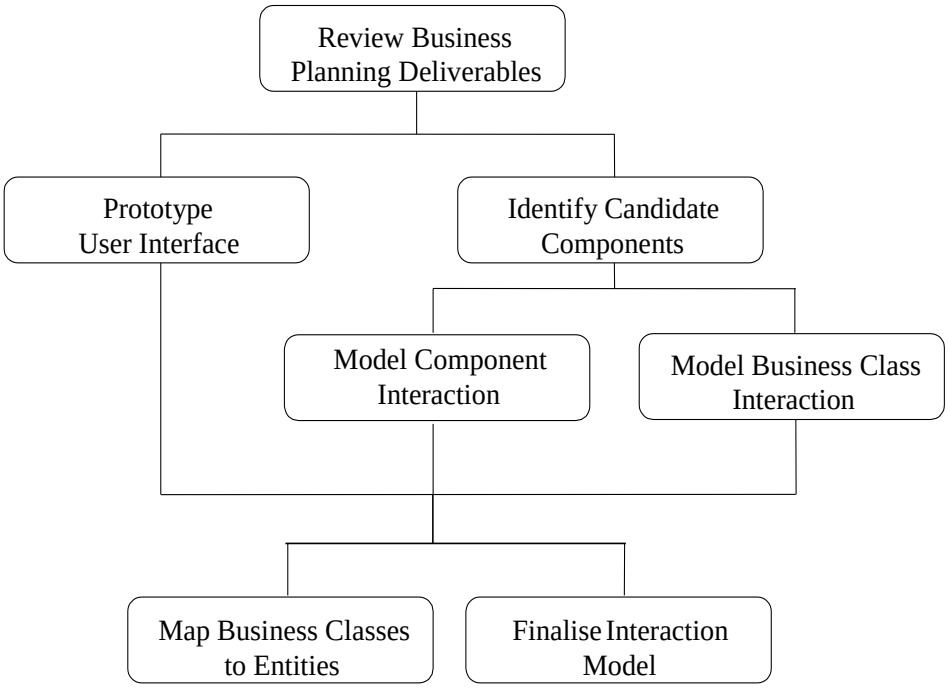
*Overview*

Business Architecture Definition focuses on gaining an increased understanding of the users' needs and defining a solution that will satisfy the needs.

System requirements as specified in the Use Cases will be analysed. The component services required to fulfil the system functionality will be identified. Component services will be realised by designing the detailed business classes and their collaboration within the components.

In order to gain advantage from component based development, services available from reusable components and the dependencies between them will be identified. Parallel to the above activities, prototypes of screens and reports will be made.

The tasks of Business Architecture Definition stage are shown in the diagram below:




## 8.5.1 Review Business Planning Deliverables

This task is required if Feasibility Study (FS) had been done as a separate phase. It aims to confirm the correctness and completeness of the deliverables produced during Business Planning. Documentation should be updated if there are changes to the requirements after the FS phase.

## 8.5.2 Prototype User Interface

This task provides a way to visualise Use Cases. It also identifies the data input/output for a Use Case and hence the data items required. Screen prototypes are built and jointly reviewed with users.

### 8.5.3 Identify Candidate Components

This task clusters logically related classes into components. A component can be treated as a black box that will deliver services of business values. If there are reusable components from other projects/systems, they will be identified and documented as well.

### 8.5.4 Model Component Interaction

This task models the interactions between components so as to support the functionality of Use Cases. It also defines the services required for each component.

### 8.5.5 Model Business Class Interaction

This task models the interactions between business classes so as to realise component services. It also defines the operations required for each business class.

### 8.5.6 Map Business Classes to Entities

This task maps the classes in class model to entities in relational model. It is required if a relational database is being used.

### 8.5.7 Finalize Interaction Model

This task finalizes component interaction and business class interaction with checking on completeness, consistency and level of detail before the model can be handed over to the Design and Build stage.

## 8.6 TECHNICAL ARCHITECTURE DEFINITION

*Objectives*

The objectives of the Technical Architecture Definition stage are:

- To define the technical architecture elements and components;
- To identify technical services requirement;
- To identify dependencies between technical components and between technical and business components;
- To define, model and test the technical components and infrastructure.

*Overview*

The activities of the Technical Architecture Definition (TAD) will be executed in parallel to the Business Architecture Definition (BAD) stage. While BAD focus on the business requirements, the TAD stage focus on technology and architecture sides.

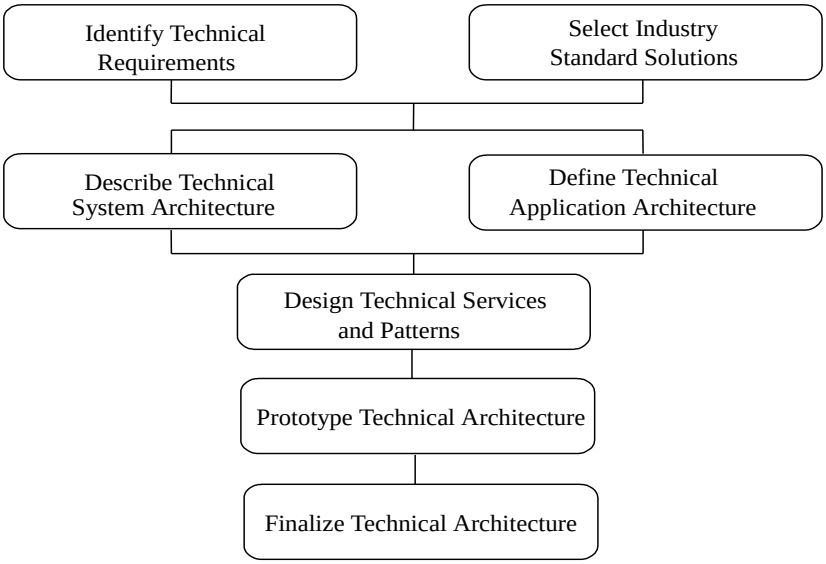
The modelling of the technical architecture will focus on defining:

- The Technical System Architecture and
- The Technical Application Architecture

The Technical System Architecture describes the hardware and system software to be used by the application system under development, testing and production. The Technical Application Architecture defines the architectural layers and their responsibilities/services for the application.

A proof-of-concept application prototype may need to be developed to test and verify the technical components and infrastructure. This is usually required for SA&D but may not be necessary for FS.

The tasks of Technical Architecture Definition stage are shown in the diagram below:




## 8.6.1 Identify Technical Requirements

This task specifies technical requirements needed to support the business application. Technical requirements may include encryption, security, authentication, transaction, access control and logging etc. Non-functional requirements are basic inputs to technical requirements.

## 8.6.2 Select Industry Standard Solutions

This task aims to study, select and adopt industry standard solutions, frameworks and best practices where possible. It involves comparison of technical requirements to the services provided by the architectural frameworks/solutions considered. Organisation standards should be taken into consideration in the selection process.

## 8.6.3 Describe Technical System Architecture

This task defines the tools, environments and software solutions that support the application. Appropriate tools are selected to support all of the different technologies needed to build the type of application in an integrated and consistent manner.

### 8.6.4 Define Technical Application Architecture

This task defines how the application is constructed by responsibility and provides a clear separation between the layers in the system. The responsibilities are mapped to different implementation techniques.

### 8.6.5 Design Technical Services and Patterns

This task designs technical services and creates patterns that developers will follow. The technical services serve as a delivery vehicle and let application built on top of them.

### 8.6.6 Prototype Technical Architecture

This task is usually required for SA&D but may not be necessary for FS projects. It encompasses the development of a "Proof-Of-Concept" application prototype. The prototype should illustrate and prove the fulfilment of major requirements, concepts and technologies as a result of integrating the Technical System Architecture and Technical Application Architecture.

### 8.6.7 Finalize Technical Architecture

This task finalizes technical architecture based on the results of the "Proof-Of-Concept" review.

## 8.7 INCREMENTAL DELIVERY PLANNING

*Objectives*

The objectives of the Incremental Delivery Planning stage are:

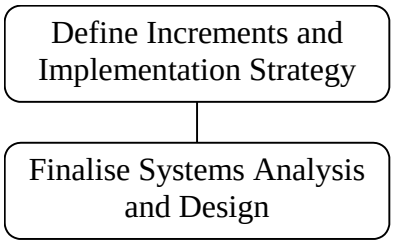
- To group Use Cases into increments which can be delivered in an incremental manner;
- To develop an Incremental Delivery Plan;
- To identify potential benefits of the development project;
- To produce an estimate of the development cost;
- To finalize the Systems Analysis and Design activities.

*Overview*

An increment delivers a number of complete Use Cases. Since each Use Case describes a way in which the system can be used, they can deliver value from the moment of deployment. Based on the prioritisation of Use Cases developed in the Business Planning stage, the most important Use Cases are delivered first.

An Incremental Delivery Plan is created at this stage to facilitate the Incremental Design and Build so that development proceeds incrementally. Impact analysis and costs and benefits analysis will be conducted. All the deliverables produced for System Analysis & Design will be finalized.

The tasks of Incremental Delivery Planning stage are shown in the diagram below:



### 8.7.1 Define Increments and Implementation Strategy

This task defines implementation/delivery strategy in terms of increments. Each increment comprises a set of complete Use Cases. Use Cases are assigned with delivery priority. The priority is derived primarily from the value to the business represented by the Use Case.

### 8.7.2 Finalize Systems Analysis and Design

This task finalizes the System Analysis and Design with cost/benefit analysis, impact analysis and estimation on subsequent implementation effort, etc.

## 8.8 INCREMENTAL DESIGN AND BUILD

*Objective*

The objectives of the Incremental Design and Build stage are:

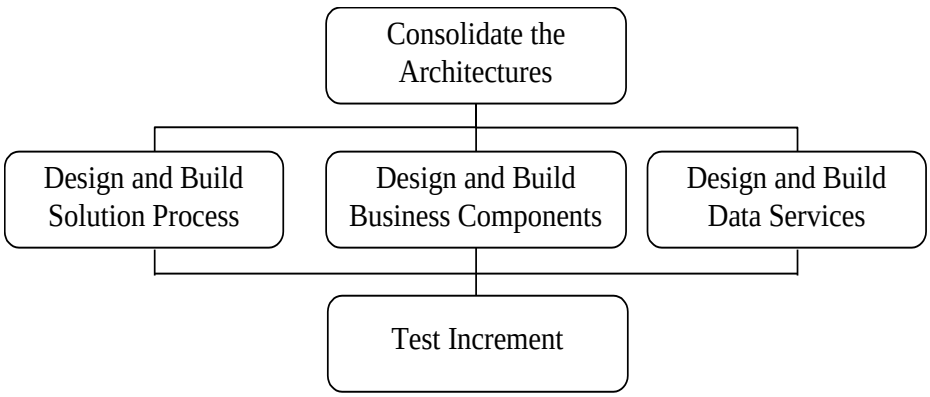
- To develop the business solutions in an incremental manner so as to deliver business value as rapidly as possible;
- To test the code created, ensuring that it is error free and behaves according to the specification given in the use cases.

*Overview*

This stage will take the final deliverables of the Business Architecture Definition as its starting point. The components and their classes will be mapped onto the technical architecture from the Technical Architecture Definition and will subsequently be optimised from reasons of performance, maintainability etc.

The optimised classes will be realised using the class concept as provided as part of the programming language. The programs will be tested, accepted by users and made ready for deployment.

The tasks of Incremental Design and Build stage are shown in the diagram below:



### 8.8.1 Consolidate the Architectures

This task consolidates Business Architecture Definition and Technical Architecture Definition before implementing the application. Each business component is reviewed in turn. Its position in the architecture, its responsibilities and the technologies to be used are identified.

### 8.8.2 Design and Build Solution Process

This task involves the building of interface objects and control objects. The solution process links up the services provided by business components to deliver functionality as specified in the Use Cases.

### 8.8.3 Design and Build Business Components

This task involves the building of business classes and interfaces of the business components that encapsulates the business logic. The business services provided by business components will be used by the Solution Process.

### 8.8.4 Design and Build Data Services

This task involves the building of database tables and data services that manage persistency of data. The data services will be used by business components so that no changes are required to the business components when the underlying storage structure is changed.

### 8.8.5 Test Increment

This task involves the testing of code created. The execution of program is tested against the behaviour as specified in the Use Cases. Testing is conducted at system level and across components. Finally, users will undertake acceptance testing to ensure their requirements and expectations are fulfilled.

## 8.9 DEPLOYMENT

*Objectives*

The objectives of the Deployment stage are:

- To deploy constructed, tested and user accepted system into production;
- To perform data conversion;
- To conduct user training.

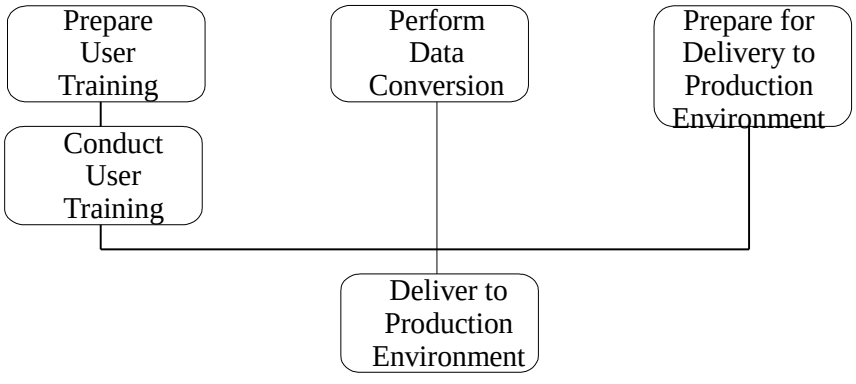
*Overview*

Incremental deployment could be adopted depending on the nature of the business, project constraints, and the benefits achievable.

If incremental deployment is adopted, the solutions built and tested for each increment are delivered into a working environment incrementally. Steps necessary to commence operation of the system are performed. This may include necessary adjustment to the hardware and system software configuration, instructions given to the operations personnel who will be operating the system, and software libraries loaded with the production versions of the application software.

If incremental deployment is not adopted, deployment will only be conducted after the last increment had conducted the final system testing and user acceptance testing, including the regression testing for all other increments.

The tasks of Deployment stage are shown in the diagram below:



### 8.9.1 Prepare User Training

This task develops the training materials and plans the training schedule. The scope of training material could be based on each increment to be deployed to the users.

### 8.9.2 Conduct User Training

This task conducts training sessions to train future users of the system on how it operates. The relevant training should be completed before each increment is placed into production.

### 8.9.3 Perform Data Conversion

This task converts the data required from existing data sources to a format accessible by the new system. This may involve suspension of transaction processing, backup of existing data and verification of data after conversion being executed.

### 8.9.4 Prepare for Delivery to Production Environment

This task compiles the Implementation Manuals. It also involves the training of operation staff, adjustment of hardware and system software configuration.

### 8.9.5 Deliver to Production Environment

This task deploys an increment or increments in the production environment. Production version of the application software is loaded into production library.

# 9. MAJOR TECHNIQUES IN OOM

## 9.1 BUSINESS PROCESS MODELLING

The Business Process Model describes the activities of the business areas that are the target of the project, whether these processes can be the subject of automation or are purely manual in nature.

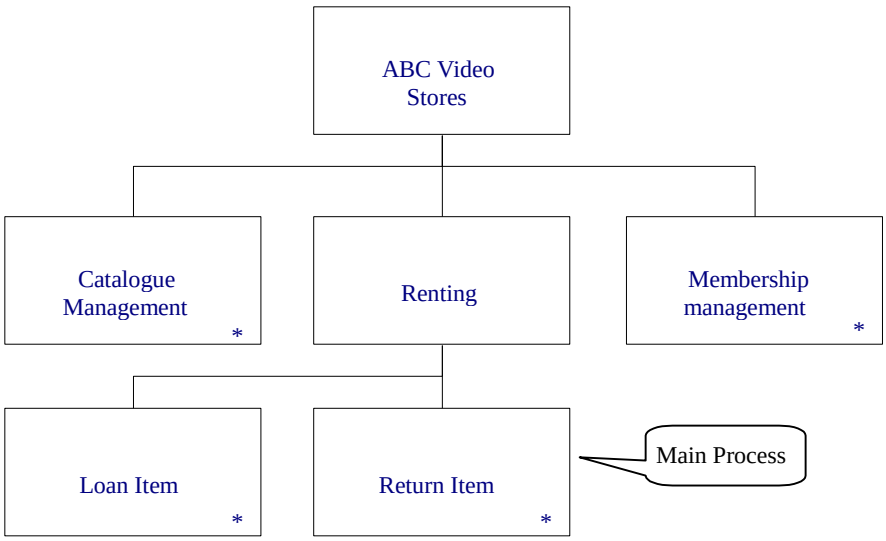
Within the OOM process, the focus of the Business Process Model is to help identify the requirements for a later phase of systems development. Typically the model will only include sufficient information and detail to help with this overall objective.

The Business Process Model is composed of the followings:

- process hierarchy;
- process threads.

The process hierarchy is a functional decomposition of the processes within the enterprise (or segment of the enterprise). The root of the hierarchy represents the operational unit being modelled, the next level the major process within it and so on. The leaf nodes represent the Main Processes.

Example of Process Hierarchy Diagram



Example of Business Process Description

| **Process** | Description |
|---|---|
| ABC Video Stores | A company that rents out video to the company members. |
| Catalogue Management | Manages the rental items in the company. |
| Renting | Manages the rental process. |
| Loan Item | The process of loaning items in the store to the members. |
| Return Item | The process of handling returned items from the members. |
| Membership Management | Manages the membership details of the company. |

A Main Process is composed of Elementary Business Processes, which can be identified by the criteria of being carried out:

- by one person;
- at one time;
- in one place.

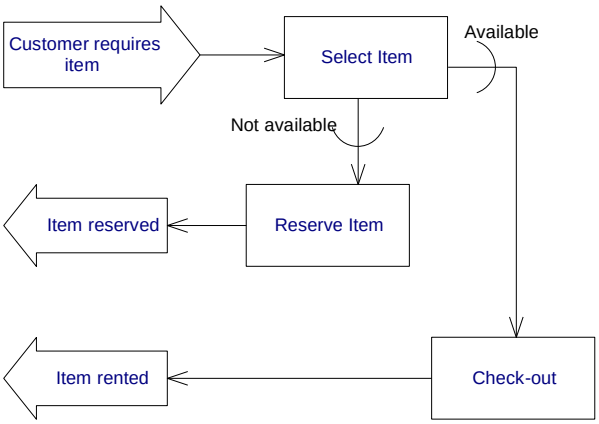
Process Threads show the flow of control between the elementary business processes. In addition to a simple linear flow of control, decision points and process breaks can be modelled.

Process threads are modelled in terms of:

- business events;
- business processes;
- transitions;
- process breaks.

Example of Process Threads Diagram

**Loan Item**




Example of Process Threads Description

| Process Name | Description |
|---|---|
| Select Item | The customer arrives at the store and walks around to look for the titles he is interested in renting. |
| Reserve Item | The customer would normally double check with the Store assistant that the item he had been looking for is not in the store, check when the item will be available and make a reservation if he is interested in renting the item after it is available.<br>If the customer wants to reserve the title, the Store Assistant will enter his membership number and the ID or Name (full or partial) of each title to be reserved. |
| Check-out | The customer arrives at the check out counter and presents his membership card. If he does not have a membership card, a temporary card will be issued for him. The membership number will be recorded and a check is done if the membership had expired. If expired, the customer will be referred to the membership desk.<br>The ID of each item is entered into the system and the item will be marked as rented. After the last item, a receipt will be printed, giving details of the rented titles, the date to return and the payment amount.<br>The customer will pay the amount via cash, cheque or credit card. If payment cannot be made, the whole rental will be cancelled. If everything is OK, the customer will leave with the items. |

## 9.2 REQUIREMENTS CATALOGUE

The Requirements Catalogue documents the overall requirements of the new system. It concentrates on the functionality to be provided by the required system from end users' point of view.

The requirements in the Requirements Catalogue should be documented in a high level way. Detailed requirements are to be specified in terms of Use Cases in later tasks.

Example of Requirements Catalogue

**Req-0001 Process Rental**

Priority:

High

Functional requirement:

Provide on-line processing of rental items including reservation of unavailable items and rental of available items.

Frequency of use:

Around 150 enquiries per day

Non-functional requirement:

Response time should be less than 5 seconds.

Proposed solution:

Implement the above on-line function as request.

## 9.3 USE CASE MODELLING

The Use Case Model describes the functional requirements that are to be implemented from the point of view of the actors to the system. Since actors are external to the implementation, the Use Case Model also defines the boundary of the implementation.

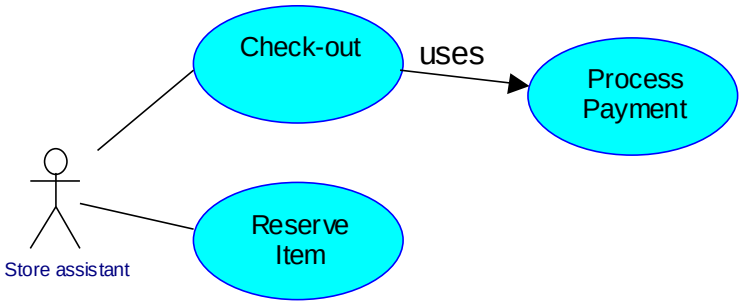
The model has two major constituents:

- actors;
- Use Cases.

Actors are the roles played by the users of the system. It may be human users, other systems which are clients of the implementation and timed events.

Use Cases describe the interaction between the actor and the implementation. They are written in technology neutral terms and in natural language, using users' terminology.

Example of Use Case Diagram




Note:

i. The "Select Item" business process in the example of Business Process Model does not have a corresponding Use Case. This is because only business processes that are going to be automated will be documented as Use Cases.

ii. Since payment processing is common to the entire application system, a separate Use Case "Process Payment" is extracted from the Use Case "Check-out" to facilitate reuse.

Example of Use Case Description

**Check-out**

**Intent:**

To check out a rental item, collect payments, and make the rented item not available for further renting.

**Used By:**

[Actors]

Store assistant

[Used Use Cases]

Process Payment

**Description:**

The actor enters the Customer No. from the membership card.

The system will check the expiry date of the membership.

If expired, customer is referred to membership counter and end Use Case.

The actor then enters the item numbers of the items the customer wants to rent.

The system will set the items to “rented” and compute the total amount to be paid.

The payment is handled using the Process Payment Use Case.

If payment is not successful, items are re-set to “available”, otherwise, a receipt is printed.

End of Use Case.

**Maximum Frequency:**

300 per day

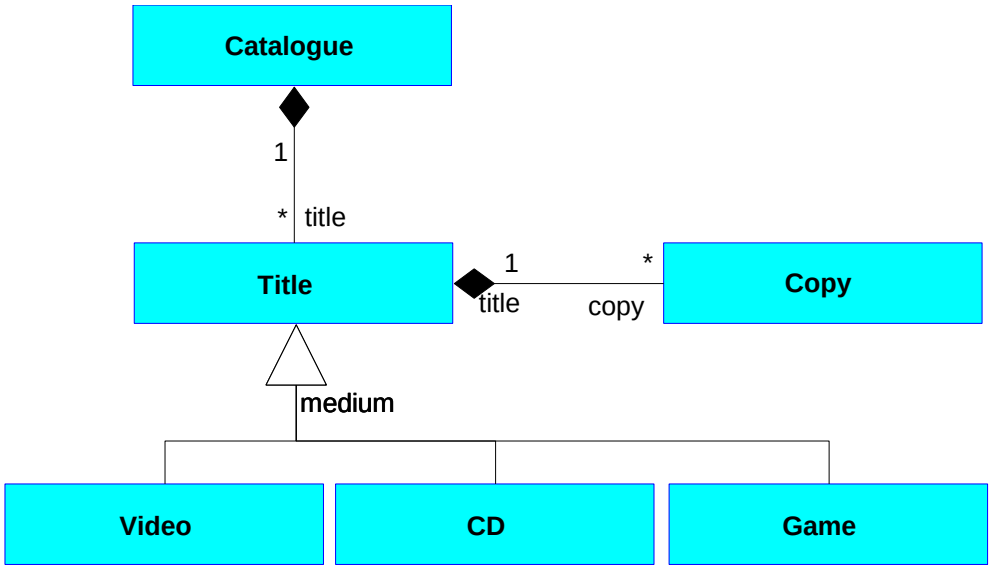
**Average Frequency:**

150 per day

## 9.4 CLASS MODELLING

The modelling of classes is one of the most important techniques within the OOM. It builds on the concepts from entity-relationship modelling in structured methods and enhances with a wider set of relationships that can be modelled to facilitate the reuse of existing classes. Classes are containers for functionality as well as data.

Example of Class Diagram




The above example illustrates that a Catalogue is composed of many Titles. A Title is composed of many Copies. We can also understand from the diagram that Video, CD and Game are kinds of Title. They all share certain common attributes and behaviors.