Chapter 3: System Development Methodology

3.1 Introduction

In this chapter, the Object-Oriented Methodology will be explored deeply in order to develop an Object-Oriented Data Automation System for Microsoft Excel Files. Here, the Unified Approach (UA) has been chosen as the methodology of system development.

Furthermore, the Unified Modelling Language (UML) will be used as the modelling tools. An explanation regarding system development phases, which are the Object-Oriented Analysis and the Object-Oriented Design phase will also be included in this section.

3.2 Object-Oriented Methodology

Object-Oriented (O-O) methodology is a different approach of system development compared to the traditional approach which is based on functions and procedures calls. O-O is based on the development of self-contained modules or objects that can be easily replaced, modified and reused. It encourages a view of the world as a system of cooperative and collaborating objects (Steve, C. and John, D. 1994).

3.2.1 Justification of Choosing O-O Technique

This project will be developed using O-O programming technique. Several justifications have been made to choose to use the O-O programming technique. Below are the justifications of choosing O-O technique.

i. Reusability of classes

   Object-Oriented programming technique provides functionality such as extends or implements that enable the reusability of existing classes. Furthermore, in this project there will be a lot of objects that need to be extended or implemented in order to establish the connection with Microsoft COM objects. Therefore, O-O technique is chosen to develop this project.

ii. Interfacing with Microsoft COM objects

   System developer could use the Interface class provided in any O-O programming language (e.g Java) to invoke the underlying objects that are developed in other programming languages such as C++. The OODA System for Microsoft Excel Files needs to communicate with all the C++ classes constructed and stored in a Dynamic Link Library (DLL) file format.

iii. Supports enterprise application

   All the classes developed using O-O technique can be extended and implemented to support application at enterprise level. The OODA System for Microsoft Excel Files can be used as an office application for administration work. The system can be installed and used at any workstation. It can also be installed in a central server where other clients (workstation) can access the program. All classes could be migrated and placed at any machine without facing any difficulties.

iv. Promotes integrity

This project needs to extend, use or implement many core classes which are already developed and have proven their stability and reliability. With all the ready-made classes, the system integrity is guaranteed. Sun Microsystems Inc. has provided the Java Development Kit (JDK) that comprises many core classes which are very useful. Furthermore, it could shorten the system development period.

v. Higher level of security

Object-Oriented language such as Java needs to be compiled before it is deployed and invoked by other classes. The binary format of the class file has managed to hide the original source code. It doesn't easily expose the source code and it could be safely distributed or installed in any machine.

3.2.2 The Advantages of Object-Orientation

The advantages of Object-Oriented system development methodology, defined by Steve.C and John.D (1994) are stated as below:

i. Higher level of abstraction

Any object that is developed using object-oriented approach should support the abstraction or encapsulation of its' attributes and methods. Changes made within an object are not exposed and hence it will not affect the development of other classes.

As long as the syntax of methods invocation is stated clearly, the process of designing, coding, testing and maintaining a program would be much simpler and manageable.

ii. Seamless transition among different phases of software development

Compare to the traditional software development approach, the methods and attributes are all defined and grouped together in one program. Moving from one phase to another phase is a complicated task because each method and attribute has to be modelled, developed and tested under different perspectives. This is because there is no standardization in defining the methods and attributes required for developing the system. However, the object-oriented approach uses a set of modelling language (UML) from the analysis phase until the coding and testing phases. This will then create a clearer and more robust environment of system development.

3.3 Object-Oriented Techniques

Knowing that the implication of using an object-oriented approach in system development will contribute to an easier way of analysis, design, coding, testing and maintenance, hence it is a need to study which of the O-O techniques that can be applied in developing this project. Nowadays, there are a few O-O techniques that are widely used by system developers. These techniques no doubt have their own uniqueness and special functionality in guiding system developers to build up an object-oriented system. However, it has some weaknesses as well. A brief explanation of the technique chosen for developing this project is included in Section 3.4.

3.4 Unified Approach (UA)

Unified Approach is an approach that combines the best practices of methodologies developed by Booch, Rumbaugh and Jacobson (Ali Bahrami, 1999). UA is not a new methodology. It consists of the combination of the processes, guidelines and with the utilization of the Unified Modelling Language (UML) notations and diagrams. It is proven to reach a better understanding of the object-oriented concepts and system development.

Eventually, each method developed by Jacobson, Booch and Rumbaugh has its strengths and weaknesses as well. The Object Modelling Technique (OMT) developed by Rumbaugh is well suited for describing the object model or the static structure of the system. OMT provides one of the strongest tool sets for the analysis and design of an O-O system. However, this model cannot fully express the system requirements.

Next, the methodology introduced by Booch helps the system developers to design system using the object paradigm. It only covers the analysis and design phases of an O-O system. It has been criticized for utilizing a large set of symbols.

Finally, Jacobson et al. methodologies, which include Object-Oriented Business Engineering (OOBE), Object-Oriented Software Engineering (OOSE) and Objectory, cover the entire life cycle and stress traceability between different phases in O-O system development. It can model the traceability of system component for both forward and backward. However these methodologies do not treat object-oriented design to the same level as Booch's methodology.

With the combination of the best practices introduced by Rumbaugh, Jacobson and Booch, a more well-designed O-O system can be produced.

3.4.1 Phases in Unified Approach

The process and components of UA are:

i. O-O analysis

ii. O-O design

iii. Layered approach

iv. Continuous testing

v. UML modelling

vi. Repository for object-oriented system development patterns and frameworks.

Figure 3.1 shows the phases in Unified Approach.

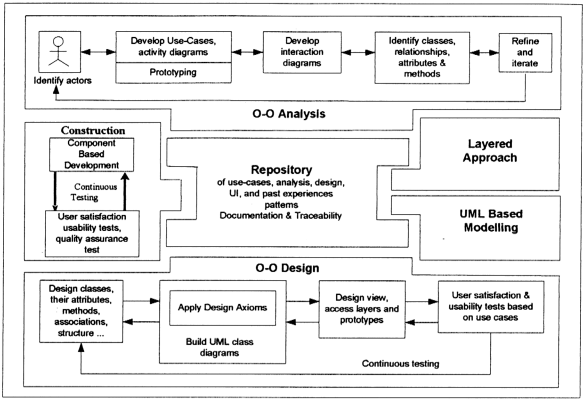
Identify actors
Develop Use-Cases, activity diagrams
Prototyping
Develop interaction diagrams
Identify classes, relationships, attributes & methods
Refine and iterate
O-O Analysis
Construction
Component Based Development
Continuous Testing
User satisfaction usability tests, quality assurance test
Repository
of use-cases, analysis, design, UI, and past experiences patterns
Documentation & Traceability
Layered Approach
UML Based Modelling
O-O Design
Design classes, their attributes, methods, associations, structure ...
Apply Design Axioms
Build UML class diagrams
Design view, access layers and prototypes
User satisfaction & usability tests based on use cases
Continuous testing

Figure 3.1 Phases in Unified Approach

3.4.2 Object-Oriented Analysis

Object-oriented analysis is a process that is responsible for extracting the user needs toward an Object-oriented system. It is a core process that must be carried out in order to produce system which can satisfy user needs. In the analysis phase, the domain problem is being identified. It is important to clearly understand the domain problem before any system development is being carried out.

In order to develop this project, the user requirements and their problems when manipulating the Microsoft Excel files must be clearly identified first. A further description of this analysis phase is included in Chapter 4.

The outcome of the object-oriented analysis phase will be a list of models that describe what the system will do. However, these models will not explain how the system will function. Besides that, the analysis is being conducted from the user's perspective and not the technology perspective. The Use-case diagram shown in Chapter 4 will explain the system flow from user's perspective.

Basically, there are five steps to perform in the Object-oriented analysis phase:

i. The actors of the system (user) is identified

ii. Create a simple business process model, which is the UML Activity diagram

iii. Develop use-case

iv. Develop the interaction diagram

v. Identify classes of object.

3.4.3 Object-Oriented Design

The object-oriented design phase introduced in UML has combined Jacobson's analysis and interaction diagram, Booch's object diagram and Rumbaugh et. al's domain models. Therefore, with the best practices extracted from three of them, all the processes involved in system development are traceable. Basically, an object-oriented system development process must be traceable across requirements, analysis, design, coding and testing phases.

In this phase, the design of the classes, attributes, methods, association structure and protocols will be produced. Basically, there will be three layers of objects that need to be developed. These layers of objects are the Access layer, View layer and Business layer.

As soon as the design of objects is completed, user satisfaction and usability tests can be carried out in order to verify and validate the system. Testing must be done by referring to the use-case diagram, which is prepared during the analysis phase. Iterative testing can refine the quality of the design. Thus, the outcome of the system development process will be able to fulfil users' needs.

3.4.4 Iterative Development and Continuous Testing

The process of an Object-oriented system development is described as an iterative process where the system is tested until the users are satisfied with it. The purpose of this continuous testing is to identify as many weaknesses in the system as possible. With this process of identifying, the end-users will find the system beneficial to their work and hence improve their quality and quantity of work.

A proper testing plan has to be developed in order to carry out the process of iterative and continuous testing. From day one of the development of this project, the coding that was built is tested again and again until it works properly. Whenever there is any error or bug, the coding have to be modified and incrementally developed until it can be transformed into the actual system application.

3.4.5 Unified Approach Proposed Repository

The UA has proposed a repository that should be able to store any objects from previous projects that might be useful in future system development. All definitions of the data elements belong to an object have to be accessible and searchable. Hence, a faster and shorter time of development process can be reached.

All the Components Object Model (COM) developed by Microsoft Corp. are made available and can be referred easily. The reason behind this is that these COM were built using C++ language, which is an Object-oriented language as well. With a good object repository, any system enhancement can be carried out and developed without any conflicts.

3.4.6 Layered Approach

In UA, it has introduced an approach which separates the functions of interface from the functions of business and the functions of data access. There are three layers that must be clearly identified and segregated. These layers are the Access Layer, Business Layer and View Layer.

The Access Layer contains all classes that are responsible for request and retrieve data from the databases or any file-based repositories. This layer doesn't interact with any object that is displaying data or receiving data from the end-users. In this project, the access layer will contain objects that execute Microsoft Excel file opening, reading, writing as well as saving data into it.

Secondly, objects in Business Layer are responsible for handling all the business logic, business flow and the guidelines of executing the system. Business layer will invoke functions or procedures provided by the Access Layer in order to retrieve or insert the business data. This layer also interacts with the View Layer and pass relevant information to it.

Finally, the View Layer is responsible for interacting with the end-users. The objects which belong to this group will accept requests from users and invoke the respective Business Layer objects in order to response to the users. The View Layer will not have a direct communication with the underlying Access Layer.

The purpose of having these three distinguished layers of object is to create a more focused and independent objects development environment. It is good to create objects that represent tangible elements of the business yet are completely independent of how they are represented to the user (through an interface) or how they are physically stored (in a database or file).

3.5 Unified Modelling Language

The methods employed and applied in the Unified Approach explained earlier include the use of Unified Modelling Language (UML) for modelling. UML is a modelling language for specifying, visualizing, constructing, and documenting the artefacts of a system-intensive process. It is emerged from the unification that occurred in the 1990s following the "method wars" of the 1970s and 1980s. It was originated from three of the most prominent methodologists in the information systems and technology industry, namely Grady Booch, James Rumbaugh, and Ivar Jacobson (the three Amigos).

3.5.1 Definition of UML

UML is a modelling language. It is used to specify, visualize, construct, and document the artefacts of a system-intensive process. Within a system-intensive process, UML is used to describe methods that are applied as a process to derive or evolve a system. Diagram 3.1 shows a brief explanation of the usage of UML (Fowler, M., Kendall, S., 1997).

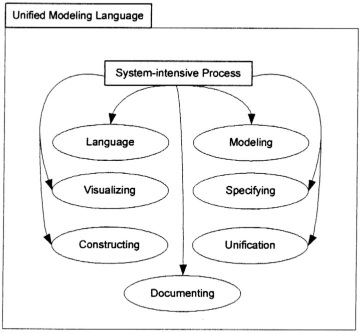

Figure 3.2 Unified Modelling Language

As a language, UML is used for communication. That is, a means to capture knowledge (semantics) about a subject and express knowledge (syntax) regarding the subject for the purpose of communication. The subject is the system under discussion.

Apart from that, UML acts as a modelling language. It focuses on understanding a subject via the formulation of a model of the subject (and its related context). The model embodies knowledge regarding the subject, and the appropriate application of this knowledge constitutes intelligence.

Regarding unification, UML unifies the information systems and technology industry's best engineering practices across types of systems (software and non-software), domains (business versus software), and life-cycle process.

UML is also applied for specifying systems. It can be used to communicate "what" is required for a system, and "how" a system may be realized. This can be done by the creation of different types of UML diagrams which will be explained later.

When applied to visualizing systems, UML can be used to visually depict a system before it is realized. This feature will help to cut down the implementation cost as the system is visually presented to the users first for clarifying specifications and avoiding misunderstanding.

Furthermore, UML helps in constructing systems where it can be used to guide the realization of a system similar to a "blueprint". UML applies in documenting systems as well. It is can be used for capturing knowledge about a system throughout the system development life cycle.

3.5.2 Justification of applying UML

The reason of applying UML in this project is mainly because of the characteristics represented by UML itself. As explained in the previous section, UML is a modelling language.

Building a model for software prior to its construction is an essential process. It is as important as having a blueprint for building a large building. A good model can act as a medium of communication from analysis phase to development phase in a system development life cycle.

The process of data automation for Microsoft Excel files comprises a lot of static and non-static classes. The static classes of object would be the COM objects which are developed and provided by Microsoft Cooperation. The non-static classes would be the Java classes specially designed for segregating an Excel file as well as combining different portions of data into one Excel file.

Due to the complexity that will occurred, visualization and modelling of the classes of object will provide a clear picture on how to design the program. Besides that, modelling can help to identify the weaknesses as well as the strengths of the design of the whole system architecture. All and all, the use of visual notation to represent or model a problem can provide several benefits relating to clarity, familiarity, maintenance and simplification.

To achieve clarity, it is always easier to use graphical representations or visual representation than from listings of codes or tables of numbers. A graphical representation helps in identifying errors and omissions. Besides that, it will also help the users to understand the system which is being represented.

Familiarity occurs when there are similarities in using form of model, which the information is actually represented. As all of the system developers are using the same standards for modelling, it will automatically create an environment of universal understanding.

Visual notation can improve maintainability of a system. The visual identification of locations to be changed and the visual confirmation of those changes will reduce errors. Therefore, it is easier and faster as well as fewer errors will occur in the process of making those changes.

Simplification of system designs can be achieved by using a higher level of representation that is clearly defined. In the context, UML is a kind of higher level of representation that helps to construct a general form of visual notation.

3.5.3 UML Diagrams

A small set of nearly independent views of a model is best to be used to represent a complex system. Each model may be expressed at different levels of fidelity. UML has defined nine graphical diagrams for modelling complex system:

i. Class diagram (static)

ii. Use-case diagram

iii. Behavior diagram (dynamic)

   a. Interaction diagram – sequence diagram, collaboration diagram

   b. Statechart diagram

   c. Activity diagram

iv. Implementation diagram

   a. Component diagram

   b. Deployment diagram

In this project, these UML diagrams will be used to model the architecture of classes of objects, which are going to be used to manipulate the Microsoft Excel files.

The first diagram, which is the Class diagram (also referred as object modelling), is the main static analysis diagram. It shows the static structure of the model. A class diagram represents a collection of static modelling elements. Besides that, it also shows the relationships among those elements.

Secondly, the use-case diagram which is introduced by Ivar Jacobson in the Object-Oriented Software Engineering (OOSE) method, represents a specific flow of events in the system. A use-case corresponds to a sequence of transactions. Each transaction will involve actors (outside of the system) and the internal objects. The interactions between them can be used to model the real implementation of the system. Before the system is actually developed, it is important to show this diagram to the end-users of the system so that they can verify and validate each system flow.

Events happen dynamically in all systems. In an object-oriented system, objects are created and destroyed, objects send messages to one another in an orderly fashion, and in some cases, external events trigger operations on certain objects. The different states of an object can be modelled using dynamic diagrams. Table 3.1 shown below will explain different usage for each type of diagrams under this category.

Table 3.1 Dynamic Diagrams

Dynamic Diagrams			
No	**Type**	**Sub type**	**Descriptions**
1.	Interaction diagram	Sequence diagram	Shows an interaction arranged in a time sequence. It is modelled within a lifeline which represents the object's existence during the interaction.
2.	Interaction diagram	Collaboration diagram	Represents collaboration, which is a set of objects related in a particular context. It shows the messages exchanged among the set of objects.
3.	Statechart diagram	--	Shows the sequence of states that an object goes through during its life in response to outside stimuli and messages.
4.	Activity diagram	--	Represents the variation or special case of a state machine. It triggers the performance of operations and the transitions of processes that describe the entire business process.

Lastly, the UML diagrams also consists of a type of diagram called implementation diagram. This type of diagram shows the implementation phase of system development. In detail, it describes the structure of the source code and the run-time implementation structure. There are two types of implementation diagrams:

i. Component diagram

   It models the physical components in the design of a system. The physical components are such as the source code, user interface and the program execution.

ii. Deployment diagram

This diagram is used together with the component diagram to show how physical modules of codes are distributed on various hardware platforms.

Figures 3.3 through 3.10 shown below are the graphical representation of each type of diagram provided in UML.

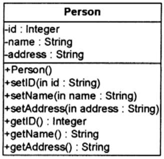

Figure 3.3 Class Diagram

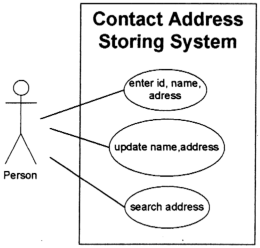

Figure 3.4 Use-Case Diagram

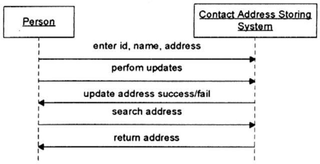

Figure 3.5 Sequence Diagram

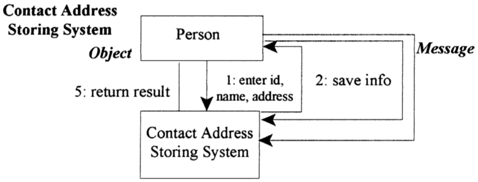
Contact Address Storing System
Object
Person
Message
1: enter id, name, address
2: save info
5: return result
Contact Address Storing System

Figure 3.6 Collaboration Diagram

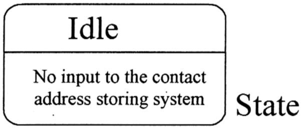

Figure 3.7 Statechart Diagram

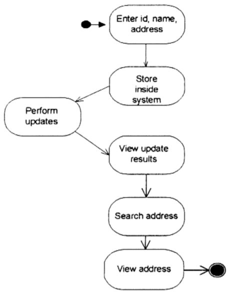

Figure 3.8 Activity Diagram

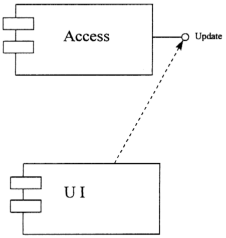

Figure 3.9 Component Diagram

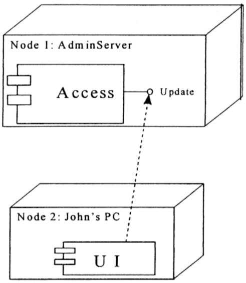

Figure 3.10 Deployment Diagram

3.6 Summary

This chapter has explained the object-oriented methodology that is going to be used to develop this project. The Unified Approach has been chosen in this project for its strength in combining the best practices, processes, and guidelines along with UML notations and diagrams for better understanding of object-oriented concepts and object-oriented system development. Furthermore, it utilizes the methods and technologies such as, layered approach and promotes repository for storing any objects that are useful for future system development.

Next chapter will cover the Object-Oriented Analysis (OOA) phase which is the most important phase for an O-O system development.