Chapter 4: Object-oriented Analysis

4.1 Introduction

In this chapter, the main focus is all the processes carried out as well as the output produced during the system analysis stage. As discussed in previous chapter, the Unified Approach (UA) chosen to develop this project requires five steps that need to be performed in an O-O analysis phase. These five steps are namely first, identify the actors. Second, produce a business process model, followed by a use-case model, then identify the system interaction with the actors and last, produce the classes of objects.

4.2 Current Practice

First of all, it is important to be able to clearly understand what are the normal practices being used by the users when they need to perform "cut & paste" onto the data contained in an Excel file. Why is this important? With the understanding of the users' normal practices, it is easier and clearer to enable the developer to identify the problems they are facing.

No doubt, Microsoft Excel application does provide a way for its users to copy (ctrl-c), or cut (ctrl-x) and paste (ctrl-v) any portion of data contained in an Excel file. However, the users have to open several Excel application windows (for each Excel file) if they need to cut/copy data from several Excel files. The situation becomes more troublesome if the amount of data extraction becomes huge. Hence, it might

slow down the computer's performance due to the limitation of the Random Access Memory (RAM).

This project suggests a way to automate the process of data extraction and data combination from several Excel files.

4.3 Identify Actors

The process of identifying actors is the first activity to carry out before any construction of objects. It is as important as identifying classes, structures, object association relations, object attributes and its behaviour (Fowler, M., Kendall, S., 1997).

Users of the system developed in this project would be those who need to deal with Microsoft Excel files. These users may come from different kinds of business fields such as accounting, financing, building tendering, and auditing. They conduct their business activities by using many Microsoft Excel files.

Regardless of the different positions hold by the users, the term that is used to represent them is Excel File Administrator. The Excel File Administrator is the only type of actor that is going to interact with the Object-oriented Data Automation System for Microsoft Excel Files.

Basically, there are two roles played by the Excel File Administrator defined in this project. These two main roles are described in the following table.

Table 4.1 Two main roles of the actor - Excel File Administrator

Role	Description
Data extraction	Extract (copy) portions of data from an Excel file and save the data in a new Excel file. The main purpose is to separate the data into different files for distribution to different groups of readers.
Data combination	Combine data from several Excel files and save it in one Excel file. The main purpose is to reduce the number of files which represent the same item. This will allow the readers to open one Excel file instead of several files to view the data.

4.4 Business Process Model using UML Activity Diagram

In this project, the business process model identified in O-O analysis stage is only a basic model. This basic model doesn't explain the details of every single process, procedure or step involved in the completion of a business process.

The main advantage of developing a business process model is to understand the system and the user requirements. It also aids in developing use cases that will be shown in section 4.5.

4.4.1 Current System Representation

The business process which represents current system implementation has to be modelled (at early stage) in order to identify the weaknesses of current system practice and possible enhancement to automate Excel file manipulation. Figure 4.1 presents an activity diagram for Excel file segregation that is practiced in current system. Figure 4.2 shows the activity diagram for Excel files combination.

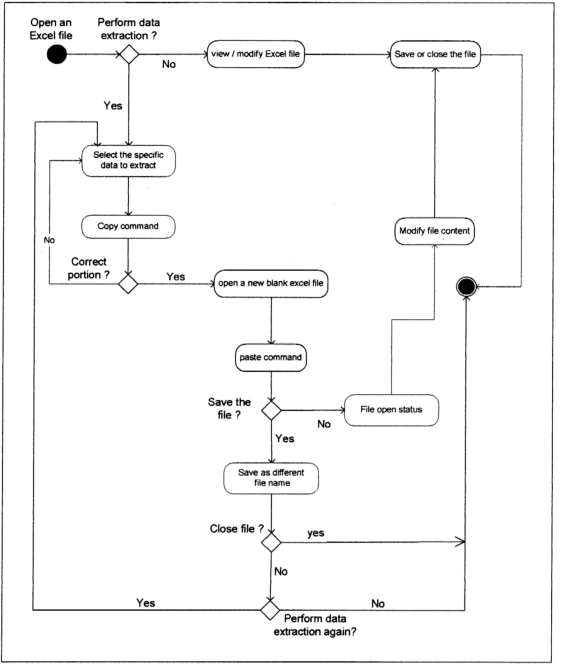

Figure 4.1 Activity diagram for Excel file segregation under current system practice

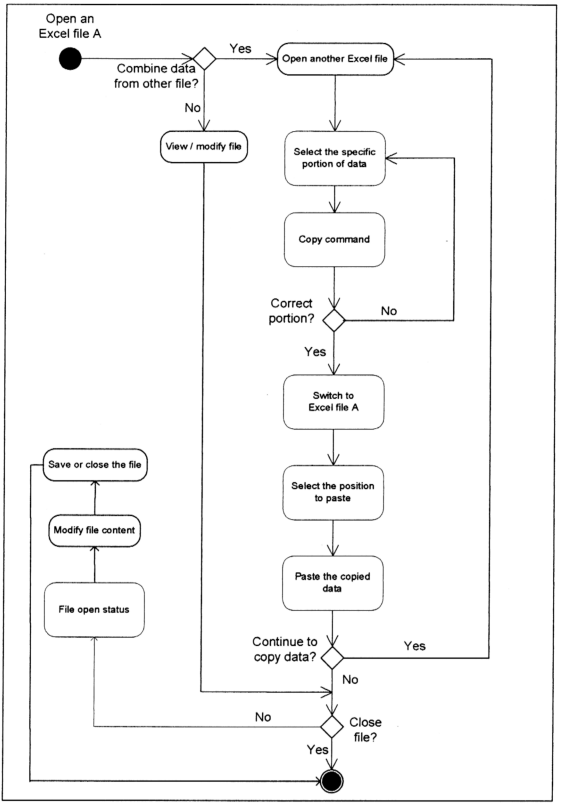

Figure 4.2 Activity diagram for Excel files combination under current system practice

4.4.2 Proposed System Representation

From the activity diagrams shown in Figures 4.1 and 4.2, it is clear that there is a need in automating both processes which are the segregation of an Excel file and the combination of data from several Excel files. This project suggests a form of automation that doesn't need a user to manually "cut & paste" portions of data in order to produce the desired output.

For the case of segregation, a user will only have to provide the source file (an Excel file), the destination file directory to store the segregated Excel files and most importantly the exact phrases contained in the source file. These phrases are needed in order to identify the portions of data that have to be duplicated in a new Excel file. Figure 4.3 and 4.4 presented below describe the activity diagrams for this process.

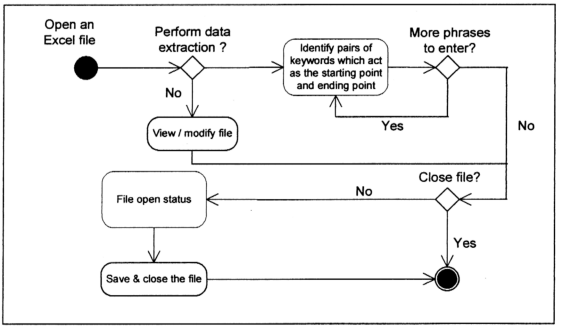

Figure 4.3 Activity Diagram for Excel file segregation – stage 1

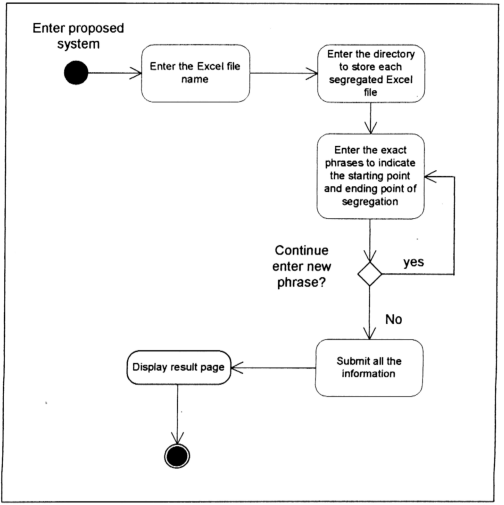

Figure 4.4 Activity diagram for Excel file segregation – stage 2

In the event of combining several Excel files into one file, this project suggests an automation process where a user doesn't have to open each file to manually "cut & paste" the data in order to achieve desired result. There are two elements that are essential as the input of the automation system. These elements are the Excel file names and the choice of the user to combine the data either in horizontal or vertical format. All data will be combined and saved as one filename.

With the OODA system, users can easily combine data from several Excel files without wasting too much time opening, copying and pasting data. Figure 4.5 describes the activities involved in combining data for Excel files.

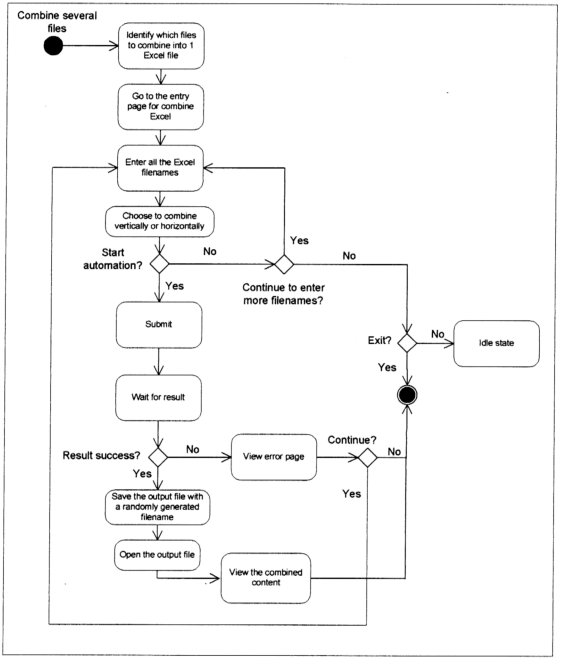

Figure 4.5 Activity diagram for combining data

4.5 Use-Case Model

In the process of implementing the O-O analysis stage, it is important to understand the system requirements by describing different types of scenarios. Use-Case diagram suggested by UML is responsible for representing interaction between users and a system. Use-Case is used for identifying responsibilities of a system to its users as well. Relationships between classes of objects involved in different subsystems are discovered during the process of generating Use-Case diagrams.

Table 4.2 below shows several use-cases and their descriptions identified in the OODA System for Microsoft Excel Files.

Table 4.2 Use-Case Names and their Descriptions

Actor:		Excel File Administrator
No.	**Use-Case Name**	**Description**
1.	Segregate data in an Excel file	The Excel file administrator interacts with OODA system for Microsoft Excel Files to automate the process of segregating data in one file and save it in several Excel files.
2.	View segregated Excel files	The Excel file administrator can view each segregated Excel files and determine correct content.
3.	Combine data from a few Excel files into one Excel file	The Excel file administrator automates the process of combining data from several Excel files and save it in one file.
4.	View combined Excel file	The Excel file administrator views the combined Excel file.
5.	Invalid phrases	If the phrases entered by the Excel file administrator are invalid (can't be located in the Excel file), an appropriate message is displayed to the administrator. This use case extends the Excel data automation process.
6.	Invalid Excel file format	If the source file provided by the Excel file administrator is in invalid format (the extension of the file is not ".xls"), an appropriate message is displayed to the administrator. This use case extends the Excel data automation process.
7.	Excel automation process	The Excel file administrator enters his choice of services to perform data automation of Microsoft Excel files.

Figure 4.6 illustrates the use-cases occurred in the OODA system for Microsoft Excel files.

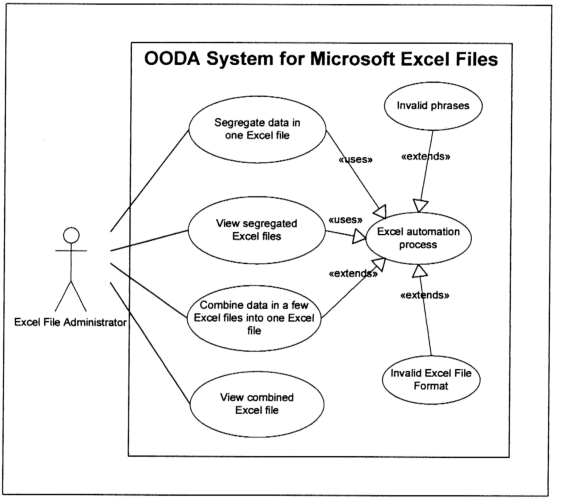

Figure 4.6 Use-Case Model

4.6 Interaction between System and Users

In the previous section, a use-case model that explains interactions between actors and the system has been developed. However, the use-cases identified earlier did not provide enough information for designing classes of objects. Therefore, it is important to carry out the process of creating sequence or collaboration diagrams as a

systematic way to think about how a use case (scenario) can take place. By doing so, identification of the objects involved in the OODA System for Microsoft Excel Files could be done easily.

Developing sequence or collaboration diagrams requires system developers to think about objects that generate the events occurred in different scenarios and therefore helps in identifying classes.

Sequence diagrams shown below have been developed to illustrate different types of scenarios for use cases listed in Table 4.2. Not all of the use cases are elaborated in sequence diagrams. Figures 4.7 through 4.10 illustrate the sequence diagrams for this system.

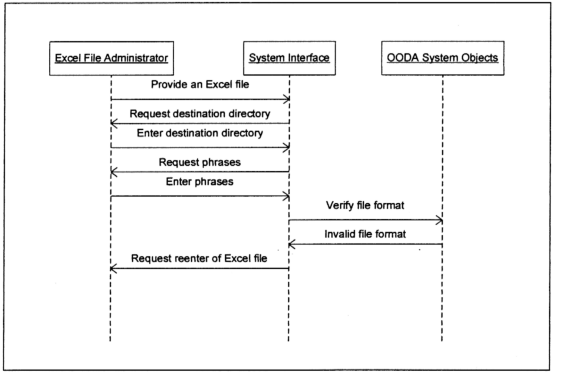

Figure 4.7 Sequence Diagram – Invalid Excel File Format

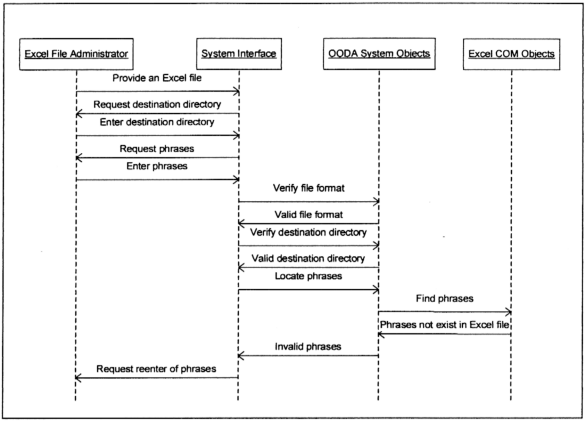

Figure 4.8 Sequence Diagram – Invalid Phrases

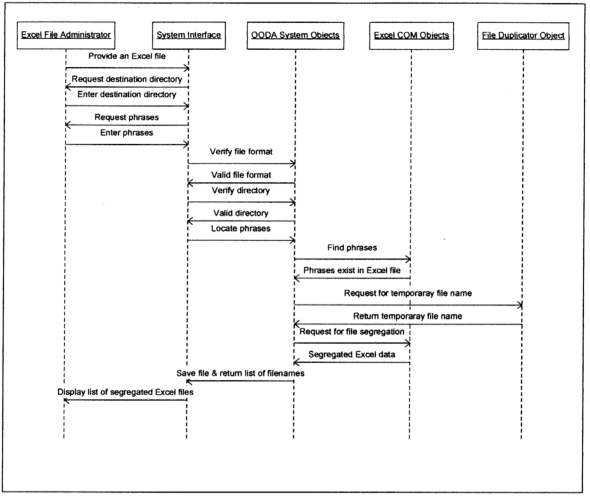

Figure 4.9 Sequence Diagram – Segregate Data in one Excel file

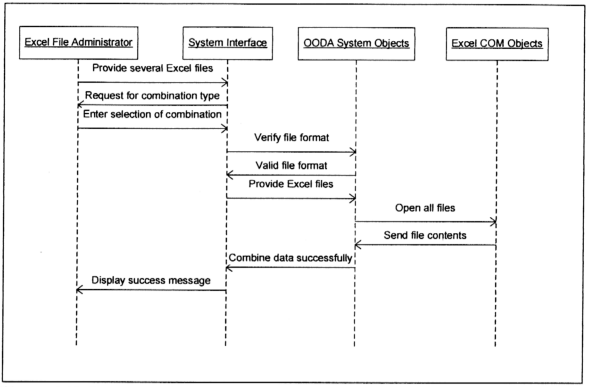

Figure 4.10 Sequence Diagram – Combine data in a few Excel files into one Excel file

The sequence diagrams provide an easier way of identifying the interactions among objects that would be needed in this project. Here, the collaboration diagrams suggested in UML ware not developed since it serves similar purposes with sequence diagrams. Section 4.7 below will describe more on the classes of object identification.

4.7 Classes of Object Identification

From the sequence diagrams developed in section 4.6, the scenarios that describe the interaction between the system actor (Excel file administrator) and the OODA system for Microsoft Excel files are now clear. The Unified Approach (UA) suggests a way of designing a system by first considering a problem-driven approach to

object-oriented analysis and not the relationships between objects, as in a data-driven approach. (Ali Bahrami, 1999)

The objects needed in this project have been identified during the stage of producing the sequence models. It is easier to think about the attributes and the methods which belong to an object class by looking at the sequence models. The sequence models will explain which events will occur in a particular time and which entity is involved.

Table 4.3 shows the low level of executable use cases that have been identified earlier.

Table 4.3 Executable Use-cases

Use Case	Description	Possible Types of Objects Involved
Segregate data in one Excel file	Actor performs data segregation on one Excel file. Each segregated portion is stored as a different Excel filename.	User interface object, System object, Java to COM object, Temporary File Object.
Combine data in a few Excel files	Actor performs data combination from a few Excel files. Combination of data can be done horizontally or vertically.	User interface object, System object, Java to COM object.
Excel automation process	The automation process which interacts directly with the Excel COM objects provided by Microsoft.	Java to COM object
Invalid Excel file format	The Excel files entered by the actor are not end with XLS extension.	Validation object
Invalid phrases	Phrases entered by the actor could not be found in the entire Excel file.	Java to COM object

From the identification of low level (executable) use cases presented in Table 4.3, the focus of classification of objects would be the four types of objects. These types of object are further explored and their contents are presented in Table 4.4.

Table 4.4 Types of Object and their specific classes/packages of object

Types of object	Specific classes of object / Package of object
User interface object	▪ svtSegregate ▪ svtCombine ▪ svtProcSegregate ▪ svtProcCombine
System object	▪ ExcelSeparatorService ▪ ExcelSeparator ▪ ExcelCombinerService ▪ ExcelCombiner
Java to COM object	▪ it.bigatti.excel8 package ▪ com.Jacob.com package
Validation object	▪ Can be included in ExcelSeperatorService and ExcelCombinerService objects ▪ Do not need to build a separated object

4.7.1 Identification of Relationships among Classes

As the types of objects have been identified, it is clear to move on to the design of the classes and the relationships among them. Basically, there are three categories of object relationships. These categories are the association relationship, super-sub relationship (generalization hierarchy) and a-part-of relationship (also known as aggregation).

Figure 4.11 to Figure 4.15 presented below show the relationships that have been identified for this project.

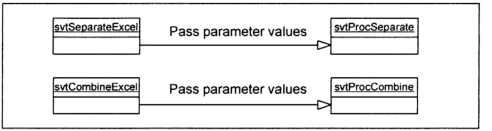

Figure 4.11 The association relationship – between Interface objects and their processing objects

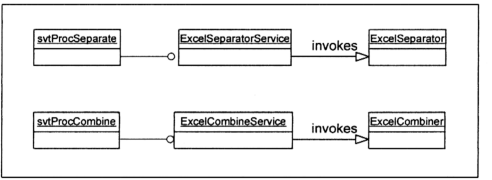

Figure 4.12 The association relationship – between Java servlet objects and Java core objects

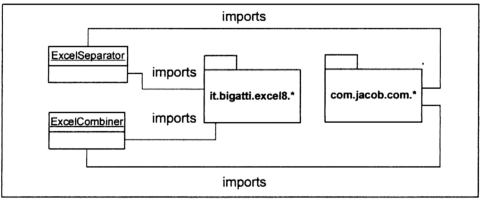

Figure 4.13 The association relationship – between Java core objects, it.bigatti.excel8 package and com.jacob.com package

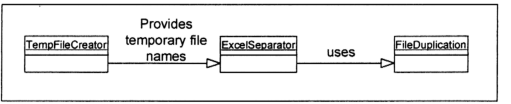

Figure 4.14 The association relationship – between TempFileCreator object, ExcelSeparator object and FileDuplication object

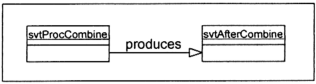

Figure 4.15 The association relationship – between svtProcCombine Java Servlet object and svtAfterCombine Java Servlet object

All figures (Figure 4.11, 4.12, 4.13, and 4.15) shown above have produced a clearer view of the classes of objects. The static class diagrams suggested in UML are presented in the following section.

4.7.2 Static Class Diagrams

For each class of object identified earlier, there must be a static class diagram to document it and store it in an object repository. The purpose of implementing this is to keep a reference of the objects for future enhancements and to be reused in future development. All the static class diagrams are shown in Figures 4.16 through 4.26.

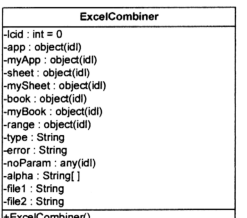

Figure 4.16 Static Class Diagram – ExcelCombiner class

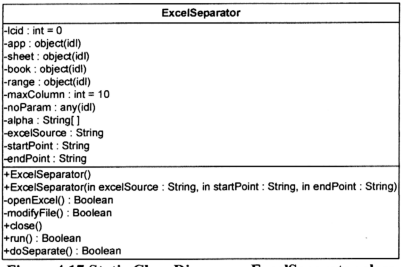

Figure 4.17 Static Class Diagram – ExcelSeparator class

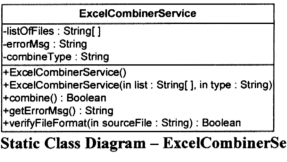

Figure 4.18 Static Class Diagram – ExcelCombinerService class

Figure 4.19 Static Class Diagram – ExcelSeparatorService class

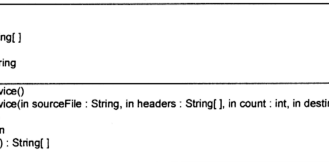

Figure 4.20 Static Class Diagram – FileDuplication class

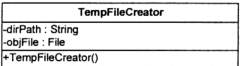

Figure 4.21 Static Class Diagram – TempFileCreator class

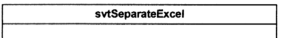

Figure 4.22 Static Class Diagram – svtSeparateExcel class

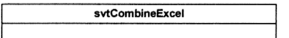

Figure 4.23 Static Class Diagram – svtCombineExcel class

Figure 4.24 Static Class Diagram – svtProcSeparate class

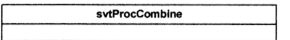

Figure 4.25 Static Class Diagram – svtProcCombine class

Figure 4.26 Static Class Diagram – svtAfterCombine class

4.8 Summary

As a conclusion, this chapter has covered the Object-Oriented Analysis process which is suggested by UML. The process has to be iterated and refined when necessary in order to produce the desired outcome of the OODA System for Microsoft Excel Files. The following chapter explains the details in conducting the Object-Oriented Design process.