Chapter 5

Analysis And System Design

5.1 System Overview

This chapter presents the analysis and the system design. The following diagram is an overview of the system requirements.

![Diagram of system requirements]

Figure 5.1: System overview
The system analysis and design is based on object oriented methodology, which leads to an easy conversion from analysis to design models, through the use of similar terms and leads to the possibility of directly converting a model developed during analysis to an implementation [IGO98]. In addition, it reduces the need for testing [STE97]. In the beginning, the analysis was started by defining the hardware on which the system will be executed. Therefore the object oriented programming language environment and Sun workstation or PC are needed for developing the system.

5.2 Classes

The methodology which has been used for analysis and system design is the Booch methodology. Because, it is more open than the others in terms of how the models and the relationships between them are applied and does not propose prescriptive steps [IGO98]. Figure 5.2 represents the main classes in the system which addresses the behavior of the system and it is considered as the first step in Booch model [GRA94].
- **Project class**: will include all the general information about the project and the selected methodology.

- **Methodology class**: which represents the software development process methods. Each project has one methodology but many activities and tasks.

- **Activity class**: this class will include the activities and tasks which represents the selected methodology. Based on this class, the project will be built and tracked.

- **Graph class**: this class is for tracking and monitoring the project. The project graph will be built based on the activities, tasks, and the project changes.

- **Problems class**: based on this information, the analysis and compromise between the projects will be addressed.

- **Change class**: this class collects the project information on the change of requirements to address the problems, responsibility, reports, and monitoring the effect of each change on the project.
- **Expert class**: this class will provide the information about each person to be used in different aspects such as assigning people, displaying information, analyzing problems and generating reports.

- **AvalChecker class**: provides the availability of each person and the availability of the people within a period of time.

### 5.3 Objects

An object diagram is used to show the existence of objects and their relationships in the logical design of a system [GRA94]. Figure 5.3 represents the main objects and their relationship in a whole system.

![Object Diagram](image)

**Figure 5.3: Object diagram**
• **Method**: Some of the attributes of this object are Waterfall, Spiral, Incremental, Rapid prototype as well as Build and fix models.

• **Phase**: the activities and the tasks for each selected methodology are the attributes of this object.

• **DrawGraph**: the attributes of this object are activities and tasks duration, description and some of the changes in requirement information.

• **DueProblem**: the description of the problems in each phase.

• **Person**: the attributes are experience, qualification, past project information, and others.

• **Analysis**: the attributes of this object are the results of the analysis such as problems in specific and all projects, problems due to change in requirements and others.

• **ChangeReq**: the attributes of this objects are the change in requirement reports, how many times the change has occurred in specific projects and others.

### 5.4 System Design

While designing the system, some modifications has been done to the classes: subclasses that have been derived include Build & Fix, WaterFall, Spiral, Incremental, and Prototyping. Some new classes have also been added such as view, equipment and report classes as seen in Figure 5.4.
- **Equipment class**: this class will provide information about the needed equipment for the project such as hardware and software.

- **Requirement class**: this class will provide the system with the project requirements of the project.

- **Calendar**: this class is used to allow the user to select and load the dates.

- **Methodology class**: five classes will be derived from this class, and presents the software process methods. These processes are Build and Fix model, Waterfall, Spiral, Incremental, and Prototyping model.

- **Report class**: this class will produce reports such as problems in all or in specific projects, information about the experts such as qualified people in specific areas or in specific programming languages and others.
- **View class**: this class will display information about the people such as working personnel on a project (full time, part time, and others), expert or skilled people in specific method, techniques, or programming languages.

The interaction diagram is used to trace the execution of a scenario in the same context as an object diagram. It is another way of representing the objects. However, it is easier to read the passing of messages in relative order [GRA94]. Therefore, Figures 5.5a and Figure 5.5b represents the scenario of two main functions of the system. Figure 5.5a represents the scenario for creating or updating the information for activity or task.

<table>
<thead>
<tr>
<th>Method</th>
<th>Phase</th>
<th>ActORTask</th>
<th>Person</th>
<th>PersonINF</th>
<th>PersonAVL</th>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>getMethod()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5.5a: Scenario for assigning personnel**
• **Method**: the attributes of this object are used methodology such as Waterfall, Spiral, Incremental, Rapid prototype as well as Build and fix models.

• **Phase**: the attribute of this object is the life cycle for each methodology such as planning, organizing, staffing, directing, controlling, design, coding and testing.

• **ActORKTask**: the attributes of this object is the phase’s activity or task that the personnel will assign in.

• **Person**: the selected personnel for the activity or task.

• **PersonINF**: the attributes are experience, qualification, past project information, and others.

• **PersonAVL**: the attribute of this object is the availability of the selected personnel.

• **Problems**: the attributes of this object are the problems that facing the selected phase.
Figure 5.5b: Scenario for changing or monitoring project requirements

- **Method**: the attributes of this object are Waterfall, Spiral, Incremental, Rapid prototype as well as Build and fix models.

- **Change**: the attributes of this object are the change in requirement report, the problems caused the change and the change date.

- **ProjectReq**: the attributes of this change are keeping track the project requirements and times of change.

- **Problems**: the attribute of this object is the problems that caused the change in project requirements.
- **Person**: the responsible personnel for causing the change in project requirements.
- **ChangeInf**: the attributes of this object are project requirements, the status of the project, problems, and the responsible personnel.
- **Graph**: the attribute of this object is the status of the project last change and the number of times the project has changed.

### 5.4.1 Database Design Examples

Microsoft Access was used for developing the database tables due to availability, easy to use and access. Table 5.1 shows a sample of the data base design representing the information about the change in requirements and Table 5.2 represents an expert’s information.

**Table 5.1: Change in requirements**

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>DATA TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pcode</td>
<td>Number</td>
<td>Project code</td>
</tr>
<tr>
<td>PN</td>
<td>Text</td>
<td>Project name</td>
</tr>
<tr>
<td>Changed</td>
<td>Number</td>
<td>Change day</td>
</tr>
<tr>
<td>Changem</td>
<td>Number</td>
<td>Change month</td>
</tr>
<tr>
<td>Changey</td>
<td>Number</td>
<td>Change year</td>
</tr>
<tr>
<td>Reson</td>
<td>Text</td>
<td>The problems causing the change in requirements</td>
</tr>
<tr>
<td>Resp</td>
<td>Text</td>
<td>Those responsible for this change in requirements</td>
</tr>
<tr>
<td>Count</td>
<td>Number</td>
<td>Times of change</td>
</tr>
<tr>
<td>Report</td>
<td>Memo</td>
<td>The new requirements</td>
</tr>
</tbody>
</table>

**Table 5.2: Database form for the expert’s information**

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>DATA TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExpName</td>
<td>Text</td>
<td>Person’s name</td>
</tr>
<tr>
<td>ExpID</td>
<td>Number</td>
<td>Person’ ID</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>Ocup</td>
<td>Text</td>
<td>Person’ occupation</td>
</tr>
<tr>
<td>Qualification</td>
<td>Number</td>
<td>Person’ qualification</td>
</tr>
<tr>
<td>WonP</td>
<td>Number</td>
<td>Working under contract</td>
</tr>
<tr>
<td>Experience</td>
<td>Number</td>
<td>Person’ experience</td>
</tr>
<tr>
<td>Qinsa</td>
<td>Text</td>
<td>Areas the person qualified in</td>
</tr>
<tr>
<td>Linsa</td>
<td>Text</td>
<td>Areas the person lack knowledge in</td>
</tr>
<tr>
<td>Taken</td>
<td>Text</td>
<td>Training taken by the person</td>
</tr>
<tr>
<td>Needed</td>
<td>Text</td>
<td>Training needed</td>
</tr>
<tr>
<td>Notes</td>
<td>Memo</td>
<td>Person’s notes and comments</td>
</tr>
<tr>
<td>Projects</td>
<td>Text</td>
<td>Project(s) names the person has worked with</td>
</tr>
<tr>
<td>Pproject_id</td>
<td>Text</td>
<td>Project(s) ID</td>
</tr>
<tr>
<td>AC_TAS</td>
<td>Text</td>
<td>The nature of the work in these projects (past projects)</td>
</tr>
<tr>
<td>Reuse</td>
<td>Text</td>
<td>The person’s reusability level that resulted from past projects</td>
</tr>
<tr>
<td>Detect</td>
<td>Text</td>
<td>The person’s detection level that reported from past projects</td>
</tr>
<tr>
<td>AC_sd</td>
<td>Text</td>
<td>The start date upon the work in each activity from past projects</td>
</tr>
<tr>
<td>AC_fd</td>
<td>Text</td>
<td>The finish date upon the work in each activity from past projects</td>
</tr>
<tr>
<td>Exp_PL</td>
<td>Text</td>
<td>The used programming language from past projects</td>
</tr>
<tr>
<td>PL_SD</td>
<td>Text</td>
<td>The start date of the programming language used</td>
</tr>
<tr>
<td>PL_FD</td>
<td>Text</td>
<td>The finish date of the programming language used</td>
</tr>
<tr>
<td>INFO</td>
<td>Text</td>
<td>Any additional information about the person</td>
</tr>
</tbody>
</table>

5.4.2 Samples Of Screen Design

Screen layout might affect the user in different aspects such as confusion, and the inefficiency in the screen layout lets the user facing difficulties in doing the job and gives more chances for mistakes to occur. Moreover, poor screen design may discourage people from using the system [WIL97]. Therefore, the focus is to build a good system which is the main concern in this work. The screen design was built based on the object oriented concept, which consists of objects and actions. The objects are the body of the screen which appear to the user; the actions are the commands that manipulate objects.
and are executed immediately when selected. The screens were designed based on the following principles which are well defined in [WIL97]:

- **Consistency** is very important for a screen design to provide a clear view of the screen description, and allow a system approach to the user environment. This is because the consistency aspect follows the same conventions and rules across all the graphical user interface designs. Therefore, the focus of this aspect becomes the main concern in building the screen system.

- **Sequentially** most of the system's screen are built to provide visual screen composition by using the sequential form which provides the sequentiality by arranging elements to guide the eye through the screen in an obvious, and logical order.

- **Captions or Labels** are clearly identified so that the users are able to determine each field in the screens. Left-justified approach has been used to control the caption/labels.

- **Data fields** are used to enter, modify, and display information. Left-justified approach is also used to control data fields.

- **Application Window** is the main window in the system and is used as a framework for the function commands and data for the system.

- **Menu** is used to provide a consistent starting point for all the system activities. The titles are chosen so as to reflect the alternatives available in the associated pull-down menu. The titles are written in lowercase format except for the first letter (e.g. File, View, Tools, and Report). In addition, reverse color bar was used to highlight the choice in the menu. This has been presented in Figure 5.6.
- **Pull-Down Menu** is used to provide access to common and frequently used application actions that takes place on a wide variety of different windows. All the items are clearly displayed except some of the unavailable items due to the current state (inaccessible procedure).

- **Pull-Down menus Leading to Another Pull-Down menu** are used to group some choices to the first pull-down. The second pull-down is indicated on the first pull-down by including a right-pointing triangle to the right of the applicable choice description. This shown in Figure 5.6.

![Prototype tool for Project Management](image)

Figure 5.6: Menu screen

- **Buttons** always visible, convenient and can provide meaningful descriptions of the actions that will be performed. They are logically organized in the work area and are used to carry out actions to be accomplished.
- **Text fields** are used to enter, modify, and display data. Text fields are also used to provide some selectable information that can be automatically chosen. For example, the user can either type the date or call the calendar class to get the required date and display it.

- **Radio Buttons** are used to highlight the selected option only and the rest are automatically un-highlighted and deselected. This has been used when the user selects the methodology for the project.

- **List Boxes** has been used in building the screen system many times. The main reason is that it is the most suitable approach to express and present some feature such as analysis, compare, and display the problems, display the resources and their information and others. The other reason is that the unlimited number of choices and it is always visible. The list is also used to perform actions by double clicking on the selected item in the list. For example when the list is used to display the problems in any aspect, double clicking on any problem will open another window holding information about the selected problems. Highlighting the selected item(s) is also provided. Moreover, the list boxes are fully authorized to provide most of the features (single and multi selected item(s)).

- **Pop-Up List Boxes** are used to allow the user to select one item from a large list when the screen space is limited. Example of this approach is shown in Figure 5.14.

- **Scroll Bar** is used to track the project graphically, and will display the non-visible information in the graph.
• **Messages** Galitz stated [WIL97] that messages, feedback, and guidance are elements for a good design. Messages are communications provided on the screen to the user. There are many kinds of messages which are displayed in different formats. Warning messages have been used to warn the user that the dates, which have been chosen for the project, are invalid (Figure 5.7). Question message has been used asking for confirmation when the project is about to be deleted. An example of a message such as shown in Figure 5.8.

![Error Date window](image)

**Invalid Dates**

![OK Help](image)

Figure 5.7: Warning message for invalid dates

![Error Date window](image)

**Warning!!**

![Yes No](image)

Figure 5.8: Question message for confirmation of deletion of project

• **Graph** there are several ways of representing graphs and the one used in this system is the Bar chart. It is used to display the activities and tasks of a specific project.
- **Help Facility** is to assist people in remembering what to do. Therefore, it has been used to describe steps and to define the purpose of the main buttons. Example on help screen shown in Figure 5.9.

**Help menu**

For each double click on the change date’s list all information about this change will be highlighted such as:

- How this change effect the project, The reason(s) of causing this change, and who are responsible.

**More button**

To highlight the reasons for causing the change.

**Requirements Button**

To display all the reports on the changes in the requirements.

![Help window](image)

Figure 5.9: The help screen of change in requirement option

- **Color** is effectively used in screen design. Studies have shown that the proper use of color will help improve performance, remind the users and will be useful for organizing information [WIL97]. Therefore, the color has been used in two aspects. Firstly, in reminding the user or describing status by displaying one line text with blue or red foreground. Secondly, the color is used to show periods of each activity and task as well as to indicate (red color) the risk by highlighting the project risk level.
5.4.3 Examples On Input and Output Screen Design

Example 1: shown in Figure 5.10 which is a project information screen where the user will fill in the general information, determine the used methodology, the needed equipment, and the requirements of the project. Project information can then be added, modified and deleted by using the Add, Update and Delete buttons.

![Project Information Screen](image)

Figure 5.10: Project information

Example 2 represents the activity screen design which will include the activities and tasks information for each stage in the selected methodology. This information such as the stage name, project name, work type (whether is activity in the stage or task in specific activity and activity or task description) and the planned duration. In addition, it will allow the user to assign the resources and the problems to the activity or the task when activating Assign Resources and Problems buttons. Moreover, the list of the activities
will be shown by activating *Show activities* button and a list of all the tasks will be shown by the *Show tasks* button. Help facility has also been provided as a result of the action of the *Help* button (Figure 5.11).

![Activity Window]

**Figure 5.11: Activity form**

Example 3: When the *Assign resources* button in the previous example (Figure 5.11) is activated, the screen in Figure 5.14 will appear to display the possible status or conditions for selecting the resources such as experience, qualification, reusability level, detection level, expertise in specific programming language, qualification in specific area, knowledge in specific methodology or method, or the person's work in a project as part time, full time, or working under contract. New expert record can be added by activating *Add* button.

Example 4: Figure 5.12 and 5.13 represent samples of the system output. Figure 5.12 is a result form of the analysis of all the problems in current and in past projects including the

System Analysis and Design

92
selected projects. Figure 5.13 however represents the report form of the problems in current and past projects.

<table>
<thead>
<tr>
<th>Project names</th>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounting system</td>
<td>Error in data representation (EOR).</td>
</tr>
<tr>
<td>Software development</td>
<td>Error in design logic (EDL).</td>
</tr>
<tr>
<td></td>
<td>Incomplete or erroneous specification (IES).</td>
</tr>
<tr>
<td></td>
<td>Lack in SQA tools for testing the specification.</td>
</tr>
<tr>
<td></td>
<td>The methodology, standards and procedure are unclear.</td>
</tr>
<tr>
<td></td>
<td>Incomplete or erroneous specification (IES).</td>
</tr>
<tr>
<td></td>
<td>Intentional deviation from specification (IDS).</td>
</tr>
<tr>
<td></td>
<td>Violation of programming standards (VPS).</td>
</tr>
<tr>
<td></td>
<td>Misinterpretation of customer communication (MCC).</td>
</tr>
</tbody>
</table>

Note: Double click on the project name in the list to display the problems.

Figure 5.12: The problems due to change in requirements
The problems in past projects due to change in requirements are:

1. Error in data representation (EDR)
2. Error in design logic (EDL)
3. Incomplete or erroneous specification (IES)
4. Lack in SOA tools for testing the specification
5. The methodology, standards and procedure are unclear.
6. Incomplete or erroneous specification (IES).
7. Intentional deviation from specification (IOS).
8. Violation of programming standards (VPS).

Figure 5.13: The report form of the problems due to change in requirements

Assign people

Experience: Non
Qualification: Undergraduate
Reusability: 10% - 20%
Detection: High

- Programming Language
- Working in specific activity in past projects
- Qualify in specific area
- Lack knowledge in specific area
- Full time
- Part time
- Assigning people to external projects
- Working under contract
- Methodology model or methods

Figure 5.14: Assign people base on certain factors