CHAPTER 2

THEORETICAL FOUNDATION & FRAMEWORK

2.1 Theoretical Foundation

This section will define and give comprehensive explanation about all relevant theories in order to support the solution design for the problems.

2.1.1 Data, Information, and Knowledge in Information System

2.1.1.1 Data

Data are described as unrefined facts that contain the description about things, events, activities, and transaction. Data can be captured, recorded, stored, and classified, but not organized to express any definite meaning [1]. Some instances of data are the employee salary, student grade, total number of books in a library, etc.

2.1.1.2 Information

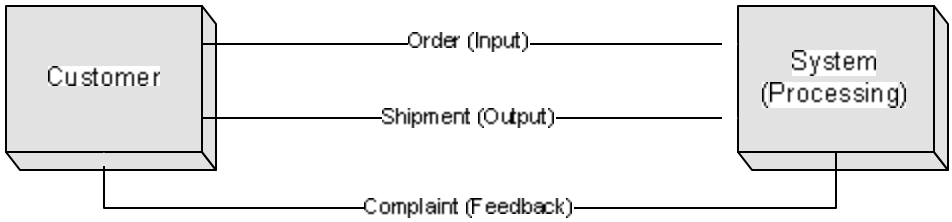
Information can be described as more refined data or cluster of data which is arranged to have a specific meaning. Collection of data is used as the source of information to convey a definite meaning [1]. Employee salary with the employee name is one of example of information. In example above, the information is assembled from two data, which are employee salary and employee name.

2.1.1.3 Knowledge

Knowledge is a refined collection of information that could convey understanding, experiences, accumulated learning, or expertise, which are applies to recent business problem or process. The organizational knowledge, which is provided by the refined information, can give the value that useful for the manager to avoid any mistakes done before and neglect any means to "reinvent the wheel." [1]

2.1.2 Information System

In order to have better understanding about information system, first the readers need to understand about the concept of a system. According to McKeown [2] "a system is a group of elements (people, machines, cells, and so forth) organized for the purpose of achieving a particular goal." Some examples of system would include information system, business system, and computer system. Every system must have the input, output, and feedback. Input means anything that enters the system, which will be processed by the system to produce the output, while feedback is the response of output that may alter the system's process. The figure below depicts an example of simple transaction system that can give a better understand about the definition of a system:



According to Turban, Rainer and Potter [1] "an information system collects, processes, stores, analyzes, and disseminates information for a specific purpose." As a system, information system also have inputs (data, instructions), outputs (reports, calculations), and feedback that may be used to control the system's operation. As stated by McKeown [2], there are three roles of information systems:

- Handling the present: means that the organization must be able to take care of its day-to-day business, primarily by processing transactions that involve customers, suppliers, and employees.
- Remembering the past: those transactions within the organization must be stored if the organization is to remember its past.
- Preparing for the future: the data on transactions are then used to prepare for the future, which results in decisions that determine the way transactions are handled in the future.

Those information system functions can be represented by a cycle (Information System cycle), because the first function provides input to the second function, while the second function provides input for third function, and the third function will give input back to the first function. Information System cycle is shown in the following figure:

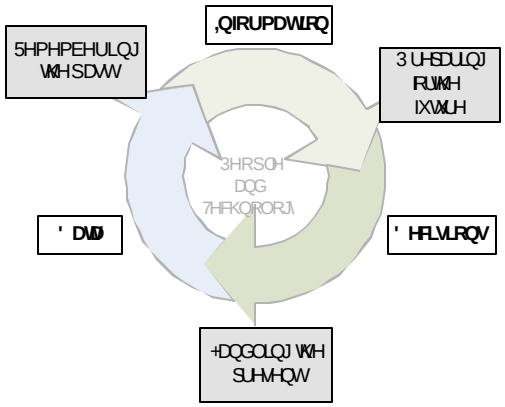


Figure 2.2 Information System Cycle [2]

As stated by Whitten, Bentley, and Dittman [3], information can be mapped to five layers:

- Presentation layer: the user interface that handles the presentation of inputs and outputs for users.
- Presentation logic layer: it is the processing that must be done to generate the presentation, for instance: editing input data and formatting output data.
- Application logic layer: this layer handles all the logic and processing required supporting the actual business application and rules.
- Data manipulation layer: includes all the commands and logic required to store and retrieve data to and from the database.
- Data layer: the stored data in a database.

2.1.3 Database Systems

As the time goes on, the amount of information keeps growing to a certain level that enforced the necessity to use tools, which are used to manage the information. Not only because its amount, but the value of information also enforced the necessity to use the tools in order to preserve the value. Moreover, the tools must be able to keep track and extract any important data quickly.

A database system is the tools that can answer the needs to manage the growing amount of information. Before the computer era, most of information is recorded by using paper based database, in which not efficient in terms of cost (paper cost, ink cost, etc) and the time needed to extract useful information.

Nowadays, most of the database systems are already using computer to manage the information, which give an efficient manner toward cost and time. Furthermore, the database system in this thesis only refers to the computerized database system. The following figure shows the simplified depiction of database system.

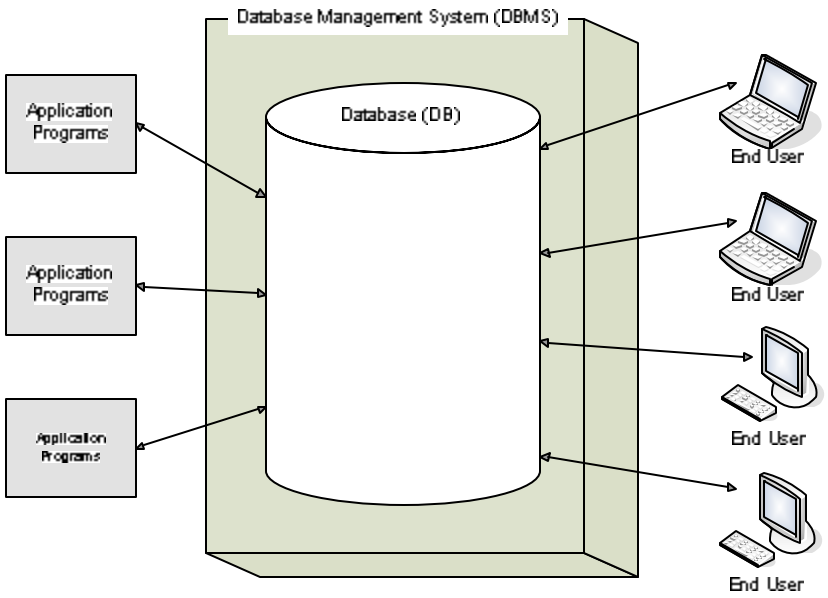


Figure 2.3 Simplified View of Database System

2.1.3.1 Database (DB)

As stated by Ramakrishnan and Gerhke [4], the definition of "a database is a collection of data, typically describing the activities of one or more related organizations." Almost similar with Ramakrishnan and Gehrke [4], according to Connolly and Begg [5], a database is "a shared collection of logically related data, and a description of this data, designed to meet the information needs of an organization." Another definition from Date [6] acknowledged that database is "a repository or container for a collection of computerized data files."

From those definitions, a conclusion can be made that a database is a collection of computerized data that are rationally related and can be used to describe a useful meaning within some connected organizations.

In general, there are three types of database models based on the structure of data in database [7]:

- Hierarchical database model: in this database model, the data are organized into a tree-like structure (figure 2.4). Data can be accessed with top-down approach, one-to-many relationship, where the top of the tree is called root and the bottom is called leaves. The main advantage of hierarchical model is that the data can be accessed quickly. Usually, this type of database is used for transaction processing and Management Information Systems (MIS) applications. Nowadays, there are few databases that are built with this kind of model because the needs of broader functions.
- Network database model: this model allows each data to be connected into several data (figure 2.5). From the definition, the author can conclude that Hierarchical database model is a special case of Network database model. Although, a Network database model could provide wider functions than the hierarchical database model, but it requires a good programming expertise, knowledge about database design, and longer time to built. In addition, this model of database is still used in some mainframes and high-volume transaction processing applications.
- Relational database model: within this model, the data are logically structured in two-dimensional tables (relations) and the data are associated with another data by using keys (figure 2.6). Recently, this is the most popular database model and commonly used to build new database systems.

Relational is the database model that the author will follow in designing the database for CV. X. Furthermore, detailed explanation about this model is given in sub-section 2.1.3.2 Relational Database Model.

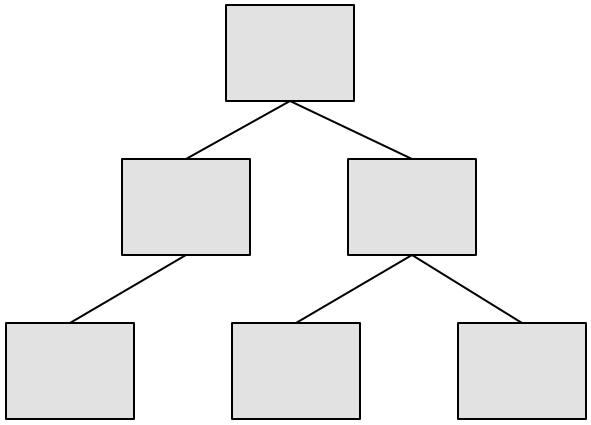


Figure 2.4 Hierarchical Database Model [7]

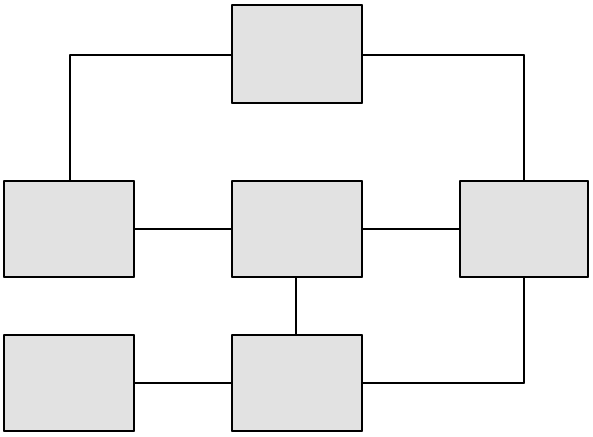
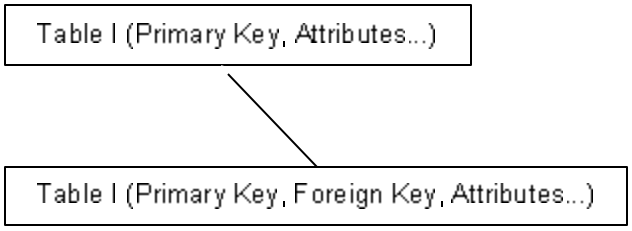


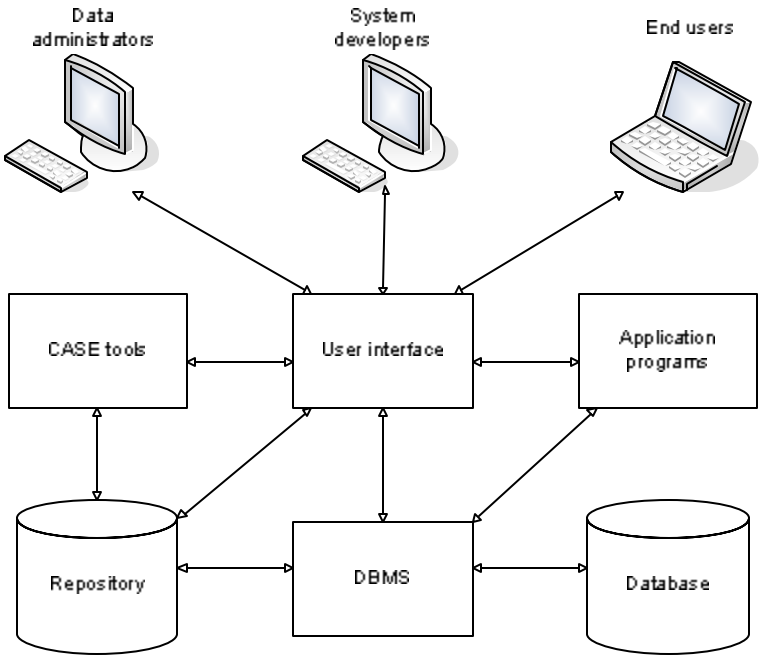
Figure 2.5 Network Database Model [7]



2.1.3.1.1 The Database Environment Components

In a typical database environment there are nine important components as stated by Hoffer, Prescott, and McFadden [7]:

- Computer-aided software engineering (CASE) tools: it is the automated tools in which used to design databases and application programs.
- Repository: it contains all knowledge base, including data definitions, data relationships, screen and report formats, and other system components.
- Database management system (DBMS): commercial software that is used to define, create, maintain, and provide controlled access to the database and repository.
- Database: collection of computerized data that are rationally related and can be used to meet the information needs within some connected organizations.
- Application programs: the applications that are used to create and maintain the database and provide information to users.
- User interface: it is the interface (language, menus, etc) that users use to interact with various system components, for example application programs, CASE tools, repository, and DBMS.
- Data administrators: the person who responsible for managing the organization information resources.
- System developers: persons who design new application programs. People like systems analysts and programmers could become the system developers.
- End users: persons, who can add, delete, modify, and retrieve information from the database.


Data administrators
System developers
End users
CASE tools
User interface
Application programs
Repository
DBMS
Database

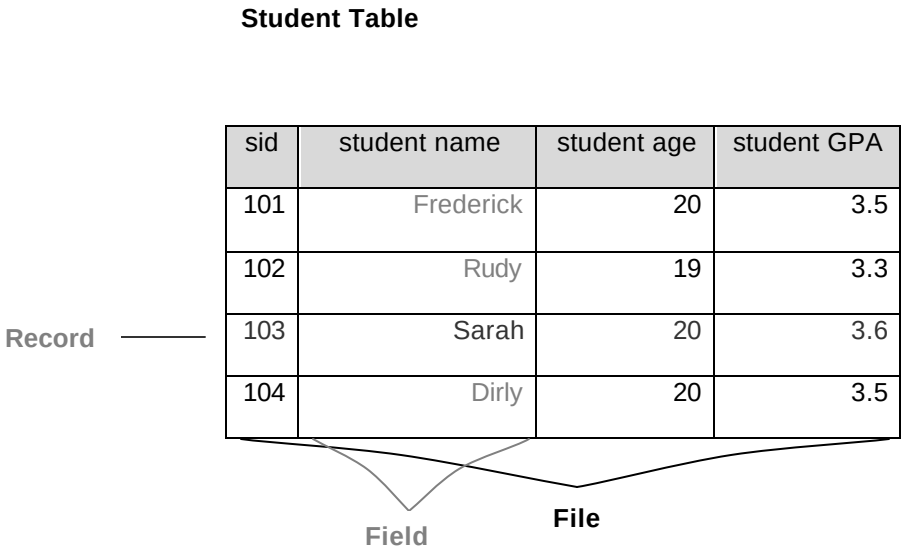


Figure 2.8 Field, Record, and File in Student Table

- Entity: "is an object in the real world that is distinguishable from other objects" [4]. An entity can be considered equivalent to a table in the relational database model. Some examples of entity are student table, employee table, course table, etc.
- Entity set: the collection of similar entities.
- Domain: it is used to determine to the data type or allowable value of an attribute. Some examples would include student name has string data type, student id has string data type, student age has integer data type and student GPA has real number data type.
- Relationship: it can be described as the connection of more than one entity.
- Relationship Set: is a collection of similar relationship.

2.1.3.2.1 Relation in Relational Database Model

Relation is the most important term for the data representation in relational database model. According to Ramakrishnan and Gehrke [4], relation can be categorized into two types, which are:

- Relation schema: this type of relation specifies the relation's name, all fields' name, and all fields' domain. Domain specifies the domain name and its associated value. The example of a relation schema is shown below:

  Students (*sid:* string, *name:* string, *age:* integer, *gpa:* real)

  Figure 2.9 Relation Schema [4]

  From example in figure 2.9 the relation's name is Students, while *sid*, *string*, *age*, and *gpa* are the field's name, and the domain name are string, integer, and real. The domain value is determined by the domain name, for example an integer has value of all positive number without any decimal digits.

- Relation instance: it is a set of records. Every record in an instance must have identical field quantity with the relation schema. In other words, an instance is a set of rows within a table, and each row has identical field quantity. Usually, a relation instance is only called a relation. An example of relation instance:

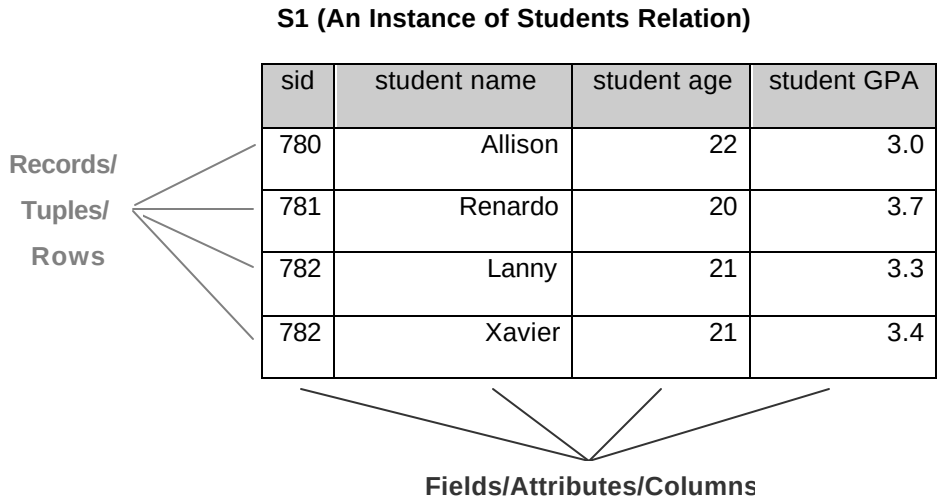


Figure 2.10 Relation Instance

In figure 2.10, the readers can observe that S1 is the instance of Students relation in which includes four records and four fields (the same with the relation schema example above). Relational model requires that each row in the table must be unique, so a relation can be defined as a set of unique rows.

2.1.3.2.2 Relations Properties

Not all two-dimensional tables of data can be defined as relations, because there are some properties that must be fulfilled in determining whether a table is a relation or not. The properties of relations are listed as follow [7]:

- The relation must have a discrete name within database.
- Every cell (intersection of a row and a column) in the relation only contains single value.
- For every field in a relation must have a distinguish name from other fields.
- For every row in a field must contain identical domain value with other rows in the field.

- Each record in a relation can be distinguished from other records or there is no more than one identical record in a relation.
- The field's sequence in the relation is insignificant, means that a field's position can be interchanged with another field's position.
- The row's sequence in the relation is also insignificant, since each row is interchangeable with another row within a relation.

Figure 2.10 is an example of table that satisfies the properties of relations.

2.1.3.2.3 Relational Keys

Key in the relational database model is an important concept and it also known as relational keys. The main functions of relational key are to associate one data into another data and identify each record within a relation uniquely. There are several relational keys, namely:

- Superkey: "is an attribute or set of attributes that can uniquely define a record in a relation. The superkey may contain unnecessary attributes that can be ignored for unique identification" [5].
- Candidate Key: is an attribute or set of attributes that can uniquely define an entity in a set of entity. Candidate key can also be regarded as the superkey that only contains essential attributes for unique identification.
- Primary key: the candidate key that is used to uniquely define an entity.
- Composite key: a candidate key that contains more than one attribute.
- Alternate key: are candidate keys that are not used for primary key.
- Foreign key: a key that is referenced from another primary key in another table.

2.1.3.2.4 Integrity Constraints in Relational Database Model

Integrity constrains ensure the accuracy and integrity of data in the database. The categorization of integrity constraints is defined as follow:

- Domain constraint: this constraint enforced that the value of every record in a field must contain the same domain type. For example a student age field with integer domain type, so the value of each record in the student age field must contain an integer value.
- Entity integrity: as stated by Hoffer, Prescott, and McFadden [7], the entity integrity rule states the following: "No primary key attribute (or component of a primary key attribute) may be null." Entity integrity is intended to give surety that a primary key must exist within every relation and a primary key must contain valid data values. Entity integrity also assure that primary key always have non-null value. Null value is assigned when the relevant value for an attribute is unknown.
- Referential integrity: according to Hoffer, Prescott, and McFadden [7], "a referential integrity rule states that if there is a foreign key in one relation, either each foreign key value must match a primary key value in another relation or the foreign key value must be null." The aim for referential integrity is to confirm the consistency between the records of two relations.
- General constraint: additional constraint that can be specified over some aspect in the relations. The users or the database administrators may add general constraint into the database only when needed. For instance, the total number of students that are enrolled in a particular course can be restricted; if the total number is 20, then the course cannot be enrolled for new students anymore.

2.1.3.3 Database Management Systems (DBMS)

Database management system (DBMS) is "software system, usually commercial, that is used to define, create, maintain, and control access to the database" [5]. Some examples of DBMS would include Microsoft SQL Server, MySQL, Oracle, and so on.

A typical DBMS would be able to perform some functions:

- Data Definition: this function allows the users to define the creation, deletion, and modification of tables in database by using the Data Definition Language (DDL). With DDL, the users can specify the data types and structures and the constraints of the data. Moreover, in order to understand the DDL, a DBMS must have a DDL compiler or DDL processor.
- Data Manipulation: this function allows the users to insert, update, retrieve, and delete data in the database. Using the Data Manipulation Language (DML), it allows users to perform some functions stated above. A DML compiler or DML processor in a DBMS is used to compile and execute the DML.
- Data Security: the DBMS must provide a facility to secure its database. DBMS must be able to prevent any illegitimate access into the database.
- Data Integrity: DBMS will ensure the data consistency within the database.
- Data Concurrency: this function could provide shared access of data in the database.
- Data Recovery: this function becomes very useful when a failure occurs in the hardware or software and the data become corrupt or inaccessible. The DBMS could restore its state into preceding consistent state.

- Data Dictionary: data dictionary provide the descriptions of the data, which called metadata or descriptors. In other word, data dictionary is the data that can define other data in the database.

The combination of Data Definition Language (DDL) and Data Manipulation Language (DML) can be described as the query language. The most common and popular query language is Structured Query Language (SQL). A detailed discussion of SQL will be presented in sub-section 2.1.3.5.

2.1.3.3.1 Relational Database Management Systems (RDBMS)

In general, a relational database management system is a DBMS that fully implements the relational database model. In designing the database systems for CV. X, the author will use a RDBMS, which conform to the relational database concept.

2.1.3.3.2 Level of Abstractions in RDBMS

In RDBMS, the data is viewed as three levels of abstraction, which are:

- Conceptual schema/logical schema: this schema describes about the detailed specification of all stored relations in the database. In relational database model, the conceptual schema can be represented by using the entity-relationship (E-R).

Figure 2.11 below shows an example of conceptual schema for university database:

Students (*sid:* string, *name:* string, *age:* integer, *gpa:* real)
Faculty (*fid:* string, *fname:* string, *sal:* real)
Courses (*cid:* string, *cname:* string, *credits:* integer)
Rooms (*rno:* integer, *address:* string, *capacity:* integer)
Enrolled (*sid:* string, *cid:* string, *grade:* string)
Teaches (*fid:* string, *cid:* string)
Meets_In (*cid:* string, *rno:* integer, *time:* string)

Figure 2.11 Conceptual Schema of University Database [4]

In figure 2.11, the relations are composed from several entities and relationships. Relations like Student, Courses, Faculty, and Rooms are the entities, while other relations such as Teaches, Meets_In, and Enrolled are the relationships.

- Physical schema: this schema emphasizes on the discussion of how the data in conceptual schema are stored in secondary storage, such as disks or tapes. The topic, such as the database technology used to define how the data can be managed in the secondary storage, become an important consideration in designing physical schema.
- External schema: this schema is designed to customize the data access for any specific needs of users or groups of users. In a database, both conceptual schema and physical schema only have one occurrence because the database only contains a set of stored relations. In other hand, the external schema may have numerous occurrences, with each occurrence could contain several views and relations from the conceptual schema. A view in the DBMS can be described as a relation with one difference that the view will not be stored into the database. An example of view is provided in figure 2.12.

CourseInfo (cid: string, fname: string, enrollment: integer)

Figure 2.12 An example of View [4]

The view in figure 2.12 is used to give information about the names of faculty members teaching courses as well as course enrollments [4]. In addition, the design of external schema relies on user requirements.

A general view of three levels of abstraction in DBMS is depicted in figure 2.13 as follow:

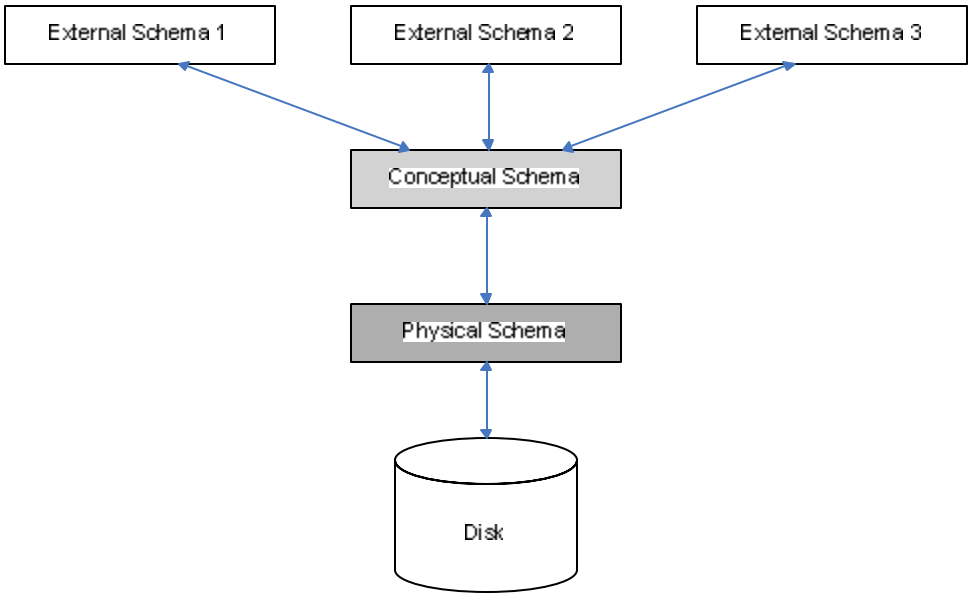


Figure 2.13 Three Levels of Abstraction in DBMS [4]

2.1.3.4 Data Modeling

According to Connolly [5], there are two main purposes of data modeling, which are "to assist in the understanding of the meaning (semantics) of the data and to facilitate communication about the information requirements." The data are modeled in order to guarantee that the following points can be understood [5]:

- Each user's perspective of the data.
- The nature of data itself, independent of its physical representations.
- The use of data across user views.

Entity-Relationship (ER) model is the most popular high-level data model that commonly used in designing a database. The graphical representation of ER model is discussed in sub-section below (Entity Relationship Diagram).

2.1.3.4.1 Entity Relationship Diagram (ERD)

ERD represents the graphical and logical structure of data and it is presented by two elements, namely entity and relationship. In terms of entity and relationship, ERD depicts the relationships between several entities and the attributes of both entities and their relationships [7].

2.1.3.4.2 ERD Notation

In order to have better understanding about the ERD notation, let's have an example of a simple ERD:

- Identifying relationships: “it is the relationship between weak entity and its owner” [7].
- Supertype and subtype: some entity can be divided into several types of entity, for example an entity named PEOPLE consists of several types, like STUDENTS, EMPLOYEE, and so forth. In the example, the PEOPLE entity is the supertype, while the STUDENTS and EMPLOYEE entities are the subtype.

  Supertype contains all general characteristics about an entity, while subtype has all of its supertype characteristics plus some specific characteristics that applies only for the subtype. Moreover, the relationship between a supertype entity and a subtype entity is analogous to the parent and child relationship.

2.1.3.4.4 Attribute

An attribute specify the characteristic of an entity. Some examples of attributes in STUDENTS entity are: student id, student name, student age, and so on. Some important terminologies that are related to attributes:

- Simple attribute: the attribute that cannot be separated into several smaller elements. Student id and student age are some examples of simple attribute.
- Composite attribute: the attribute that can be separated into several smaller elements. An example of composite attribute is student address, which can be separated into street_address, city, state, and postal_code [7].
- Multivalued attribute: it is the attribute in an entity that could contain more than one value, for instance there is an attribute named skill in EMPLOYEE entity [7]; it is obvious that an employee could possess some skills.

- Derived attribute: this type of attribute has value that can be calculated from another attribute value, for example in EMPLOYEE entity, an attribute named Years_Employed can be calculated from another attribute value of Date_Employed and the current time [7].
- Identifier: an attribute that is able to uniquely identify an entity. Identifier can be considered as the equivalent of primary key in the relational keys. Employee id is one example of an identifier for EMPLOYEE entity.
- Composite identifier: a composite attribute that can uniquely identify an entity. For instance in FLIGHT entity, there Flight_ID attribute, which can be separated into two smaller elements (Flight_Number and Date [7]).

2.1.3.4.5 Relationships

Relationship is the association between two or more entities. According to Hoffer, Prescott, and McFadden [7], an instance of relationship "is an association between (or among) entity instances where each relationship instance includes exactly one entity from each participating entity type." An example of relationship instance: an employee who completes a course; figure 2.15 below could help the readers understand about the relationship instance concept.

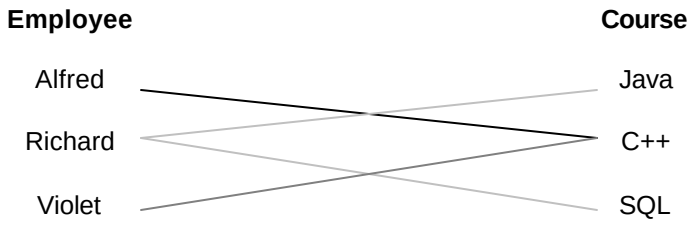


Figure 2.15 Relationship Instance [7]

Figure 2.15 depicts the information about the employees that has completed some courses. Each line in figure 2.15 represents a relationship instance between an employee and a course.

Relationship can be classified by the degree of a relationship. The degree of a relationship could show the readers about how many entities participate in a relationship. Generally, there are three types of relationship, based on the degree of a relationship, which are:

- Unary: this type of relationship only has single degree, means that only one entity that participate in the relationship. Unary relationship is often called recursive relationship, because "the same entity participates more than once in different roles" [5].
- Binary: it consists of two entities that participate in a relationship.
- Ternary: this relationship has three entities in which participate in a relationship.

Relationship can also be divided by its cardinality. The cardinality shows the number of instance of the participating entities in a relationship. Based on its cardinality, the relationship can be categorized into three types, which are:

- One-to-one: only one instance for each of the participating entities that participate within a relationship.
- One-to-many: this type of relationship associates one instance of an entity with zero, one, or many instances of another entity in a relationship.
- Many-to-many: it associates zero, one, or many instance of an entity with zero, one, or many instances of another entity in a relationship.

2.1.3.5 Normalization

Normalization is the process of removing anomalies within a set of relations in order to create well-structured relations. Normalization is done with several steps, which each step can be corresponded by a normal form. The definition of a normal form is "a state of a relation that results from applying simple rules regarding functional dependencies (or relationships between attributes) to that relation" [7]. There are three stages of normal form that are commonly used, namely:

- First normal form: the first normal form will remove any multivalued attributes.
- Second normal form: this normal form will remove any partial functional dependencies. A relation will reach this stage only after it is in first normal form and any partial functional dependencies have been removed. A functional dependency can be defined as "the constraint between two attributes or two sets of attributes in a relation" [7]. For example, if A and B are attributes of relation R, B is functionally dependent on A (denoted A → B), if each value of A is associated with exactly one value of B [5]. By understanding the definition of functional dependency, now the author can discuss about the partial functional dependency. According to Hoffer, Prescott, and McFadden [7], partial functional dependency is "a functional dependency in which one or more nonkey attributes are functionally dependent on part (but not all) of the primary key."
- Third normal form: this stage of normal form can be reached if a relation already in its second normal form and any transitive dependency has been removed. By its definition, a transitive dependency is a functional dependency between more than one nonkey attributes.

2.1.3.6 Structured Query Language (SQL)

SQL is a language is used for querying data from the database. Because SQL conforms to the relational database concept, consequently a relation is usually denoted by a table in SQL. SQL language is actually composed from two subsets, which are Data Definition Language (DDL) and Data Manipulation Language (DML).

DDL is responsible for the process of creating, deleting, and modifying the definitions of tables. Figure below shows a simple example of creating a table in a database.

```
CREATE TABLE employee
(
        eid int,
        ename varchar(100),
        age int,
        salary real,
        PRIMARY KEY (eid)
)
```

Figure 2.16 Creating a Table

In figure above, CREATE TABLE is a keyword to create a table and the following word (”employee”) represent the table’s name. Moreover, the opening bracket and closing bracket represent the preliminary point and concluding point of the table’s attributes definition. Employee table in figure 2.16 contains four attributes with its domain constraint. PRIMARY KEY is the keyword used to define a primary key for the table, which is “eid”.

DML allows users to insert, delete, update, and query data (records) into/from database. Let’s view the example of querying data from the database.

```
SELECT S.sid, S.sname
FROM Student S
WHERE S.sid = 710
```

Figure 2.17 Querying Data

SELECT keyword is used to identify the fields (columns) to be attained in the results FROM keyword determine the table from which the records must be attained. WHERE keyword specifies identification conditions on the tables determined in the FROM clause. The uppercase s (“S”) symbolizes an instance of Student table. In figure 2.17, it can be observed that the SQL syntax is trying to gather information about student id and student name from Student table, in which the student id is 710.

2.1.3.7 Database Backup and Recovery

Database backup and recovery is an important feature, because it could helps to recover the corrupted database when undesired failure happens to the system. The recovery is done by recovering the database to previous consistent state, which is called the checkpoint. The database backup process must be done regularly, usually once per day, in order to keep the most updated checkpoint.

2.1.4 System Analyst and Design

All modeling tools used in analyzing and designing the business process are discussed in this subsection.

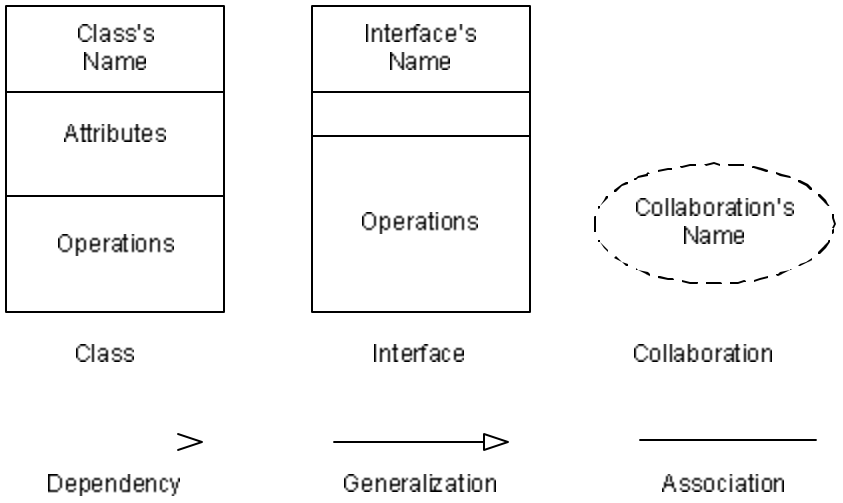
2.1.4.1 Class Diagram

Class diagram is a diagram that depicts the object-oriented model static structure. A class diagram could contain several things [9]:

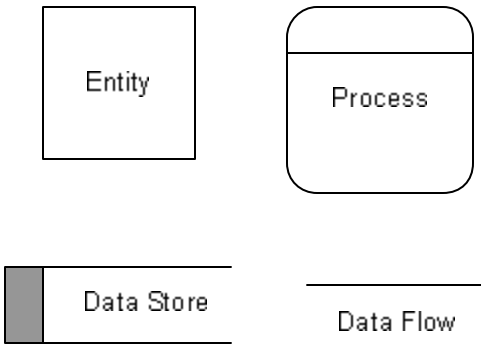
- Classes: “it is the description of a set of objects that share the same attributes, operations, relationships, and semantics” [9]. A class is denoted by a rectangle with three parts: the upper part contains the name of the class, the middle part is used to specify the attributes, and the lower part specifies the operations of the class.
- Interfaces: it is a unique class that only contains a set of operations, without any attributes.
- Collaborations: “is a collection of classes, interfaces, and other elements that collaborate for some purpose that is bigger than the sum of all its elements” [9]. The collaboration is represented by an ellipse with dashed line.
- Relationships: represent the association between things, like classes. Generally, the relationship can be divided into three classifications, which are:
    - Dependency: “is a using relationship that states that a change in specification of one thing may affect another thing that uses it, but not necessarily the reverse” [9]. Dependency is denoted with a dashed directed line that is directed to the thing being depended on.
    - Generalization: shows the relationship between the supertype classes and subtype classes (parent-child relationship). The supertype classes contains all common attributes and operations of the class, while the subtype classes inherits the supertype class’s attributes and operations with some specific attributes and operations that specifically could describe the subtype class.

Graphically, the generalization is denoted by directed line with a large open arrowhead, pointing to the parent [9].

- Association: “it shows the relationship between instances of classes” [10]. This type of relationship is graphically represented by a line.



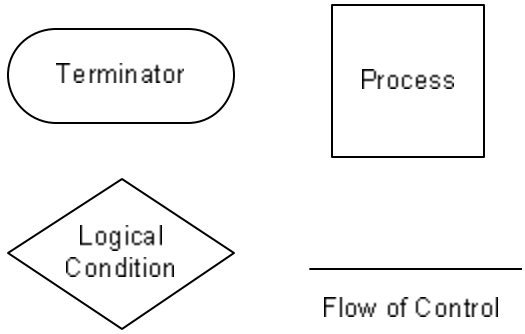
- Open-ended boxes: this symbol is used to denote the data store (files or database).
- Arrows: symbolize the data flows (input or output) to and from the processes.



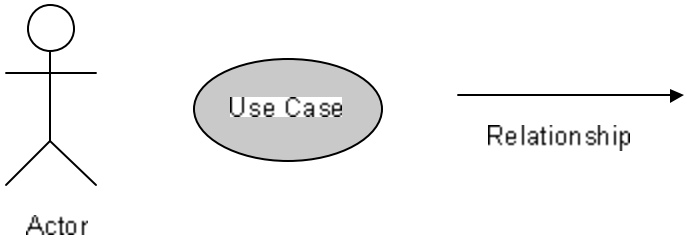
- The flow chart is not capable to show processes that have dramatically different timing, whilst a DFD is able to show such thing. A DFD could show the processes that operate hourly, daily, weekly, and on demand.

The notations of the flow chart are stated as follow:

- Rounded (and elongated) rectangle: this notation is used as the starting or finishing point of a flow chart.
- Square: represent the process.
- Diamond: symbolize the logical condition/selection (if-then-else).
- Arrow: “symbolize the flow of control” [11].



- Use case: as stated by Whitten, Bentley, and Dittman [3], use case is “a behaviorally related sequence of steps (a scenario), both automated and manual, for the purpose of completing a single business task. “ A use case is denoted by an ellipse.
- Actor: it is anything that initiates and interacts with use case in order to complete some tasks. A stick figure is used to illustrate an actor in use case diagram.
- Relationship: this element is denoted by an arrow and it is used to connect two elements in the use case diagram.



2.1.5.1 Microsoft .NET Framework

.NET framework is a new framework/API (Application Programming Interface) designed by Microsoft for the Windows platform programming. .NET framework can be considered as a complete solution that provides all tools to help the developers in programming their application. As stated by Troelsen [12], .NET framework has the following features:

- Full interoperability with existing code: it allows the existing code to be used with newer .NET framework.
- Complete and total language integration: .NET supports cross-language inheritance, cross-language exception handling, and cross-language debugging.
- A common runtime engine shared by all .NET-aware languages: .NET has a well-defined set of types that each .NET-aware language "understands."
- A base class library: this library provides shelter from the complexities of raw API calls and offers a consistent object model used by all .NET-aware languages.
- A truly simplified deployment model: under .NET, there is no need to register a binary unit into the system registry. Furthermore, .NET allows multiple versions of the same *.dll to exist in harmony on a single machine.

The features provided by .NET framework can be made possible because of the following three elements:

- Common Language Runtime (CLR): it is "the runtime execution environment" [13]. Some functions of CLR are to "locate, load, and manage .NET types on your behalf" [12].

Moreover, CLR also handles some of low-level aspects, like the memory management and security checks. A code that runs under the control of CLR is often called a managed code.

- Common Type System (CTS): it specifies "all possible data types and programming constructs supported by the runtime, specifies how these entities can interact with each other, and details how they are represented in the .NET metadata format" [12].
- Common Language Specification (CLS): it defines the standard of the subset of common types and programming constructs that are agreed by all .NET programming language. A combination of CLS and CTS is used to guarantee the language interoperability.

2.1.5.2 C# Programming Language

C# is a programming language that is designed especially for the .NET framework. All C# code can only run within the .NET framework and C# can be said as the successor of previous programming languages, namely C, C++ and Java. In addition, C# is claimed as "a simple, modern, object-oriented, and type-safe programming language" [13].

According to Robinson [13], some features of C# would include:

- Full support for classes and object-oriented programming, including interface and implementation inheritance, virtual functions, and operator overloading.
- A consistent and well-defined set of basic types.
- Built-in support for automatic generation of XML documentation.
- Automatic cleanup of dynamically allocated memory.

- The facility to mark classes or methods with user-defined attributes. This can be useful for documentation and can have some effects on compilation (for example, marking methods to be compiled only in debug builds).
- Full access to the .NET base class library, as well as easy access to the Windows API
- Pointers and direct memory access are available if required, but the language has been designed in such a way that you can work without them in almost all cases.
- Support for properties and events in the style of Visual Basic.
- Just by changing the compiler options, you can compile either to an executable or to a library of .NET components that can be called up by other code in the same way as ActiveX controls (COM components).
- C# can be used to write ASP.NET dynamic Web pages and XMLWeb services.

2.1.5.3 Microsoft Visual Studio .NET

Microsoft Visual Studio .NET is a fully integrated development environment (IDE) that designed specifically for .NET framework and it contains a set of useful supporting tools needed by the .NET programmer. Some of key features of Visual Studio .NET are listed as follow [13]:

- Text editor: this feature allows the programmer to write all of programming languages that are supported by .NET. This text editor is actually quite sophisticate because it could covers some features, such as color-coded keywords, lay out code in indenting lines, syntax error check, and so on.

- Design view editor: enables to place the user interface and data-access controls in the project, while Visual Studio .NET will automatically adds the required C# code in the source code.
- Supporting windows: it allows viewing and modifying some aspects of the project, like the classes of the source code and the available properties for Windows Forms and Web Forms classes.
- The ability to compile from within the environment. Instead of having to run the C# compiler from the command line, a menu option can simply be selected to compile the project and Visual Studio .NET will call the compiler automatically and pass all the relevant command-line parameters to the compiler.
- Integrated debugger: the debugger helps in finding the problems within the code by setting breakpoints in order to watch several variables from within the environment.
- Integrated Microsoft Developer Network (MSDN) help: enables the users to access the MSDN documentation from within the Visual Studio .NET.
- Access to other programs: Visual Studio .NET can be used to access other utilities for examining and modifying some aspects related to the computer or network, without any necessity to leave the IDE.

2.1.5.4 ADO (Active Data Object) .NET

ADO .NET is a component of .NET framework that useful for the database access and manipulation. ADO .NET provides support for some variety of data providers, including Object Linking and Embedding Database (OLEDB), Open Database Connectivity (ODBC), and Oracle.

Data provider is an intermediary component that connects .NET framework with the source of data or in other words, data provider is the database connector for .NET framework (ADO .NET). ADO .NET encompasses a comprehensive set of classes that enables the users to access and manipulate the database efficiently.

2.1.5.5 Microsoft SQL Server Express

Microsoft SQL Server Express is a relational DBMS (DBMS) that is developed by Microsoft and it implements the ANSI standard Structured Query Language (SQL) for data definition and manipulation. Microsoft SQL Server Express is actually a free distributable version of Microsoft SQL Server. Although it is a free version, SQL Server Express edition supports up to 1 CPU, 1GB of addressable memory, and 4GB of database size.

2.1.6 Local Area Network (LAN)

A LAN connects several numbers of computers within an isolated area, for example a building, a house, a university, etc. The computers that are connected in a LAN could communicate (sending/receiving data) with other computers in that LAN. Moreover, several devices or software could be attached into the LAN, which will enables each computer in that LAN to use those shared devices (scanner, printer, etc) and/or software.

2.1.7 Client and Server

This subsection covers about the definition of a client, a server, and the idea of client/server architecture.

2.1.7.1 Client

According to Hoffer, George, and Valacich [16], client is “the (front-end) portion of the client/server database system that provides the user interface and data manipulation functions.” A client could be a personal computer (PC), notebook, PDA, and so on. Usually, the client has limited computing capability because it only used to present the user interface and probably some (little) processing.

2.1.7.2 Server

Server is a computer system that possesses powerful computing capability and designed to perform a specific application or some applications. A server often has its own special design of software and hardware to support its functionality. Although in practice a PC can be used as a server, the computing capability is not as powerful as the “true” server.

2.1.7.3 Client/Server Architecture

As stated by Sommerville [17], the client/server architecture is “a distributed system model which shows how data and processing are distributed across a range of processors.”

If the clients and servers are positioned relatively close, then a LAN can be used in this architecture to interconnect those computers. Based on the distributed data and processing in client/server architecture, there are several types of server available [3]:

- Database server: this server hosts one or more shared databases, executed the database commands, and services for the clients. This type of server will be used in designing the business system for CV. X.
- Transaction server: it ensures that all database transaction updates succeed or fail as a whole.
- Application server: provides application logic and services for the clients.
- Messaging/groupware server: hosts several work group services, such as e-mail, calendaring, and so on.
- Web server: this type of server hosts the internet or intranet website service.

Generally, there are three types of client/server architecture [3] based on the layers that are distributed in the information system, namely:

- Distributed presentation: the presentation and presentation logic layers are reallocated from the server into the client.
- Distributed data: some of the data and data manipulation layers are sited in the server, while the application logic, presentation logic, and presentation layers are stored in the client. Distributed data and distributed presentation are also called the 2-tier client/server computing.
- Distributed data and application: in this type, the data and data manipulation layers are placed on their own server(s), the application logic is placed on its own server, whereas the presentation logic and presentation are located on the clients. Also called three-tiered or n-tiered client/server computing.

2.1.8 Overview of Business System

The overview about various business systems within the scope of this thesis, are the main topic of discussion of this subsection.

2.1.8.1 Human Resource System

According to Noe, Hollenbeck, Gerhart, and Wright [18], human resource information system is "a system used to acquire, store, manipulate, analyze, retrieve, and distribute human resource information." In other word, a human resource system can be used to handle all matters relating to the employees. The human resource system for CV. X will be able to handle information about the employees' payroll payment, attendance record, and biodata.

2.1.8.2 Cost and Bonus Systems

Cost can be described as resource that must be sacrificed in order to obtain specific objective. After the resource is used up, then that resource will become unavailable to be used anymore. In business term, the resource is usually defined to be equivalent with money and the objective can be regarded as product or service that is obtained. Moreover, the main function of a cost system is to record all costs within a company.

Bonus is something received freely from another person or company. Bonus is usually given in two types, which are money or merchandise. Furthermore, bonus system will record all bonuses received by the company.

2.1.8.3 Report

Report is defined "as organized information, distilled, and focused on answering specific questions" [20]. Report could represent and convey useful information that can be used for the decision making analysis. A report could include several elements, such as text, number, graph, and illustration. In addition, this thesis will utilize the integrated Report Viewer in Visual Studio to generate all reports that are related to Human Resource system and Cost system.

2.1.9 Security

The coverage of this subsection is about the algorithm needed to secure a stored text, like password or any other confidential text.

2.1.9.1 Secure Hash Algorithm (SHA)

SHA-1 is a hash algorithm that could take an input (message) of length less than 2^{64} bits and generates an output of 160-bit, which is called message digest. SHA-1 is often described as the successor of MD5 (Message Digest algorithm number five), which is one the most popular hash algorithm. As stated by Stallings [21], the comparison between SHA-1 and MD5 is discussed as follow:

- Security against brute-force attacks: SHA-1 produces a 160-bit output, while MD5 produces a 128-bit output. With brute-force attacks, MD5 have the possibility of 2^{128} operations to produce the original message from a given message digest, but SHA have 2^{160} possibilities.

Moreover, in order to produce two messages with same message digest, it requires 2^{64} operations for MD5 and 2^{80} operations for SHA-1. From those statements above, the author could conclude that SHA-1 will stand longer against the brute-force attacks.

- Security against cryptanalysis: cryptanalysis is the technique to decipher algorithm and usually it uses the algorithm's characteristics. Since the design process, MD5 already vulnerable to this kind of attack, in contrast SHA-1 isn't vulnerable to this attack.
- Speed/performance: SHA-1 is slower than MD5, because SHA-1 has bigger number of steps (80 compared to 64) and used bigger size of buffer (160-bit compared to 128-bit) to produce a message digest.
- Simplicity and compactness: both SHA-1 and MD5 are simple to describe and implement. Furthermore, they do not require large programs or substitution tables.
- Little-endian versus big-endian architecture: in order to interpret a message as a 32-bit words sequence, MD5 uses the little-endian scheme, while SHA-1 uses the big-endian scheme. Both approaches have no significant advantage.

Besides SHA-1, there are other types of SHA, which are SHA-256, SHA-384, and SHA-512. The comparison between all variant of SHA is shown in a table 2.1.

Characteristics	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160-bit	256-bit	384-bit	512-bit
Message size	$<2^{64}$ bits	$<2^{64}$ bits	$<2^{128}$ bits	$<2^{128}$ bits
Number of steps	80	80	80	80

Table 2.1 Comparison of All SHA Variant [21]

2.2 Theoretical Framework

Within this section, the main discussion will cover about Waterfall model of System Development Life Cycle (SDLC) that the author uses in constructing the design of solution.

2.2.1 Waterfall

Waterfall is one of the first model of SDLC that encompass several phases, namely system initiation phase, planning phase, system analysis phase, system design phase, coding phase, testing phase, implementation phase, and maintenance phase. In this model, the phases must be completed in a sequence or in other word a phase can be executed only after the previous phase has been completed. Because those phases are executed in sequence, output of a phase becomes the input of next phase. Furthermore, it is possible to revisit one phase before the current phase if needed.

The phases in Waterfall model can be customized and adjusted for the project's need. Usually, the large-scaled project follows all of the phases, while medium-scaled and small-scaled project could use a subset of the phases. In order to suit the needs of the author's thesis, the author decided to use a subset of the Waterfall phases, which is shown in figure below.

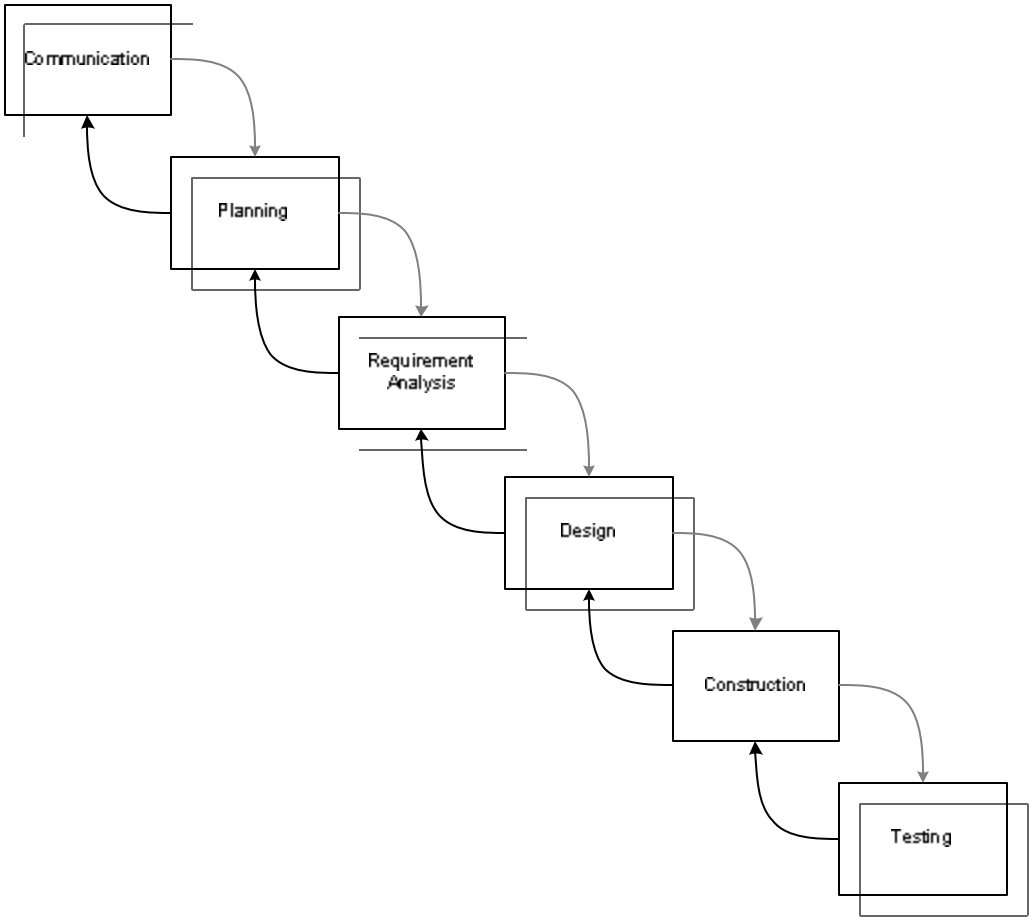


Figure 2.22 The Customized Waterfall Phases

2.2.1.1 Communication Phase

This phase can be described as initiation point of the project. In this phase, the requirement elicitation process is done to capture the business problems and needs. From the requirement elicitation process, a proposal document is produced to record the company background, the problem perceived, the project's scope, and the application's objective. In summary, the input for this phase is requirement elicitation, while the output is a proposal document.

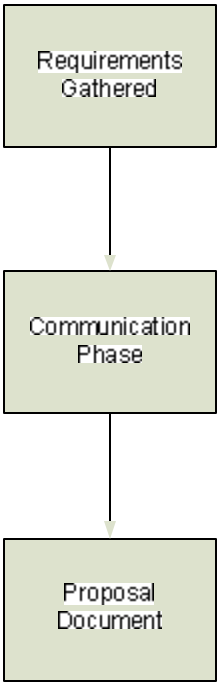


Figure 2.23 Communication Phase

2.2.1.2 Planning Phase

The proposal document from communication phase is used as the input for this phase. Based on the proposal document, a suitable system development life cycle is chosen, the project plan, and the project's schedule plan are generated. Moreover, the project plan comprises information about project's objective/benefit, project's organization, project scheduling (Gantt Chart), and Work Breakdown Structure (WBS) [3] [11]. Generally, this phase's input is the proposal document and the outputs are the system development life cycle model, Gantt Chart, and project plan document.

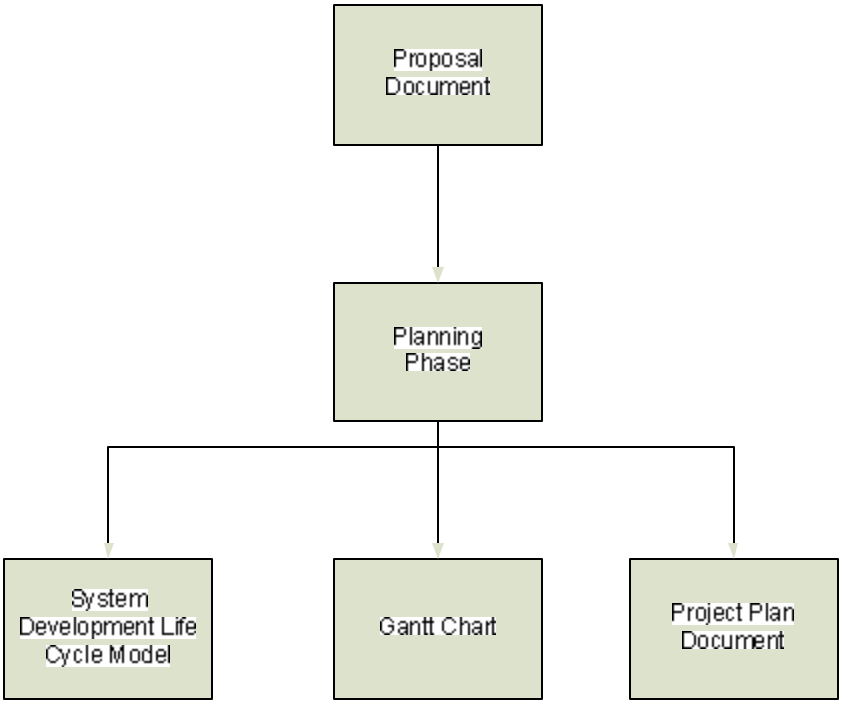


Figure 2.24 Planning Phase

2.2.1.3 Requirement Analysis Phase

In this phase, the input is the proposal document from the communication phase. Besides the proposal document, this phase also requires a thorough requirement elicitation process to gain better understanding about current business problems. This phase produce the requirement document that contains information about current system business process, existing problems, and the proposed solution.

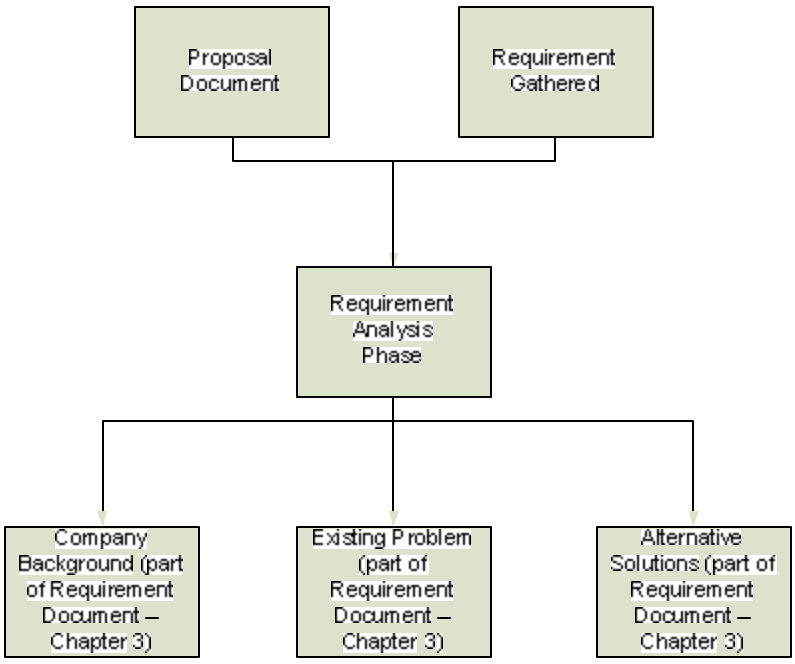


Figure 2.25 Requirement Analysis Phase

2.2.1.4 Design Phase

The requirement document from previous phase will be used in this phase to design the new business system. The design will includes the specification of new business system both logical and physical design. Logical design specifies the software system of the new business system, which consist of the data flow, business process flow, object relations, and users' interaction with business system. The logical design is graphically represented by several diagrams, including Entity Relationship Diagram (ERD), Data Flow Diagram (DFD), Flow Chart, Use Case Diagram, and Class Diagram. While logical design concern about the software, in contrast, the physical design concern about the hardware infrastructure. In addition, the design phase will produce a design document that contains the design specification for new business system.

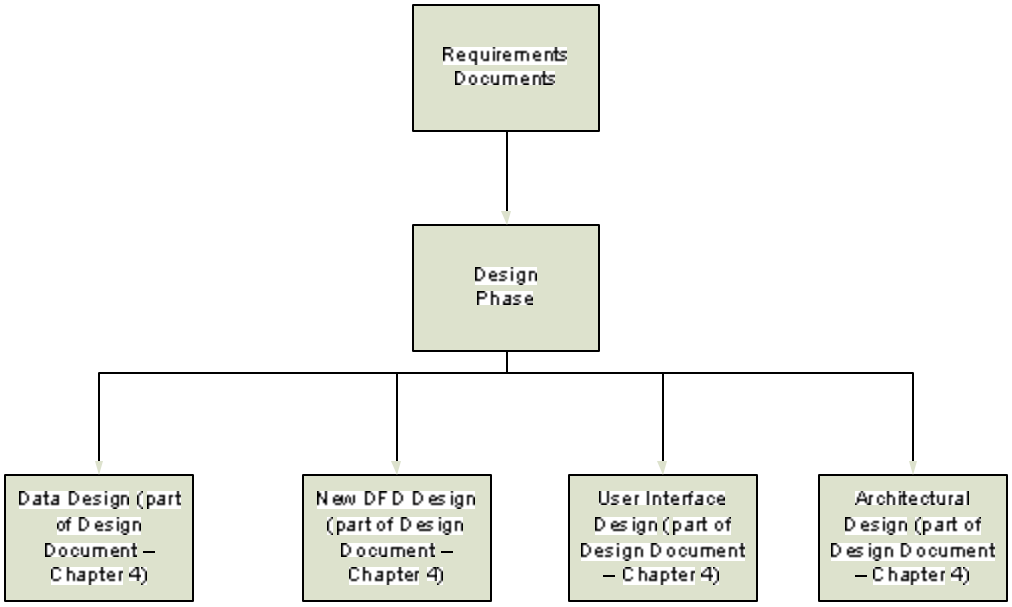


Figure 2.26 Design Phase

2.2.1.5 Construction Phase

This phase capture the design document from design phase, which will be transformed into functional software modules. This is the phase where coding activity take place, the software is integrated, and the user help document is written. Furthermore, there are three outputs produced by this phase, namely: functional software modules, integration of the software, and user help document.

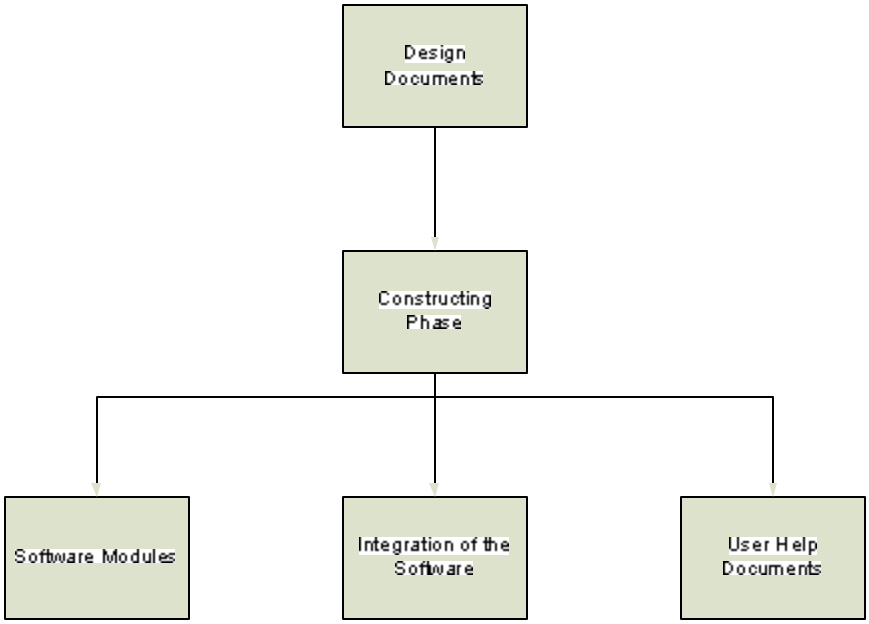


Figure 2.27 Constructing Phase

2.2.1.6 Testing Phase

In this phase, a verification process is performed to confirm whether the new business system able to meet the expected requirements. It will also try to identify any occurrence of error in the new business system and conduct the cost estimation document. A cost estimation document could give the readers a better understanding about the costs spent. In summary, testing phase requires the functional software modules, integration of the software, and user help document as input, while it will produce the testing documents, operation procedures, and cost estimation document as the output.

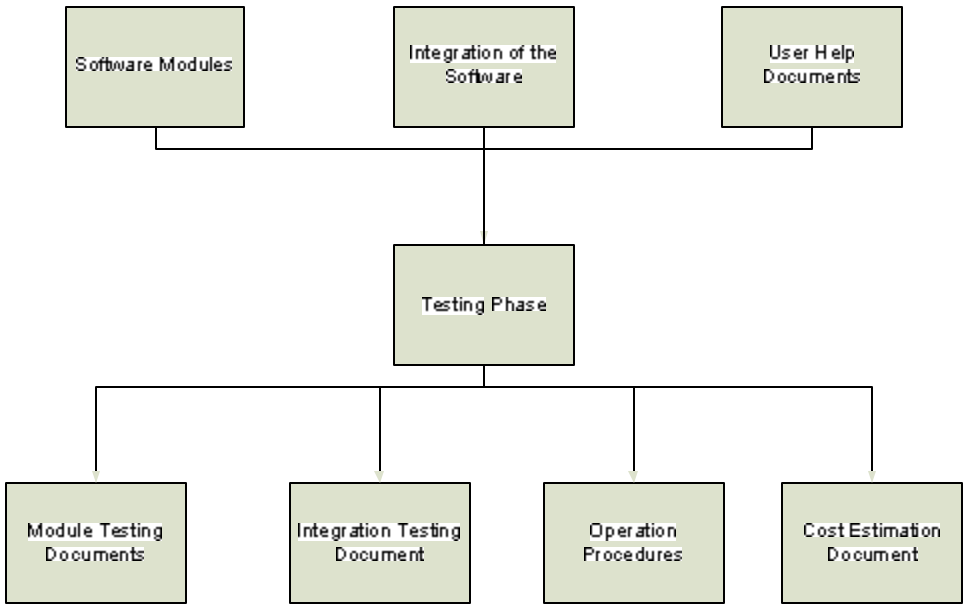


Figure 2.28 Testing Phase