CHAPTER 2

THEORETICAL FOUNDATION

2.1 THEORETICAL FOUNDATION

In this part, relevant theories is summarized and described comprehensively. Below are theoretical terms that would be used to support the thesis.

2.1.1 Information Systems

System is a collection of interrelated components that function together to achieve outcome. ***Information system*** is a collection of interrelated components that collect, process, store, and provide as output the information needed to complete business task. Both definitions are quoted from Satzinger, Jackson, and Burd (2002:6).

A system can be divided into several components, which is referred as ***functional decomposition***. *Functional decomposition* is dividing a system into components based on subsystems that in turn are further divided in subsystems (Satzinger, Jackson, and Burd, 2002:7).

2.1.2 Data and Information

Data are facts, texts, graphics, images, sound, and video segments that have meaning in the user's environment. ***Information*** is data that have been processed in such a way as to increase the knowledge of the person who uses the data. (Hoffer, Prescott, McFadden, 2002:5).

2.1.3 Database

A ***database*** is an integrated collection of stored data that is centrally managed and controlled. A database typically stores information about dozens or hundreds of entity types or classes. (Satzinger, Jackson, and Burd, 2002:394). An ***entity*** is any person, place, or thing with characteristics or attributes that will be included in the system. (Morris and Murphy, 2003:4).

2.1.4 Systems Development Life Cycle

Systems Development Life Cycle (SDLC) is the general term that is used to describe the method and process of developing a new information system. (Satzinger, Jackson, and Burd, 2002:33). ***Phase*** is a division of the SDLC where similar activities are performed. (Satzinger, Jackson, and Burd, 2002:34). Activities of every project can be classified into basic five *phases*, which are planning phase, analysis phase, design phase, implementation phase, and support phase. (Satzinger, Jackson, and Burd, 2002:35-39).

1. Planning Phase

Planning phase is the initial phase of the SDLC whose objective is to identify the scope of the new system and plan the project. (Satzinger, Jackson, and Burd, 2002:35). Five activities that are identified in the planning phase are:

- Define the problem
- Produce the project schedule
- Confirm project feasibility

- Staff the project
- Launch the project

2. Analysis Phase

Analysis phase is one phase of the SDLC whose objective is to understand the user needs and develop requirements. (Satzinger, Jackson, and Burd, 2002:36). Six primary activities that are considered part of this phase are:

- Gather information
- Define system requirements
- Build prototype for discovery of requirements
- Prioritize requirements
- Generate and evaluate alternatives
- Review recommendations with management

3. Design Phase

Design phase is the phase of SDLC where the system and programs are designed. (Satzinger, Jackson, and Burd, 2002:37). Seven major activities must be done during design phase are:

- Design and integrate the network
- Design the application architecture
- Design the user interfaces
- Design the system interfaces
- Design and integrate the database
- Prototype for design details
- Design and integrate the system controls

4. Implementation Phase

Implementation phase is the phase of the SDLC where the new system is programmed and installed. (Satzinger, Jackson, and Burd, 2002:38). Five major activities in implementation phase are:

- Construct software components
- Verify and test
- Convert data
- Train user and document the system
- Install the system

5. Support Phase

Support phase is the phase of SDLC that occurs after the system is installed. (Satzinger, Jackson, and Burd, 2002:39). Three major activities occur during support phase are:

- Maintain the system
- Enhance the system
- Support the users

2.1.5 Approaches to System Development

According to Satzinger, Jackson, and Burd, 2002:77, there are two general approaches to system development, that form the basis of virtually all methodologies, which are the ***traditional approach*** and ***object-oriented approach*.**

1. The Traditional Approach

Traditional Approach, which is also known as ***structured system development***, is a system development using structured analysis,

structured design, and structured programming techniques. (Satzinger, Jackson, and Burd, 2002:78).

Structured analysis is a technique that helps the developer define what the system needs to do (the processing requirements), what data the system needs to store and use (data requirements), what inputs and outputs are needed, and how the functions work together overall to accomplish tasks. (Satzinger, Jackson, and Burd, 2002:81).

Structured design is a technique providing guidelines for deciding what the set of programs should be, what each program should accomplish, and how the program should be organized into a hierarchy. (Satzinger, Jackson, and Burd, 2002:80).

2. The Object-Oriented Approach

Object oriented approach is an approach to system development that views an information system as a collection of interacting objects that work together to accomplish tasks. (Satzinger, Jackson, and Burd, 2002:84).

2.1.6 Methodologies, Models, Tools, and Techniques

System development methodology is comprehensive guidelines to follow for completing every activity in the system development life cycle, including specific models, tools, and techniques. (Satzinger, Jackson, and Burd, 2002:74)

Model is representation of some important aspect of the real world. (Satzinger, Jackson, and Burd, 2002:74). ***Tool*** is software support that helps

create models and other components required in the project. (Satzinger, Jackson, and Burd, 2002:75). ***Technique*** is collection of guidelines that help an analyst complete a system development activity and task. (Satzinger, Jackson, and Burd, 2002:76)

2.1.7 System Requirement

System requirements are all of the capabilities and constraints that the new system must meet. (Satzinger, Jackson, and Burd, 2002:112). In order to define system requirements, we need to gather all necessary information. Based on Satzinger, Jackson, and Burd, 2002:121, there are several techniques for information gathering, which are:

- Review existing reports, forms, and procedure descriptions
- Conduct interviews and discussions with users
- Observe and document business processes
- Build prototypes
- Distribute and collect questionnaires
- Conduct joint application design (JAD) sessions
- Research vendor solutions

2.2 THEORETICAL FRAMEWORK

Here, a coherent model, which shows the relationships between variables, should be formulated to seek the solution. The model should clarify how the design of solution may be constructed.

2.2.1 Data Flow Diagram

A ***data flow diagram*** (DFD) is a graphical system model that shows all of the main requirements from an information system in one diagram including inputs and outputs, processes, and data storage. (Satzinger, Jackson, and Burd, 2002:195). Five data flow diagram symbols are (Satzinger, Jackson, and Burd, 2002:196)

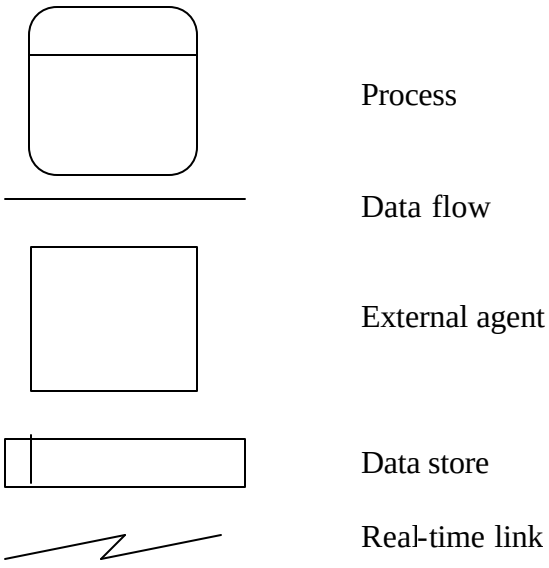

Figure 2. 1 DFD Symbols

1. **Process**
   Process is a symbol on DFD that represents an algorithm or procedure by which data inputs are transformed into data outputs. It is usually drawn as circle or process.
2. **Data Flow**
   Data flow is an arrow on a DFD that represents data movement among processes, data stores, and external agents.
3. **External Agent**
   A person or organization, outside the system boundary, that supplies data inputs or accepts data outputs. It is usually drawn as a rectangle.
4. **Data Store**
   A place where data are held pending future access by one or more processes.
5. **Real-time Link**
   Real-time Link is a communication back and forth between an external agent and a process as the process is executing.

High-level processes (a more general view of the system) on one DFD can be decomposed into separate lower-level (a more detailed view of one process). ***Levels of abstraction*** are any modeling technique that breaks the system into hierarchical set of increasingly more detailed models. (Satzinger, Jackson, and Burd, 2002:197).

2.2.2 Context Diagram

A ***context diagram*** is a DFD that summarizes all processing activity within the system in a single process symbol. All external agents and all data flows into and out of the system are shown in one diagram. (Satzinger, Jackson, and Burd, 2002:198).

A ***DFD fragment*** is a DFD that represents the system response to one event within a single process symbol. The fragments show details of interactions among the process, external agents and internal data stores. (Satzinger, Jackson, and Burd, 2002:200). All of the DFD fragments for a system or subsystem can be combined on a single DFD called the ***event-partitioned system model*** *or* ***diagram 0***. (Satzinger, Jackson, and Burd, 2002:202).

2.2.3 Entity-Relationship Diagram

Entity relationship diagram (ERD) is a graphical model of the data needed by a system, including things about which information is stored and the relationships among them, produced in structured analysis and information engineering. (Morris and Murphy, 2003: 4)

Entity relationship diagram symbolize data entities use rectangles and lines connecting the rectangles show the relationships among data entities. Types of relationships that can exist between two entities are shown below (Morris and Murphy, 2003: 4-5):

Figure 2. 2 Entity Relationship Symbols

1. **One-to-one**

   In a one-to-one relationship, each occurrence of data in one entity is represented by only one occurrence of data in the other entity.

2. **One-to-many**

   In a one-to-many relationship, each occurrence of data in one entity can be represented by many occurrences of the data in the other entity.

3. **Many-to-many**

   In a many-to-many relationship, data can have multiple occurrences in both entities.

2.2.4 C Programming Language

C is a general-purposed programming language. The language is not tied to any one operating system. It has been called a *system programming language* because it is useful to writing compilers and operating systems. It has been used equally well to write major programs in many different domains. (Kernighan and Ritchie, 2003, pg. 1) Some strength that C programming language has are (Kernighan and Ritchie, 2003, pg. 1-2):

- *C provides a variety of data types*

  The fundamental types are characters, integers, and floating-point numbers of several sizes. There is also a hierarchy of derived data types, which are created with pointers, arrays, structures, and unions

- *C provides the fundamental control-flow constructions required for well-structured programs*

  C provides a statement grouping, decision making (if-else), selecting one of a set of possible cases (switch), looping with termination test at the top (while, for) and at bottom (do), and early loop exit (break).

- *C is a low level language*

  C deals with the same sort of objects that most computers do namely characters, numbers, and addresses.

2.2.5 PHP Web-Programming Language

PHP is a server-side scripting language designed specifically for the Web. Within an HTML page, you can embed PHP code that will be executed every time the page is visited. You PHP's code is interpreted at the Web server and generate HTML or other output that visitor will see. (Welling and Thomson, 2003, pg. 2-3) PHP's Strengths are (Welling and Thomson, 2003, pg. 4-5)

- *High performance*

  PHP is very efficient. Using a single inexpensive server, you can serve millions of hits per day.

- *Database Integration*

  PHP has native connections available to many database systems. Using *Open Database Connectivity Standard (ODBC)*, you can connect to any database that provides an ODBC driver.

- *Built-in libraries for many common Web tasks*

  Because PHP was designed for use on the Web, it has many built-in functions for performing many useful Web-related tasks. You can generate GIF images on-the-fly, connect to other network services, send

email, work with cookies, and generate PDF documents, all with just a few line of code.

- *Low cost*

  PHP is free. You can download the latest version at any time form http://www.php.net for no charge.

- *Ease of learning and use*

  The syntax of PHP is based on other programming languages, primarily C and Perl. If you already know C or Perl, or a C-like language such as C++ or Java, you will be productive using PHP almost immediately.

- *Portability*

  PHP is available for many different operating systems. You can write PHP code on the free Unix-like operating systems such as Linux, FreeBSD, commercial UNIX versions such as Solaris and IRIX, or on different versions of Microsoft Windows. Your code will usually work without modification on a different system running PHP.

- *Availability of source code*

  You have access to the source code of PHP. Unlike commercial, closed-source products, if there is something you want modified or added to the language, you are free to do this. You do not need to wait for the manufacturer to release patches. You do not need to worry about the manufacturer going out of business or deciding to stop supporting a product.

2.2.6 MySQL Database Tool

MySQL is very fast, robust, *relational database management system (RDBMS)*. A database enables you to efficiently store search, sort, and retrieve data. The MySQL server controls access to your data to ensure that multiple users can work with it concurrently, to provide fast access to it, and ensure that only authorized users can obtain access. Hence, MySQL is a multi-user, multi-threaded server. It uses SQL (*Structured Query Language*), the standard database query language worldwide. (Welling and Thomson, 2003, p3). MySQL's Strengths are (Welling and Thomson, 2003, p6)

- *High performance*

  MySQL is undeniably fast.

- *Low cost*

  MySQL is available at no cost, under an Open Source License, or at low cost under a commercial license if required for your application.

- *Easy to configure and learn*

  Most modern databases use SQL. If you have used another RDBMS, you should have no trouble adapting to this one. MySQL is also easier to set up than many similar products

- *Portable*

  MySQL can be used on many different UNIX systems as well as under Microsoft Windows.

- *The source code is available*

  You can obtain and modify the source code for MySQL.