Chapter 3

Structured Methodologies

Introduction

Structured methodologies (or structured systems analysis and design) have been used to document, analyze, and design information systems since the 1970s. **Structured** refers to the fact that the techniques are step by step, with each step building on the previous one. Structured methodologies are top-down, progressing from the highest, most abstract level to the lowest level of detail – from the general to the specific.

Structured development methods are process-oriented, focusing primarily on modeling the processes, or actions that capture, store, manipulate, and distribute data as the data flow through a system. These methods separate data from processes.

The primary tool for representing a system's component processes and the flow of data between them is the **data flow diagram (DFD)**. The data flow diagram offers a logical graphic model of information flow, partitioning a system into modules that show manageable levels of detail. It rigorously specifies the processes or transformations that occur within each module and the interfaces that exist between them.

Another tool for structured analysis is a **data dictionary**, which contains information about individual pieces of data and data groupings within a system. The data dictionary defines the contents of data flows and data stores so that systems builders understand exactly what pieces of data they contain. **Process specification** is another tool that describes the transformation occurring within the lowest level of data flow diagrams. They express the logic for each process. We can express the logic using structured English, decision table, decision tree etc.

In structured methodology, software design is modeled using hierarchical structure charts. The **structure chart** is a top-down chart, showing each level of design, its relationship to other levels, and its place in the overall design structure. The design first considers the main function of a program or system, then breaks this function into subfunctions, and decomposes each subfunction until the lowest level of detail has been reached. A structure chart may document one program, one system (a set of programs) or part of one program.

The Need for Structured Methodology

Most information systems development organizations use structured methodology because it offers the following advantages:

- Improve project management & control
- Make more effective use of experienced and inexperienced development staff
- Develop better quality systems with low cost
- Make projects resilient to the loss of staff
- Enable projects to be supported by computer-based tools such as computer-aided software engineering (CASE) systems
- Establish a framework for good communications between participants in a project

- Enable projects to deliver the product on time because it allows to plan, manage and control a project well
- Respond to the changes in the business environment while project progresses
- Improves the overall productivity by encouraging on-time delivery, meeting business requirements, ensuring better quality, and using human resources effectively.

Role of CASE in Data Modeling

Data models are stored in a repository. CASE (computer-aided systems engineering) provides the repository for storing the data models and its detail description. Most CASE tools support computer-assisted data modeling and database design. CASE supports in drawing and maintaining the data models and their underlying details.

Using CASE product, you can easily create professional, readable data models without the use of paper, pencil, erasers, and templates. The models can be easily modified to reflect corrections and changes suggested by end users – you don't have to start over. Also, most CASE products provide powerful analytical tools that can check your models for mathematical errors, completeness, and consistency. Some CASE products can even help you analyze the data models for consistency, completeness, and flexibility.

Also, some CASE tools support reverse engineering of existing files and database structures into data models. The resulting data models represent physical data models that can be revised and reengineered into a new file or database, or they may be translated into their equivalent logical models. The logical data models could then be edited and forward engineered into a revised physical data model and subsequently a file or database implementations.