Chapter 6
Decision Support System Development

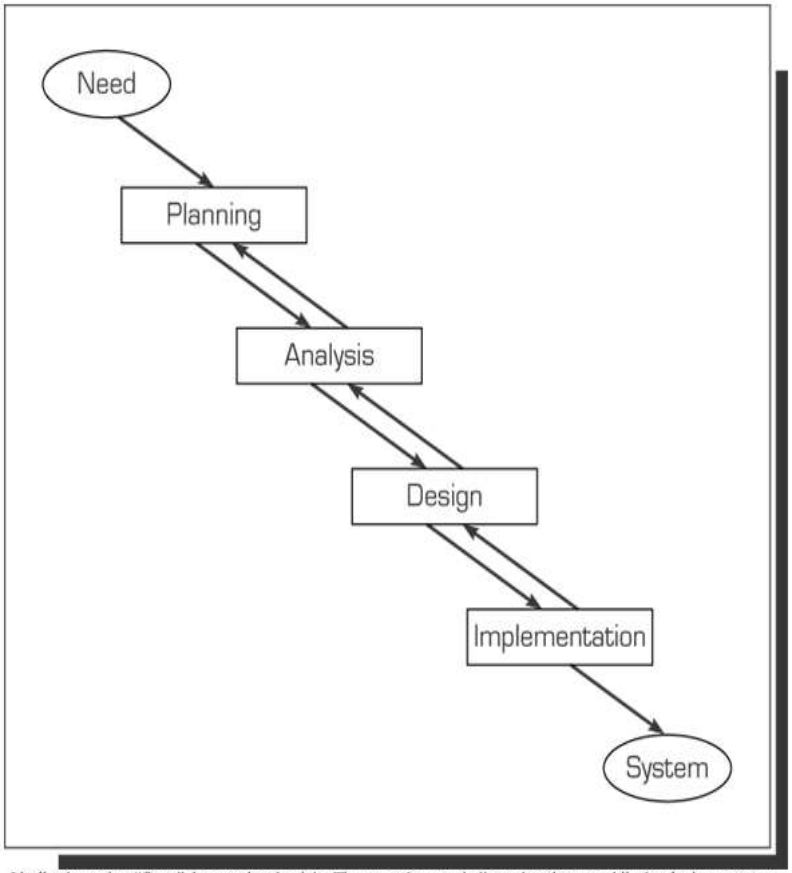
Learning Objectives

- DSS user interface design and management.
- Understand the concepts of systems development.
- Learn **PADI**, the phases of SDLC.
- Describe prototyping.
- Understand which factors lead to DSS success or failure.
- Learn the importance of project management.
- Describe the three technology levels of DSS.
- Understand the learning process involved in DSS development.
- DSS hardware, software, and technology Levels.
- decision support system construction methods

Traditional Systems Development Life Cycle (SDLC)

- A structured approach for managing the development of information systems.
- Four phases (PADI)
  - **P**lanning
  - **A**nalysis
  - **D**esign
  - **I**mplementation
- Cyclical
- Can return to other phases

Figure 6.1 The Traditional System Development Life Cycle (SDLC)



Ideally, the project "flows" down and to the right. The upward arrows indicate that changes while developing a system can return the process to an earlier stage. This is also known as *waterfall development*.

The Traditional SDLC

Major Phase	*Minor Step*	*Deliverable*
Planning: Why build the system?	1. Identify business value	System request
	2. Analyze feasibility	Feasibility study
	3. Develop work plan	Work plan
	4. Staff project	Staffing plan
	5. Control and direct project	Project plan Project charter Project management tools CASE tool Standards list Project "binders" or files
Analysis: Who, what, when, and where will the system be?	6. Analyze problem	Risk assessment Analysis plan
	7. Gather information	Information
	8. Model process(es)	Process model
	9. Model data	Data model
Design: How will the system work?	10. Design physical system	Design plan Architecture design Infrastructure. design
	11. Design architecture	Interface design
	12. Design interface	Data storage design
	13. Design database and files	Program design
	14. Design program(s)	Test plan
Implementation: System delivery	15. Construction	Programs Documentation
	16. Installation	Conversion plan Training plan

Source: Based on Dennis and Wixom (2000).

CASE Tools

These tools are essentially information systems, for systems analysts and can help manage every aspect of developing a system.

- Upper CASE : CASE tools that assist in the analysis phase in creating system diagram
- Lower CASE CASE tools that mange the diagram and generate code for the database tables
- Integrated CASE (I-CASE)tools

do both.

Some CASE tools are designed to handle strictly object-oriented systems by supporting the construct of the universal modeling language (UML).

Other Analysis And Design Tools

- RAD design tools
  - Enterprise class repository and collaboration
  - UML modeling
- Analysis and design tools
  - Rational RequisitePro helps identify requirements.
  - Rational ClearQuest provides reliable and efficient project metrics. It includes reports, charts, and a Web interface.

Code debugging and testing Tools

- It is better to adopt methods that help identify problems early in the development process
  - Code debugging tools
  - Automated testing and quality assurance tools

Successful Project Management

- Establish a baseline
- Define scope of project
- Manage change and scope creep
- Get support from upper management
- Establish timelines, milestones, and budgets based on realistic goals
- Involve users
- Document everything

Project Failures

- Lack of stakeholder involvement
- Incomplete requirements
- Scope creep
- Unrealistic expectations
- Project champion leaves
- Lack of skill or expertise
- Inadequate human resources
- New technologies

Project Management Tools

- Project management software can allow:
  - Collaboration among disparate teams
  - Resource and program management
  - Portfolio management
    - Tools to analyze and collectively manage a group of current or proposed projects based on numerous key characteristics to determine the optimal resource mix for delivery .
  - Web enabled
  - Aggregates and analysis project data

Other Development Methodologies

- Parallel development Methodologies.
- Rapid application development Methodologies.
  - Phased development
  - Prototyping
  - Throwaway prototyping
- Agile Development and extreme programing Methodologies.

Parallel development Methodologies

- In a parallel development, the design and implementation phases split into multiple copies following the analysis phase. Each of this copies involves development of a separate subsystem or subproject. They all come together in a single implementation phase in which a system integrator puts the pieces together in a cohesive system.

- Part of DSS implementation is handled in parallel development Methodologies
  - The following four components can essentially be developed in parallel.:
    1. database
    2. model base
    3. user interface
    4. knowledge

Rapid application development Methodologies. (RAD)

- RAD
  - Quick development allowing fast, but limited functionality i.e. system can be developed quickly and users can obtain some functionality as soon as possible. It includes:
    - Phased development
      - System broken up into a series of versions
      - Sequential serial development with incremental functionality
        - » Disadvantage: incomplete
        - » Advantage: each series is apart of the final system
    - Prototyping
      - Rapid development of portions of projects for user input and modification
      - Small working model or may become functional part of final system
    - Throwaway prototyping
      - simple development platforms
      - Used when the project idea is not clear

Prototype vs. Throwaway Prototype

Figure 6.3 Prototyping Development Process: A Rapid Application Development (RAD) Method

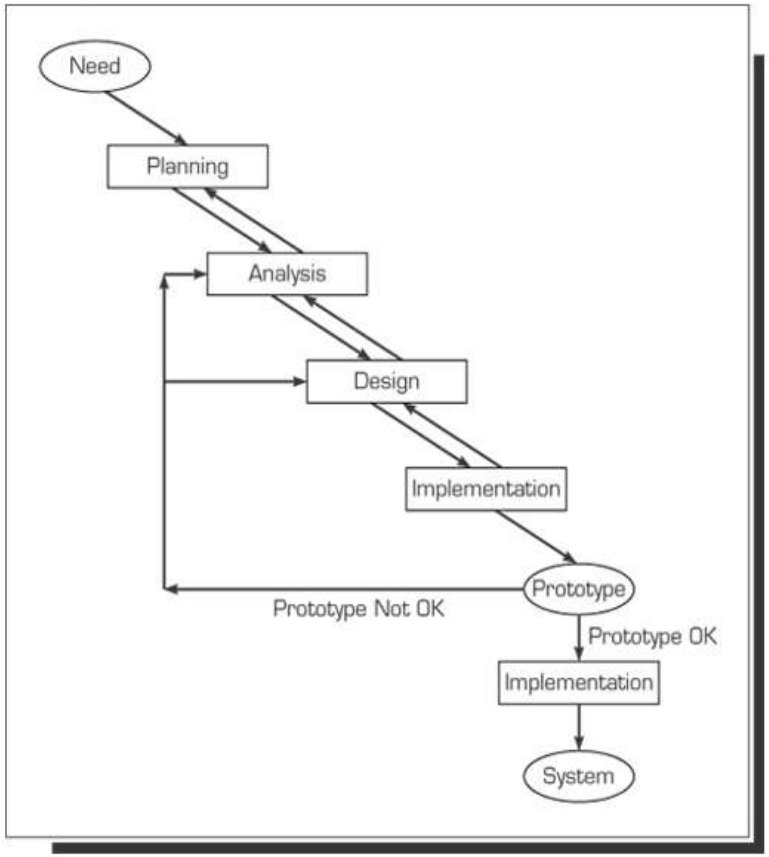
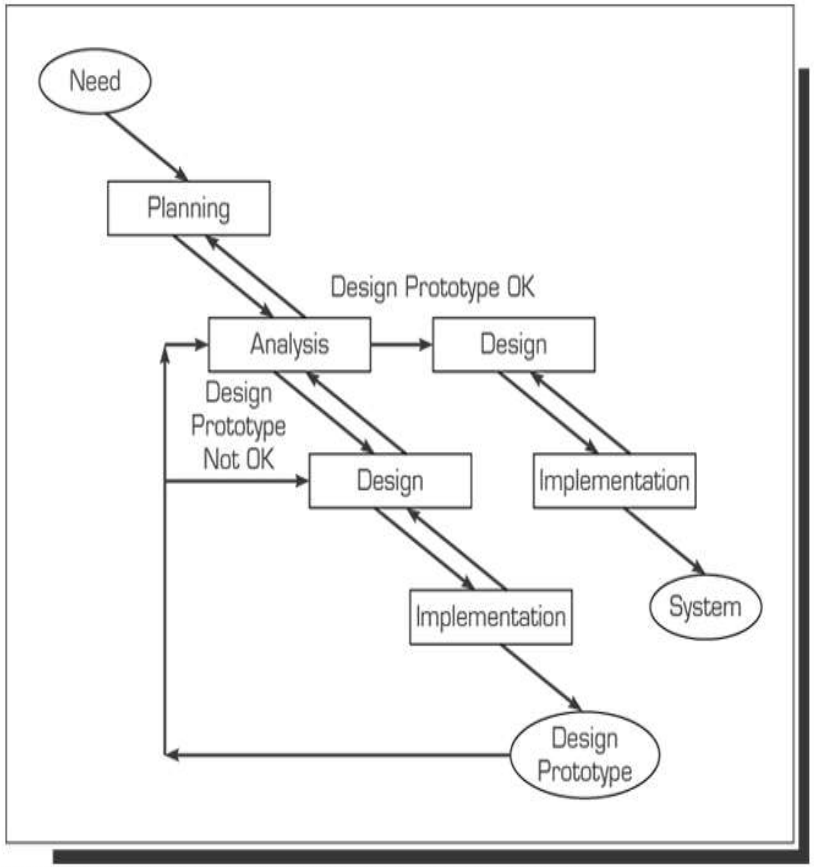


Figure 6.4 Throwaway Prototyping Development Process: Another Rapid Application Development Method



The design prototype helps the team work out details used in the system that is developed.

Agile Development

- A new form of rapid prototyping which attempts to bypass much of the formalism inherent in the system-development life cycle, and even that of prototyping

- Used for:
  - Unclear or rapidly changing requirements
  - Speedy development
- General Characteristics of the Agile:
  - Heavy user input
  - Self organized team
  - Incremental delivery with short time frames
  - Tend to have integration problems
  - Create simple content
  - Test oriented
  - Several design models

Agile Development

- Extreme Programming (XP) is the most popular example of agile processes, yet there are many other ones. its features include:
  - User *stories* or needs that determine software features are usually written on index cards and are the basis of requirements for a project.
  - Simple functionality tests are written by users before coding begins.
  - Coding is broken down into very small segments of functionality that can be completely coded in two or three days.
  - Programmers work in pairs; projects rotate frequently among teams, giving all programmers an understanding of the entire project.
  - Code is frequently refactored and revised to improve quality and performance.

Prototyping : The DSS Development Methodology

- DSS development is done through proto typing for the following reasons:
  - Users and managers are involved in every phase and iteration.
  - Learning is explicitly integrated into the design process
  - *Prototyping essentially bypasses the formal* life-cycle
  - A key criterion associated with prototyping is the short interval between iterations.
  - *The initial prototype must be low-cost*

DSS Prototyping

- Short steps
  - Planning
  - Analysis
  - Design
  - Prototype
- Immediate stakeholder feedback to ensure that the development is proceeding correctly
- Iterative
  - In development of prototype
  - Within the system in general
  - Evaluation is integral part of the development process
    - Control mechanism

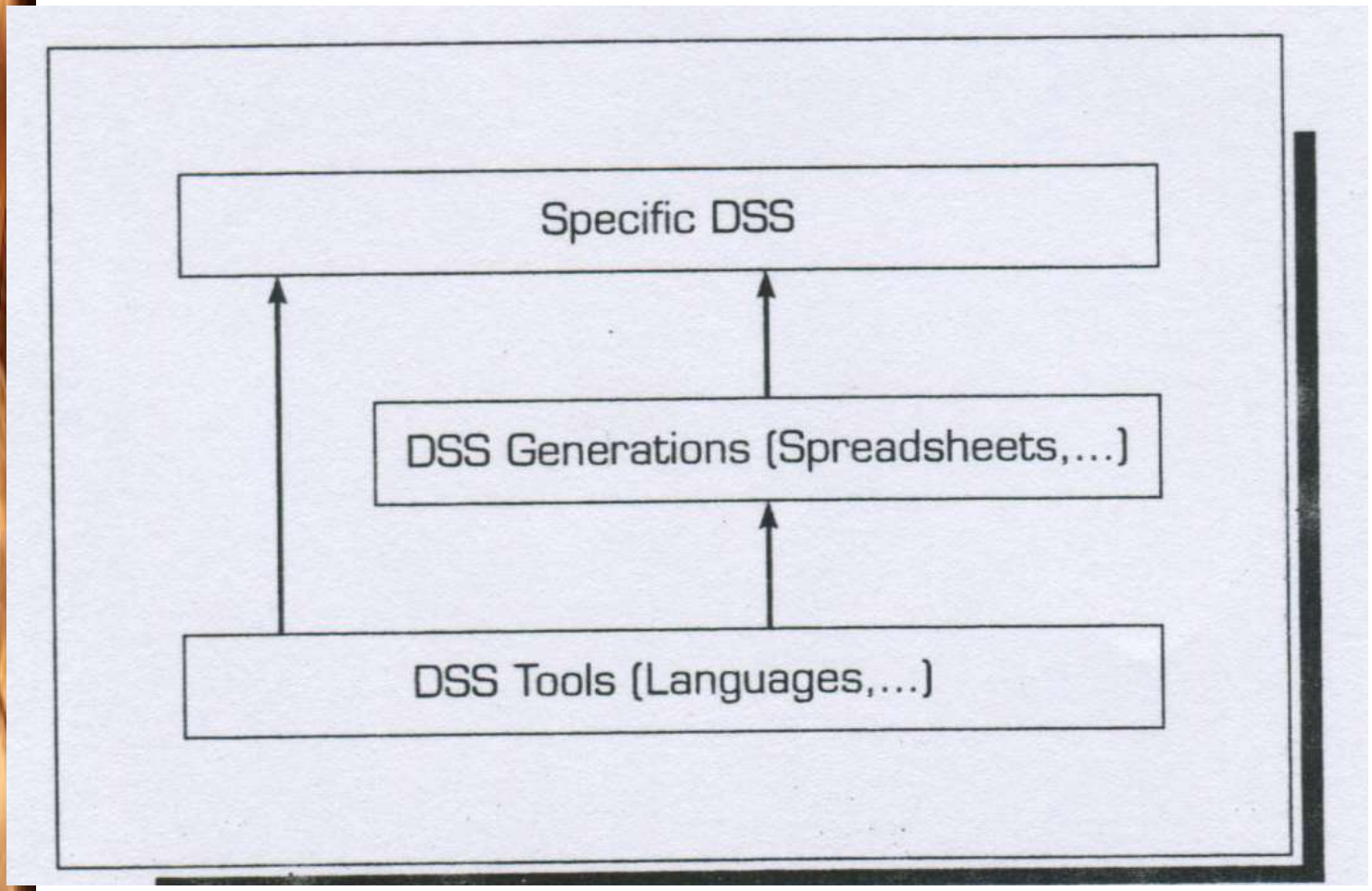
DSS Prototyping

- Advantages
  - User and management involvement
  - Learning explicitly integrated
  - Prototyping bypasses information requirement
  - Short intervals between iterations
  - Low cost
- Disadvantages
  - Changing requirements
  - May not have thorough understanding of costs
  - Dependencies, security, and safety may be ignored
  - Higher costs due to multiple productions

DSS Technology Levels

- DSS primary tools
  - Fundamental elements
    - Programming languages, graphics, editors, query systems
- DSS generator (engine)
  - Integrated software package for building specific DSS
    - Modeling, report generation, graphics, risk analysis
  - Examples: Excel, OLAP systems, Lingo.
- Specific DSS
  - DSS application that accomplishes the work
- DSS primary tools are used to construct integrated tools (generators) that are used to construct specific tools

DSS Technology levels



DSS Development Tool Selection

- Hardware
  - PCs to multiprocessor mainframes
- Software
  - Involves multiple criteria (when selecting software)
  - Off the shelf software rapidly updated; many on market
  - Price changes are frequent
  - The desirability of staying with a few vendors
  - Technical, functional, end-user, and managerial issues are all considered

Team developed DSS

- Team developed DSS requires substantial effort to build and manage it.
- The systems are constructed by a team composed of users, DSS developers, technical support experts, and IS personnel.
- Developing a DSS with a team is a complex, lengthy, costly process.
- Since early 2000s, tools and generators have improved, smaller teams can handle complex DSS development.

End user developed DSS

- Decision-makers and knowledge workers develop to solve problems or enhance productivity
  - Advantages
    - Short delivery time
    - The prerequisites of extensive and formal user requirements specifications are eliminated
    - Reduced implementation problems
    - Low costs
  - Risks
    - Quality may be low
    - May have lack of documentation
    - Security risks may increase

Developing DSS: Putting the system together

- Two important concepts:
  - The use of highly automated tools throughout the DSS development process
  - The reuse of prefabricated components.
- DSS is much more than just a DBMS, MBMS, GUI, interface, and knowledge component. There are interfaces among the components and with outside systems.
- The system core includes a development language or a DSS generator.