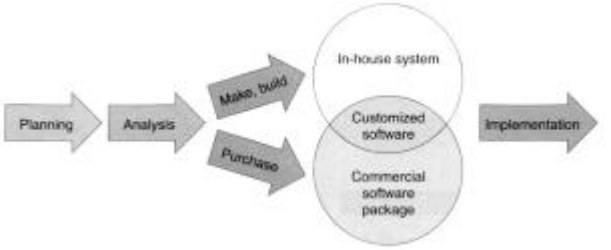
Transition to System Design

Peter Lo

Evaluating Software Alternatives

- Examine software alternatives and select an overall strategy for the proposed system to prepare for the transition to the systems design phase.



Developing Software In-House

- Satisfy unique business requirements, to minimize changes in business procedures and policies.
  - Satisfy Unique Business Requirements
  - Minimize Chances in Business Procedure
  - Meet Constraints of Existing Systems
  - Meet Constraints of Existing Technology
  - Develop Internal Resources and Capabilities

Purchasing a Software Package

- A commercially available software package could satisfy system requirements.
  - Lower Cost
  - Less Time to Implement
  - Proven Reliability and Performance Benchmarks
  - Less Technical Development Staff
  - Future Upgrades Proved by the Vendor
  - Other Companies as Resources

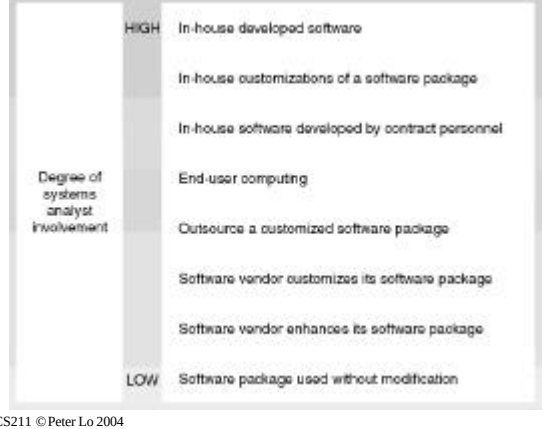
Customizing Software Packages

- Acquire a package that can be customized to meet the needs of an organization.
  - Purchase a basic package that vendor will customized to suit your needs
  - Negotiate directly with the software vendor to make enhancements to meet your needs by paying extra charge
  - Purchase the package and make your own modification.

Other Software Alternatives

- Other possibilities include using an application service provider, outsourcing, and developing end-user applications.
  - Application Services Providers
  - Outsourcing
  - User Application

Selecting a Software Alternative



Steps in Evaluating and Purchasing Software Packages

1) Evaluate the Information System Requirements
   - Identify the Key Features of the system
   - Estimate Volume and Future Growth
   - Specify any Hardware Constraints
   - Prepare a Request for Proposal or Quotation
2) Identify Potential Software Vendors
3) Evaluate Software Package Alternatives
4) Make the Purchase
5) Install the Software Package

Completion of Systems Analysis

- To complete the systems analysis phase, SA must prepare the System Requirements Document and present to management.

System Requirements Document

- The system requirements document (or software requirements specification):
  - Contains the requirements for the new system
  - Describes the alternatives that were considered
  - Makes a specific recommendation to management.

Presentation to Management

- To obtain approval for the development of the system and to gain management's full support, including necessary financial resources.
- Depending on their decision, the next task as a Systems Analyst will be one of the following:
  - Develop an In-House System
    - Begin the systems design phase for the new system.
  - Modify the Current System
    - Begin the systems design phase for the modified system.

Presentation to Management

  - Purchase or Customize a Software Package
    - Negotiate the purchase terms with the software vendor for management approval.
  - Perform Additional Systems Analysis Work
    - Management might want you to investigate certain alternatives.
  - Stop all Further Work
    - The decision might be based on your recommendation, a shift in priorities or costs, or for other reasons.

Transition to Systems Design

- If management decides to develop the system in-house, then the next phase of the Software Development Life Cycle (SDLC) – the systems design phase can begin.
- As proceed to the next phase, must be certain that performed a thorough and accurate systems analysis and communicated the results in system requirements document.

Systems Design Overview

- The logical design is completed during the systems analysis phase of the SDLC.
- To create the logical design, investigated, analyzed, and documented the input, processing, and output requirements and constraints.
- In the systems design phase, ready to begin work on the physical design of the system.

Relationship between Analysis and Design

- During systems design, you should return to the analysis phase only in very limited situations.
- You might Return to Fact-Finding:
  - If you discover that you overlooked an important fact,
  - If users have significant new needs, or
  - If legal or governmental requirements change.
- You might Return to Requirements Analysis:
  - If you encounter unforeseen design issues or problems.

Prototyping

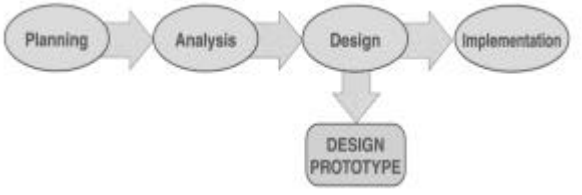
- A prototype is an early, rapidly constructed working version of the proposed information system.

Approaches to Prototyping

- Prototyping involves a repetitive sequence of analysis, design, modeling and testing.
- The end product of **System Prototyping** is a full-featured, working model of the information system, ready for implementation.

Approaches to Prototyping

- Systems analysts also use **Design Prototyping** or **Throwaway Prototyping** to verify user requirements, after which the prototype is discarded and implementation continues.
- The end product of design prototyping is a user-approved design prototype that documents and benchmarks the features of the finished system.



Benefits of Prototyping

- Avoid misunderstanding.
- Create accurate specification.
- Evaluate a working model more effectively.
- Develop testing and training procedures before the finished system is available.
- Reduces the risk and potential financial exposure that occur when a finished system fails to support business needs

Potential Problems of Prototyping

- The rapid pace of development can create quality problems, which are not discovered until the finished system is operational.
- Other system requirements, such as reliability and maintainability, cannot adequately be tested using a prototype.
- In very complex systems, the prototype becomes unwieldy and difficult to manage.

Prototyping Tools

- Application Generators
  - An application generator also called a code generator.
- Report Generators
  - A report generator, also called a report writer, is a tool for designing formatted reports rapidly.
- Screen Generators
  - A screen generator, or form generator, is an interactive software tool that helps you design a custom interface, create screen forms, and handle data entry format and procedures.

Limitations of Prototypes

- A prototype is a functioning system, but it is less efficient than a fully developed system.
- The final version of the system typically demands higher-level performance than the prototype can provide.
- It is a model, rather than a completed system, the prototype will have slower processing speeds and response times.
- The prototype also might lack security requirements, exception and error handling procedures, and other required functions.

Overview of Systems Design

Step	Activity	Description
1	**Review system requirements.**	Become familiar with the logical design.
2	**Design the system.**	
	• User interface	Design an overall user interface, including screens, commands, controls, and features that enable users to interact with an application.
	• Input processes	Determine how data will be input to the system and design necessary source documents.
	• Input and output formats and reports	Design the physical layout for each input and output screen and printed report.
	• Data	Determine how data will be organized, stored, maintained, updated, accessed, and used.
	• System architecture	Determine processing strategies and methods, client/server interaction, network configuration, and Internet/intranet interface issues.
3	**Present the systems design.**	Create the systems design specification document, in which you describe the proposed system design, the anticipated benefits of the system and the estimated development and implementation costs.

Systems Design Objectives

- The goal of systems design is to build a system that is effective, reliable, and maintainable.

Systems Design Considerations

- User Considerations:
  - Of the many issues you must consider during systems design, your most important goal is to make the system acceptable to users, or user-friendly.
- Data Considerations:
  - Data entry and storage considerations are important parts of the system design.
- Processing Considerations:
  - Use a modular design, Design modules that perform a single function. Independent modules provide greater flexibility.

Designing and Using Codes

- A code is a set of letters or numbers that represents a data item.
- Codes can be used to simplify output, input, and data formats.

Types of Coding

- Sequence codes
  - Numbers or letters assigned in a specific order.
- Block sequence codes
  - Use blocks of numbers for different classifications.
- Alphabetic codes
  - use alphabet letters to distinguish one item from another based on a category, an abbreviation, or an easy-to-remember value, called a mnemonic code.

Types of Coding

- Significant digit codes
  - Distinguish items by using a series of subgroups of digits. ZIP codes, for example, are significant digit codes.
- Derivation codes
  - Combine data from different item attributes, or characteristics, to build the code.
- Cipher codes
  - Use a keyword to encode a number.

Types of Coding

- Action codes
  - Indicate what action is to be taken with an associated item.
- Self-checking codes
  - Use a check digit to verify the validity of a numeric code.

Developing a Code

- Keep codes concise
- Allow for expansion
- Keep codes stable
- Make codes unique
- Use sortable codes
- Avoid confusing codes
- Make codes meaningful
- Use a code for a single purpose
- Keep codes consistent