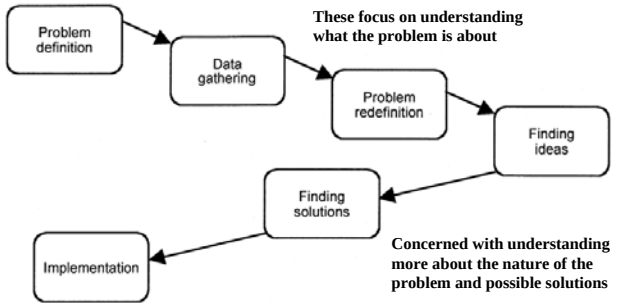
Software Development Methodology

Chapter 2A

In this lecture you will learn:

- The stages in the waterfall life cycle
- About prototyping and incremental life cycles
- The importance of project management
- How users may be involved in a project
- The role of CASE tools in systems development

General Problem Solving Model



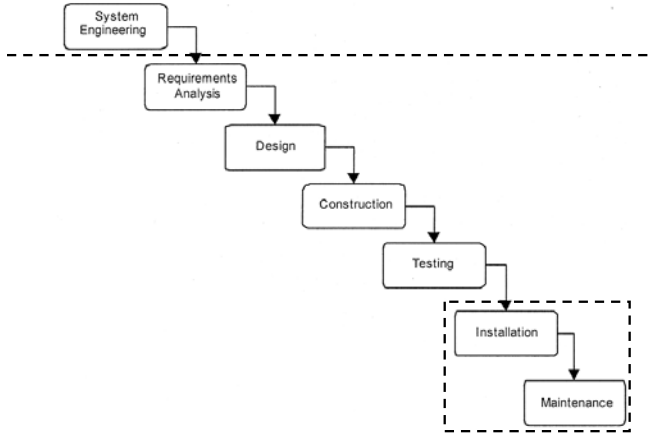
Project Life Cycles

- A distinction should be made between
  - Systems development which incorporates human, software and hardware elements
  - Software development which is primarily concerned with software systems
- Two important phases are
  - Strategic Information Systems Planning
  - Business Modelling

Strategic Information Systems Planning - Information systems work within the context of an organization and must satisfy its current requirements as well as providing a basis from which future needs can be addressed. In order to do this, strategic plans are developed for the organization as a whole and within their context a strategic view of information systems needs can be formed. For example, in the Agate case study a strategic decision may be made to target multinational companies for international advertising campaigns. This has consequences for campaign management and its supporting information systems.

Business modelling - In order to determine how an information system can support a particular business activity it is important to understand how the activity is performed and how it contributes to the objectives of the organization. Campaign management is an important business function for Agate and it should be modelled in order to determine how it is carried out, thus providing some of the parameters for subsequent information systems development.

Traditional Waterfall Life Cycle Model



The traditional life cycle for information systems development is also known as the waterfall life cycle model. So called because of the difficulty of returning to an earlier phase

The model shown here is one of several more or less equivalent alternatives. Typical deliverables are shown for each phase

Systems Engineering

- Software is always part of a larger system or business at work, work begins by establishing requirements for all system elements and then allocating some subset of these requirements to software. Essential for interfacing correctly with external components, e.g. databases, hardware
- Deliverables: High Level Architectural Specification

Requirement Analysis

- Analysis of requirements is now focused on software alone. Requirements for both system and the software are documented and reviewed with the customer
- Deliverables: Requirements Specification, Functional Specification, Acceptance Test Specifications

Design

- The design process translates requirements into a representation of the software. The multi-step process that focuses on four distinctive attributes of the program: Data structures, Software architecture, Procedural detail & Interface characteristics
- Deliverables: Software Architecture Specification, System Test Specification, Design Specification, Sub-system Test Specification, Unit Test Specification

Coding

- Design is translated to machine readable form. The software design is realized as a set of programs or program units
- Deliverables: Program Code

Testing

- Focuses on the logical internals of the software for the intent of finding errors. Debugging will follow the conduct of successful testing.
- Deliverables: Unit test Report, Sub-system Test Report, System Test Report, Acceptance Test Report, Completed System

Installation

- Install the system to client production environment
- Deliverables: Installed System

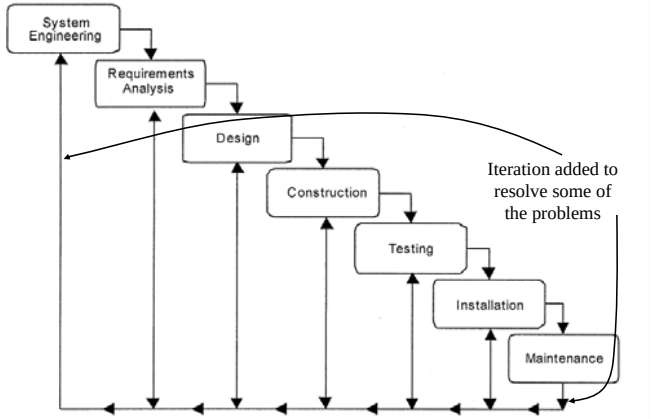
Maintenance

- After the installation and implementation of the software in the actual environment, errors/changes will invariably occur because software must accommodate changes in the real and external environment.
- Deliverables: Change requests, Change request report

Problems with Traditional Life Cycle

- Real projects rarely follow such a simple sequential life cycle
- Iterations are almost inevitable
- Time elapses between system engineering and the final installation
- The design is unresponsive to business changes during the project

Traditional Life Cycle with Iteration



Strengths of Traditional Life Cycle

- Tasks in phases may be assigned to specialized teams
- Project progress evaluated at the end of each phase
- Manage projects with high levels of risks

Prototyping

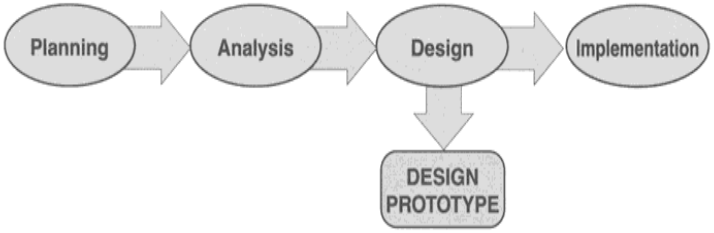
- The process of building an experimental system quickly and inexpensively for demonstration and evaluation so that users can better determine information requirements.
- By interacting with the prototype, users can get a better idea of their information requirements.

System Prototyping

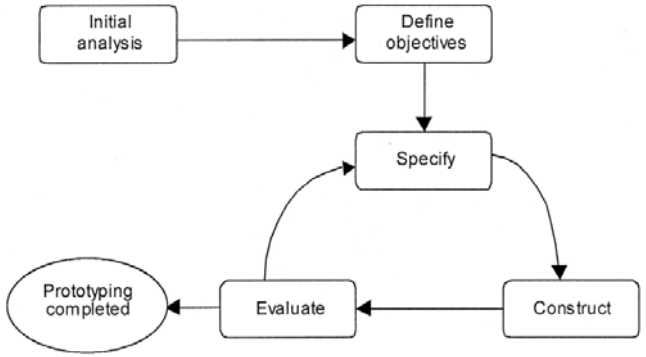
- Prototyping involves a repetitive sequence of analysis, design, modeling and testing.
- The end product of **System Prototyping** is a full-featured, working model of the information system, ready for implementation.

Design Prototyping

- Design Prototyping (also called Throwaway Prototyping) is used to verify user requirements.
- The end product of **Design Prototyping** is a user-approved design prototype that documents and benchmarks the features of the finished system.



Prototyping Life Cycle



U08182 © Peter Lo 2009 11

Perform an Initial Analysis - All software development activity utilizes valuable re-sources. Embarking upon a prototyping exercise without some initial analysis is likely to result in an ill-focused and unstructured activity producing poorly designed software.

Define Prototype Objectives - Prototyping should have clearly stated objectives. A proto-typing exercise may involve many iterations, each iteration resulting in some improv-ement to the prototype. This may make it difficult for the participants in a prototyping exercise to determine if there is sufficient value to continue the prototyping. However, with clearly defined objectives it should be possible to decide if they have been achieved.

Specify Prototype - Although the prototype is not intended for extended operation it is important that it embodies the requisite behaviour. It is almost certainly the case that the proto-type will be subject to modification and this will be easier if the software is built according to sound design principles.

Construct Prototype - Since it is important that prototype development is rapid, the use of a rapid development environment is appropriate. For example, if an interactive system is being prototyped, environments such as Delphi™ or Visual Basic® can be most effective.

Evaluate Prototype and Recommend Changes. The purpose of the prototype is to test or explore some aspect of the proposed system. The prototype should be evaluated with respect to the objectives identified at the beginning of the exercise. If the objectives have not been met then the evaluation should specify modifications to the prototype so that it may achieve its objectives. The last three stages are repeated until the objectives of the prototyping exercise are achieved.

Advantages of Prototyping

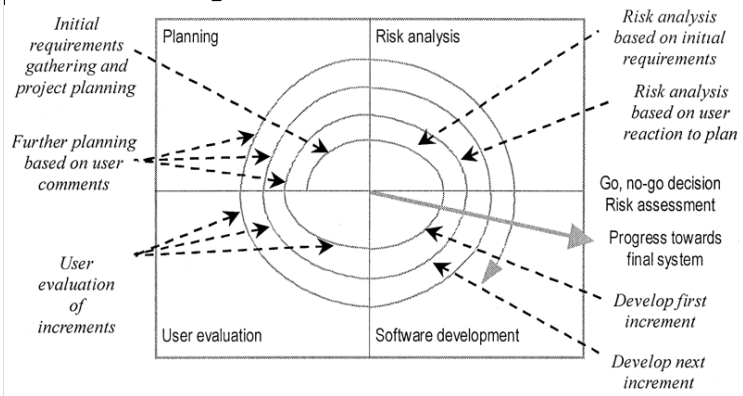
- Early demonstrations of system functionality help identify any misunderstandings between developer and client
- Client requirements that have been missed are identified
- Difficulties in the interface can be identified
- The feasibility and usefulness of the system can be tested, even though, by its very nature, the prototype is incomplete

U08182 © Peter Lo 2009 12

Problems in Prototyping

- The client may perceive the prototype as part of the final system
- The prototype may divert attention from functional to solely interface issues
- Prototyping requires significant user involvement
- Managing the prototyping life cycle requires careful decision making

Spiral Model & Incremental Development



Boehm's (1988) spiral model can be viewed as supporting incremental delivery. However, Gilb (1988) argues that the spiral model does not fully support his view of incremental development, as there are aspects of systems development that it does not emphasize or include. These are as follows:

- The production of high-value to low-cost increments;
- The delivery of usable increments of 1% to 5% of total project budget;
- A limit to the duration of each cycle (e.g. one month);
- A measure of productivity in terms of delivered functionality or quality improvements;
- An open-ended architecture that is a basis for further evolutionary development.

Activities of Spiral Model

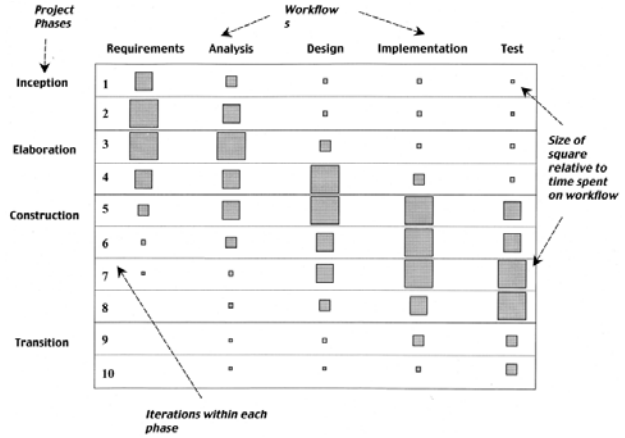
- Each spiral consist of four main activities:
  - **Planning** – Setting project objectives, defining alternatives
  - **Risk Analysis** – Analysis of alternatives and the identification and solution of risks.
  - **Engineering** – Equivalent to the build phase of the SDLC with coding and testing
  - **Customer Evaluation** – Testing of the product by customers

Software is developed in a series of incremental releases.

•During early iterations, the incremental release might be a paper model or prototype.

•During later iterations, increasingly more complete versions of the engineered systems are produced.

Unified Software Development Process



•Captures many elements of best practice

•Main phases

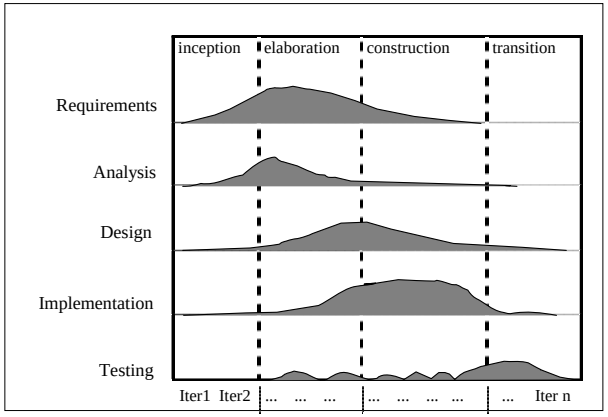
Inception is concerned with determining the scope and purpose of the project

Elaboration focuses requirements capture and determining the structure of the system

Construction's main aim is to build the software system

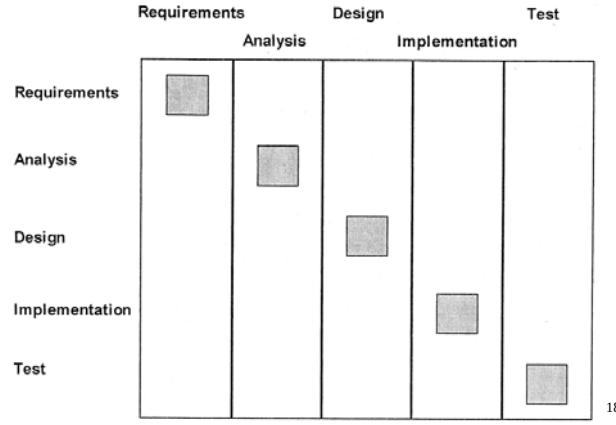
Transition deals with product installation and rollout

Distribution of Workflows over Phases



U08182 © Peter Lo 2009 17

Phases and Activities in a Simplified Waterfall Process



18

Difference from Waterfall Life Cycle

- In a waterfall life cycle project the phases and the workflows are linked together
- In the Requirements phase, only Requirements workflow activities are carried out
- All Requirements activity should be completed before work starts on Analysis
- In an iterative life cycle project it is recognised that some Requirements work will be happening alongside Analysis work

Computer-Aided Systems Engineering (CASE)

- Computer-aided systems engineering is a technique that uses powerful programs, called CASE tools, to help systems analysts develop and maintain information systems

Typically Features of CASE

•CASE tools
•Checks for syntactic correctness
•Repository support
•Checks for consistency and completeness
•Navigation to linked diagrams
•Layering
•Requirements tracing
•Report generation
•System simulation
•Performance analysis
•Code generation