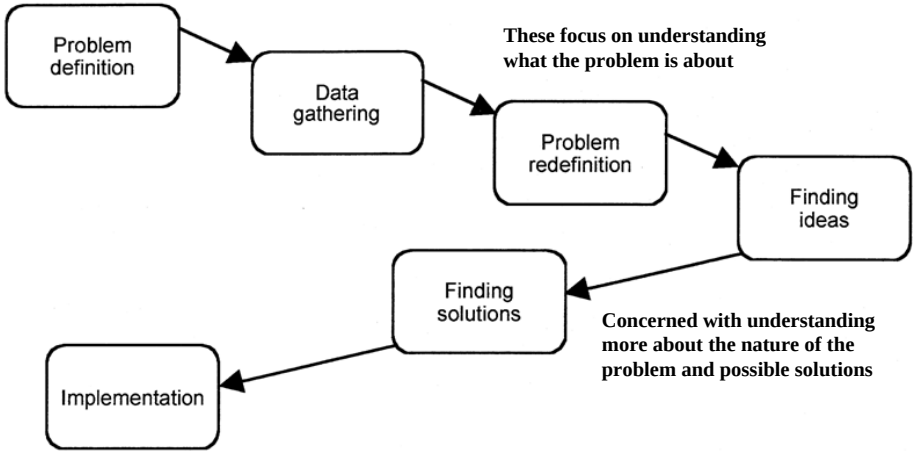
Avoiding the Problems

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.

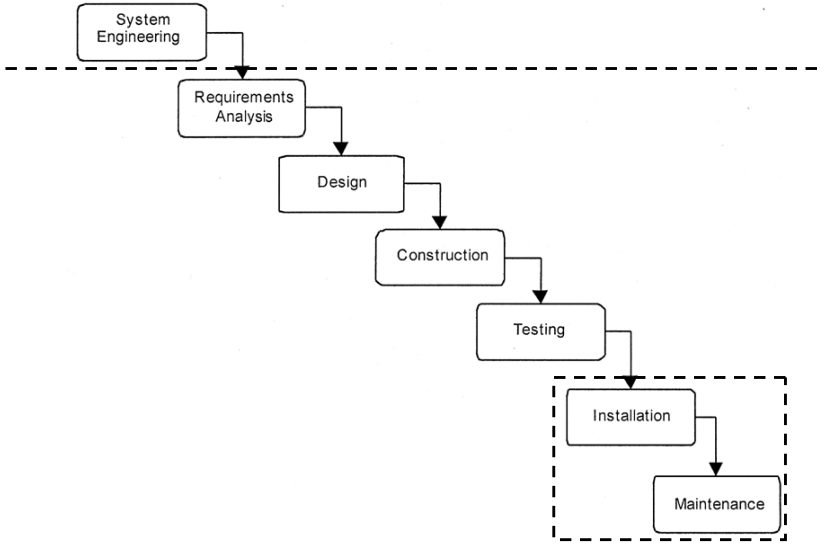
Business Modeling

- 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 modeled in order to determine how it is carried out, thus providing some of the parameters for subsequent information systems development.

Waterfall Life Cycle

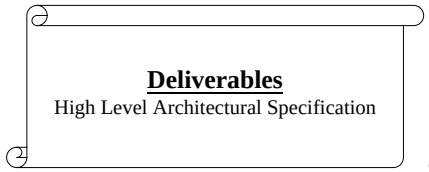
- The traditional life cycle for information systems development is also known as the waterfall life cycle model
  - So called because of the difficulty of returning to an earlier phase
- The model shown here is one of several more or less equivalent alternatives
  - Typical deliverables are shown for each phase

Traditional Waterfall Life Cycle Model



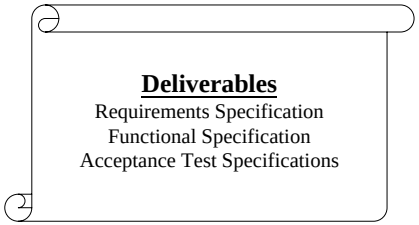
Systems Engineering

- 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



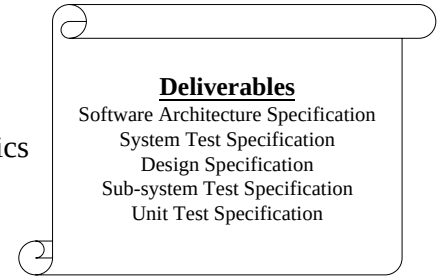
Requirement Analysis

- Analysis of requirements is now focused on software alone.
- Requirements for both system and the software are documented and reviewed with the customer.



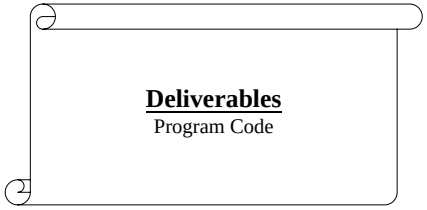
Design

- 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



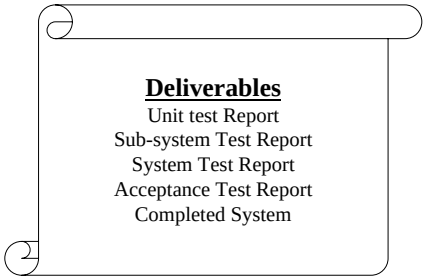
Coding

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



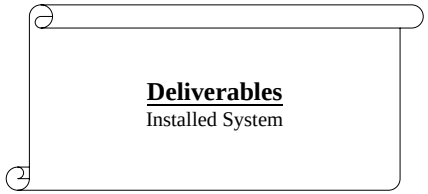
Testing

- Focuses on the logical internals of the software for the intent of finding errors.
- Debugging will follow the conduct of successful testing.



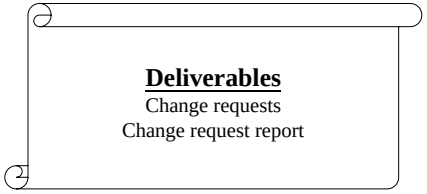
Install

- Install the system to client production environment



Maintenance

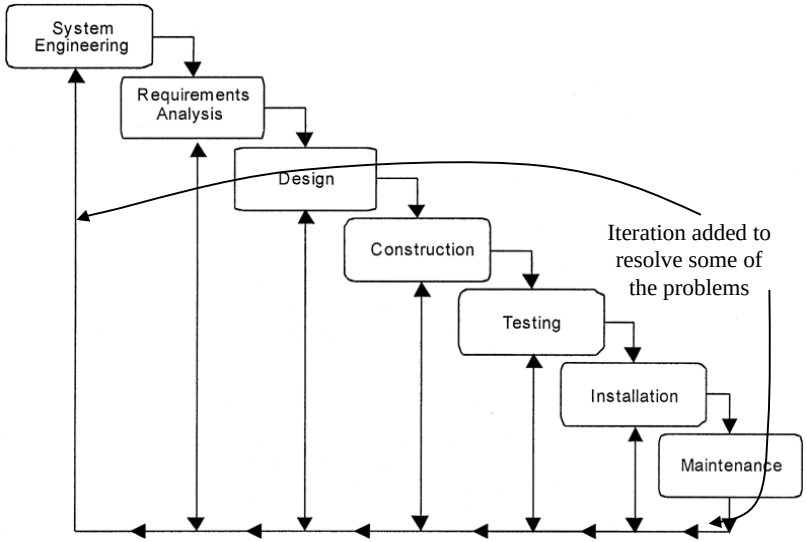
- After the installation and implementation of the software in the actual environment, errors/changes will invariably occur because software must accommodate changes in the real and external environment.



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

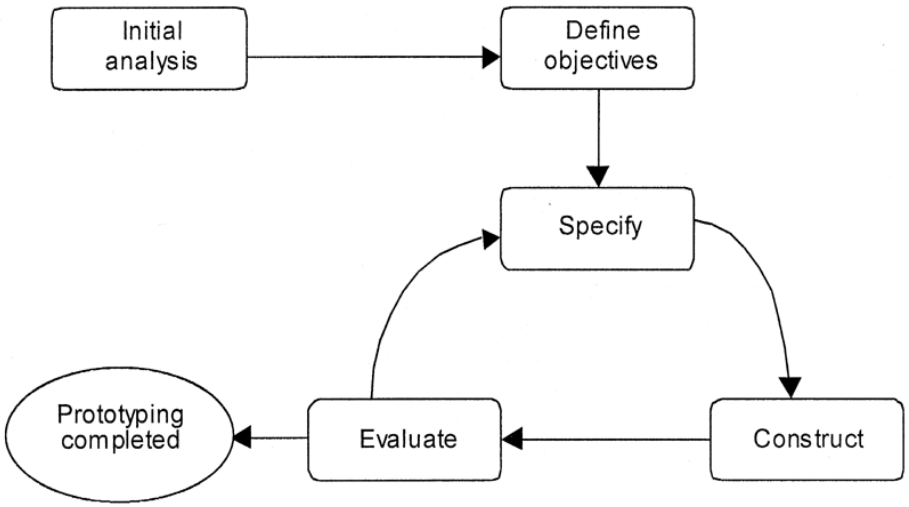
Class Exercise

- What are the advantages and the disadvantages of the traditional waterfall life cycle?

Prototyping

- A prototype is an early, rapidly constructed working version of the proposed information system.
- **Design Prototyping** or **Throwaway Prototyping** is used to verify user requirements.

Prototyping Life Cycle



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 improvement 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 behavior.
- 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.

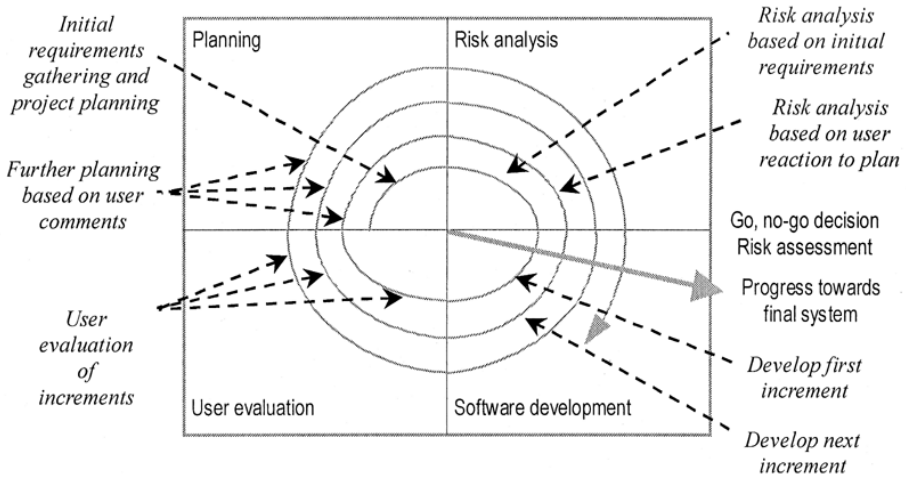
Advantages of Prototyping

- 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

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



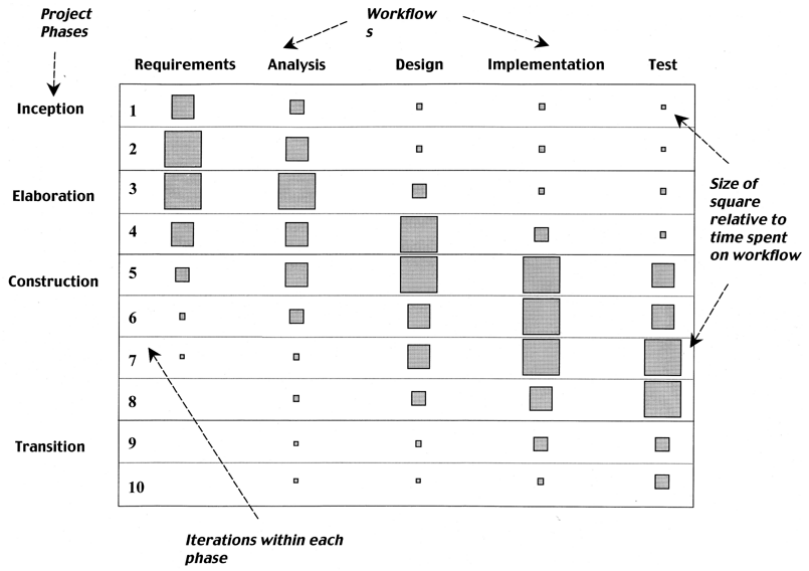
Spiral Model

- Boehm's (1988) spiral model can be viewed as supporting incremental delivery.
- 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.

Gilb's Argument against Spiral Model

- 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.

Unified Software Development Process



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

Class Exercise

- How does prototyping differ from incremental development?

User Involvement

- Users can be involved in various ways
  - As part of the development team
  - Via a consultative approach
  - In fact gathering

Class Exercise

- What are the different ways of involving users in the system development activity? What are potential problems with each of these?

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

Class Exercise

- How do “Syntactic”, “Consistency” and “Completeness” differs from each other?

Class Exercise

- What does requirements traceability mean?

Class Exercise

- Why is it not enough for a diagram to be syntactically correct, consistent and complete?

Class Exercise

- What is the purpose of a repository?