Avoiding the Problems

Based on Chapter 3 of Bennett, McRobb and Farmer:

Object Oriented Systems Analysis and Design Using UML, (3rd Edition), McGraw Hill, 2005.

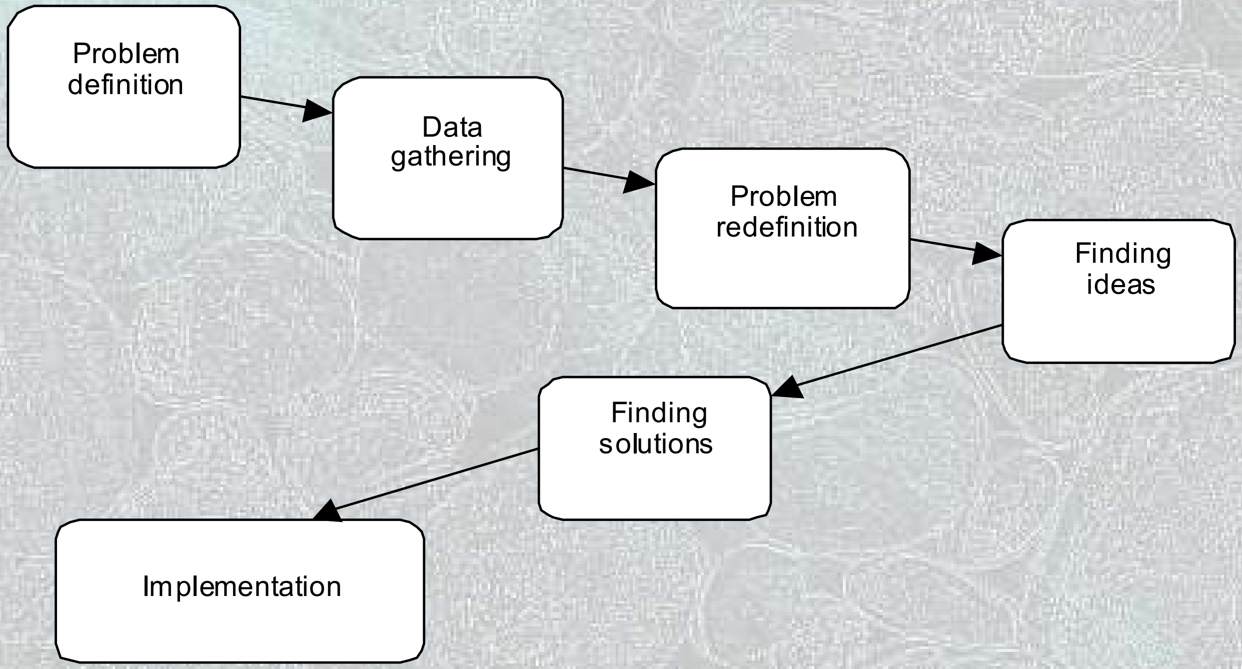
In This Lecture You Will Learn:

- 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.

Problem Solving Model

- Main phases are
  - Data gathering
  - Problem redefinition
    - These focus on understanding what the problem is about
  - Finding ideas
    - Concerned with understanding more about the nature of the problem and possible solutions
  - Finding solutions
  - Implementation

Problem Solving Model



General problem solving model (adapted from Hicks, 1991).

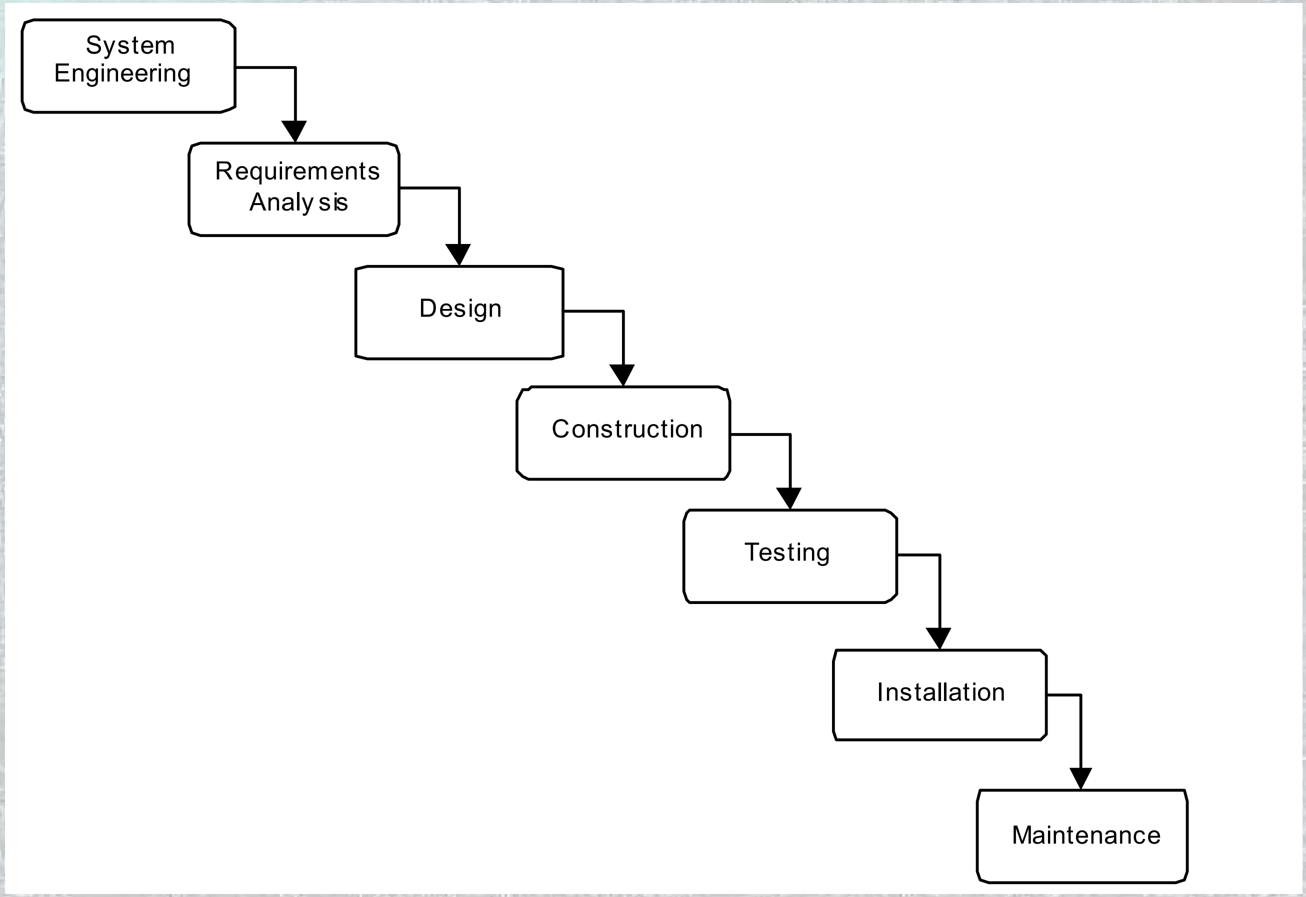
Project Life Cycles

- 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

Waterfall Life Cycle

- The traditional life cycle (TLC) 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 Life Cycle



TLC Deliverables

- Systems Engineering
  - High level architectural specification
- Requirements Analysis
  - Requirements specification
  - Functional specification
  - Acceptance test specifications

Life cycle deliverables (adapted from Sommerville, 1992).

TLC Deliverables

- Design
  - Software architecture specification
  - System test specification
  - Design specification
  - Sub-system test specification
  - Unit test specification

Life cycle deliverables (adapted from Sommerville, 1992).

TLC Deliverables

- Construction
  - Program code
- Testing
  - Unit test report
  - Sub-system test report
  - System test report
  - Acceptance test report
  - Completed system

Life cycle deliverables (adapted from Sommerville, 1992).

TLC Deliverables

- Installation
  - Installed system
- Maintenance
  - Change requests
  - Change request report

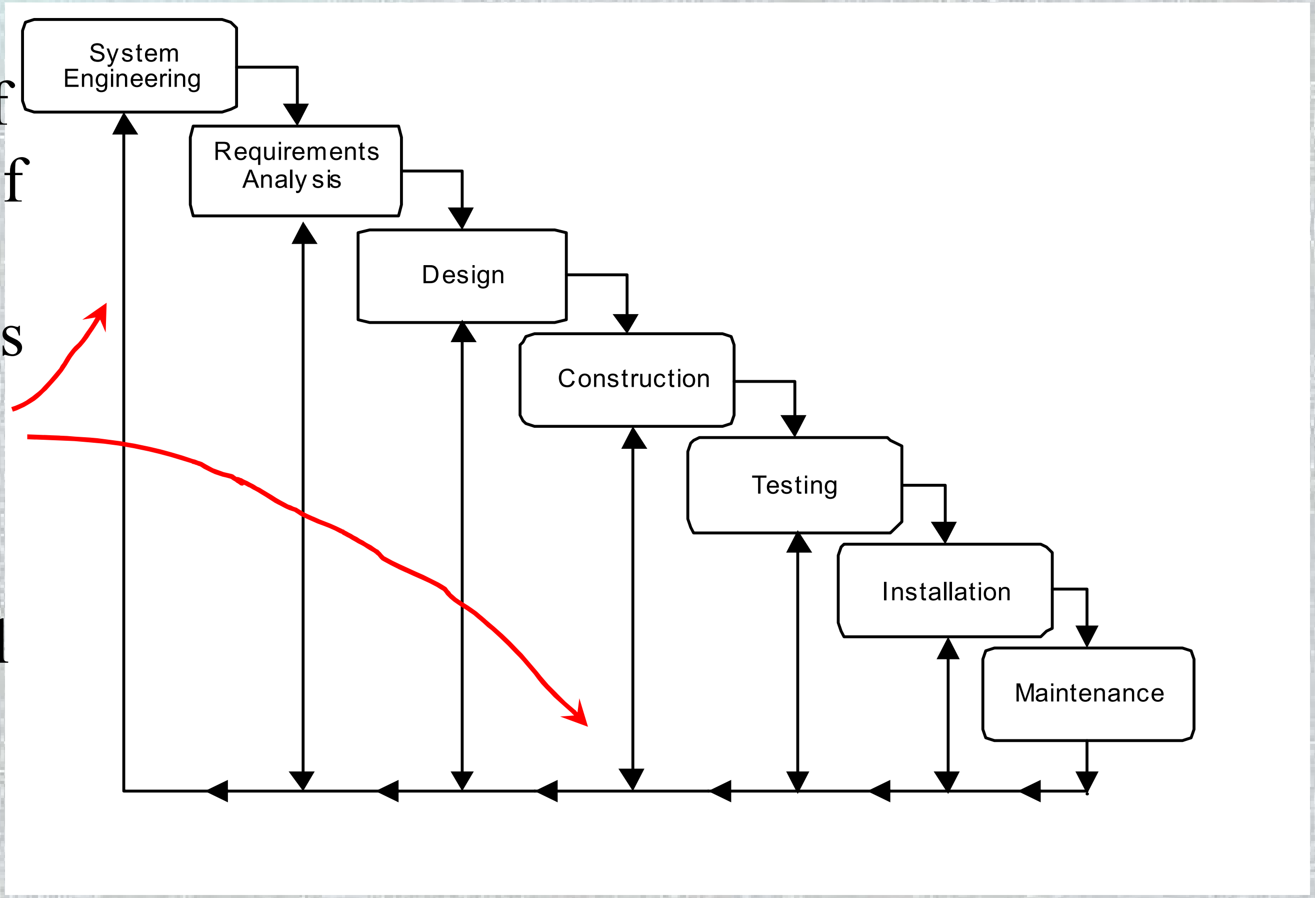
Life cycle deliverables (adapted from Sommerville, 1992).

Problems with TLC

- Real projects rarely follow such a simple sequential life cycle
- Lapsed time between systems engineering and the final installation is long
- Iterations are almost inevitable in real projects but are expensive & problematic with the TLC
- Unresponsive to changes during project as iteration is difficult

TLC with Iteration

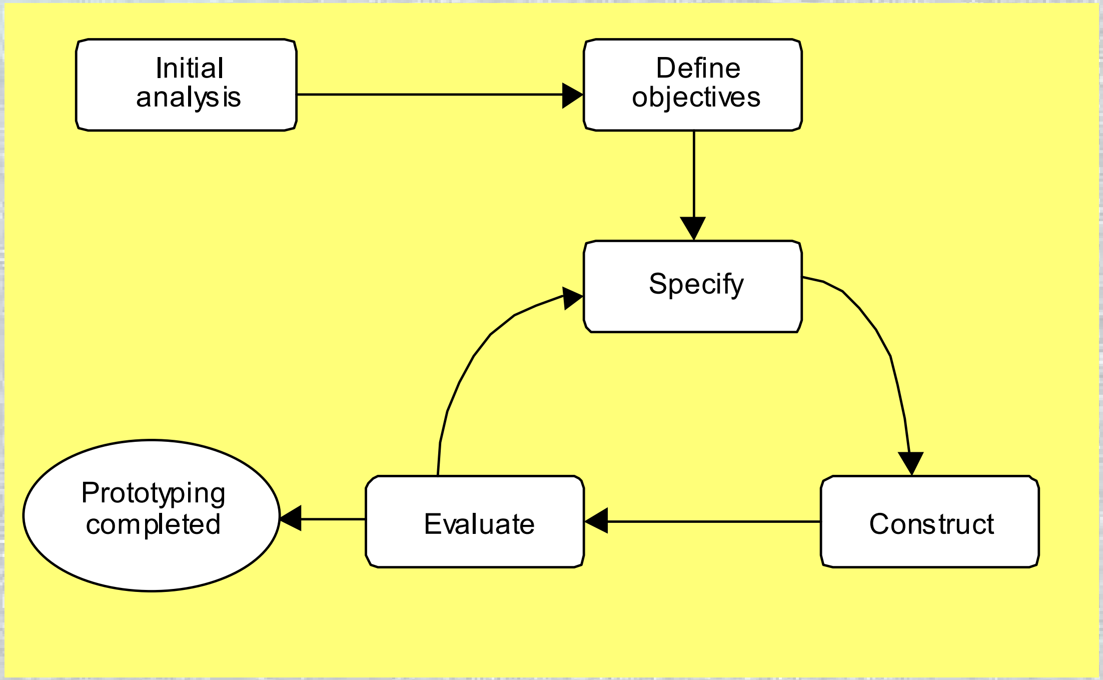
The cost of this form of iteration increases as the project progresses making it impractical and **not** effective



Strengths of TLC

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

Prototyping Life Cycle



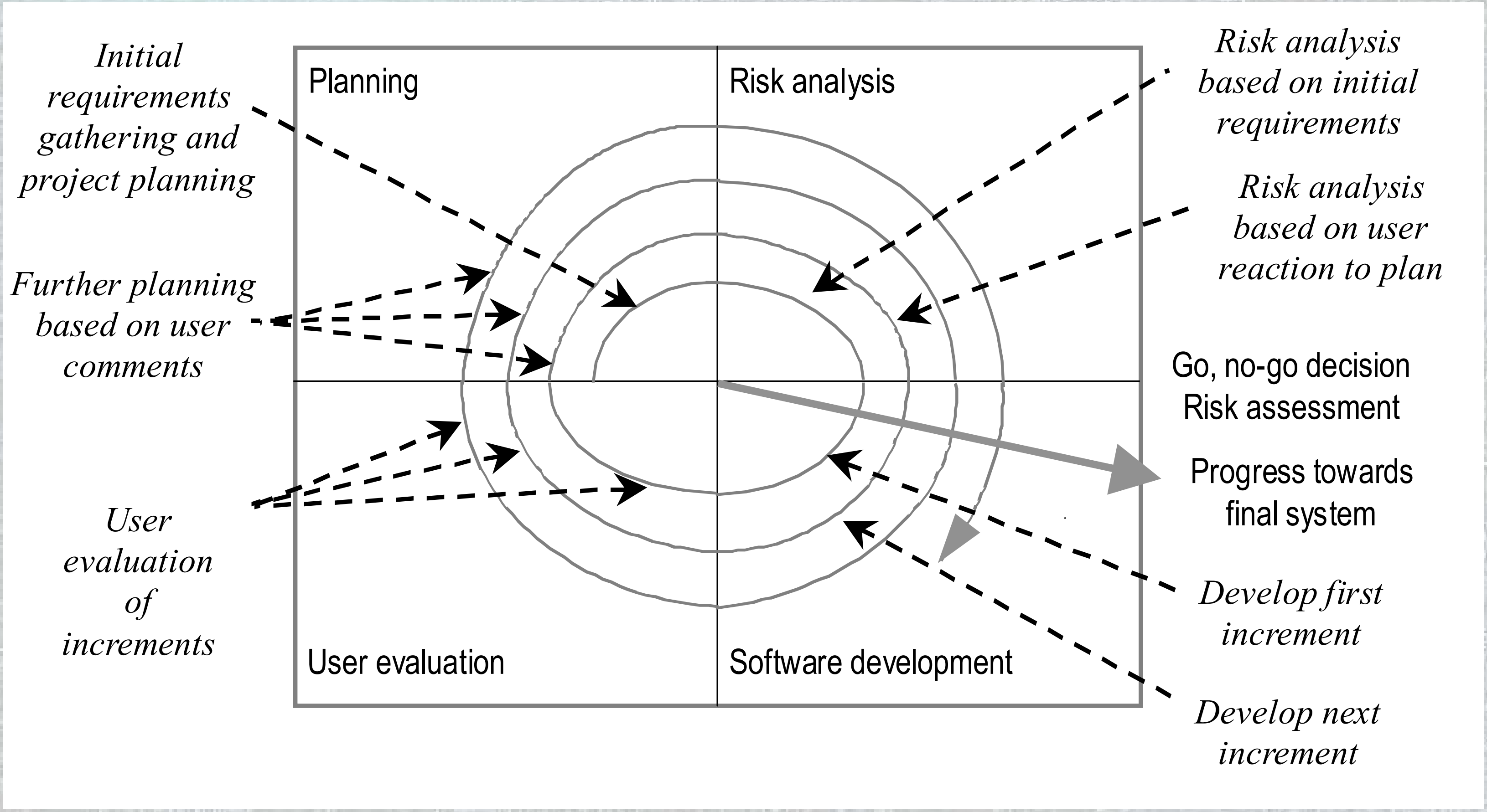
Prototyping – Advantages:

- 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

Prototyping – Problems:

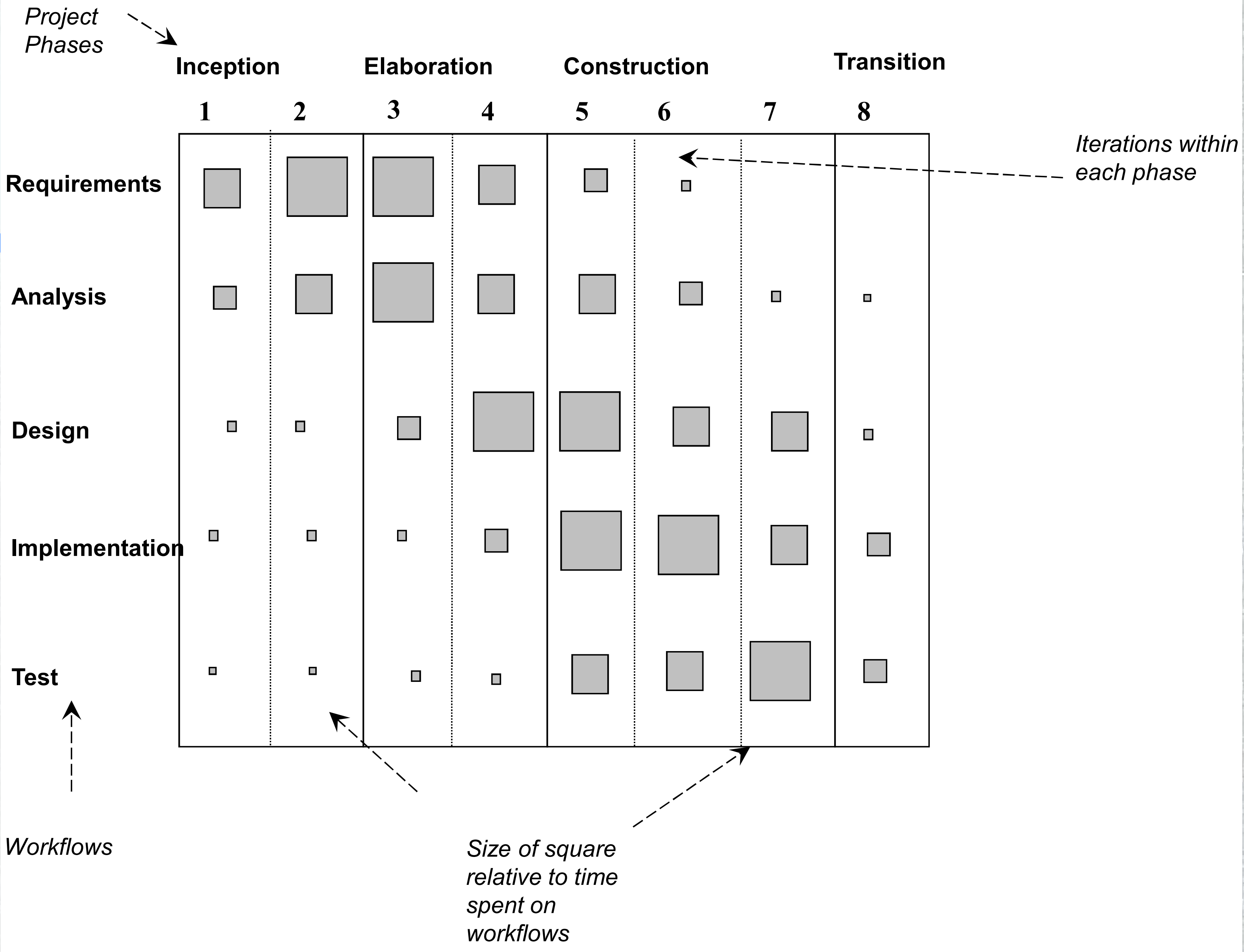
- The client may perceive the prototype as part of the final system
- The prototype may divert attention from functional to solely interface issues
- Prototyping requires significant user involvement
- Managing the prototyping life cycle requires careful decision making

Spiral Model & Incremental Development


Initial requirements gathering and project planning
Further planning based on user comments
User evaluation of increments
Planning
Risk analysis
User evaluation
Software development
Risk analysis based on initial requirements
Risk analysis based on user reaction to plan
Go, no-go decision
Risk assessment
Progress towards final system
Develop first increment
Develop next increment

Unified Software Development Process

- Captures many elements of best practice
- The phases are:
  - *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.


Project Phases
Inception
Elaboration
Construction
Transition
1
2
3
4
5
6
7
8
Iterations within each phase
Requirements
Analysis
Design
Implementation
Test
Workflows
Size of square relative to time spent on workflows

User Involvement

- User’s can be involved at various levels
  - As part of the development team (DSDM)
  - Via a consultative approach
  - In fact gathering

Agile Approaches

- Iterative lightweight approach
- Accepts that user requirements will change during development
- XP and DSDM are considered agile
- Non-agile approaches can be viewed as plan-based

Agile Approaches

Manifesto for Agile Software Development

We are uncovering better ways of developing software
by doing and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

The Manifesto for Agile Software Development

Computer Aided Software Engineering

- CASE tools typically provide a range of features including:
  - checks for syntactic correctness;
  - repository support;
  - checks for consistency and completeness;
  - navigation to linked diagrams;

Computer Aided Software Engineering

- Features of CASE tools continued
  - layering;
  - traceability;
  - report generation;
  - system simulation;
  - performance analysis;
  - code generation.

Summary

In this lecture you have learned about:

- 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.

References

- Hicks (1991)
- Sommerville (1992, 2004) and Pressman (2004)
- Jacobson, Booch and Rumbaugh (1999)
- Chapters 5 and 21 of Bennett, McRobb and Farmer include more detail about the Unified Process

(For full bibliographic details, see Bennett, McRobb and Farmer)