CS4125
SYSTEMS ANALYSIS
SPRING SEMESTER 2010-2011

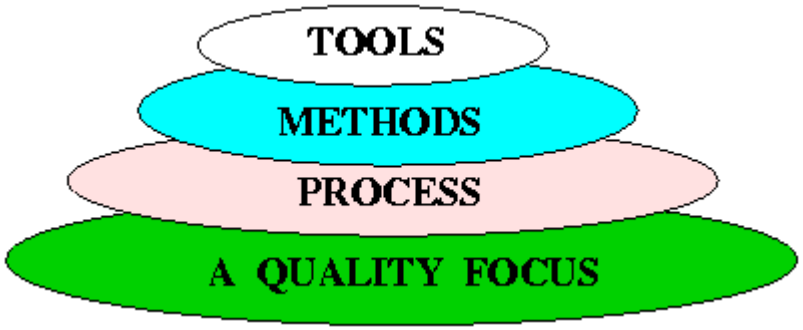
J.J. Collins
Dept of CSIS
University of Limerick

Lecture T: W10L1

Software Lifecycles

1. Software Process

- The terms "*software lifecycle*" and "*software process*" have similar definitions.
- The foundation for software engineering is the process layer.
- Process defines a framework for a set of key process areas (KPAs) that must be established for effective delivery of software engineering technology.



Adapted from Roger S. Pressman. Software Engineering: A Practicioner's Approach, 7th Edition. McGraw-Hill. 2009.

1. Software Process

- The KPAs form the basis for:
  - Management control of software projects.
  - Establish the context in which technical methods are applied.
  - Work products (models, documents, data, forms, reports, etc.) are produced.
  - Milestones established.
  - Quality is ensured.
  - Change is properly managed.

1. Software Process

- Umbrella (optional) activities in the process:
  - Software project tracking and control.
  - Formal technical reviews.
  - Software quality assurance.
  - Software Configuration management.
  - Document preparation and production.
  - Reusability management.
  - Measurement.
  - Risk Management.

2. Tower of Babel

- All software development projects follow 2 distinct processes:
  - The management process schedules work, plans deliveries, and monitors progress.
  - The development process creates working software from requirements.

Chessman and Daniels (2001). UML Components, Chapter 2: The Development Process. Addison-Wesley.

- Assuming that you have these processes written down in manuals, you would consult your management process manual if help was required with the setting of milestones.
- Likewise, you would refer to your development process manual if help was required with allocating operations to interfaces.
- The software process literature contains examples that include either or both of these processes.
  - Dynamic Systems Development Method (DSDM) is a management process,
  - Catalysis is a development process, and
  - The Rational Unified Process (RUP) covers both.

3. SEI's CMM

- The Software Engineering Institute (SEI) at Carnegie-Mellon have developed a comprehensive model that may be used by software houses to evaluate their process maturity.
- The SEI uses an assessment that results in a five point (level) grading scheme.
- The grading scheme determines compliance with the Capability Maturity Model (CMM)
- The CMM has defined 18 KPAs, and each level complies with an increase number of these KPAs.

3. SEI's CMM

SEI's Process Maturity Grading Scheme Level Description

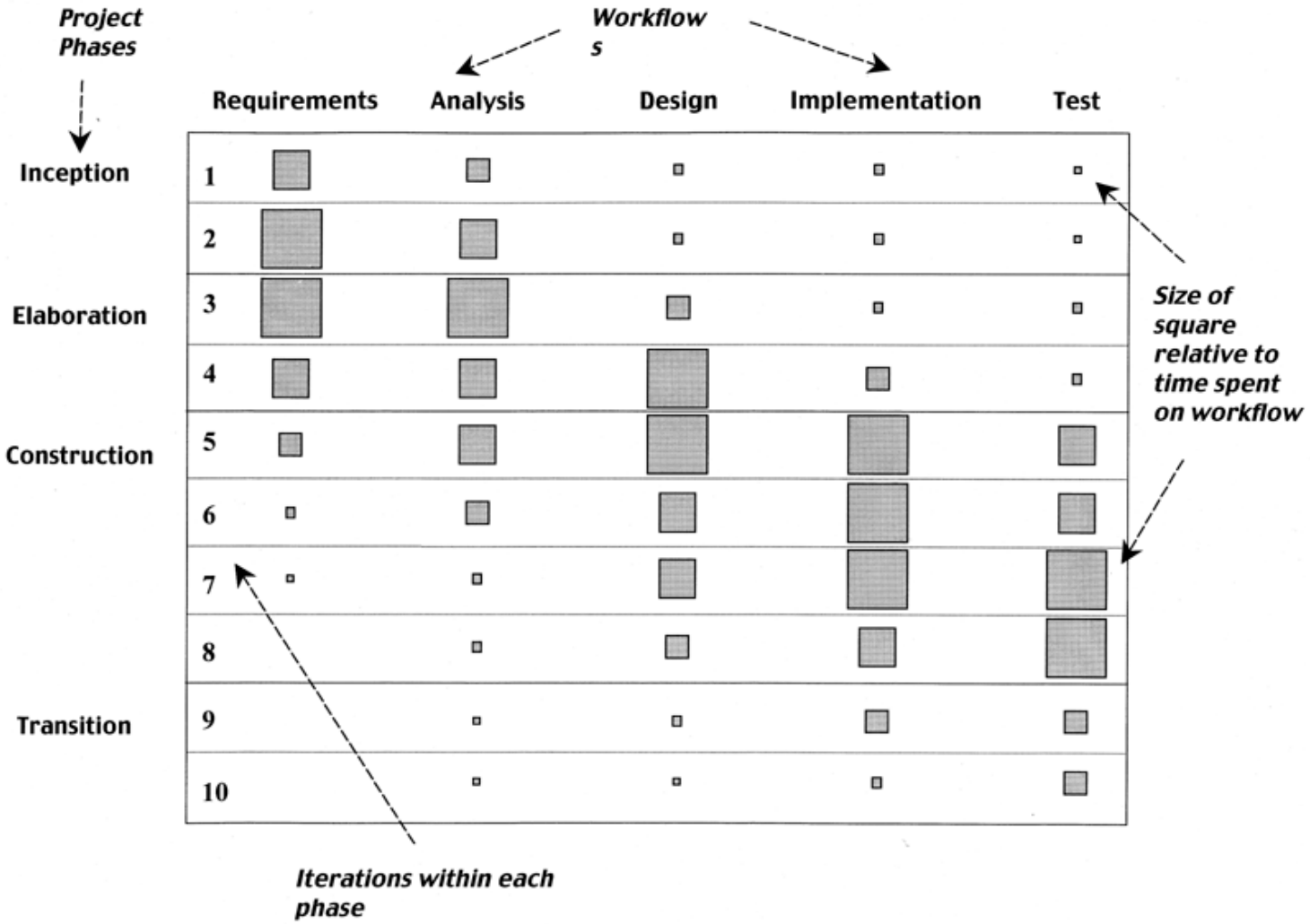
1. **Initial**: process is ad hoc and even chaotic.
2. **Repeatable**: basic project management processes are established to track costs, schedules, and functionality. Will successfully complete projects similar to previously successful ones.
3. **Defined**: process for both management and engineering (development) activities is documented, standardised, and integrated into organisation, used by all projects.
4. **Managed**: metrics captured at all stages of the process. Both the software process and product have a quantitative model that is controlled using metrics.
5. **Optimising**: continuous process improvement is enabled by quantitative feedback from the process and from testing innovative concepts and leading edge technologies.
    1. Many manufacturing companies are at this level, and have created predictive models of resource and process performance that determines product quality.

4. Rational Unified Process (RUP)

The three amigos.

- Grady Booch: Object Oriented Design (OOD). Set up Rational.
- James Rumbaugh: Object Modelling Technique (OMT). Joined Rational Rose in 1994.
- Ivar Jacobson: Object Oriented Software Engineering (OOSE). Rational takes over Objectory in 1995.
- Early focus on unified software development process – not doable because of lack of common syntax.
- Focus turned to developing a unified modelling language – UML.
- Then: Unified Software development Process (USDP).
- Rational Unified process (RUP) is a refinement of USDP
- Many other extensions
  - Agile Unified Process (AUP),

4. Rational Unified Process (RUP)



Unified Software Development Process (USDP) / Rational Unified Process (RUP)

4. Rational Unified Process (RUP)

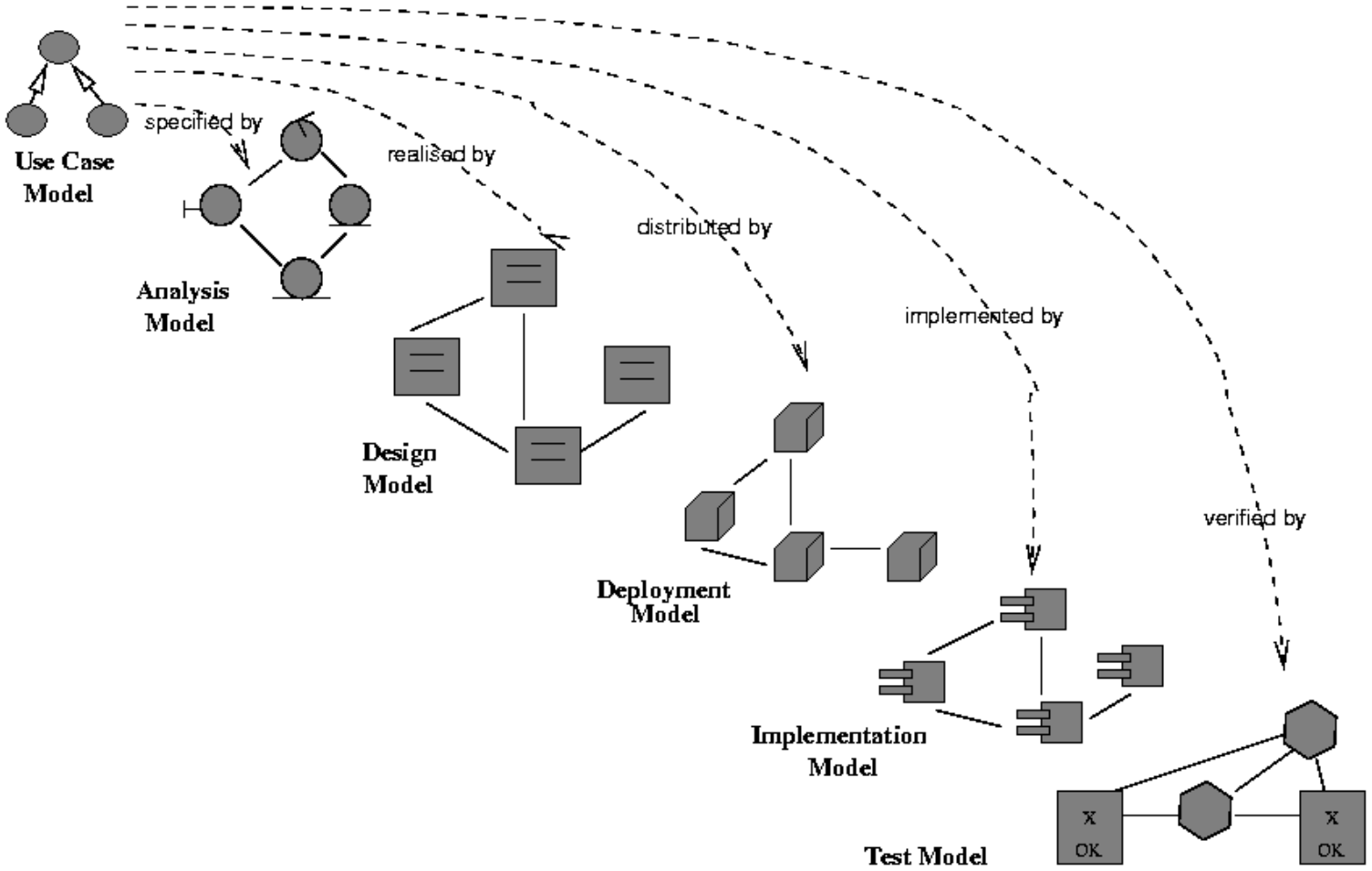
Key Principles.

- Manage requirements / Use case driven.
- Architecture centric and component-based.
- Use iteration and incremental development to control risk.

- Additional Principles in RUP
  - Visual Modelling
  - Continuously verify software quality.
  - Configuration and change management
  - The 4 Ps - people, project, product and process.

4. Rational Unified Process (RUP)

- **Representations of the Software Product.**



4. Rational Unified Process (RUP)

- Use case model.
- An analysis model:
  - Refine the use cases in more detail.
  - To make an initial allocation of the behaviour.
- A design model that defines:
  - Static structure of the system as sub-systems, classes and interfaces.
  - Use case realised as collaborations among the sub-systems, classes and interfaces.
- An implementation model which includes components and the mapping of the classes to components.
- A deployment model - mapping of components to nodes.
- A test model - describes test cases that are used to verify use cases.
- A representation of the system architecture.
- May also have a domain or business model.

4. Rational Unified Process (RUP)

Use Cases.

- Use cases capture functional requirements.
- Use case model represents traditional functional specification.
- What is the system supposed to do for each user.
- Unified process proceeds through a series of workflows that derive from the use cases.
- Use cases and their realisations developed in tandem with the system architecture.

4. Rational Unified Process (RUP)

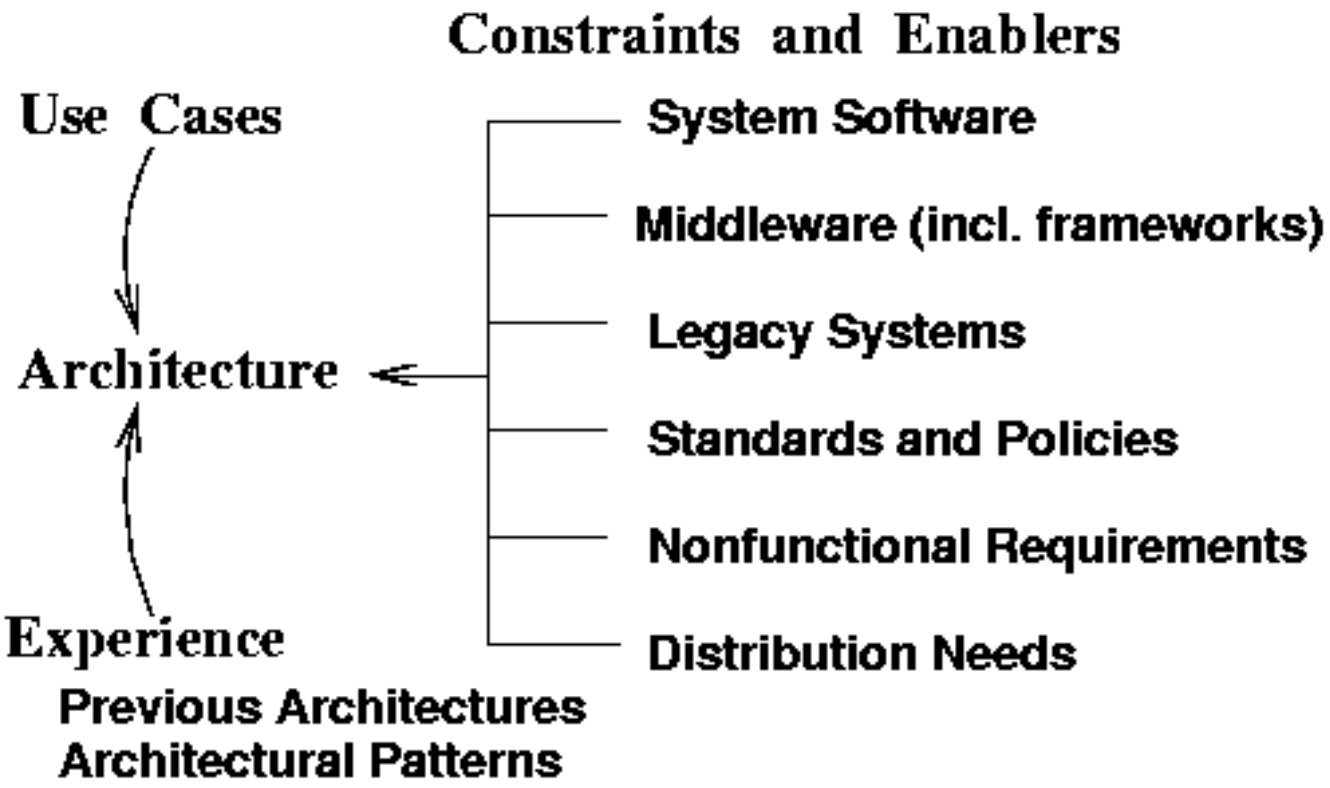
Architecture

- Every product has both function and form that is captured in its architecture.
- Software architecture encompasses decisions about:
  - The organisation of a software system.
  - The structural elements, their interfaces, and their behaviour.
  - The composition of the structural and behavioural elements into progressively larger subsystems.
  - The architectural style that guides the organisation.
  - Concerned with usability, functionality, performance, reuse, economic and technological constraints, and aesthetics.
- A software system must have an architecture, defines a common context.

4. Rational Unified Process (RUP)

- Use cases, when realised, must fit into the architecture.
- To create the architecture, the architect must work from an understanding of the key use cases - 5% / 10% of total.
- The architect must:
  - Create a rough outline of the architecture starting with focus on form that is not use case specific.
  - Next, work with a subset of the identified use cases that represent key functions of the system.
  - As the use cases mature, more of the system architecture is refined until it is deemed stable.

4. Rational Unified Process (RUP)



Role of Use Cases and Other Factors in the Evolution of Architecture

4. Rational Unified Process (RUP)

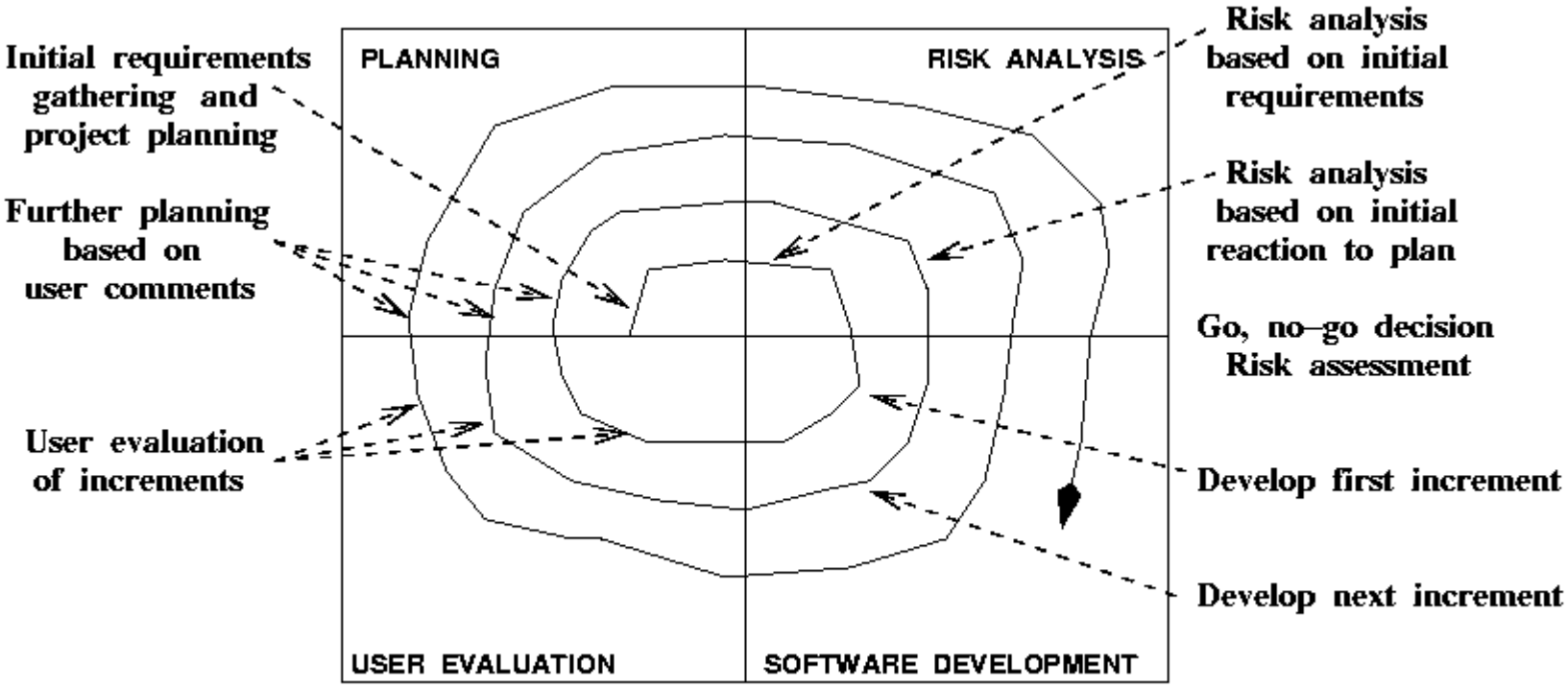
- The architect is not only guided by the significant use caes, but by other factors such as:
  - Platform - computer architecture, OS, DBMS,
  - Middleware.
  - Deployment considerations
  - Legacy systems
  - The reusable building blocks available - frameworks for GUI
  - Non-functional requirements - performance, security policy.
  - Standards, i.e. using OMG's Interface Definition Language (IDL) to specify interfaces.

4. Rational Unified Process (RUP)

Iterative and Incremental.

- Iterations refer to steps in the workflow, and increments, to growth in the product.
- At each iteration developers deal with:
  - A group of use cases that extend the usability of
  - the product developed so far.
  - The most important risks.
- Benefits of iteration:
  - Controlled iteration reduces the cost risk to expenditure on a single increment.
  - Reduces risk of failure.
  - Speeds up process by the use of short range well defined goals.
  - User requirements cannot be specified up front. Need to be refined.

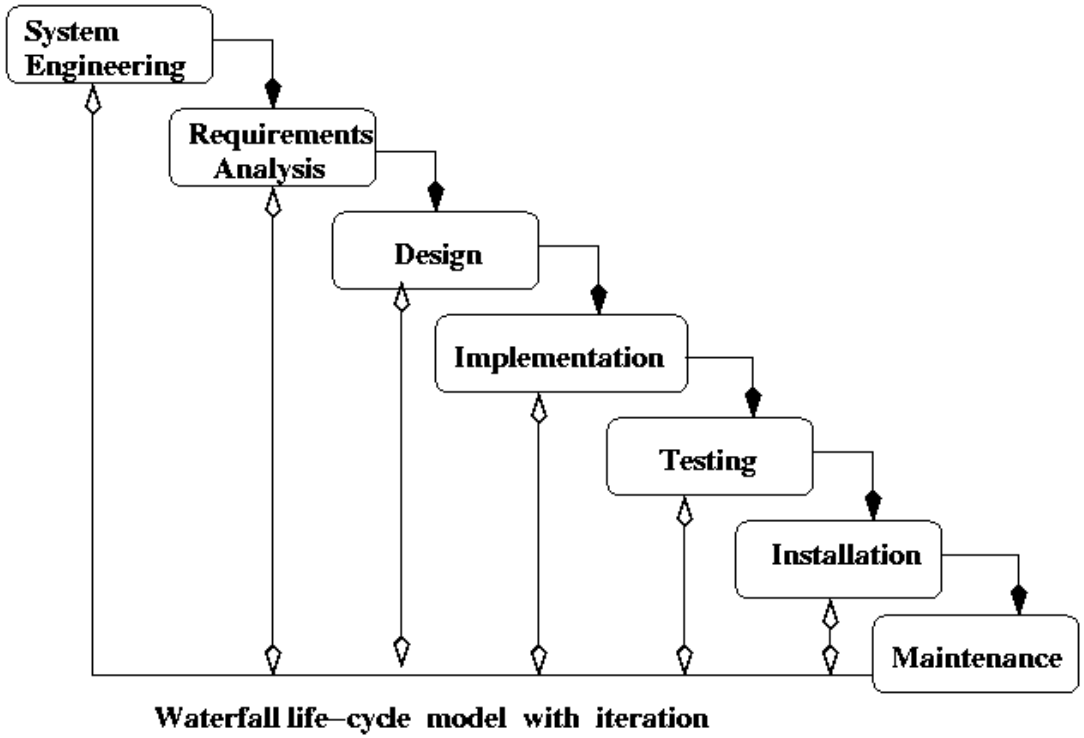
5. Software Lifecycles: Spiral



Spiral model (Boehm, 1988)

6. Waterfall with Feedback

A software lifecycle depicts the phases that software passes through from inception to demise.



Waterfall life–cycle model with iteration

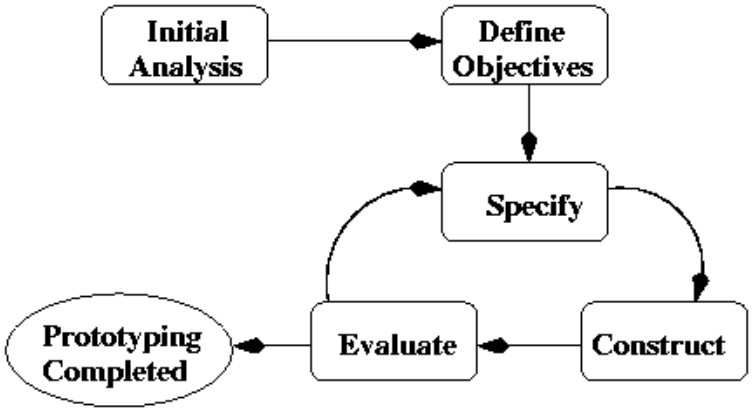
Q: benefits and liabilities ???

7. Prototyping

- A prototype is a partially developed system that is built quickly to explore requirements, and is not intended as the final working system. e.g. automotive industry, aircraft etc. use prototypes.
- Many software life cycle models involve iterations to refine requirements.
- In waterfall model, user sees system for first time upon final delivery of product.
- Prototyping approach tries to resolve potential misunderstandings and ambiguities in the requirements that the user may have.
- A prototype may have the following characteristics:
  - Lack full functionality.
  - Limited data processing.
  - Exhibit poor performance.
  - Limited quality assurance.
- Commonly uses rapid development tools e.g. Interface and report generators with links to office docs such as spreadsheets, and/or Visual Basic.

7. Prototyping

- Various objectives. Evaluate:
  - User requirements.
  - Human computer interface (HCI) - how is data captured from and presented to the user.
  - Graphical user interfaces (GUI).
  - Can implementation platform support processing requirements.
  - Language support, database management system, communications.
- Managing prototyping lifecycle and number of iterations - difficult.
- What are advantages and disadvantages of prototyping ????



Prototyping life–cycle

8. Agile Manifesto and Extreme Programming

- Principles of the Agile Manifesto
  - Customer involvement
  - Incremental delivery
  - People not process
  - Embrace change
  - Maintain simplicity
- Extreme Programming (XP)
  - On-site customer
  - Incremental planning, continuous testing and integration, small releases, simple design
  - Test Driven Development (TDD)
  - Refactoring
  - Pair programming
  - Collective Ownership and sustained pace.
  - Mentoring.

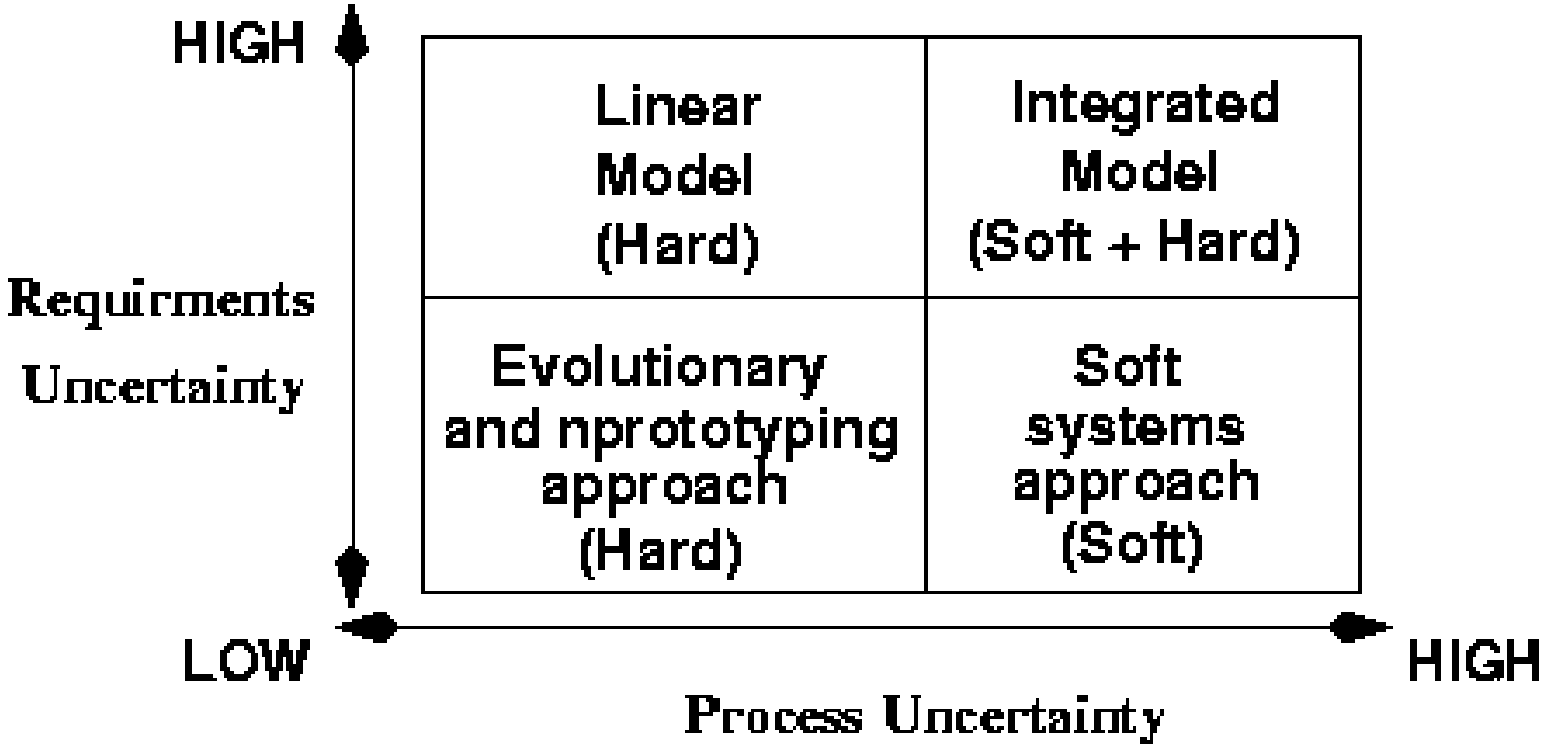
9. Development Philosophies

Hard systems view	Soft systems view
The activity of IS development is all about building a technical system that is made only of software and hardware.	An IS also comprises the social context in which the technical system (software and hardware) will be used.
Human factors are chiefly important from the perspective of the software's usability and acceptability. Politics and group behaviour are only an issue for project managers.	A new IS impacts on interpersonal communication, social organization, working practices and much more, so human and social factors are paramount.
Organizations exist only to meet rational objectives, through the application of rational principles of business management. It is possible to be both rational and objective about the requirements for a new system.	Organizations are made up of individuals with distinct views and motivations, so any picture of requirements is subjective. It is not always possible even to reach a consensus. In practice, this means that the powerful decide, not the wise.
When requirements are uncertain or unclear, it is up to management to decide. Setting objectives is a principal role of management, and others should follow their lead.	If management has not accommodated the full range of views in the organization, encouraging managers to decide on the requirements may be completely counter-productive.

9. Development Philosophies

- Both approaches can compliment each other, with a focus on soft methodologies at the start of the lifecycle which then changes to hard methodologies for design and implementation.
- The contingency framework developed by Flynn (1998) which helps to guide selection based on requirements uncertainty and process uncertainty.
- Process uncertainty refers to the degree of doubt about how to build the proposed system.

9. Development Philosophies



Reading

- Sections 3 & 4 in chapter 3 and chapter 21 from Bennett et al.
- OR
- Chapters19 and 20 in Stevens and Pooley