COM 6030 Software Analysis and Design

Lecture 2- Software Process Models and Project Management

Module homepage http://vista.shef.ac.uk

Introduction

1. Software engineering: definition, products, processes
2. Software engineering – models, management

Outline

- Software processes
- Software engineering models
- Software process models: waterfall, evolutionary, formal development, reuse/component development
- Management spectrum
- Software teams
- Managing people, project, product and processes

Reading: Sommerville chapter 3; Pressman chapter 2

(II) Software processes

Software process – component of SE activity – is a set of activities and associated results which lead to a software product.

- Complex activity requiring an abstract representation – model/process paradigm
- Software process reflects the problem solving stages:
  - Status quo – current state of the problem
  - Problem definition – problem identification
  - Technical development – technology to solve the problem
  - Solution integration – deliver the solution

(III) SE models

Models are used to abstractly represent various entities, transformations, processes, activities.

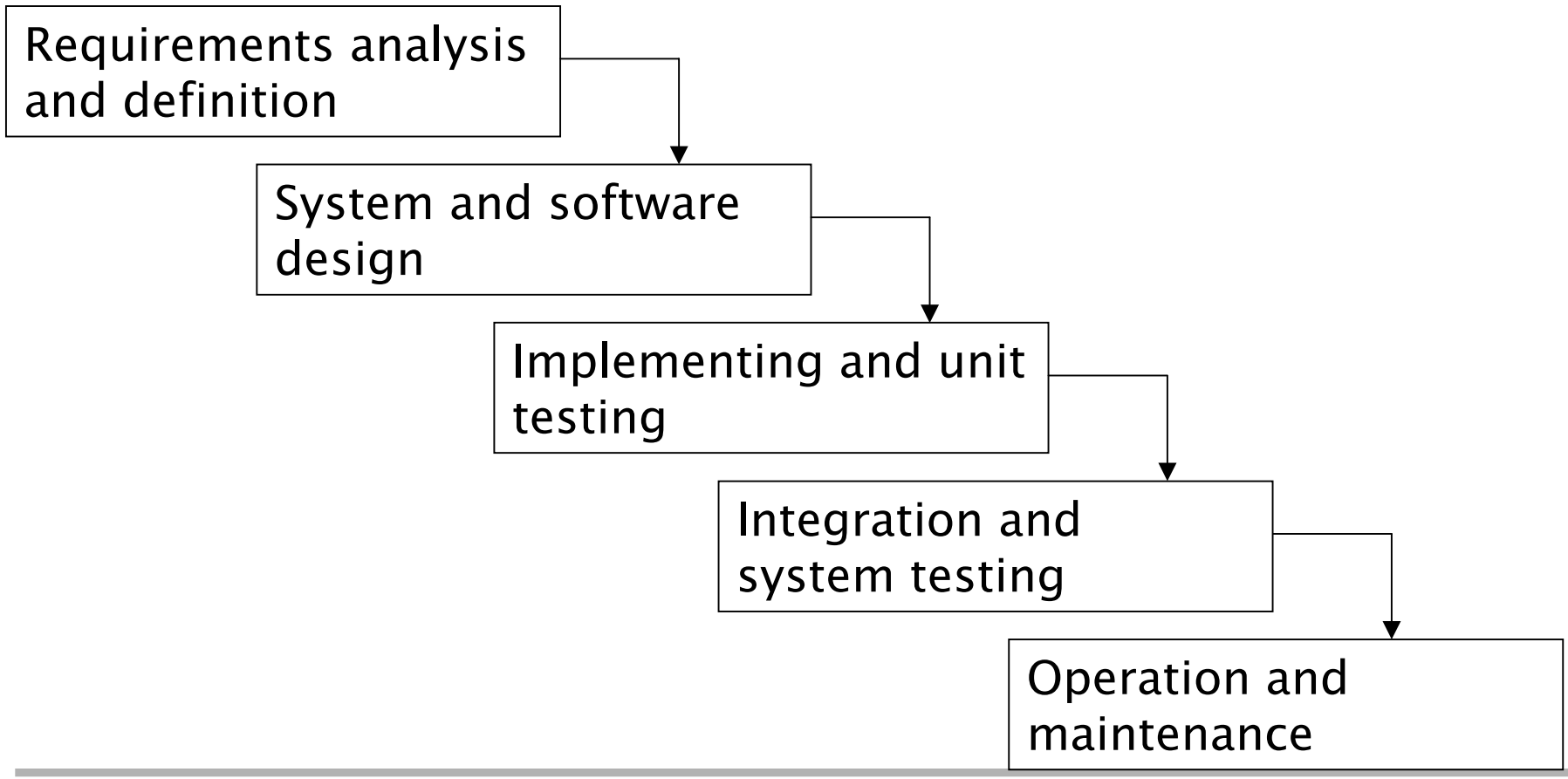
- SE process models:
  - Architectural perspective
  - Flow of activities
  - Linear vs iterative
- Software processes depend on project type, complexity aspects, technology, budget, people knowledge
- Models of various activities (specification, design), entities (data, transformations), processes (structural, dynamic)

SE process models

- SE process models:
  - Waterfall
  - Evolutionary
  - Formal systems development
  - Reuse/Component-based
  - Iterative
    - Incremental
    - Spiral
  - Others

Waterfall model

Waterfall model maps linearly fundamental activities



Waterfall: +'s and –'s

Advantages:

- Help project management, documentation
- Complex systems are well-defined and structured
- Reflects engineering practice
- Encourages a discipline of modelling

Limitations

- Often stages overlap
- Rather inflexible
- Imposes early commitments to rigid requirements set
- Impossible to deal with changes

Evolutionary development

Initial implementation refined into final system

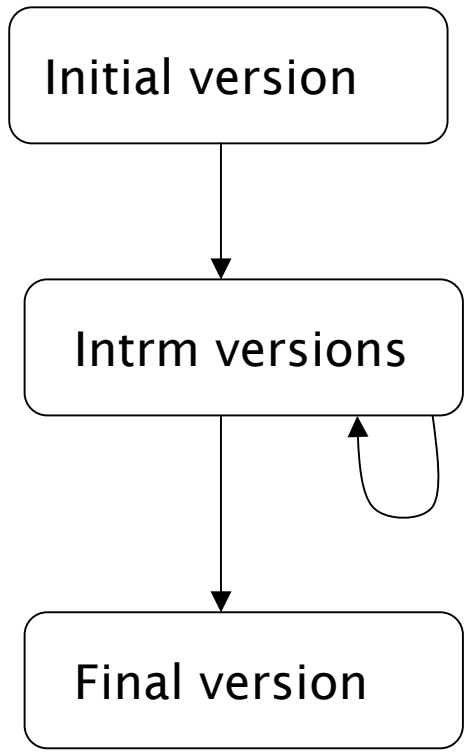
- Exploratory development: the system evolves through stages into a final software product
- Throw-away prototyping

Advantages:

- Flexible and suitable for small systems
- Helps in early stages or when little information known about a system

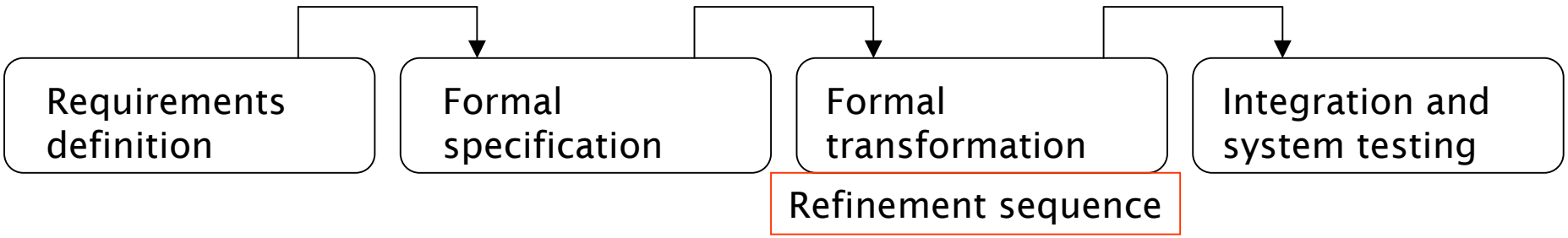
Limitations:

- Process is not visible
- Poorly structured systems
- Special tools/techniques required



Formal systems development

An approach based on formal mathematical transformations of a system specification to an executable program; it consists of:



Advantages

- Precise and error-free
- Suitable for safety-critical systems
- Correctness proofs

Limitations

- Require specialised expertise
- No obvious quality over other non-formal approaches
- Introduce extra-complexity

Reuse-oriented development

Reuse of components from other projects; essential in evolutionary approach; it consists of

- Component analysis – already existing components identified
- Requirements modification – requirements vs information about components
- System design and reuse – design framework vs existing design
- Development and integration – new components integrated with existing frameworks/templates

Advantages:

- Rapid and efficient development process
- Reduces costs and risks

Limits

- Systems that do not meet real expectations
- Loose control over the inherited components

Iterative development

Complex software systems require hybrid approaches (combination of various models) and iterations over certain stages

- Incremental development
  - Outline requirements
  - Assign requirements to increments
  - Define system architecture
  - Develop increment
  - Validate increment
  - Integrate increment
  - Validate (partial) system
  - Final system

Advantages

- Gap between system specification and system delivery is reduced
- Early increments may be used as prototypes
- Lower risk of project failure
- Highest priority services delivered first

Limitations

- Increments should be small
- Difficult to map customer's requirements into increments

Spiral development

The software process instead of being a sequence of activities is now a spiral; each loop of the spiral represents a phase of software process; each loop has four components:

- Objective setting – specific objectives defined
- Risk assessment and reduction
- Development and validation – an appropriate model is considered
- Planning – when a new phase is requested

Other models

- *Rapid Application Development (RAD)* – an incremental software development process model emphasizing an very short development cycle; useful when requirements are well-understood, modular
- *Concurrent development model* represents associated activities as states in a concurrent statecharts – it is a paradigm for a client/server applications
- *XP* (agile)

Management spectrum

Software engineering management refers to four P's: people, product, process and project

People:

- Key players in software projects:
  - senior managers
  - project (technical) managers
  - practitioners
  - customers
  - end-users
- Software team (practitioners)
  - Team organization (skills, cohesion)
  - Team communication

Software teams

Software team should have:

- A long/medium/short term plan (+recovery)
- A good mixture of skills (management, client interaction, technical, analysis, design, modelling)
- Coherent set of documents corresponding to software processes (analysis, design, test), team members inter-relationships (minutes, agendas, tasks, plans, charts)

Software product management

Requires quantitative estimates and an organized plan (mostly when solid information is unavailable !) .

- Software scope is defined (context, information objectives, function and performance)
- Problem decomposition (partitioning or problem elaboration)

Software process management

The suitable process model is decided for

- The customers requesting the product
- The characteristics of the product
- The project environment

Actions

- Preliminary plan developed
- Process decomposition

Software project management

Software project management requires assessing the risks involved and understanding the problems that may threaten the project; 10 facts undermining a project

- Software people don't understand customer's needs
- The project scope poorly defined
- Changes poorly managed
- The technology changes
- Business is changing
- Unrealistic deadlines
- Resistant users
- Losing the sponsorship
- Lack of skilled people in software team
- Managers/Practitioners do not use best practices

Summary

- Software engineering processes and their associated models are presented
- Software life cycle is represented in different modelling paradigms
- Characteristics of the main software process models are analysed
- Management spectrum of activities covering people, product, process and project is briefly discussed
- The role of software teams is emphasized
- Learned about the need of modelling and engineering approach to software production