Software Engineering COMP 201

Lecturer: **Sebastian Coope**

Ashton Building, Room G.18

E-mail: ***coopes@liverpool.ac.uk***

COMP 201 web-page:

http://www.csc.liv.ac.uk/~coopes/comp201

Lecture 2 – Software Processes

Processes (building a house)

Tasks

- What to build?
- Where to build?
- How much money?
- Get the money
- Design the build (Detailed plans, timescales etc.)
- Get permissions?
- Start with foundations
- Build up from foundations
- Fully test everything
- When basic structure complete, make sure it is looks right
- Show to customer
- Re-adjust to customer's feedback

Questions?

- Does the previous list apply to Software?

Task order (early fix is easy)

What is a Process … ?

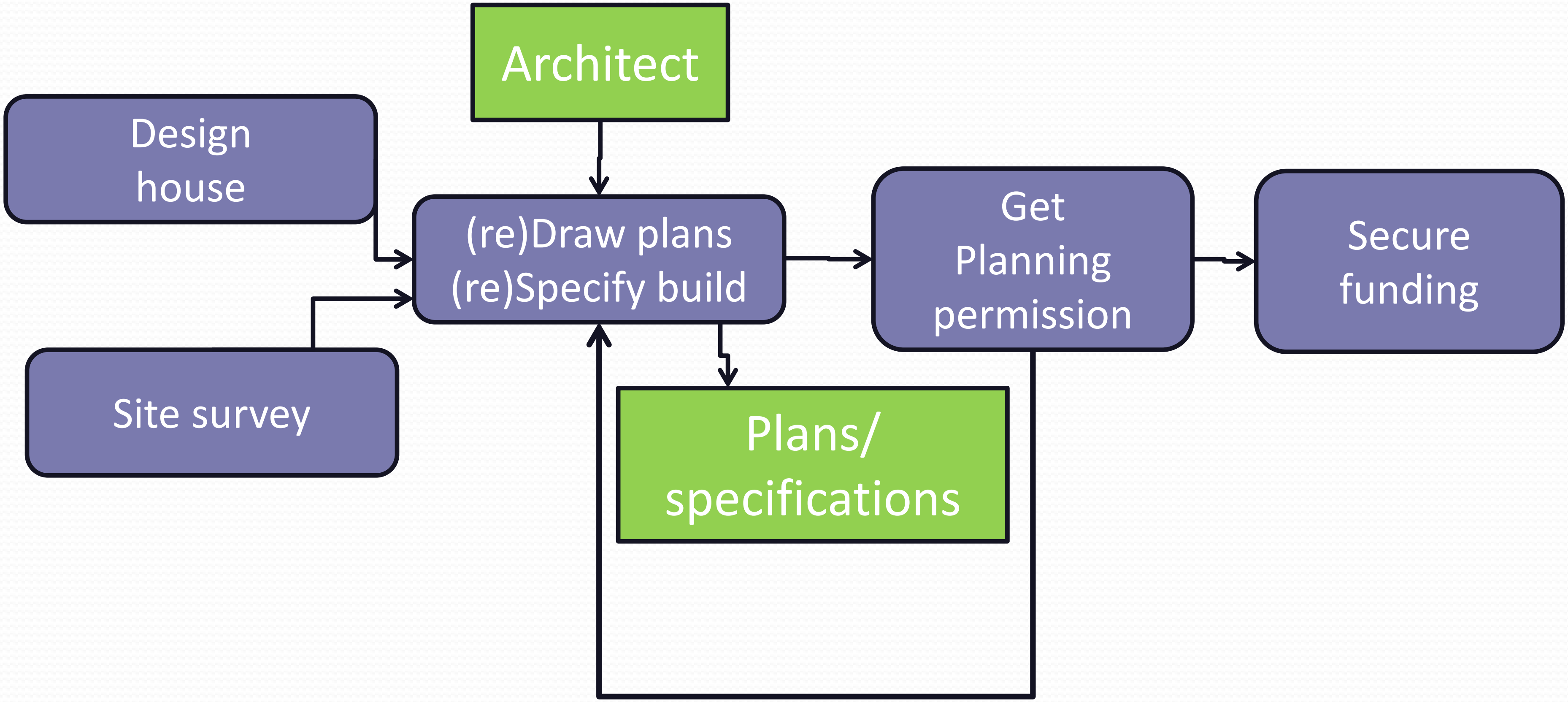
- When we provide a service or create a product we always follow a sequence of steps to accomplish a set of tasks
  - You do not usually
    - put up the drywall before the wiring for a house is installed or
    - bake a cake before all the ingredients are mixed together
- We can think of a series of activities as a **process**
- During this lecture we shall see some examples of software development processes that are used to ensure software is developed in a systematic way using tried and tested techniques

What is a Process … ?

- Any process has the following characteristics
  - It prescribes all of the major activities (what to do)
  - It uses resources and produces intermediate and final products (what is made)
  - It may include sub-processes and has entry and exit criteria (detailed work flow)
  - The activities are organized in a sequence
    - when to do it
  - Constraints or controls may apply to activities

    (budget constraints, availability of resources , etc.)

Process
Building development



Software Processes

When the process involves the building of some product we refer to the process as a **life cycle**

Software development process – software life cycle

Coherent sets of activities for

- Specifying,
- Designing,
- Implementing and
- Testing software systems

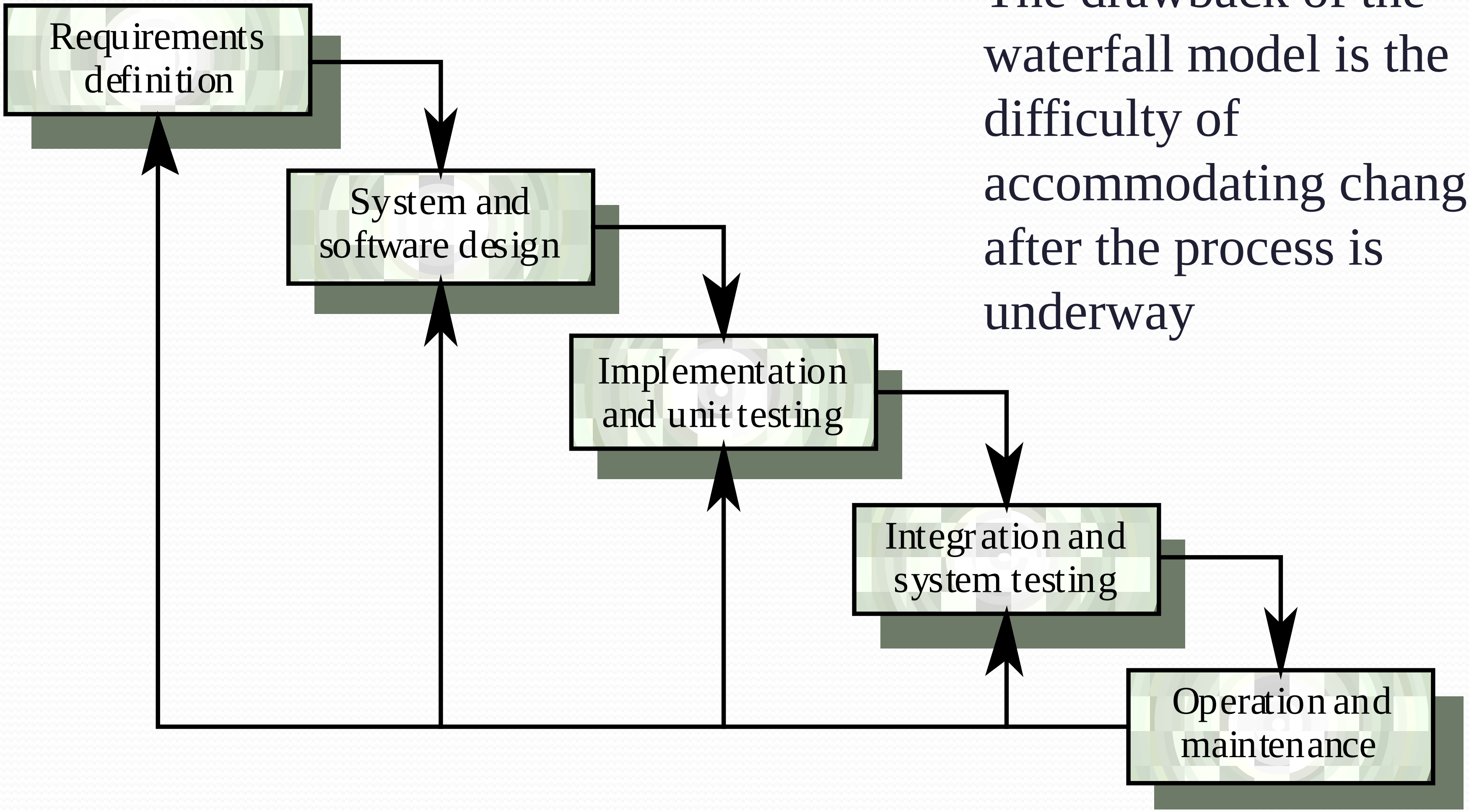
Processes and software

- Software (unlike buildings/bridges etc.)
  - Can be changed at anytime
  - Is often required to change often after construction
- Benefits
  - Software can be improved almost without limit
- Leading to problems
  - Software often gets faults as it evolves
  - Software cost is hard to manage
  - Problems with user’s experience and expectations
  - Software complexity grows exponentially with size (not linear)

Software Process Models

- **The Waterfall Model (classic engineering, example bridge building)**
  - Separate and distinct phases of specification and development
- **Evolutionary Development (more like product engineering)**
  - Specification and development are interleaved
- Agile and Scrum
  - Used widely in industry today

Waterfall Model


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

The drawback of the waterfall model is the difficulty of accommodating change after the process is underway

Waterfall Model Problems

- **Inflexible partitioning** of the project into distinct stages
- This makes it difficult to respond to changing customer requirements
- Therefore, this model is only appropriate when the (**final**) requirements are well-understood (rare in software)
- Waterfall model describes a process of stepwise refinement
  - Based on hardware engineering models
  - Widely used in military and aerospace industries

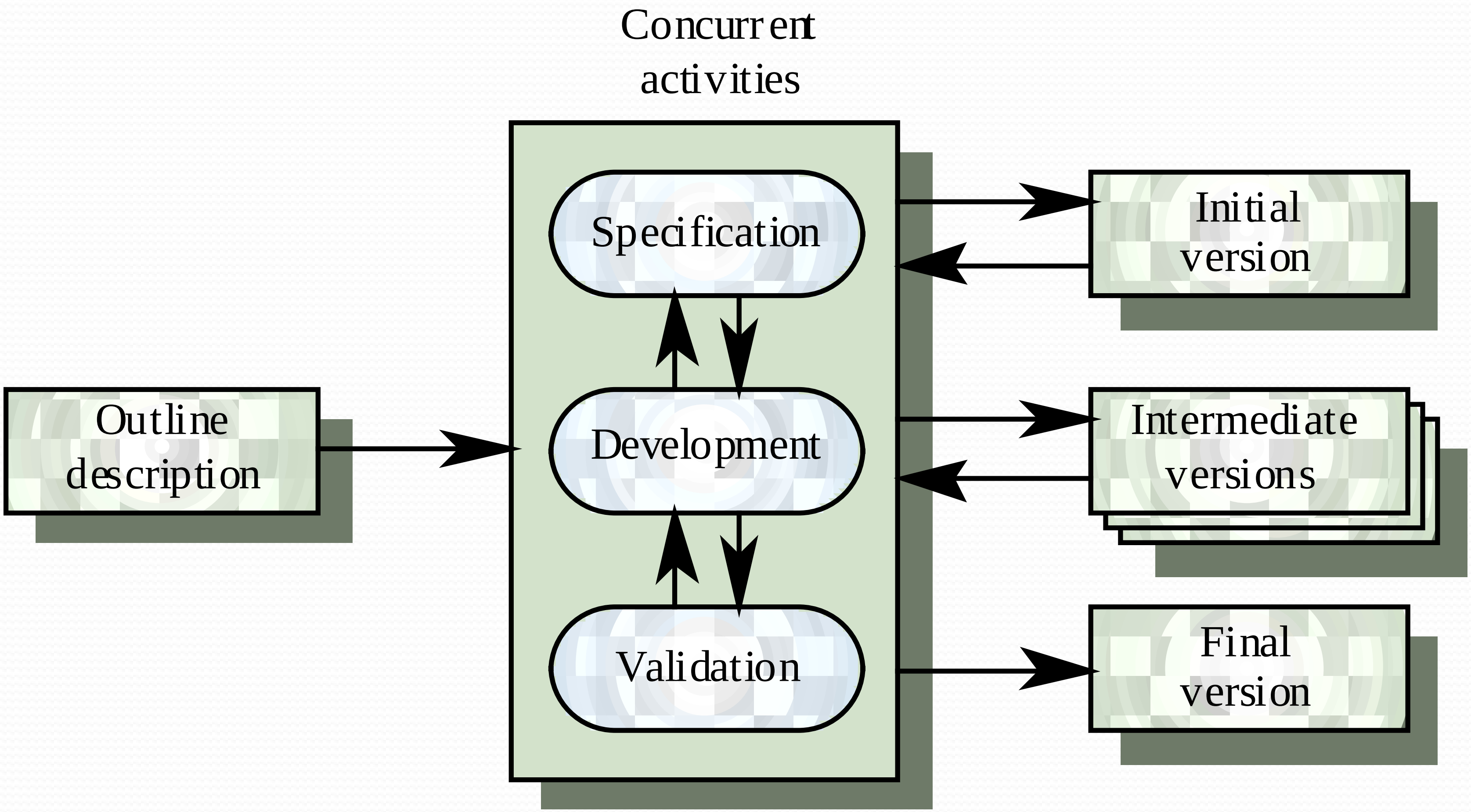
Reality check!

- Practically no one in industry follows the waterfall method as shown exactly here to produce software
- Why bother, then?
  - Each stage is an important step in software development
  - It’s easy to remember
  - The sequence is important
    - Spec. before Design
    - Design before coding etc.
  - Many industry practises could do with improvement!

Evolutionary Development

- Rather than using the waterfall model we may use **Evolutionary development** which is based upon the idea of developing an **initial implementation** , exposing it to the user and refining it based upon their response.
- **Exploratory development**
  - Objective is to work with customers and to evolve a final system from an initial outline specification.
  - Should start with *well-understood requirements*.
  - The system evolves by adding new features as they are proposed by customer.
- **Evolutionary development involves Exploratory development**

Evolutionary Development



Evolutionary Development

- **Problems**
  - Lack of process visibility
  - Systems are sometimes poorly structured
- **Applicability**
  - All types of system but rare in safety critical

Reality check

- In reality all modern development has a degree of evolutionary development BUT is hybrid (see SCRUM later)
- Sometimes the specification is mostly completed at the start and then added to as the project proceeds
- The evolution cycles pre-determined
  - 500 use cases
  - In phase 1 develop code use cases 1-50
  - In phase 2 develop use cases 51-100
  - In phase 3 develop use cases 101-200
  - In phase 4 develop use cases 201-500

Agile development

- Lightweight approach to software development
- Example include Scrum and XP
- Focused on
  - Code development as code activity
  - Test driven development (tests developed before code)
  - Often use pair programming
  - Iterative development
  - Self-organised teams (people sign up for tasks)

Incremental Development (Scrum)

- Rather than deliver the system as a single delivery, **the development and delivery is broken down into increments** (SCRUM ***sprints***) with each increment delivering part of the required functionality
- **User requirements are prioritised** and the highest priority requirements are included in early increments
- **Once the development of an increment is started, the requirements are frozen** though requirements for later increments can continue to evolve

Incremental Development Advantages

- **Customer value** can be delivered with each increment so system functionality is available earlier
- **Early increments** act as a prototype to help elicit requirements for later increments
- **Lower risk of overall project failure**
- **The highest priority system services** tend to receive the most testing

In Reality

- Most software processes involve
  - Prototyping
  - Iterative building
- Why
  - It reduces risk of making the wrong product
  - It allows the software to undergo more testing
  - It produces working product as we go along, so less chance of inventory loss

Process is context dependent

- Nuclear power station/air traffic control
  - Highly formalised processes
  - Detailed testing specifications
- Web development for small website
  - Prototyping
- Web development for large website
  - SCRUM development, storyboarding

Final question

- The specification is by a long margin
    - The MOST critical phase of any software engineering project
- Why?

Lecture Key Points

- Software processes are the activities involved in producing and evolving a software system. They are represented in a software process model
- General activities are specification, design and implementation, validation and evolution
- Generic process models describe the organisation of software processes
- Iterative process models describe the software process as a cycle of activities