COMP201
Software Engineering
Lecture 2 – Software Processes

Lecturer: **T. Carroll**

Ashton Building, Room G.14

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

See Vital for all notes

Recap

Recap from Lecture 1

- SE is an engineering discipline concerned with all aspects of software development
- Software products consists of both the actual program and the associated documentation
- We have seen reasons for requiring solid SE principles
- Software engineers must act in an ethical and professional manner at all times

Processes

Building A House

Aknowledgement:
https://fullfact.org/media/_versions/housebuilding-england_social_media.jpg

Tasks Required

- 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
- Make sure it is looks right
- Show to customer
- Re-adjust to customers feedback

Tasks Required

- 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
- Make sure it is looks right
- Show to customer
- Re-adjust to customers feedback

Similar process for making a piece of Software?

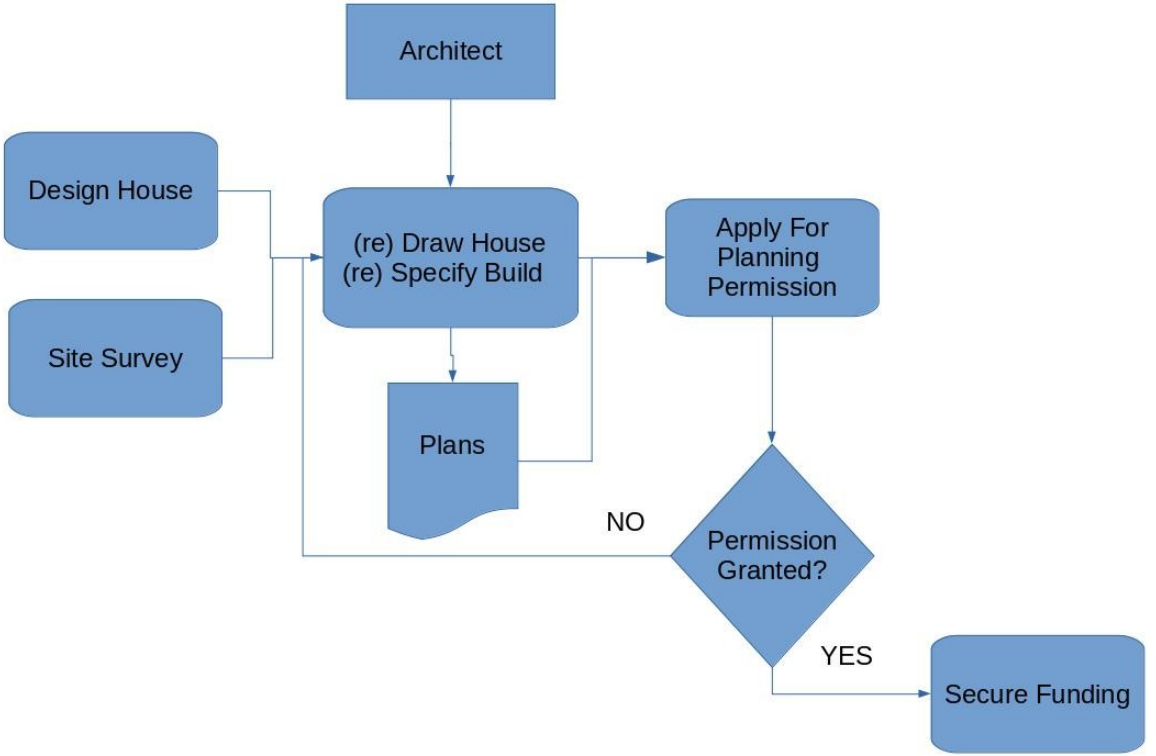
What is A Process?

- 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

Characteristics Of A Process

- Prescribes all of the major activities
- Uses resources and produces intermediate and final products
- May include sub-processes and has entry and exit criteria
- Activities are organized in a sequence
- Activities may be constrained or controlled
  -Eg: budget constraints, availability of resources , etc...

Process For Building A House



Processes And Software

- Software Can be changed at any time, even after construction

- Benefit:
  - Software can be improved almost without limit
- Problems:
  - Software often gets faults as it evolves
  - Software cost is hard to manage
  - Problems with user’s experience and expectations

Recap from Lecture 1 - What Is A Software Process?

- A Software Process is a set of activities whose goal is the development or evolution of software
- Fundamental activities in all software processes are:
  - Specification - what the system should do and its development constraints
  - Development - production of the software system (design and implementation)
  - Validation - checking that the software is what the customer wants
  - Evolution - changing the software in response to changing demands

Waterfall Model

Waterfall Model

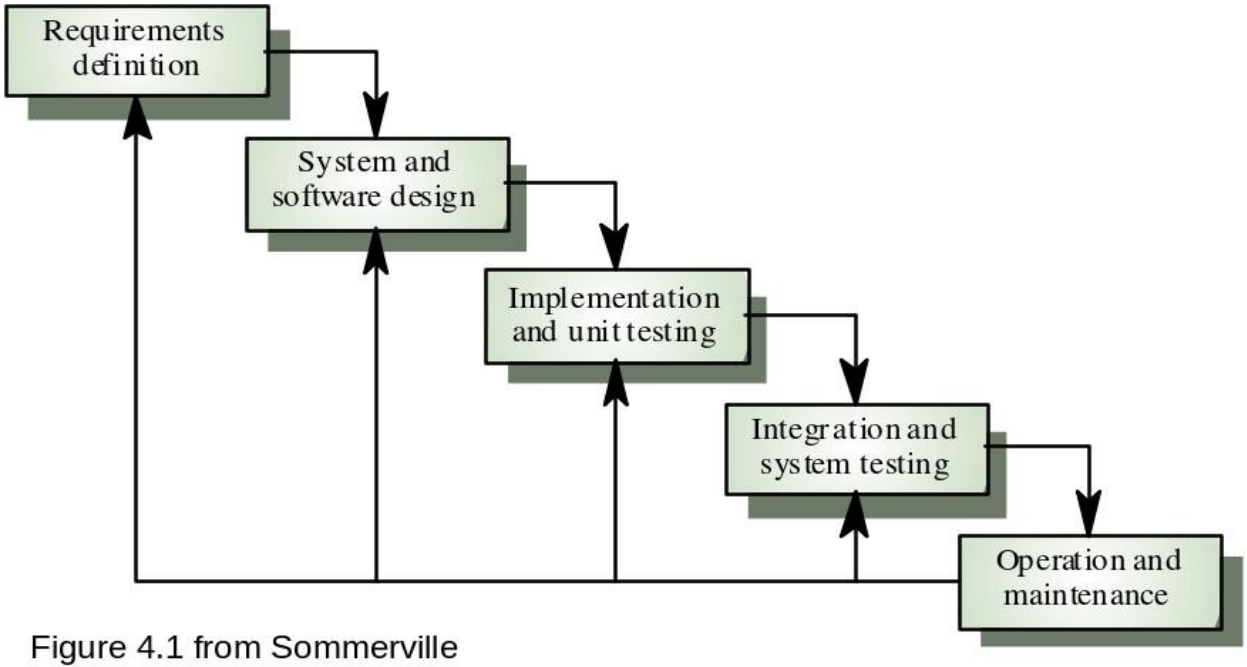


Figure 4.1 from Sommerville

Separate distinct phases, each resulting in “signed off” documents

Waterfall Model

- Clear distinction of the different aspects of the process
- A process of step-wise refinement (the sequence is good to remember)
- Operation and Maintenance often the longest phase
- Phases often overlap, and (in reality) are edited in a post hoc fashion
- Waterfall method only really applicable when the final requirements are well understood (this is rare)
- Rarely used in industry, but common in military and aerospace applications

Waterfall Model Problems

- The phases are *inflexible,* making it difficult to respond to change
- No fabrication step
  - - Program code is another design level
  - - Hence, no “commit” step – software can always be changed…!
- No body of experience for design analysis (yet)
  - -Most analysis (testing) is done on program code
  - -Hence, problems are often not detected until late in the process
- Waterfall model takes a static view of requirements, ignoring changing needs
- Lack of user involvement once specification is written
- Unrealistic separation of specification from the design
- Doesn’t accommodate prototyping, reuse, etc.

Evolutionary Model

Evolutionary Development

- Evolutionary development interleaves stages of development
- Creates an initial implementation, exposes to the user, and then refines…
- Two approaches:
  - Exploratory Development
  - Throwaway Prototyping

Exploratory Development

- Works with the customer to *explore* their requirements, refining towards the final system
- Starts with *well understood* requirements
- Features/requirements are then added by the customer
- Implementation refined, etc...

Throwaway Prototyping

- Evolutionary development uses **Throwaway Prototyping** to understand system requirements
  1) Starts with *poorly understood* system requirements
  2) Develop a **quick and dirty** system
  3) Expose to user, get comments
  4) Refine until adequate system produced
- Good when detailed requirements are not available

Evolutionary Model

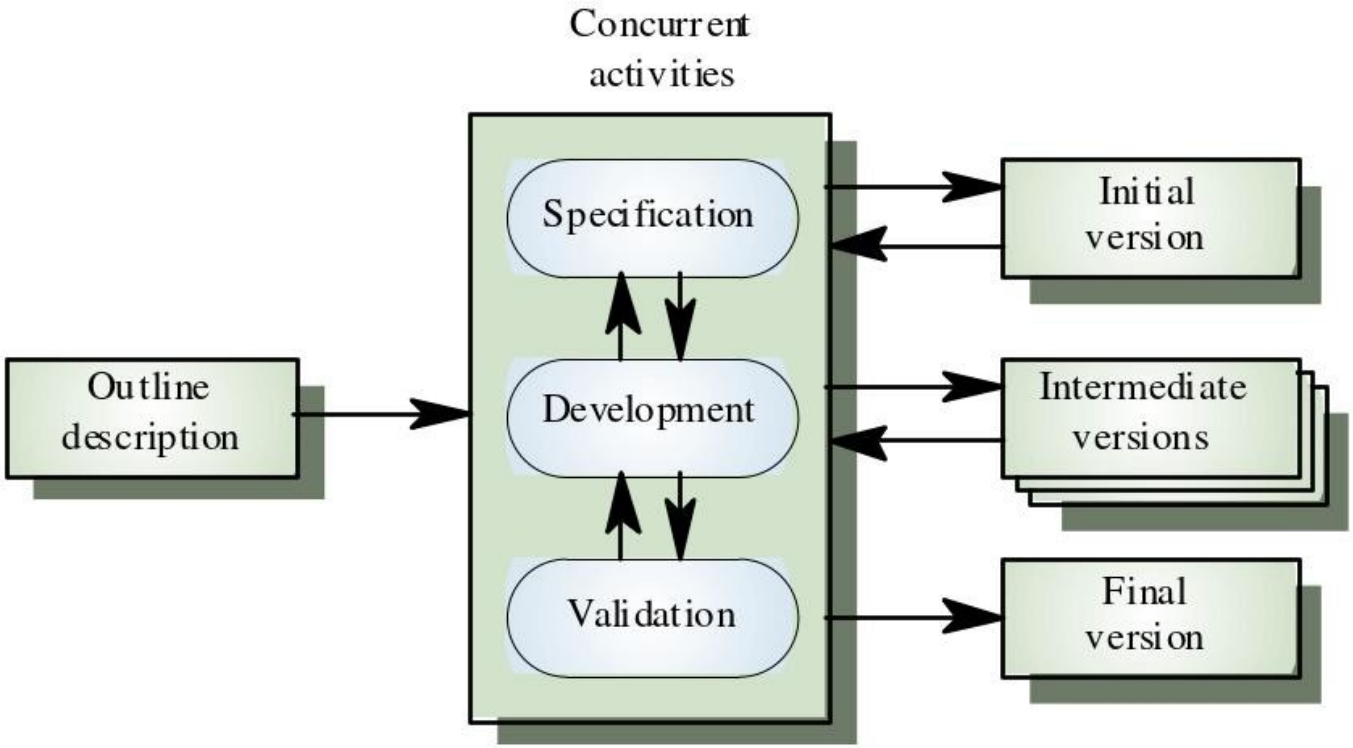


Figure 4.2 in Sommerville

Evolutionary Model

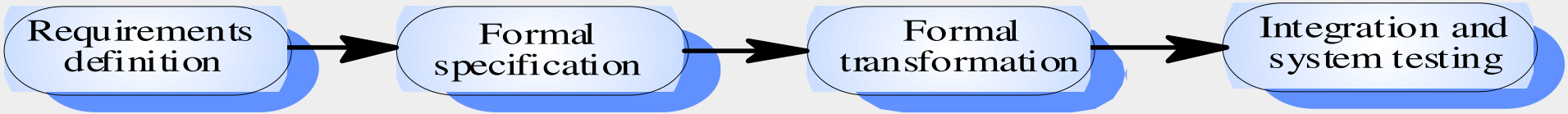
- Can have a lack of process visibility
- Can lead to poorly structured systems, especially in complex, long-life systems
- Specialist skills (in languages or prototyping) may be required
- In reality, **all** modern development has a degree of the evolutionary model
- Sometimes, the specification is mostly completed at the start and then added to
- Applicable to most systems, but rare in safety critical systems

Formal System Development

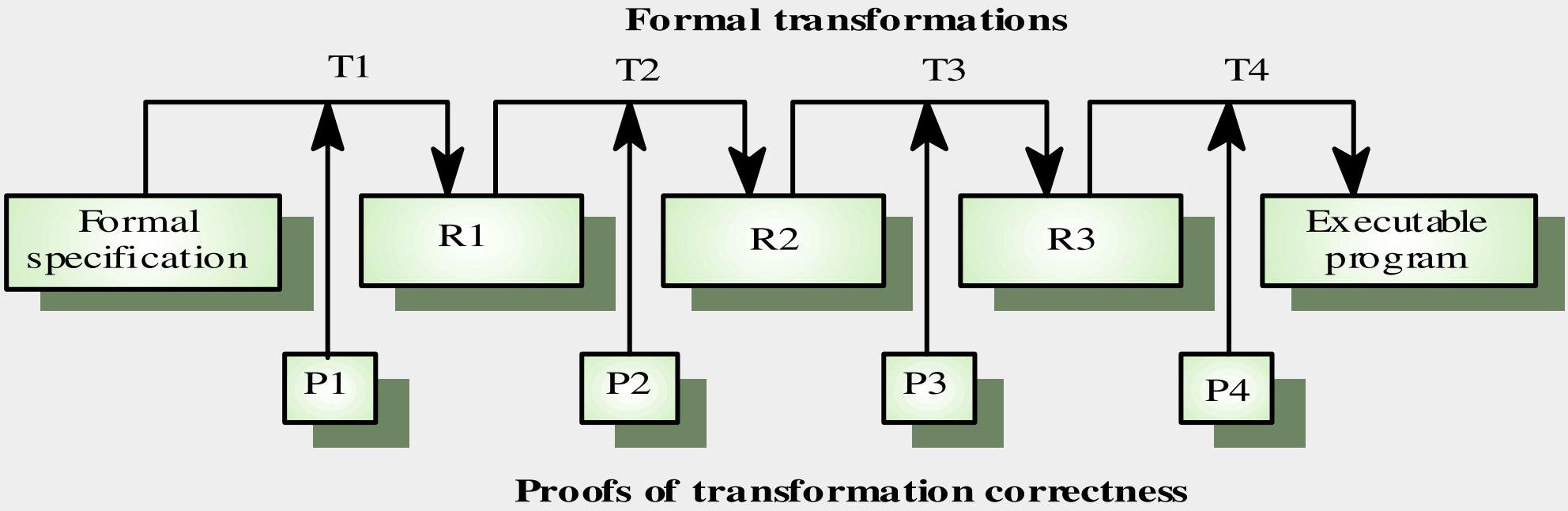
Formal Systems Development

- Based on the transformation of a mathematical specification through different representations to an executable program
- Transformations are ‘correctness-preserving’ so it is straightforward to show that the program conforms to its specification
- Embodied in the ‘Cleanroom’ approach *(originally developed by IBM)* to software development

Formal Systems Development



Formal Transformations



Example code (in Z)

Example: Banking System

WithdrawMoney

$$
\begin{array}{l}
\Delta BankAccount \\
dollarAmount? : \mathbb{N} \\
centAmount? : \mathbb{N} \\
\hline
dollarAmount? \leq dollars \\
dollarAmount? = dollars \Rightarrow centAmount? \leq cents \\
centAmount? > cents \\
\quad \Rightarrow (\quad dollars' = dollars - dollarAmount? - 1 \\
\qquad \wedge \quad cents' \quad = cents - centAmount? + 100) \\
centAmount? \leq cents \\
\quad \Rightarrow (\quad dollars' = dollars - dollarAmount? \\
\qquad \wedge \quad cents' \quad = cents - centAmount? \quad)
\end{array}
$$

Formal Systems Development

- Problems
  - Need for specialised skills and training to apply the technique (Higher initial cost)
  - Difficult to formally specify some aspects of the system such as the user interface
  - Can be more time consuming than other approaches (increased time to market)
  - Many stake holders cannot understand the specification
- Applicability
  - Critical systems especially those where a safety or security case must be made before the system is put into operation

Process Iteration

Process Iteration

- **Modern development processes take iteration as fundamental,** and try to provide ways of managing, rather than ignoring, the risk
- **System requirements ALWAYS evolve in the course of a project** so process iteration where earlier stages are reworked is always part of the process for large systems
- **Iteration** can be applied to any of the generic process models

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

Incremental Development

- Rather than deliver the system as a single delivery, **the development and delivery is broken down into increments** 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

Recap

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
  - Agile development, storyboarding

Recap

- 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

Do you agree with the following:

Think about it for next lecture...

"The specification is the MOST critical phase of any software engineering project."