

Introduction

COMP6226: Software Modelling Tools and Techniques for
Critical Systems

Dr A. Rezazadeh (Reza)

Email: ra3@ecs.soton.ac.uk or ar4k06@soton.ac.uk

September 24

Overview

- Your background and previous knowledge
- Frequently asked questions about software engineering
- Professional software development
 - What is meant by software engineering.
- Software engineering fundamentals
- Software failures
- Cost of fixing requirements errors

Your Background and Previous Knowledge

- Can you please tell us about your undergraduate modules/work experience before joining this course?
 - Involvement in Software projects/activities?
 - Using tools, notation, methods
 - Your view on Software Engineering

Frequently asked questions about software engineering

Question

What is software?

What are the attributes of good software?

What is software engineering?

What are the fundamental software engineering activities?

What is the difference between software engineering and computer science?

Question

What is the difference between software engineering and system engineering?

What are the key challenges facing software engineering?

What are the costs of software engineering?

What are the best software engineering techniques and methods?

What differences has the web made to software engineering?

Frequently asked questions about software engineering

Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

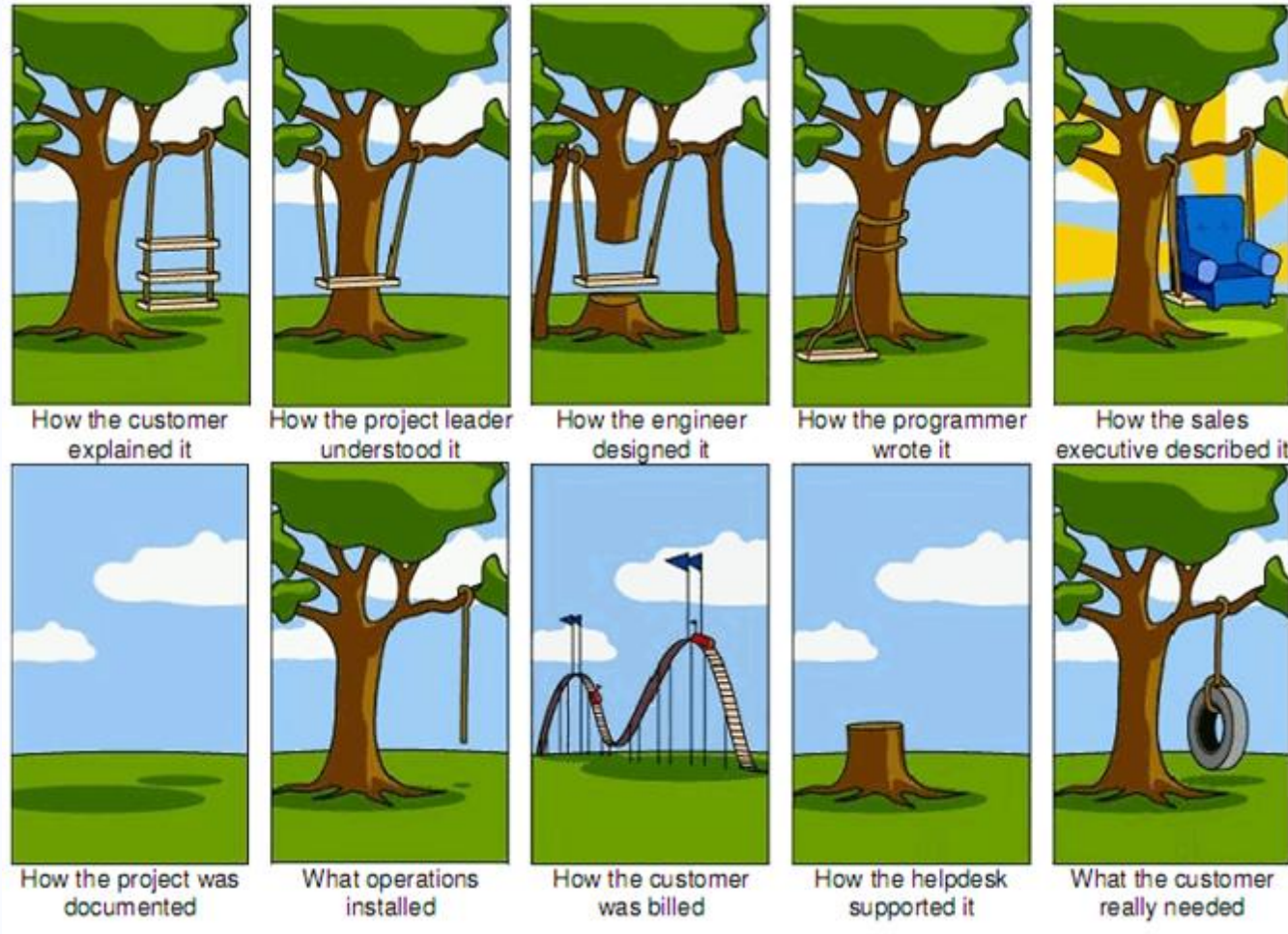
Frequently asked questions about software engineering

Question	Answer
What are the key challenges facing software engineering?	Coping with increasing Complexity , demands for reduced delivery times and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs , 40% are testing costs . For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. Lectures.
What differences has the web made to software engineering?	The web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

Project Cartoon - The Project Management Tree



<https://pmac-agpc.ca/project-management-tree-swing-story>

Software engineering

- Software engineering is an engineering discipline that is concerned with **all aspects** of **software production** from the **early stages of system specification** through to **maintaining** the system after it has gone into use.
- Engineering discipline
 - Using appropriate **theories and methods** to **solve** problems bearing in mind organizational and financial constraints.
- All aspects of software production
 - Not just technical process of development. Also, **project management** and the **development** of **tools, methods** etc. to support software production.

Importance of software engineering

- More and more, **individuals** and **society** rely on advanced software systems. We need to be able to produce **reliable** and **trustworthy** systems **economically** and **quickly**.
- The economies of ALL developed nations are dependent on software.
- More and more systems are **software controlled**
- Software engineering is concerned with **theories**, **methods** and **tools** for professional software development.
- Expenditure on software represents a significant fraction of **Gross National Product (GNP)** in all developed countries.

Software engineering – Different Aspects

- Not all aspects of SE is covered in this module
 - Process management aspects are covered in your other module:
COMP6204: Software Project Management Secure Development
 - Here we talk about the different **products** of the software process management and tools that you can use to produces these outputs.
 - **Process vs Products**










Software engineering fundamentals


- Some **fundamental principles** apply to all types of software system, irrespective of the development techniques used:
 - Systems should be developed using a **managed** and **understood development process**. Different processes are used for different types of software.
 - **Dependability** and **performance** are important for all types of system.
 - Understanding and managing the software **specification** and **requirements** (what the software should do) are important.
 - Where appropriate, you should **reuse** software that has already been developed rather than write new software.


Software Engineering – Important aspects

- Although the fundamental ideas of software engineering are applicable to all types of software system.
 - There are many **different types** of systems, and each requires **appropriate software engineering tools** and **techniques** for their development.
- This is why in this module we cover both **conventional** and **critical** systems


Software Failure


BBC  Sign in  Home  News  Sport  Weather  iPlayer  Sounds  Bitesize 

 Search





EasyJet flights cancelled after software failure
Passengers have been left waiting at airports after an IT problem caused flight cancellations.

26 May 2022 ·  News · Sussex






Software blamed for LSE failure
The London Stock Exchange says a computer software glitch was the reason trading was paralysed on Monday.

9 September 2008 ·  News · Business



The Failure of the Future
Johnny Pitts examines failed utopias to decipher if they can help us reimagine the future.

 Listen Now 

 Sounds · BBC Radio 4

Observation

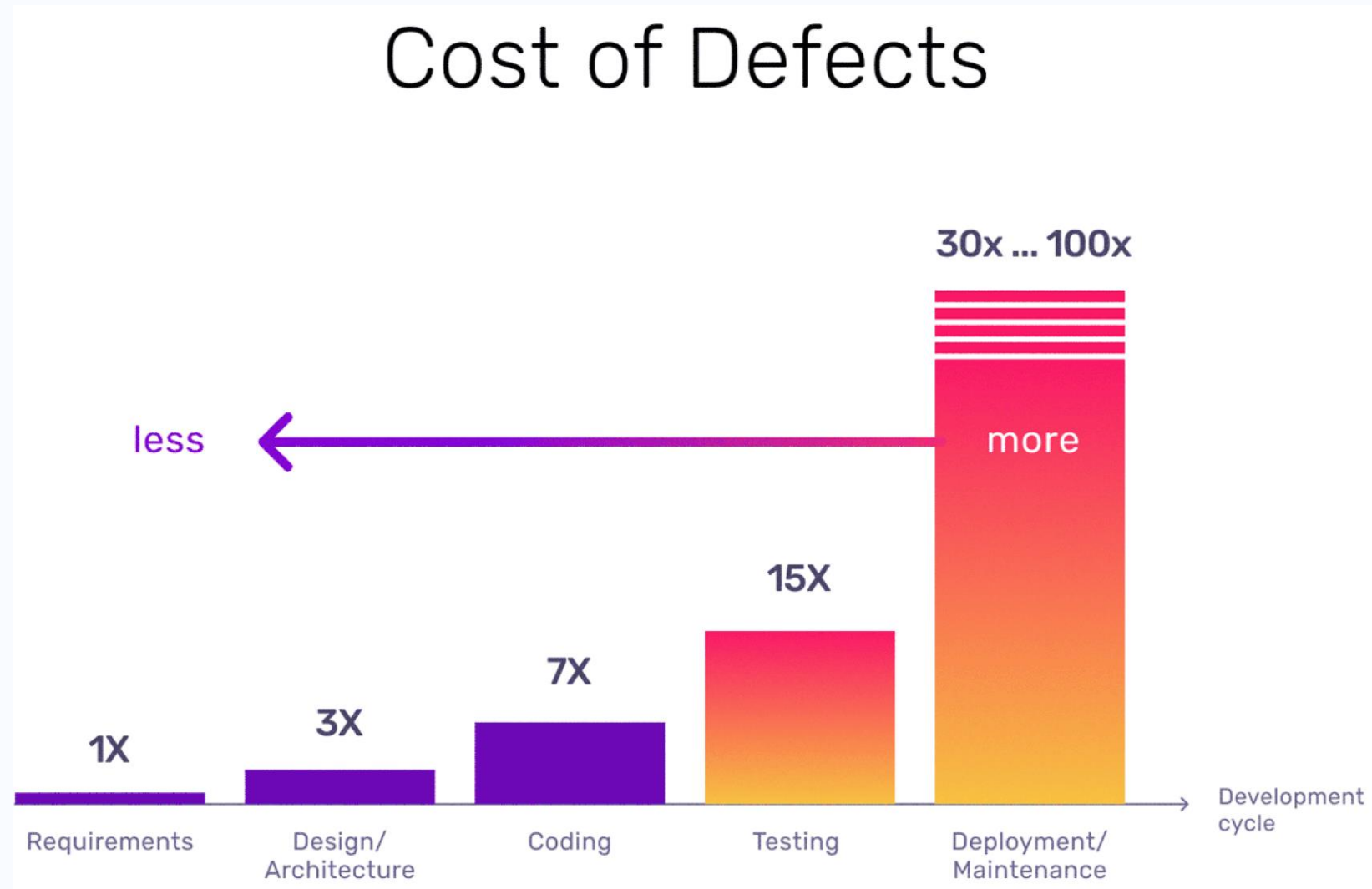
- There will be **errors** and **failures** in any software system **created** by **humans**. Developing software requires **thorough testing** under different and **realistic** conditions, which is the organization's responsibility.
- They should ensure that the software they are promoting can help its **users** and **not harm** them. Many brands can prevent **software failure** by regularly **monitoring** their **technology** and software.
- That said, by looking at some of the **top software failures**, other companies can **learn from** these **mistakes**.

Confusion and misunderstanding in software development

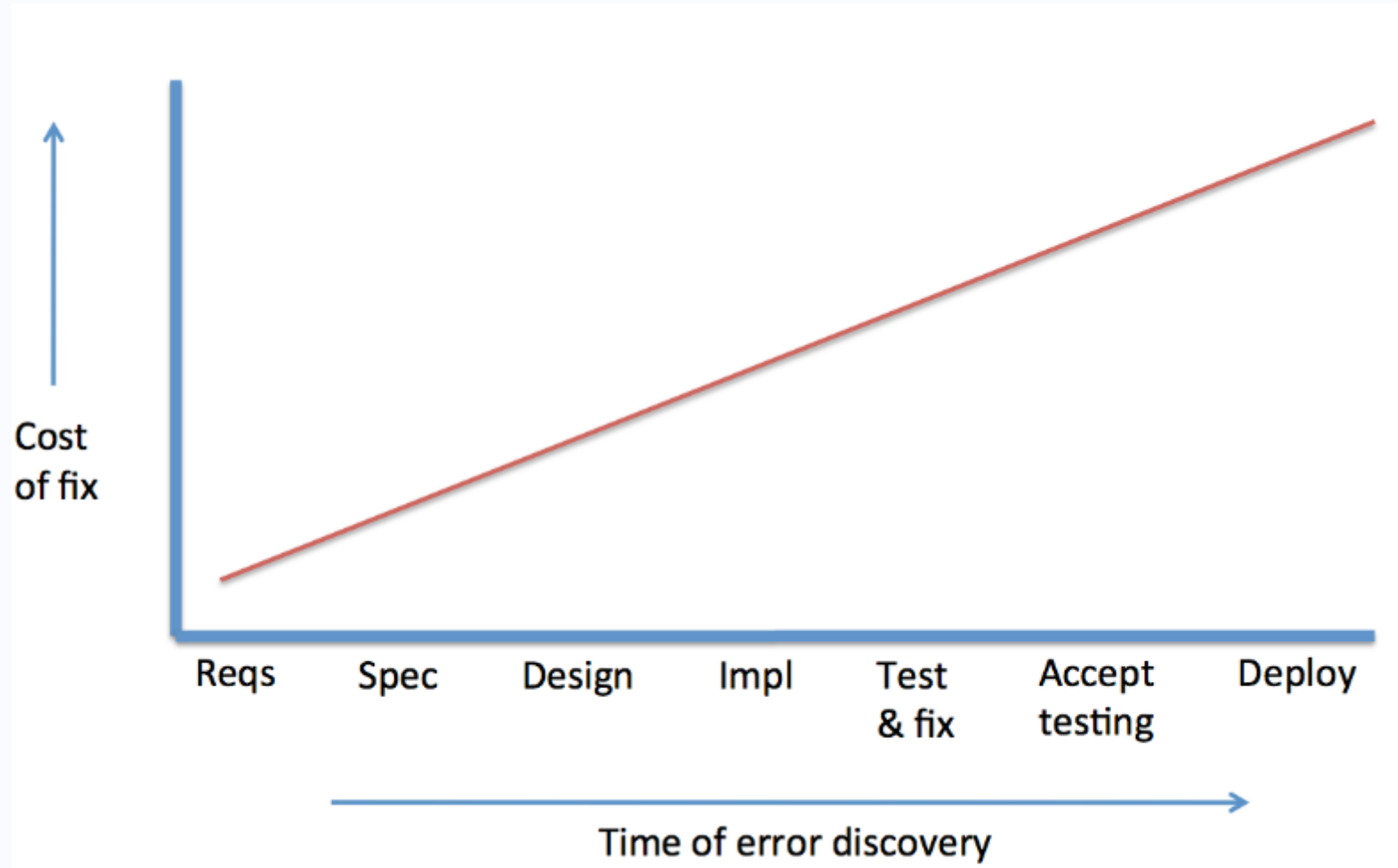
Some complaints commonly heard on large projects

- *“We don't know what we are supposed to be doing.”*
- *“Nobody told us it had changed.”*
- *“It may be what we asked for, but it's not what we want.”*

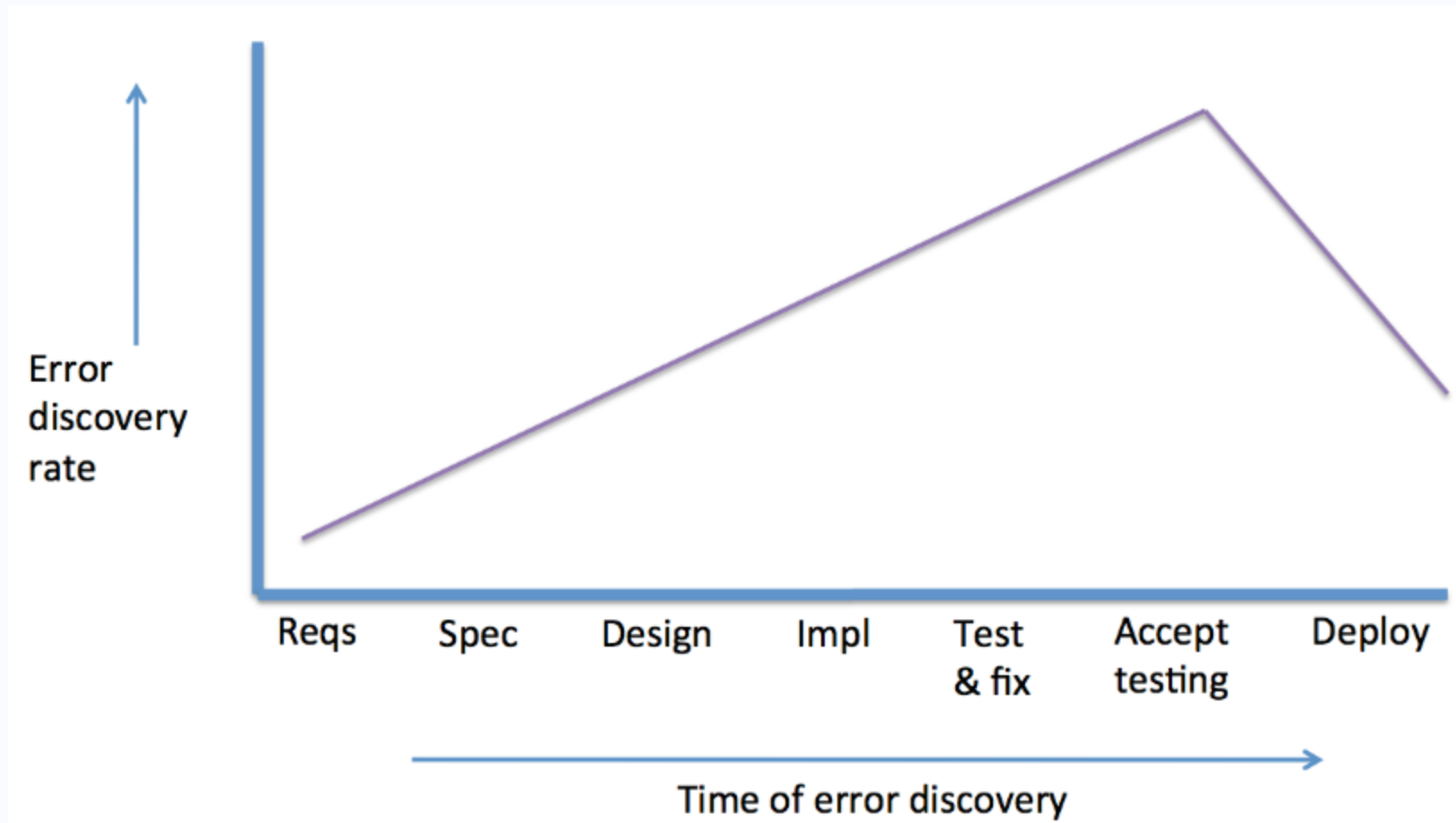
Relative Cost of Fixing Defects



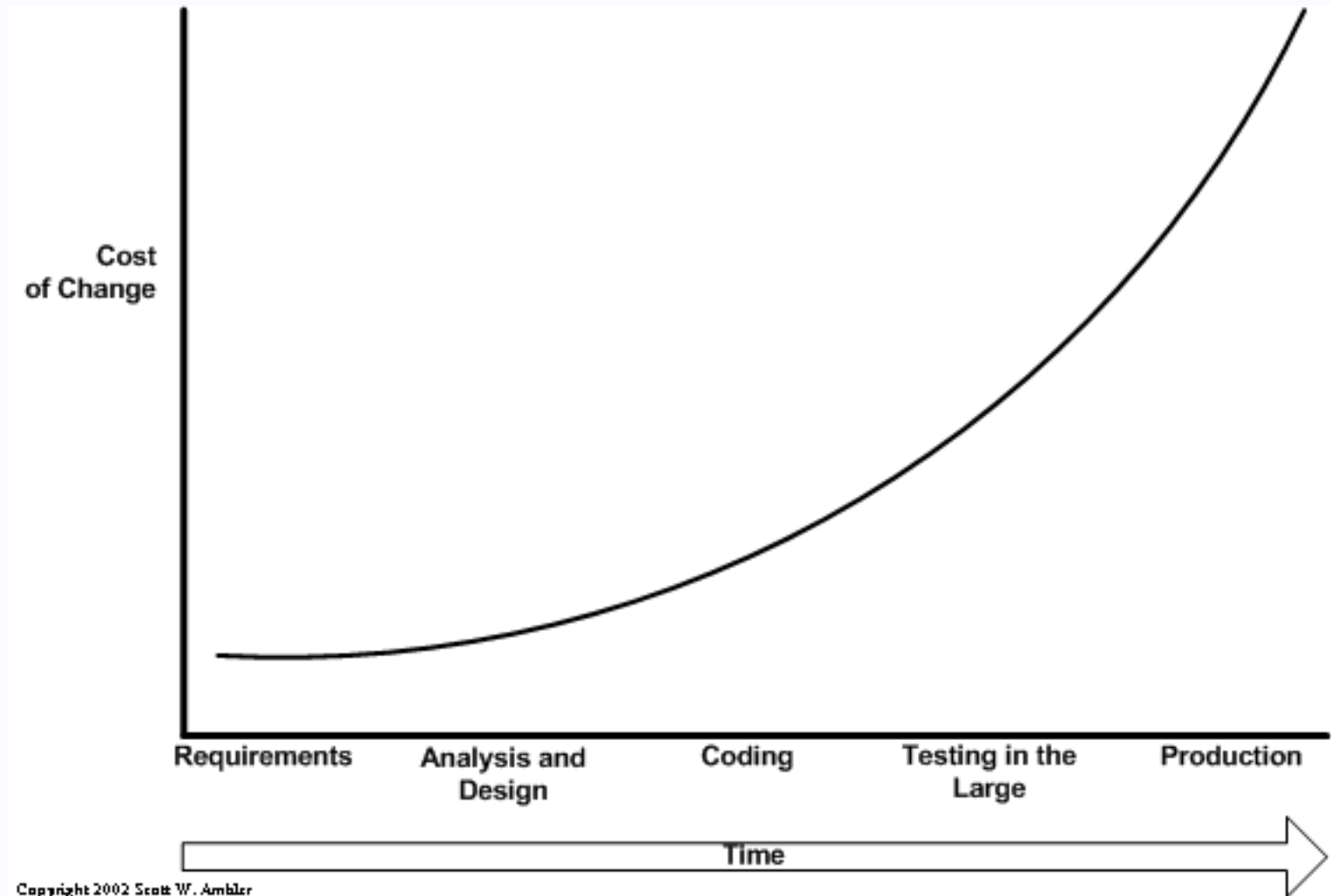
Cost of error fixes grows - difficult to change



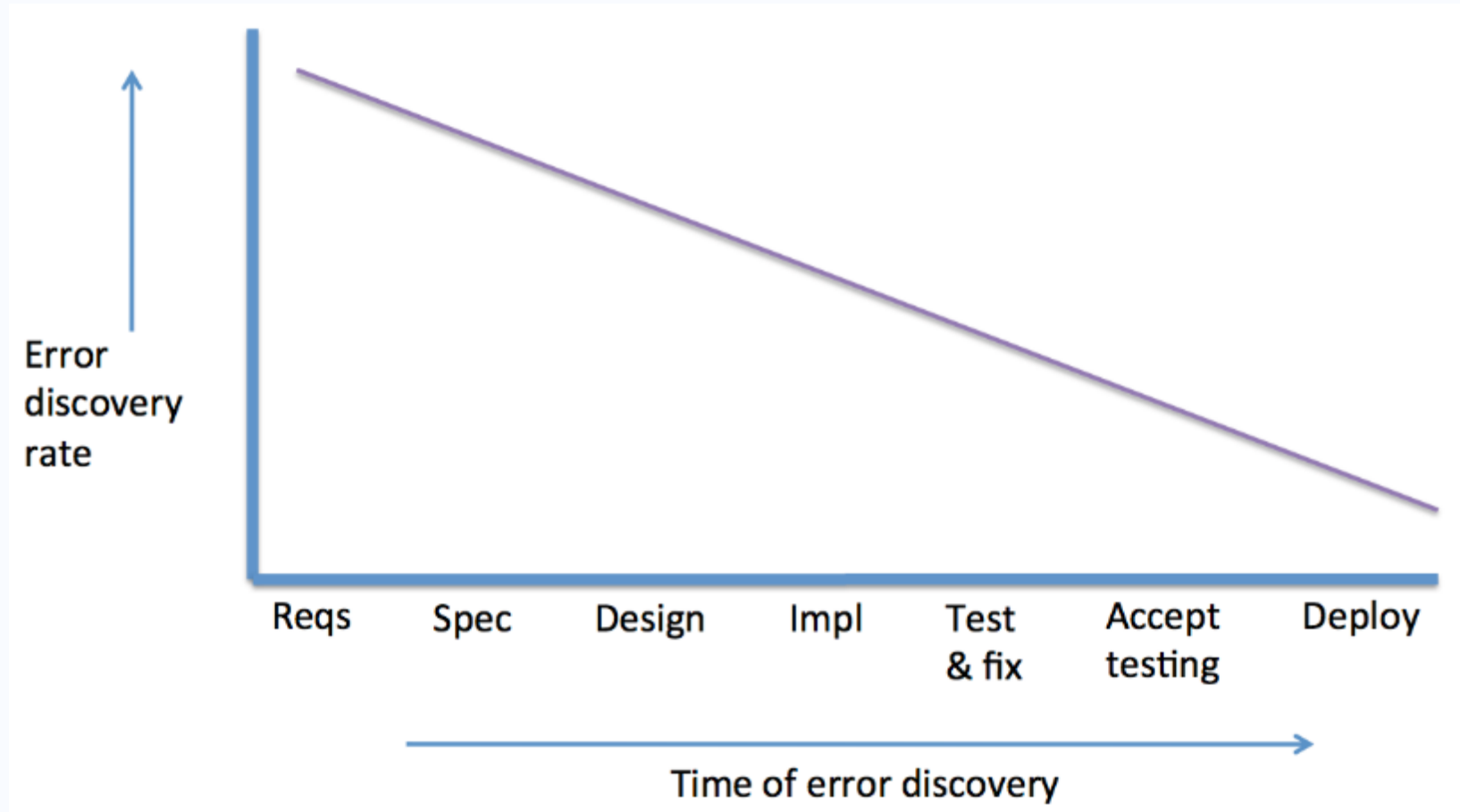
Rate of error discovery



The exponential cost of fixing bugs (Cost \times Rate of Errors)



Invert error identification rate - Is it possible?



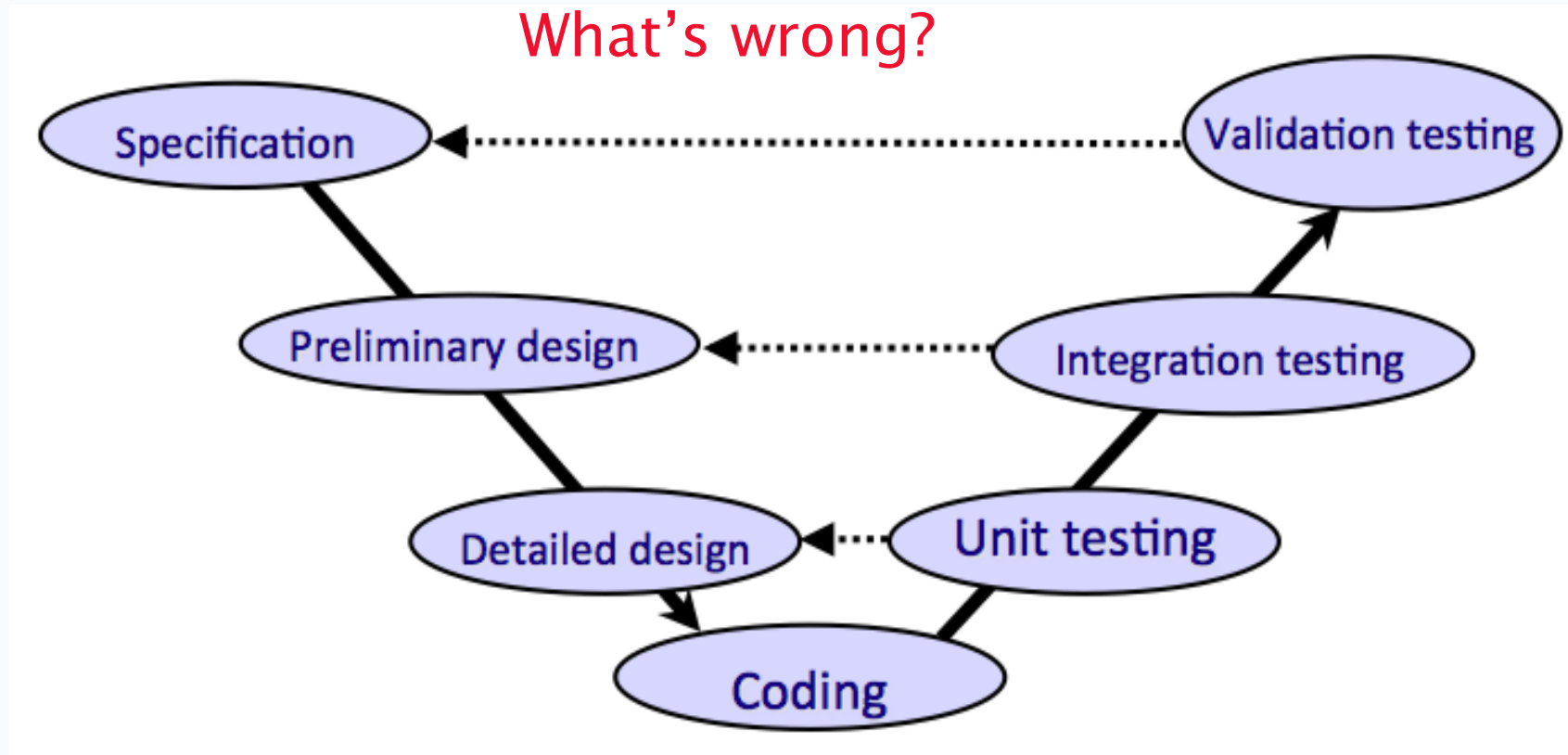
Errors, faulty design

- Errors in software can be time-consuming, expensive and possibly life-threatening.
- Many problems result from faulty specification and design of systems.

Why is it difficult to identify errors?

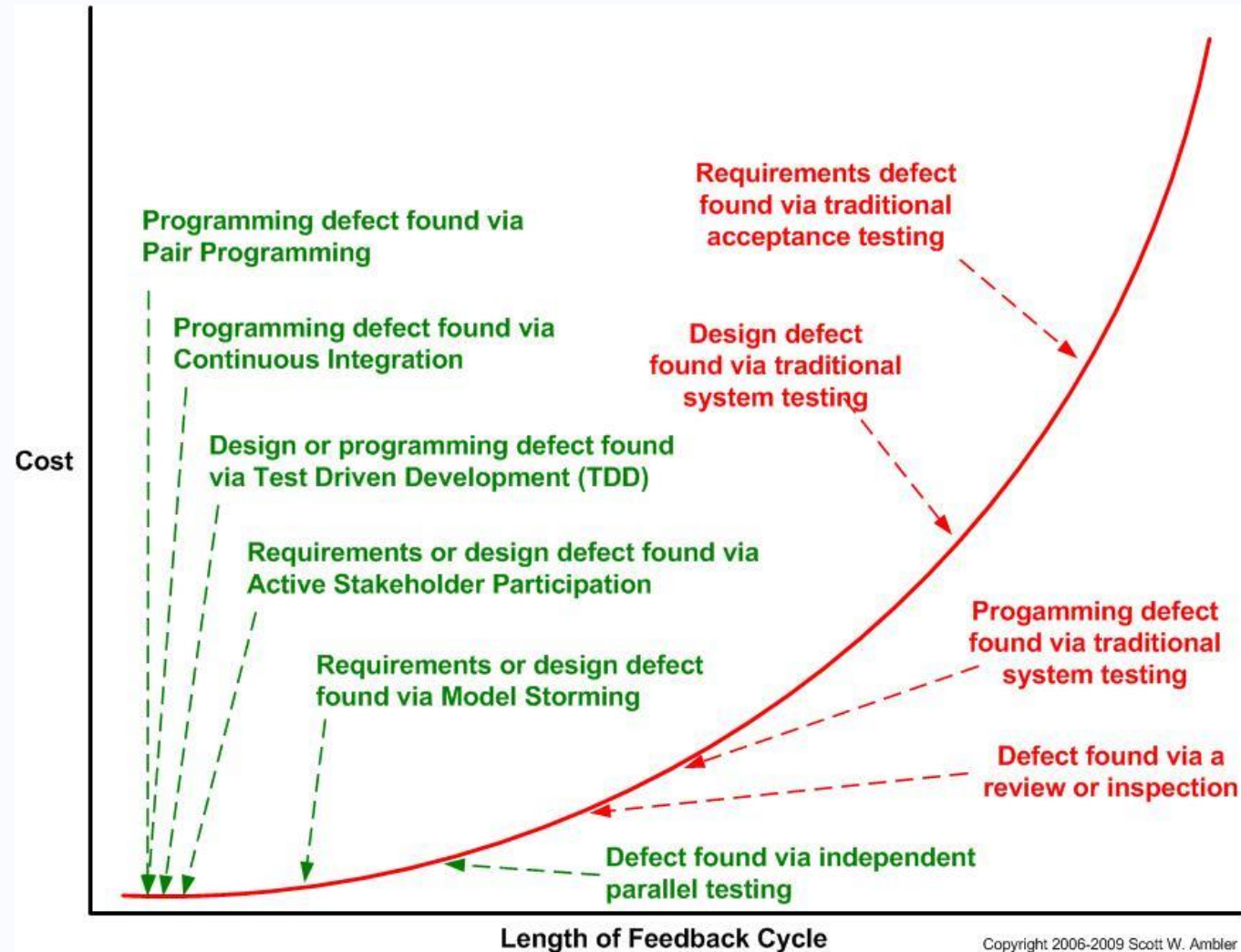
- Lack of precision
 - Ambiguities
 - inconsistencies
- Too much complexity
 - complexity of requirements
 - complexity of operating environment
 - complexity of designs

The V Model for development



Many specification errors are detected only after a lot of development has been undertaken

Comparing the Feedback Cycle of Various Inspection Techniques



Need for precision and abstraction at early stages (front-loading)

- Precision through early-stage models
 - Amenable to analysis by tools
 - Identify and fix ambiguities and inconsistencies as early as possible
- Mastering complexity through abstraction
 - Focus on what a system does (its purpose)
 - Incremental analysis and design

Validation and Verification





Early-stage models can help with:

- **Validation**
 - Ensure that the specification of the software system satisfy the requirements of the stakeholders (customers, users, administrators, ...)
- **Verification**
 - Ensure that the finished product satisfies the specification.

This Module

- Conventional Software modelling and Development
 - To build on your current knowledge and understanding of SE practices
 - Use **UML** and relevant tools to develop software artefacts/models
- Verified Design with Event-B
 - **Formal** modelling at **early stages** to prevent errors in understanding requirements and environment
 - Verify **conformance** between high-level specifications and designs using incremental approach
 - **Rodin**: **open-source toolset** for modelling, verification and simulation

Teaching & Learning Activities

W1	Mon 30th Sep	Tue 1st Oct	Wed 2nd Oct	Thu 3rd Oct	Fri 4th Oct
10am					
11am	COMP6226 L1 - Soft Model Tool & Tech Crit Sy 59P / 1007 				COMP6226 L2 - Soft Model Tool & Tech Crit Sy 59P / 1005 
12pm					
1pm					
2pm	COMP6226 L3 - Soft Model Tool & Tech Crit Sy 05 / 2011 (L/T G) 				
3pm					
4pm					
5pm		COMP6226 T - Soft Model Tool & Tech Crit Sy 59P / 1007 			

Assessment

- 50% Exam (2 hours)
- 50% Coursework (two separate coursework)
 - First CW: Small group requirements elicitation and analysis, modelling and documentation with UML plus database and UI design
 - 2^{ed} CW: Small group design-to-code coursework with Event-B

Module Website

- **COMP6226: Software Modelling Tools and Techniques for Critical Systems (2021)**
 - <https://secure.ecs.soton.ac.uk/module/2425/COMP6226/33200/>
 - <https://secure.ecs.soton.ac.uk/noteswiki/w/COMP6226>

Resources and Books

- **Software Engineering (10th Edition)** – 2016 by [Ian Sommerville](#)
- **Software Engineering: A Practitioner's Approach** – 2014 by [Roger Pressman and Bruce Maxim](#)
- A concise introduction to software engineering. Springer, 2008 by [Pankaj Jalote](#)
- *Object-Oriented Software Engineering: Using UML, PattBerns, and Java.* 2010, Bernd Bruegge & Allen H. Dutoit
- **The Unified Modeling Language**
<http://www.uml-diagrams.org/>
- Event- B: www.event-b.org

YOUR QUESTIONS