

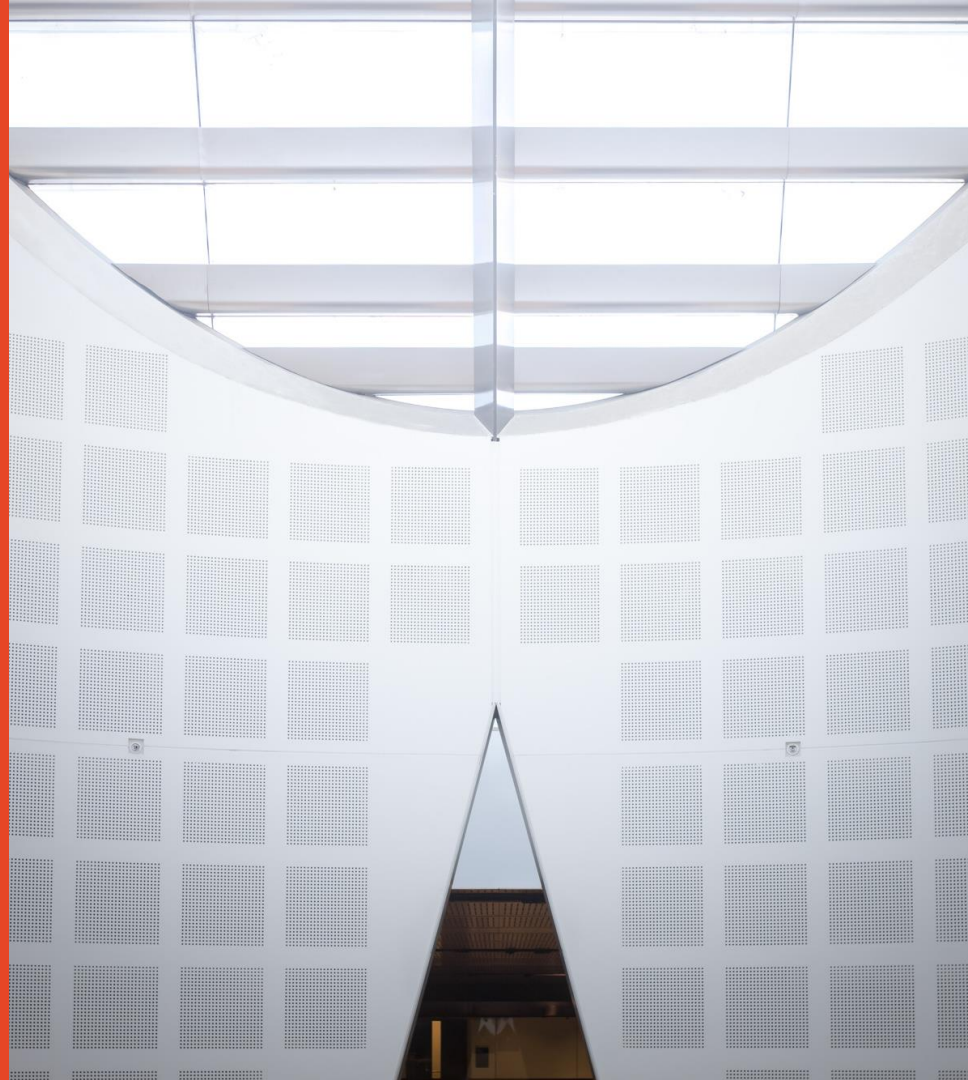
Agile Software Development Practices SOF2412 / COMP9412

Introduction

Agile Software Development

Dr. Basem Suleiman

School of Information Technologies



Copyright warning

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been reproduced and communicated to you by or on behalf of the University of Sydney pursuant to Part VB of the Copyright Act 1968 (**the Act**).

The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

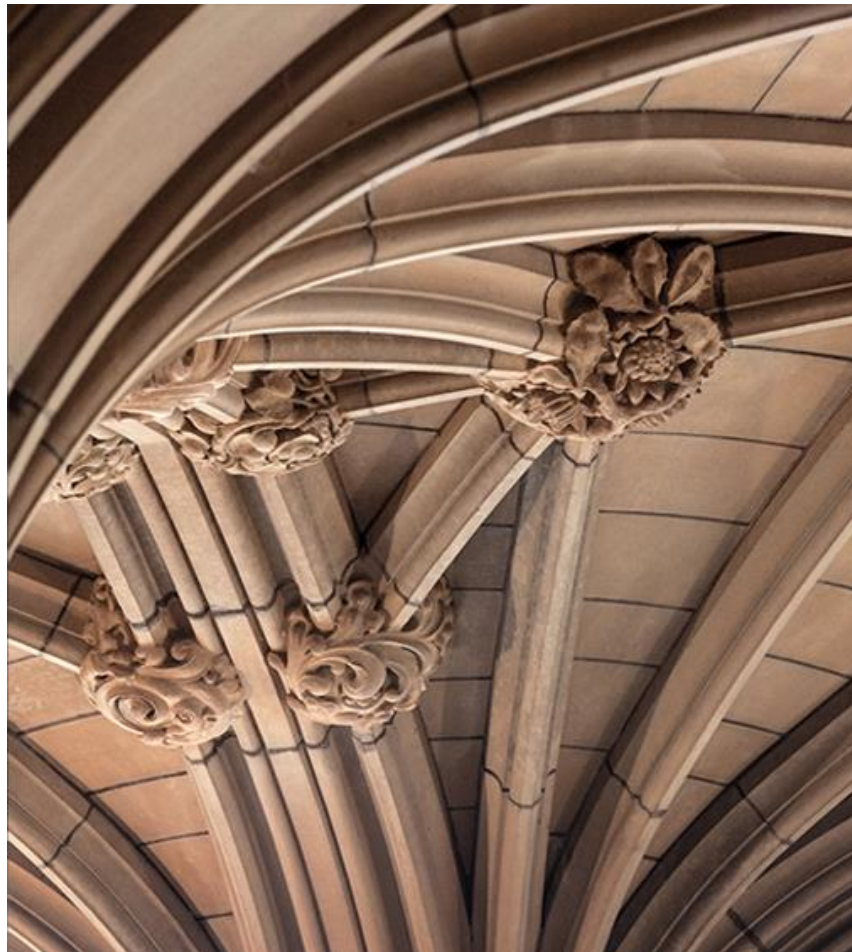
Do not remove this notice.

Agenda

- Administrivia/Introduction
- Software Engineering, Software Development, SD process, SDLC, SDLC/SD models
- Agile, agile software development, software development methods (waterfall, agile, spiral)
- Agile principles, agile practices

Administrivia

To help you get to grips with what's coming



About the Teaching Staff

Course Coordinator and Lecturer:

Dr. Basem Suleiman (basem.suleiman@Sydney.edu.au)

Supporting Academics:

A/Prof Bernhard Scholz and Dr. Ying Zhou

Teaching Associates:

Dr. Farnaz Farid and Dr. Hamza Osop

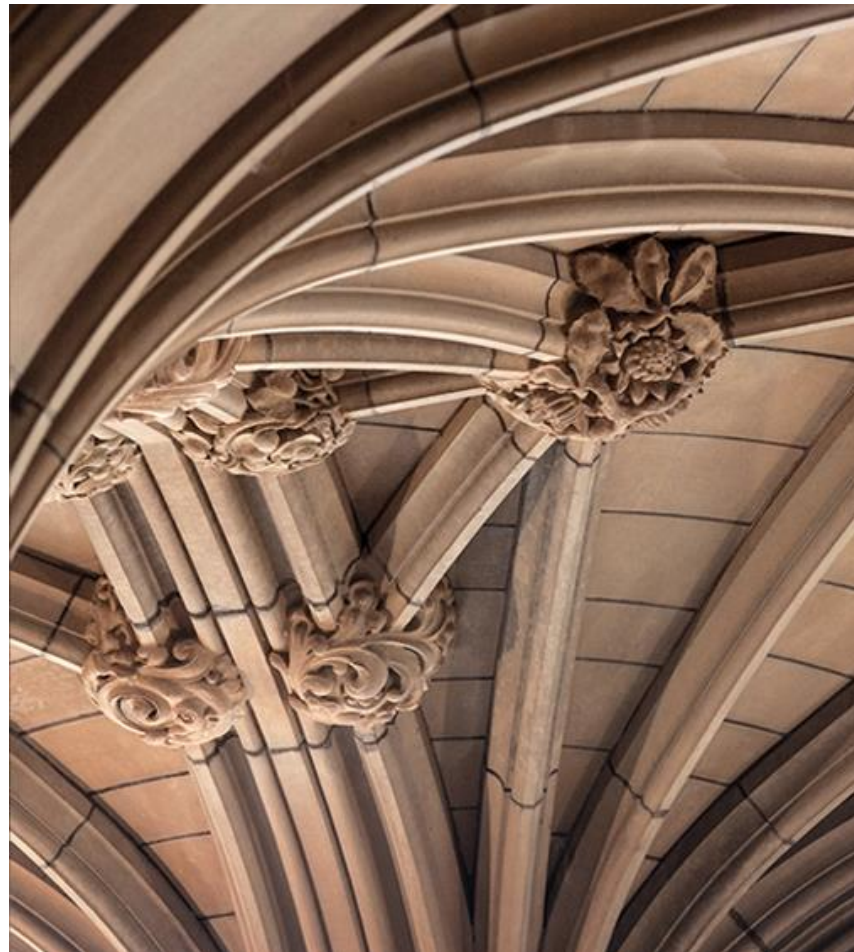
Tutors:

Tianzou (David) Wang (tianzou.wang@Sydney.edu.au)

Sadiq Sani (sadiq.sani@Sydney.edu.au)

INDUCTION

WHS, Assistance, Support and Policies



General Housekeeping – Use of Labs

- Keep work area clean and orderly
- Remove trip hazards around desk area
- No food and drink near machines
- No smoking permitted within University buildings
- Do not unplug or move equipment without permission



EMERGENCIES – Be prepared

➔ www.sydney.edu.au/whs/emergency

The screenshot shows the University of Sydney's Safety Health & Wellbeing website. The header includes the university logo and the text 'SAFETY HEALTH & WELLBEING'. A navigation bar contains links for 'SAFETY HEALTH & WELLBEING', 'UNIVERSITY HOME', 'STAFF INTRANET', and 'CONTACTS'. Below this is a search bar and a 'GO' button. A secondary navigation bar lists various topics: 'Policy & strategy', 'Responsibilities', 'Managing WHS', 'A-Z info', 'Health and wellbeing', 'Consultation', 'Incident/hazard reporting', 'Workers comp.', and 'Emergency'. The main content area is titled 'You are here: Home / WHS / Emergency'. It features a left sidebar with a list of emergency-related topics, a central section titled 'WHAT TO DO IN AN EMERGENCY' with a list of preparedness items, and a right sidebar titled 'EMERGENCY CONTACT NUMBERS' providing contact information for police, fire, ambulance, and other useful numbers.

EMERGENCY

- > What to do in an emergency
- > First aid
- > Incident & accident reporting
- > Chief building wardens
- > Emergency management
- > Building emergency procedures
- > Handling of suspicious packages
- > Chem Alert (MSDS)
- > Mercury spills

WHAT TO DO IN AN EMERGENCY

Emergencies can occur at any time, and can arise from a number of causes including fire, medical emergencies, chemical spills, gas leaks, bomb threats and physical threats. The first priority in any emergency situation is the safety of all people who may be in danger.

- [Be prepared](#)
- [Fire alarms](#)
- [Emergency response](#)
- [Medical emergencies](#)
- [People with disabilities](#)
- [Hazardous material incidents](#)
- [Gas leaks](#)
- [Phone threats](#)
- [Unattended bags or other suspicious items](#)
- [Emergency lockdown](#)
- [Personal safety on campus](#)
- [Personal threats](#)
- [Suspicious behaviour](#)

EMERGENCY CONTACT NUMBERS

POLICE, FIRE, AMBULANCE:

- | Dial 0-000 from a University phone; if you are calling from an external line or mobile phone, dial 000. Be prepared to give your name and location, and details of the emergency.

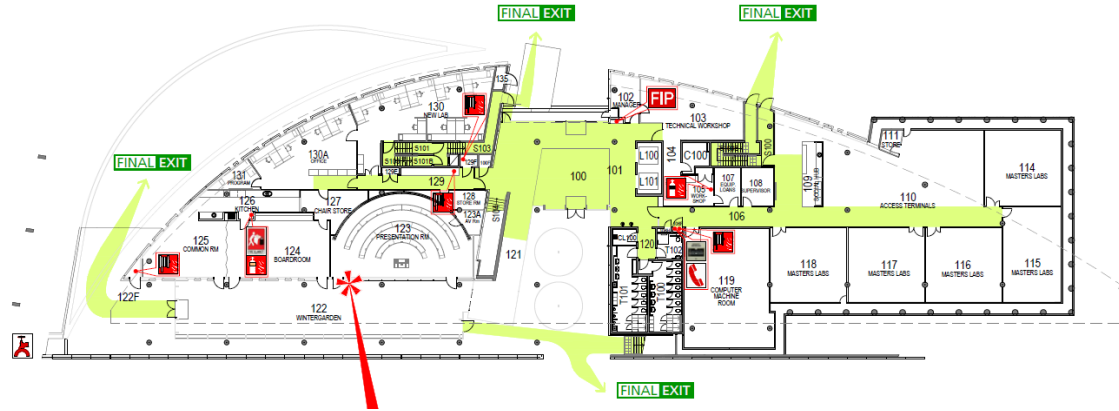
OTHER USEFUL NUMBERS

- | **University Security Service: 9351-3333**
This is an emergency number only.
- | [Chief fire wardens](#)
- | [Nominated first aid officers](#)

Be prepared

EMERGENCIES

WHERE IS YOUR CLOSEST SAFE EXIT ?



EMERGENCIES

Evacuation Procedures

ALARMS

 **BEEP... BEEP...** Prepare to evacuate

1. Check for any signs of immediate danger.
2. Shut Down equipment / processes.
3. Collect any nearby personal items.




 **WHOOOP... WHOOOP...** Evacuate the building

1. Follow the **EXIT** exit signs.
2. Escort visitors & those who require assistance.
3. DO NOT use lifts.
4. Proceed to the assembly area.

EMERGENCY RESPONSE

1. Warn anyone in immediate danger.
2. Fight the fire or contain the emergency, if safe & trained to do so.

If necessary...

3. Close the door, if safe to do so.
4. Activate the **"Break Glass"** Alarm  or 
5. Evacuate via your closest safe exit. **EXIT** 
6. Report the emergency to 0-000 & 9351-3333

MEDICAL EMERGENCY

– If a person is seriously ill/injured:

1. **call an ambulance 0-000**
2. **notify the closest Nominated First Aid Officer**

If unconscious– send for Automated External Defibrillator (AED)

AED locations.

NEAREST to SIT Building (J12)

- Electrical Engineering Building, L2 (ground) near lifts
- Seymour Centre, left of box office
 - Carried by all Security Patrol vehicles

3. **call Security - 9351-3333**
4. **Facilitate the arrival of Ambulance Staff (via Security)**



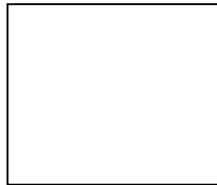
Nearest Medical Facility

University Health Service in Level 3, Wentworth Building

First Aid kit – SIT Building (J12)

kitchen area adjacent to Lab 110

School of IT Safety Contacts



CHIEF WARDEN

Name: Greg Ryan

Mobile: +61 411 406 322



FIRST AID OFFICERS



Name: Will Calleja
Location: 1 West
Phone: 9036 9706



Name: Katie Yang
Location: 2E-227
Phone: 9351 4918

**Orally REPORT all
INCIDENTS & HAZARDS
to your SUPERVISOR**

OR

Undergraduates: to Katie Yang
9351 4918

Coursework

Postgraduates: to Cecille Faraizi
9351 6060

SIT School Manager: Shari Lee
9351 4158

Assistance

- There are a wide range of support services available for students
- Please make contact, and get help
- You are not required to tell anyone else about this
- If you are willing to inform the unit coordinator, they may be able to work with other support to reduce the impact on this unit
 - eg provide advice on which tasks are most significant

DISABILITY SERVICES

Do you have a disability?

- You may not think of yourself as having a 'disability' but the definition under the **Disability Discrimination Act** is broad and includes temporary or chronic medical conditions, physical or sensory disabilities, psychological conditions and learning disabilities.
- The types of disabilities include:
 - Anxiety, arthritis, asthma, asperger's disorder, ADHD, bipolar disorder, broken bones, cancer, cerebral palsy, chronic fatigue syndrome, crohn's disease, cystic fibrosis, depression, diabetes, dyslexia, epilepsy, hearing impairment, learning disability, mobility impairment, multiple sclerosis, post traumatic stress, schizophrenia , vision impairment, and much more.
- Students needing assistance must register with Disability Services –
 - it is advisable to do this as early as possible.
- <http://sydney.edu.au/study/academic-support/disability-support.html>

Other support

- Learning support
 - <http://sydney.edu.au/study/academic-support/learning-support.html>
- International students
 - <http://sydney.edu.au/study/academic-support/support-for-international-students.html>
- Aboriginal and Torres Strait Islanders
 - <http://sydney.edu.au/study/academic-support/aboriginal-and-torres-strait-islander-support.html>
- Student organization (can represent you in academic appeals etc)
 - <http://srcusyd.net.au/> or <http://www.supra.net.au/>
- Please make contact, and get help
- You are not required to tell anyone else about this
- If you are willing to inform the unit coordinator, they may be able to work with other support to reduce the impact on this unit
 - eg provide advice on which tasks are most significant

Do you have a disability?

You may not think of yourself as having a 'disability' but the definition under the **Disability Discrimination Act (1992)** is broad and includes temporary or chronic medical conditions, physical or sensory disabilities, psychological conditions and learning disabilities.

The types of disabilities we see include:

Anxiety // Arthritis // Asthma // Autism // ADHD

Bipolar disorder // Broken bones // Cancer

Cerebral palsy // Chronic fatigue syndrome

Crohn's disease // Cystic fibrosis // Depression Diabetes //

Dyslexia // Epilepsy // Hearing impairment // Learning
disability // Mobility impairment // Multiple sclerosis // Post-
traumatic stress // Schizophrenia // Vision impairment
and much more.

Students needing assistance must register with Disability Services. It is advisable to do this as early as possible.

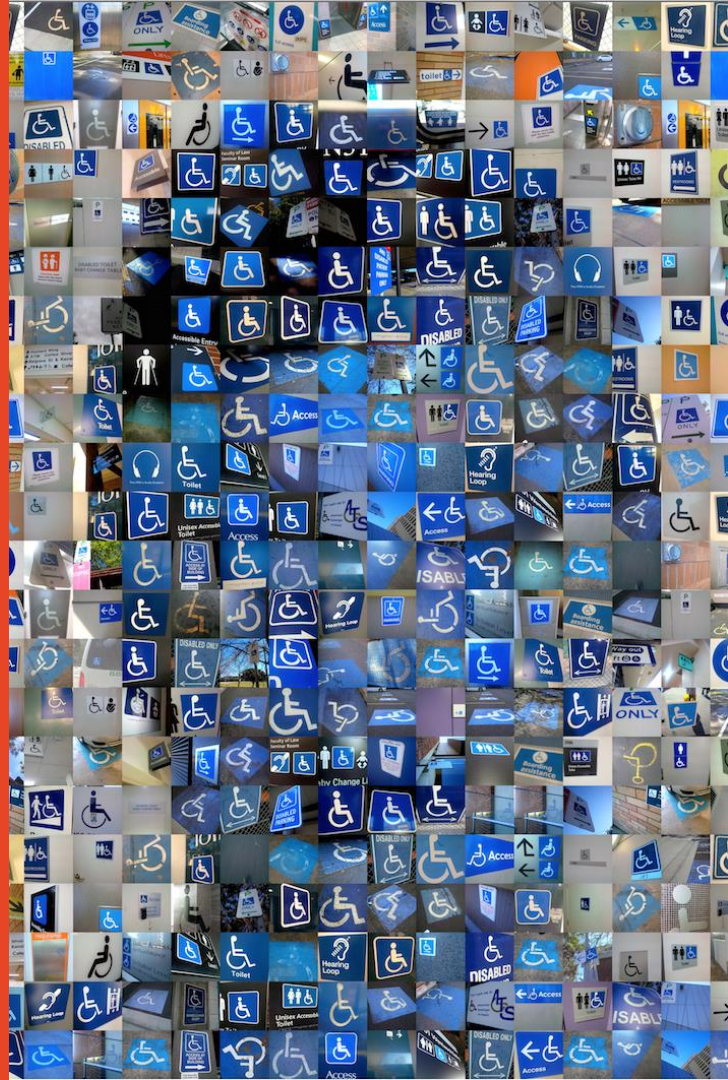
Please contact us or review our website to find out more

<http://sydney.edu.au/study/academic-support/disability-support.html>



THE UNIVERSITY OF
SYDNEY

Disability Services Office
sydney.edu.au/disability
02-8627-8422



Special Consideration (University policy)

- If your performance on assessments is affected by illness or misadventure
- Follow proper bureaucratic procedures
 - Have professional practitioner sign special USyd form
 - Submit application for special consideration online, upload scans
 - Note you have only a quite short deadline for applying
 - http://sydney.edu.au/current_students/special_consideration/
- Also, notify coordinator by email *as soon as anything begins to go wrong*
- There is a similar process if you need special arrangements eg for religious observance, military service, representative sports

Academic Integrity (University policy)

- “The University of Sydney is unequivocally opposed to, and intolerant of, plagiarism and academic dishonesty.
 - Academic dishonesty means seeking to obtain or obtaining academic advantage for oneself or for others (including in the assessment or publication of work) by dishonest or unfair means.
 - Plagiarism means presenting another person’s work as one’s own work by presenting, copying or reproducing it without appropriate acknowledgement of the source.” [from site below]
- <http://sydney.edu.au/elearning/student/EI/index.shtml>
- Submitted work is compared against other work (from students, the internet etc)
 - Turnitin for textual tasks (through eLearning), other systems for code
- Penalties for academic dishonesty or plagiarism can be severe
- Complete self-education AHEM1001 (required to pass INFOxxxx)

Advice

- Metacognition
 - Pay attention to the learning outcomes in CUSP
 - Self-check that you are achieving each one
 - Think how each assessment task relates to these
- Time management
 - Watch the due dates
 - Start work early, submit early
- Networking and community-formation
 - Make friends and discuss ideas with them
 - Know your tutor, lecturer, coordinator
 - Keep them informed, especially if you fall behind
 - Don't wait to get help
- Enjoy the learning!

Passing this unit

- To pass this unit you must do all of these:
 - Get a total mark of at least 50%
 - Get at least 40% for your progressive mark
 - Get at least 45% for your exam mark

Advice for doing well in this unit

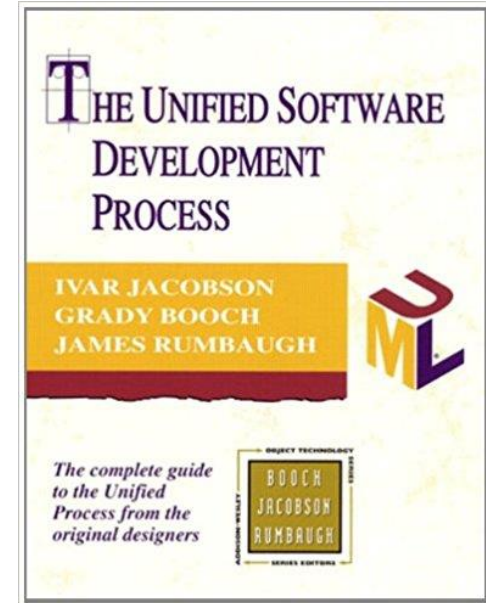
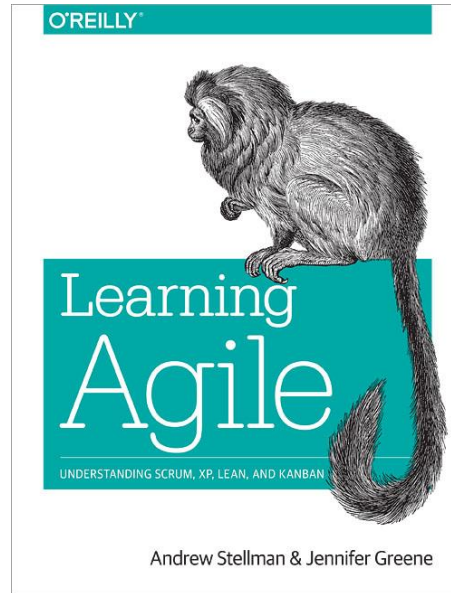
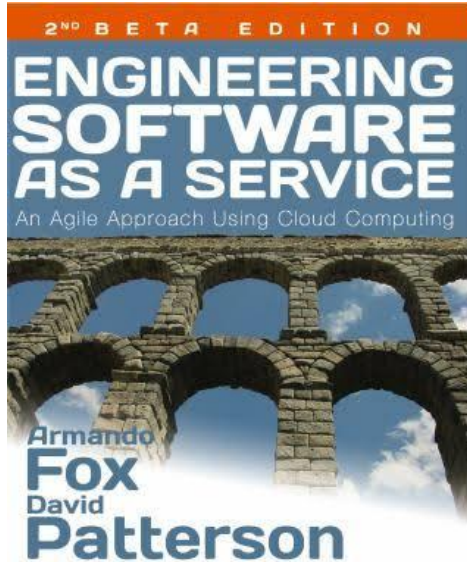
- To do *well* in this unit you should
 - Organize your time well
 - Devote 10 hours a week in total to this unit
 - Read.
 - Think.
 - Practice.

Prerequisites

- This course has the following prerequisites:
 - INFO1113 OR INFO1103 OR INFO1105 OR INFO1905
- This means that we expect students who enroll in this course to be:
 - Object-Oriented Programming (in Java) and/or
 - Data structures and algorithms (some Java programming)
- Prohibitions
 - COMP9412

Main Resources (Text books)

- We recommend the following textbooks



Lab / Tutorial Work

- Labs are available on Canvas
- 2-hour lab/tutorial work
 - Check your schedule and allocation on the timetable
- Great opportunity for interactive and hands-on learning experience
- Weekly quizzes (week 3-12)
- Respect your tutors and value their feedback
- Tutors will supervise your learning, provide you guidance
 - Not to debug your code, or solve the problems for you

Tools you will use

- Git/Github
- Graddle
- Jenkins
- Trello/Slack
- Others

Feedback

- Talk to us (e-mail) if
 - You have problems or are struggling,
 - You can't understand the contents,
 - You become ill and can't make a tutorial or quiz, or
 - You think there's something else wrong
- A discussion forum is setup:
 - This semester we are using Ed for discussions
 - Please use Ed for technical questions so that everybody can benefit from the questions and answers

Feedback to you!

- When you submit work, we have to mark it;
- We try to make this feedback as fast as possible
- Progressive marks will be recorded on Canvas

Feedback to you will take many forms: verbally by your tutor, as comments accompanying hand marking of your assignment work, and automated quiz answers. Do pay attention to this feedback, it's expensive stuff.

Assessment

What (Assessment)*	When (due)	How	Value
Group Project	Multiple Weeks	Multiple deliverables	40%
Weekly Quizzes	Multiple weeks	Online (Canvas)	10%
Exam	Exam period	Individual exam	50%

*Check latest updates of the unit outline on CUSP

Online Quiz

- Online Quiz, and are entered into Canvas. No notes nor other teaching material are permitted, i.e., closed book. For identification purposes, you need to present your student cards to your tutors. You need to stay in your assigned lab
- Format: to be shared on Canvas
- Quiz covers recent tutorials and lectures
- When: during your lab class in week 3 to 12
- **Duration:** 60 minutes
- Marks: 10%

Group Project

- Teams of 4 students
- Two phases project over multiple weeks
- Group work on a software project using:
 - Software development tools
 - Scrum practices
- Marks: 40%
- Group members may be rotated regularly

Topics Overview

- Always check CUSP for the latest version*

Note that the "Weeks" referred to in this Schedule are those of the official university semester calendar
<https://web.timetable.usyd.edu.au/calendar.jsp>

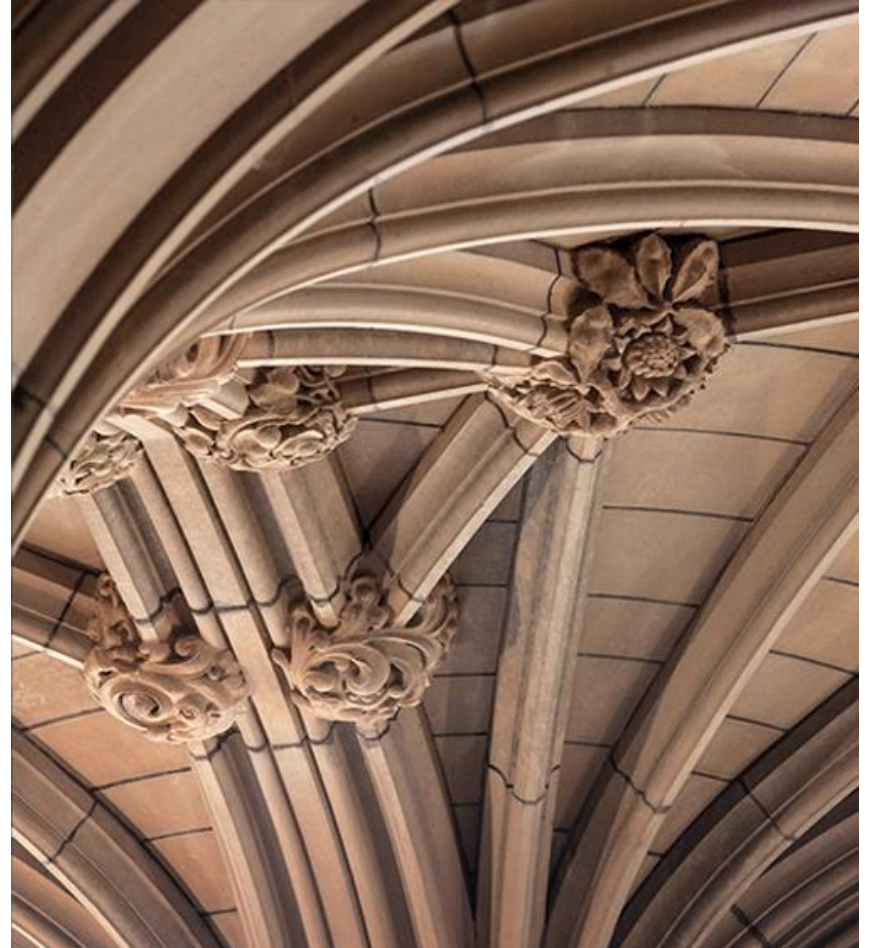
Week	Description
Week 1	Introduction and Overview; SDLC methods; Agile vs planning-heavy
Week 2	Version Control: Git Basics
Week 3	Version Control: Git Team Collaboration
Week 4	Project Automation; Software Configuration, Deployment Process
Week 5	Continuous Integration (CI)
Week 6	Software Quality Assurance; Software Testing
Week 7	Group Dynamics; Tools and Technologies for Teamwork: Planning and Issue Tracking
Week 8	Scrum Method; Expressing Requirements
Week 9	Scrum Method; Team Structures
Week 10	Project Reflection (Lessons learned and discussion of experiences so far in the project)
Week 11	Scrum Method; Estimation and Planning
Week 12	Industry Speakers
Week 13	Review
Exam Period	Assessment Due: Exam

*SOFT2412: Agile Software Development Practices (2018 - Semester 2)

*COMP9412: Agile Software Development Practices (2018 - Semester 2)

Software Engineering

What and Why?



Software is Everywhere!

- Societies, businesses and governments dependent on SW systems
 - Power, Telecommunication, Education, Government, Transport, Finance, Health
 - Work automation, communication, control of complex systems
- Large software economies in developed countries
 - IT application development expenditure in the US more than \$250bn/year¹
 - Total value added GDP in the US²: \$1.07 trillion
- Emerging challenges
 - Security, robustness, human user-interface, and new computational platforms

¹ Chaos Report, Standish group Report, 2014

² softwareimpact.bsa.org

Why Software Engineering?

Need to build high-quality software systems under resource constraints

- Social
 - Satisfy user needs (e.g., functional, reliable, trustworthy)
 - Impact on people's lives (e.g., software failure, data protection)
- Economical
 - Reduce cost; open up new opportunities
 - Average cost of IT development ~\$2.3m, ~\$1.3m and ~\$434k for large, medium and small companies respectively³
- Time to market
 - Deliver software on-time

³Chaos Report, Standish group Report, 2014

Software Failure - Ariane 5 Disaster⁵

What happened?

- European large rocket - 10 years development, ~\$7 billion
- Unmanaged software exception resulted from a data conversion from 64-bit floating point to a 16-bit signed integer
- Backup processor failed straight after using the same software
- Exploded 37 seconds after lift-off



Why did it happen?

- Design error, incorrect analysis of changing requirements, inadequate validation and verification, testing and reviews, ineffective development processes and management

⁵ <http://iansommerville.com/software-engineering-book/files/2014/07/Bashar-Ariane5.pdf>

London Ambulance Failure⁶

What happened?

- Computer aided dispatch software system in 1992
- Project cancelled and re-designed, then built by inexperienced software company
- Vehicle location system unable to track ambulances and their statuses
- Lost calls, huge delays, ambulances did not reach patients on time
- 46 lives were lost!!



Why did it happen?

- Contracted company has never developed safety critical real-time software
- Flawed software and management **process**
 - no stakeholders involvement, no quality assurance, no configuration management, no written test plans, no tracked changes
- No **test plans** during the **software process** (11 months project!)

⁶ <http://erichmusic.com/writings/technology/1992-london-ambulance-cad-failure.html>

What is the difference between SW Developers and SW Engineers?

Form groups of three and discuss for 5min



What is Software Engineering?



<http://www.purplesoft.com.au/wp-content/uploads/2017/03/software.jpg>

Software Engineering

“An engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining it after it has gone into use.”

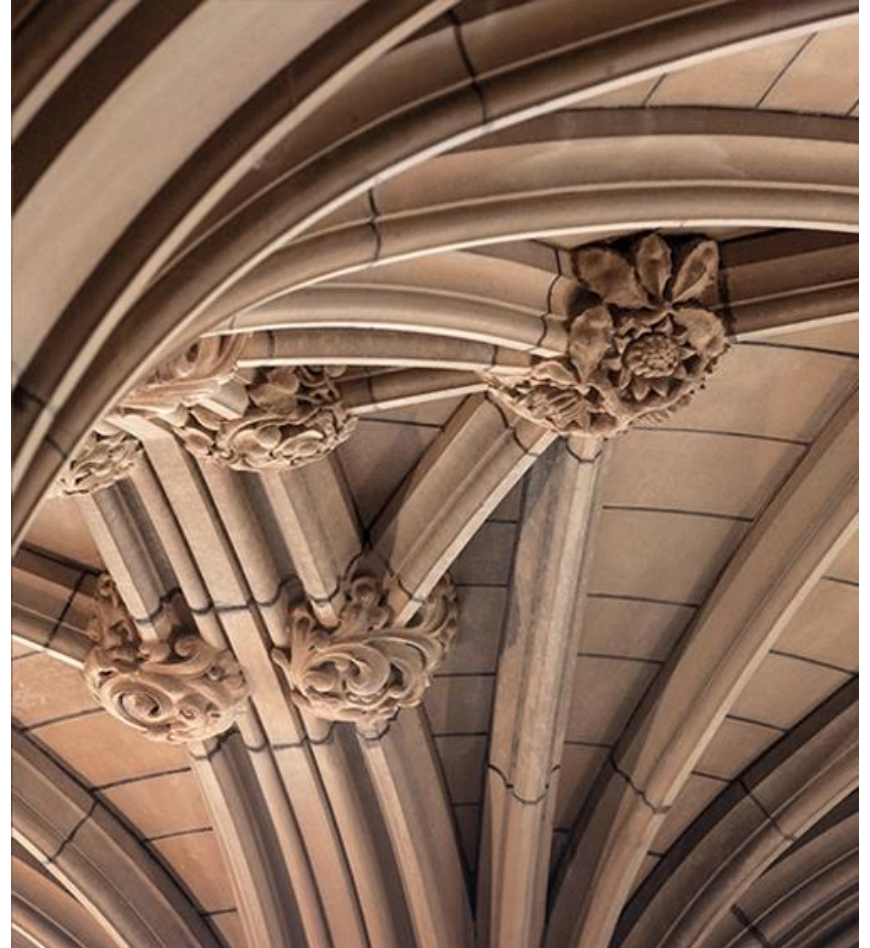
- **NOT** programming/coding! a lot more is involved
- Theories, methods and tools for cost-effective software production
- Technical process, project management and development of tools, methods to support software production
- System Engineering (Hardware & Software) - software often dominates costs

Software Engineering

“The Roman bridges of antiquity were very inefficient structures. By modern standards, they used too much stone, and as a result, far too much labour to build. Over the years we have learned to build bridges more efficiently, using fewer materials and less labour to perform the same task.” ! - Tom Clancy (The Sum of All Fears)

- The art of managing social, economical and technical factors
 - Efficient and effective **development processes** and management
 - Delivering software on-time and on-budget with all the necessary requirements
- The art of analysing and managing complexity
 - Ability to understand complex systems
 - Apply various abstraction and analytical thinking

Software Process



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 bake a cake before all the ingredients are mixed together
- We can think of a series of activities as a **process**
- Any process has the following characteristics
 - It prescribes all of the major activities
 - It uses resources and produces intermediate and final products
 - It may include sub-processes and has entry and exit criteria
 - The activities are organized in a sequence
 - Constrains or control may apply to activities
(budget control, availability of resources)

The Software Process

- Software Development Process
 - A structured set of activities required to develop a software system
 - Defines how various activities are to be done, and in what order
 - It defines a Lifecycle for a Software Development project
 - Includes processes, a set of tools, definitions of the artefacts, etc.
- Is there a universally applicable software engineering process?
 - There are many different types of software systems
 - Companies/engineers claim that they follow “methodology X”, but many times they only do some of what the methodology says
 - Most SW companies developed/customized their own SW process

The Software Process

- Although there are many software development processes, they all include common SWENG activities
 - **Specification** – defining what the system should do;
 - **Design and implementation** – defining the organization of the system and implementing the system;
 - **Validation** – checking that it does what the customer wants;
 - **Evolution** – changing the system in response to changing customer needs
- SW processes are complex and, rely on people making decisions and judgements
- Activities are complex and include sub-activities
 - E.g., requirements validation, architectural design, unit testing

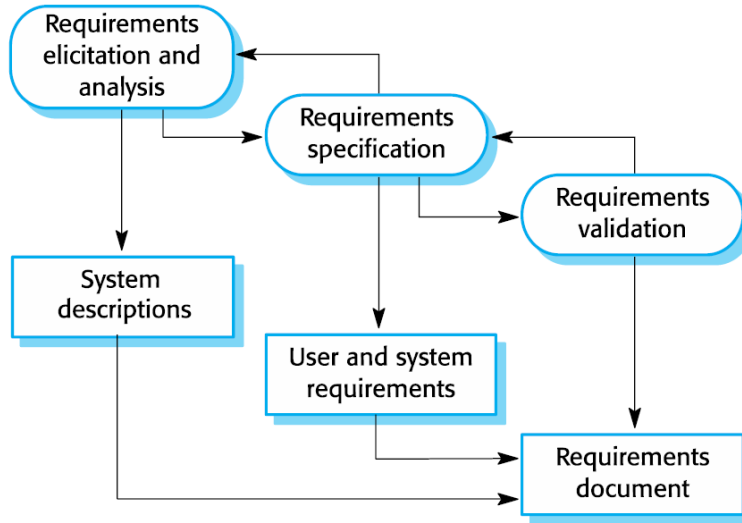
Software Engineering Body of Knowledge

- Software Requirements
- Software Design / Modelling
- Software Construction
- Software Testing
- Software Maintenance
- Software Configuration Management
- **Software Engineering Process**
- Software Engineering Tools and Methods
- Software Quality

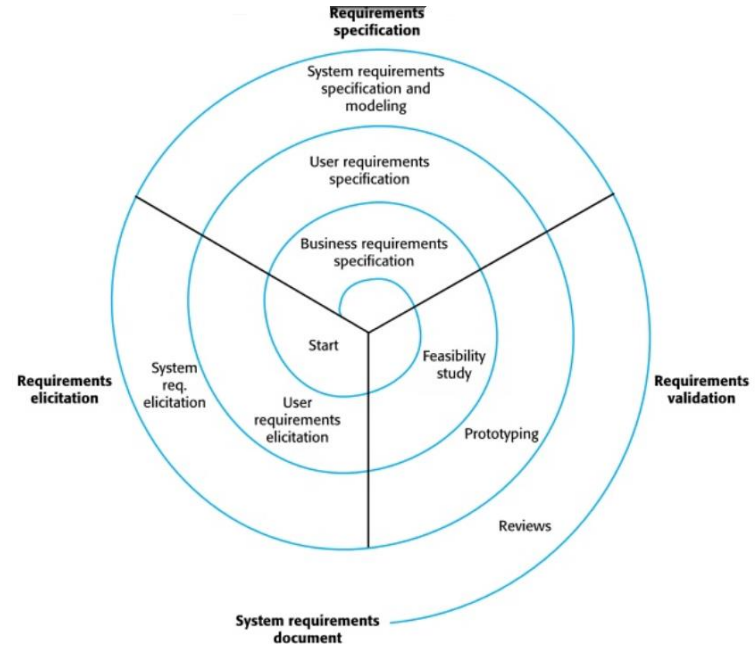


Software Engineering Body of Knowledge (SWEBOK) <https://www.computer.org/web/swelok/>

Requirements Engineering Process



Requirements engineering
main activities and deliverables



Requirements engineering
spiral view

Software Requirements Mystery



How the customer explained it



How the project leader understood it



How the engineer designed it



How the programmer wrote it



How the sales executive described it



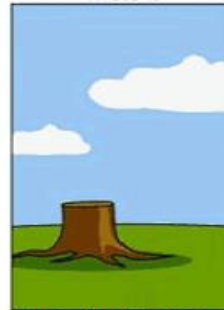
How the project was documented



What operations installed



How the customer was billed



How the helpdesk supported it



What the customer really needed

https://cdn-images-1.medium.com/max/1600/1*xzS-UkYtNOgzPvpkHGwRbQ.png

Planning in Software Development

- SW development processes is classified in terms of planning
- **Plan-driven (plan-and-document heavy-weight)**
 - All of the process activities are planned in advance and progress is measured against this plan
 - Plan drives everything and change is expensive
- **Agile processes (light-weight)**
 - Planning is incremental and continual as the software is developed
 - Easier to change to reflect changing requirements
- In practice, most practical processes include elements of both plan-driven and agile
- Each approach is suitable for different types of software
 - There are no right or wrong software processes

Software Process Models

- Also known as Software Development Lifecycle (SDLC)
- High-level abstract (simplified) representation of a process
- It presents a description of a process from some particular perspective
 - Describe the activities and their sequence but may not describe the roles of people involved in these activities

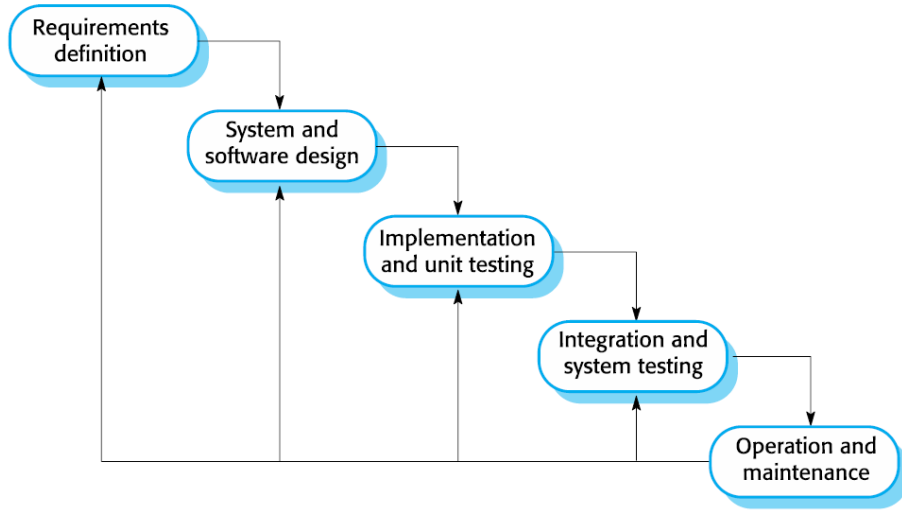
Representative Software Process Models

- **Waterfall Model**
 - Development process activities as process phases
- **Spiral Model**
 - Incremental development risk-driven
- **Agile Model**
 - Iterative incremental process for rapid software development
- **The Rational Unified Process (RUP or UP)**
 - Bring together elements of different process models
 - Phases of the model in timer (dynamic perspectives), process activities (static perspective), good practices (practice perspective)

Waterfall Model Phases

- There are separate identified phases (non-overlapping):
 - Requirements analysis and definition
 - Produces a Requirements document
 - System and software design
 - Requirements document is used to produce a Design document
 - Implementation and unit testing
 - Design document is used to produce code and test it for system components
 - Integration and system testing
 - Software components are integrated and the resulting system is tested
 - Operation and maintenance
- Intensive documenting and planning
- Easy to understand and implement
- Identified deliverables and milestones
- Discovering issues in earlier phases should lead to returning to earlier phase!

Waterfall Model – Heavy-Weight Model



Development activities	Teams
Divide the work into stages	A separate team of specialists for each stage
At each stage, the work is passed from one team to another	Some coordination is required for the handoff from team to team - using “documents”
At the end of all of the stages, you have a software product ready to ship	As each team finishes, they are assigned to a new product

Waterfall Model Problems

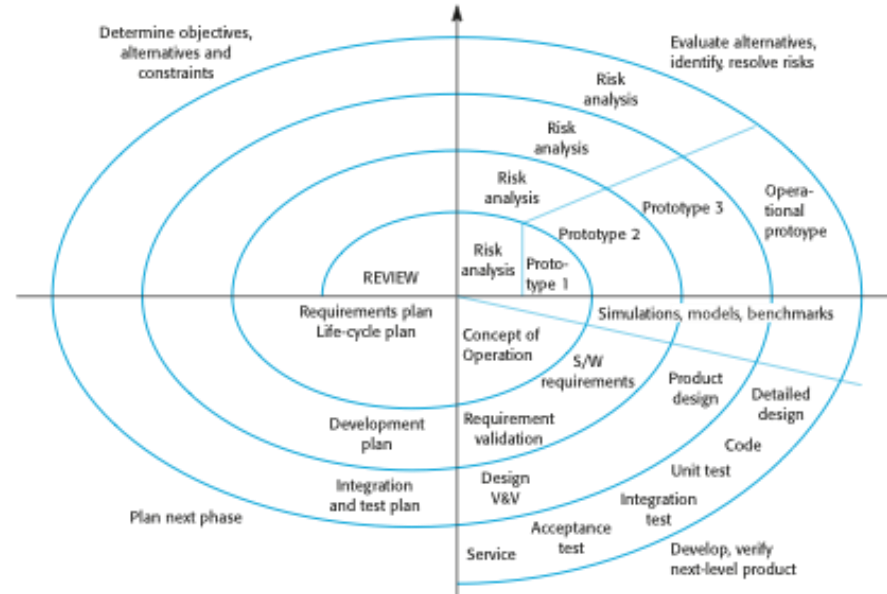
- The **main drawback** of the waterfall model is the **difficulty of accommodating change** after the process is underway
 - A phase has to be complete before moving onto the next phase
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements
 - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process
 - Few business systems have stable requirements
- The waterfall model is mostly used for **large systems engineering projects** where a system is developed at several sites
 - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work

Spiral Model – Heavy-weight Model

- Process is represented as a **spiral** rather than as a sequence of activities with backtracking
- Each **loop** in the spiral represents a **phase** in the process
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required
- **Risks** are explicitly assessed and resolved throughout the process
 - This was the motivation behind developing the Spiral Model - Risk

Spiral Model of the Software Process

- SW is done in iterations, each enhancing previous system
- Each iteration does successively:
 - Objectives/alternatives and constraints
 - Identify and resolve risks
 - Develop and test (next-level product)
 - Plan next iteration
- Spiral model has been very influential in helping people think about **iteration** in SW processes and introducing the **risk-driven approach**
- In practice, however, the model is **rarely** used as published for **practical** SW development



Software Failures – Budget, Schedule, Requirements

Project	Duration	Cost	Failure/Status
e-borders (UK Advanced passenger Information System Programme)	2007 - 2014	Over £ 412m (expected), £742m (actual)	Permanent failure - cancelled after a series of delays
Pust Siebel - Swedish Police case management (Swedish Police)	2011 - 2014	\$53m (actual)	Permanent failure – scraped due to poor functioning, inefficient in work environments
US Federal Government Health Care Exchange Web application	2013 – ongoing	\$93.7m (expected), \$1.5bn (actual)	Ongoing problems - too slow, poor performance, people get stuck in the application process (frustrated users)
Australian Taxation Office's Standard Business Reporting	2010 - ongoing	~\$1 bn (to-date), ongoing	Significant spending on contracting fees (IBM & Fjitsu), significant scope creep and confused objectives

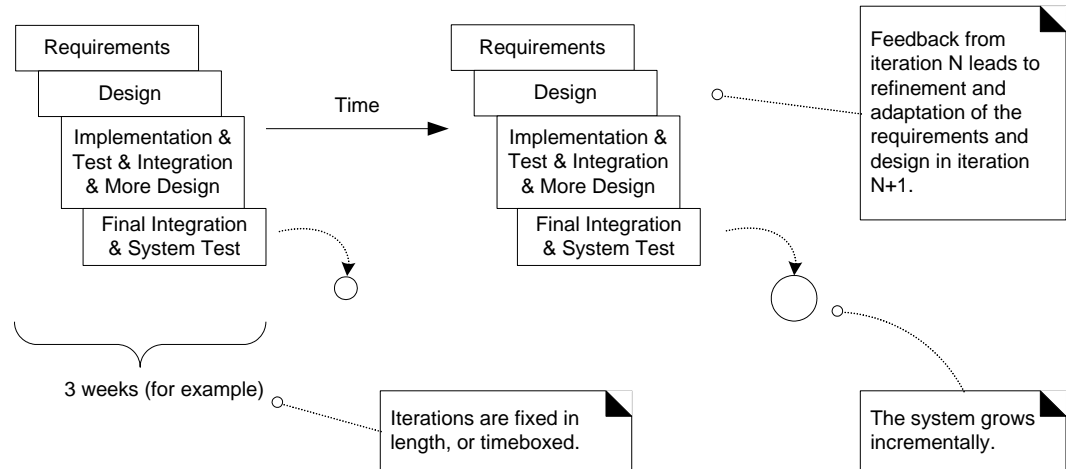
https://en.wikipedia.org/wiki/List_of_failed_and_overbudget_custom_software_projects

Software Evolution

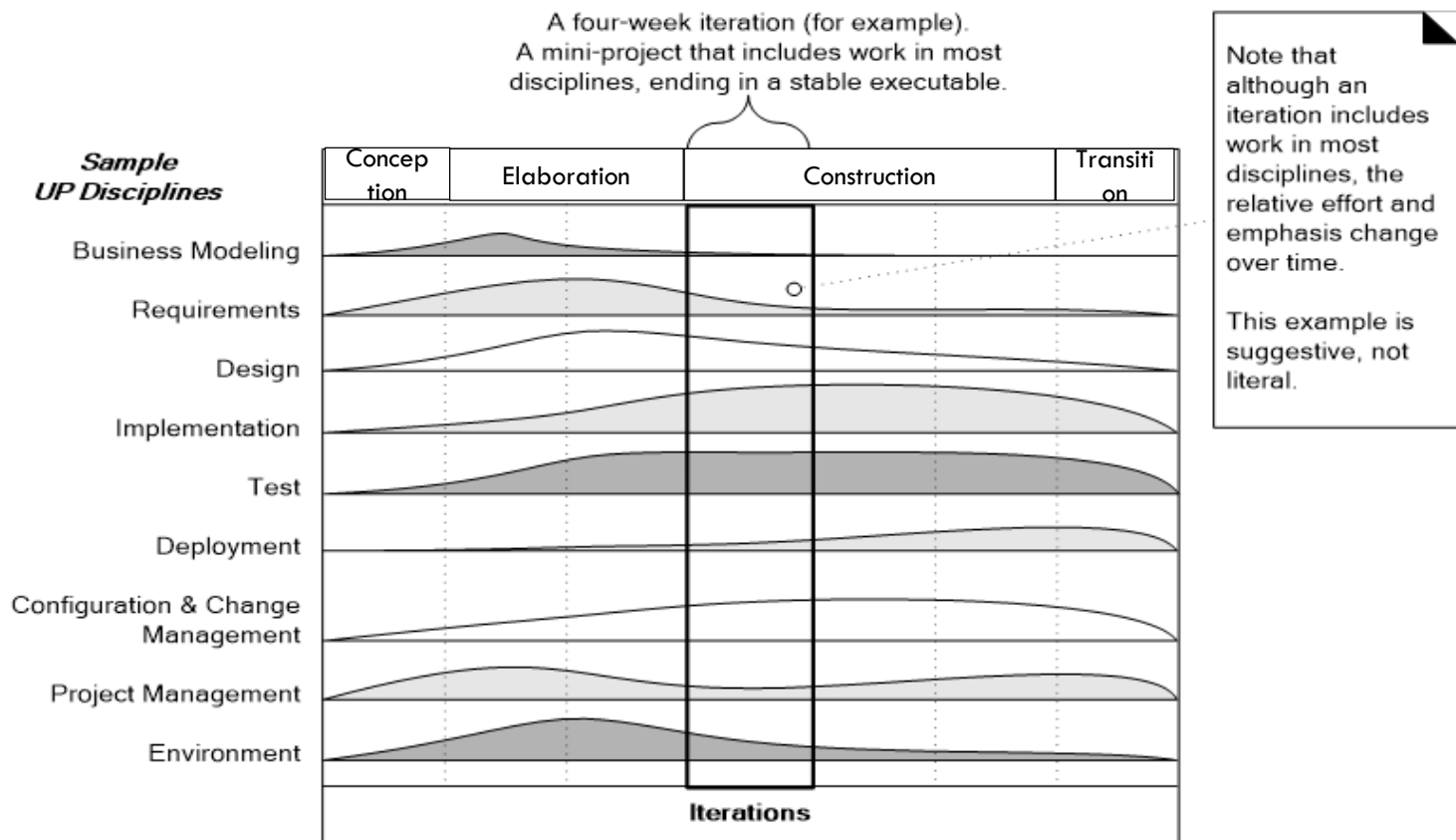
- Software is inherently flexible and can change
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change
- Business software needs to respond to rapidly changing market
 - Time-to-market
- Plan-driven software development processes are not suitable for certain types of SW systems

Rational Unified Process (RUP or UP)

- Software development process utilizing iterative and risk-driven approach to develop OO software systems
- Iterative incremental development
- Iterative evolutionary development



UP Phases and Disciplines



Agile Development Model

Agile manifesto, principles and practices



Project Failure – the trigger for Agility

- One of the primary causes of project failure was the **extended period of time it took to develop a system**
- Costs escalated and requirements changed
- Agile methods intend to **develop systems more quickly with limited time spent on analysis and design**

Why is Agile Development Important?

- The world is a lot different today. A large feature set might only increase costs for the customer
 - There is a constant introduction of new technology
 - New players enter the market,
 - New requirements are added
 - “Small is Beautiful”
 - If we are listening to the customer, we will reduce our chances of being “blindsided” by a smaller, more flexible competitor
 - Anything that helps reduce maintenance costs will contribute to the bottom line

Agile Manifesto (2001) – An Eloquent Statement of Agile Values or Goals

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Agile Manifesto: <http://agilemanifesto.org/>

© 2001, the above authors. This declaration may be freely copied in any form, but only in its entirety through this notice.

Agile Process

- Agile advocates believe:
 - Current software development processes are too heavy-weight or cumbersome
 - Too many things are done that are not directly related to software product being produced
 - Current software development is too rigid
 - Difficulty with incomplete or changing requirements
 - Short development cycles (Internet applications)
 - More active customer involvement needed

Agile Process

- Agile methods are considered
 - **Light-weight**
 - **People-based** rather than Plan-based
- Several agile methods
 - No single agile method
 - Extreme Programming (XP), SCRUM
- No single definition
- Agile Manifesto closest to a definition
 - Set of principles
 - Developed by Agile Alliance

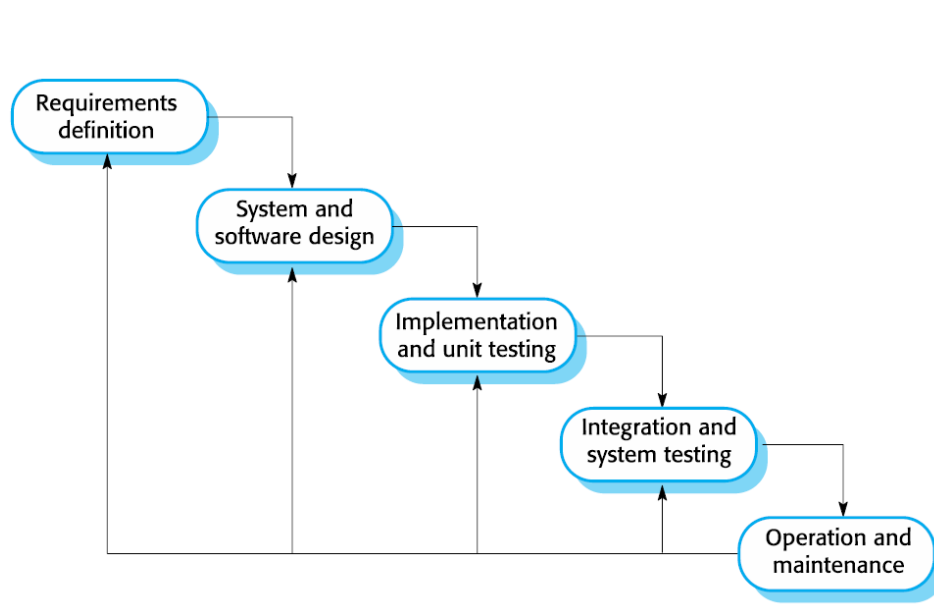
Agile Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software .	5. Build projects around motivated individuals . Give them the environment and support they need, and trust them to get the job done.	9. Continuous attention to technical excellence and good design enhances agility.
2. Welcome changing requirements , even late in development. Agile processes harness change for the customer's competitive advantage.	6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation .	10. Simplicity --the art of maximizing the amount of work not done--is essential.
3. Deliver working software frequently , from a couple of weeks to a couple of months, with a preference to the shorter timescale .	7. Working software is the primary measure of progress .	11. The best architectures, requirements, and designs emerge from self-organizing teams .
4. Business people and developers must work together daily throughout the project.	8. Agile processes promote sustainable development . The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	12. At regular intervals, the team reflects on how to become more effective , then tunes and adjusts its behavior accordingly.

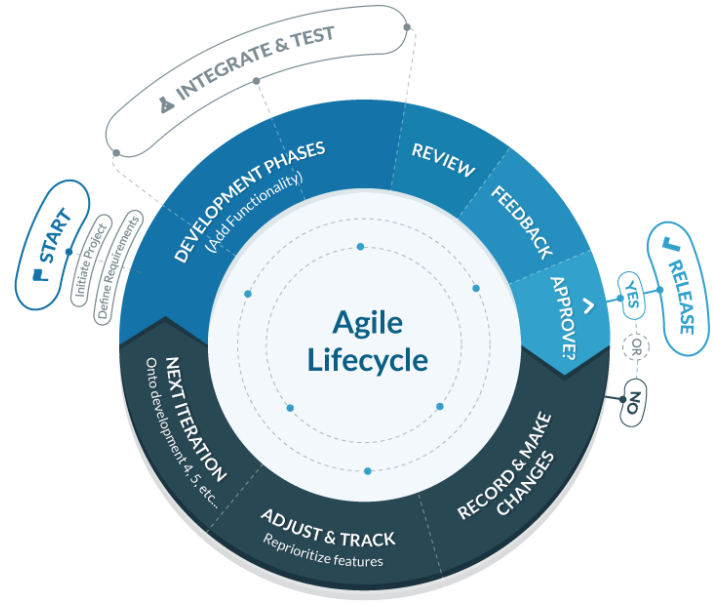
Agile Practices

- There are many Agile practices:
 - short time-boxed iterations
 - continuous integration
 - daily unit testing
 - regular retrospectives
 - direct communication between developers and the customer or a customer surrogate
 - a single list of features and tasks
 - short-term estimation of development tasks
 - information radiators
 - Refactoring
- Will you use every Agile practice? Maybe not... they are not all required
- What is required? Agile values...

Software (Development) Process Models



Waterfall model
plan-driven development



Agile model
Incremental & iterative development

Requirements process

- There is no “standard way” to do requirements in Agile development
 - Could be a normal “Software Requirements Document”
 - But it is better to be more lightweight
 - Based on user stories
 - In each iteration, elaborate a small set of the functional requirements (the high-priority behavior)
 - For some key requirements, create some acceptance tests at the same time as you write the requirements

Agile – questions and challenges?

- **Documentation** – it is still important in an Agile project.
 - If it is the only kind of communication in your project, it isn't good
 - Real working code is more valuable than documents – less ambiguous
 - Documents – easy to leave something out, easy to misinterpret
- **Development plans** – also important in an Agile project
 - the format of an Agile development schedule is a bit different from a conventional project plan.
 - Development plan includes “iterations”
 - Each iteration gives the team a chance to incorporate what they learn, rather than just following a non-adaptive plan

Agile only works with the best developers

- Every project needs at least one experienced and competent lead person. (Critical Success Factor)
- Each experienced and competent person on the team permits the presence of 4-5 “average” or learning people.
- With that skill mix, agile techniques have been shown to work many times.

Agile won't work for all projects

- There are claims that agile methods are probably best suited to small/medium-sized business systems
- Agile is an **attitude prioritizing**:
 - Project evaluation based on delivered code
 - Rapid feedback
 - People as a value center
 - Creativity in overcoming obstacles
- Not every team
 - Values the Agile value set
 - Can set up the needed trust and communication

Mutual Respect

- *Developer*: “Testers are failed programmers, they shouldn’t be called engineers”
- *Tester*: “Developers are only able to produce bugs, the world would be better with more testers and less developers”
- *Business Analysts*: “I don’t know why I bother talking to testers and developers, they are total idiots”

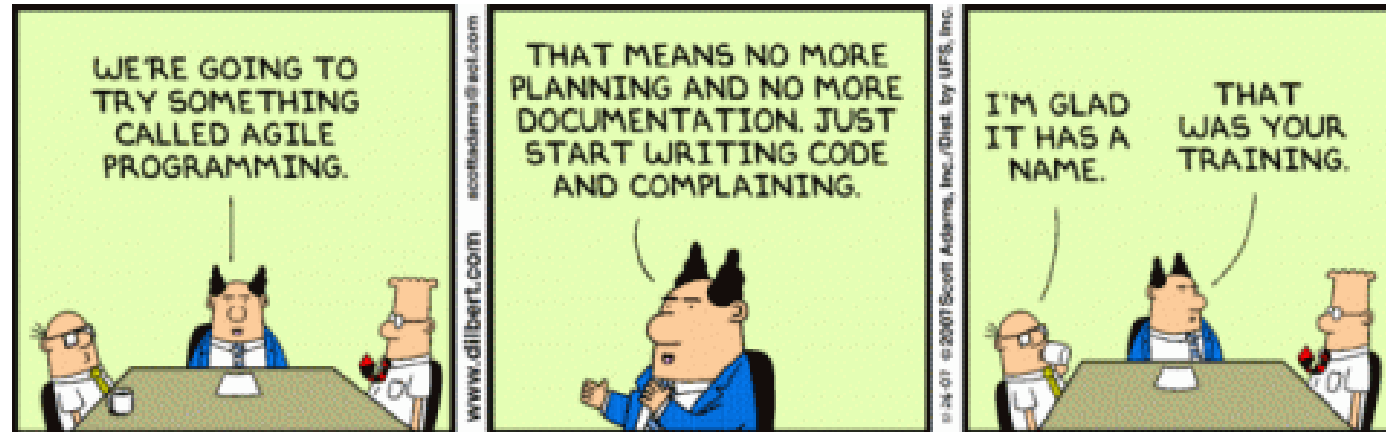
Not My Responsibility

- *Developer*: “It works in our environments, it’s operations responsibility to make it work in production”
- *Tester*: “Listen, it worked in User Acceptance Testing, it must be a configuration issue, or a missing firewall hole and nothing I could have spotted during testing...”
- *Customer*: “Hello! Nothing works here...”

How hard is it to be Agile?

- “Don’t do Agile, be Agile”
 - Just doing “development in iterations” isn’t enough
- Agile Development is about:
 - Keeping the process lightweight
 - Making real progress in each iteration
 - Communicating – face-to-face when possible
 - Actively gathering customer input – early and often
 - Being willing to make minor changes to your process

Thinking Agile!



<https://www.eylean.com/blog/2016/04/best-agile-comic-strips/>

References

- Armando Fox and David Patterson 2015. Engineering Software as a Service: An Agile Approach Using Cloud Computing (1st Edition). Strawberry Canyon LLC
- Andrew Stellman, Margaret C. L. Greene 2014. Learning Agile: Understanding Scrum, XP, Lean and Kanban (1st Edition). O'Reilly, CA, USA.
- Ivar Jacobson, Grady Booch, and James Rumbaugh. 1999. The Unified Software Development Process. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Ian Sommerville. 2016. Software Engineering (10th ed.) Global Edition. Pearson, Essex England