Software Design and Construction 1

SOFT2201 / COMP9201

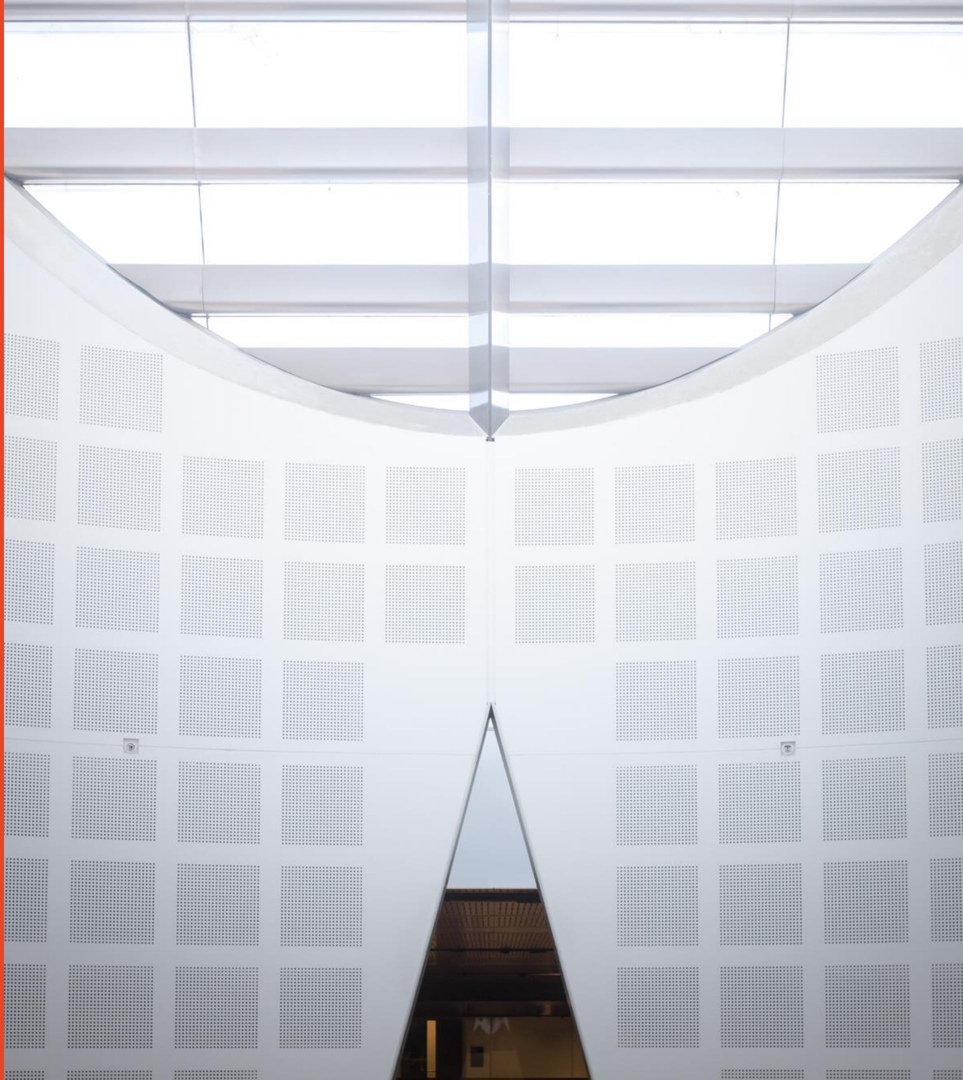
Introduction

Software Modeling

Dr. Basem Suleiman

School of Information Technologies

THE UNIVERSITY OF SYDNEY



Copyright Warning

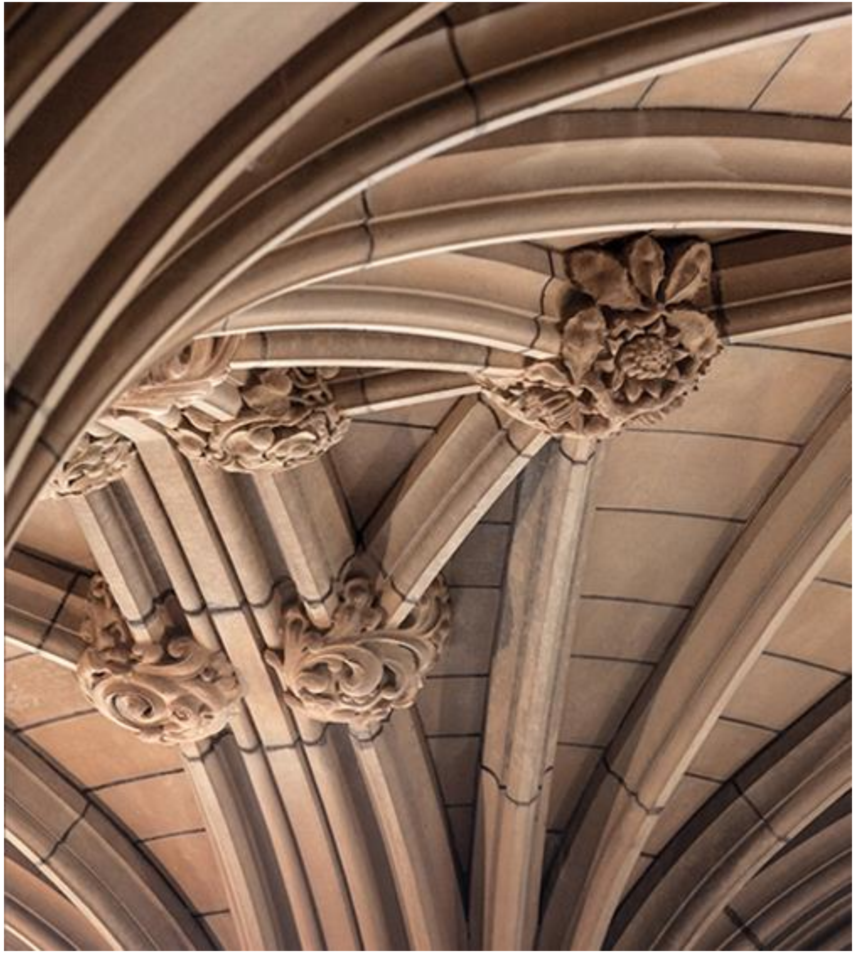
Agenda

- Administrivia/Introduction
- Software Engineering
  - Why, What
- Software Modeling
  - What, Why and How
- The Unified Process

Course Administration & Introduction

To help you get to grips with what's coming

THE UNIVERSITY OF SYDNEY



About the Teaching Staff

Course Coordinator and Lecturer:
Dr. Basem Suleiman (basem.suleiman@Sydney.edu.au)

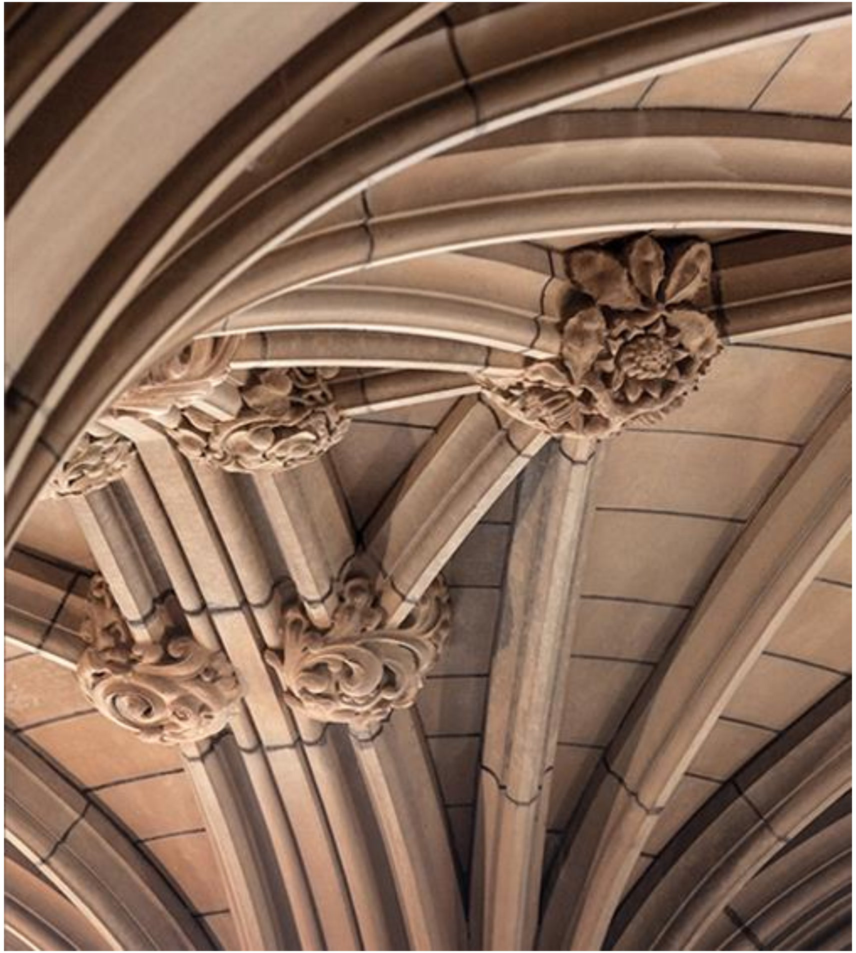
Teaching Associates:
Dr. Farnaz Farid and Dr. Hamza Osop

Tutors:
Greg McLellan (gmcl8868@uni.sydney.edu.au)
Joseph Godbehere (jgod5665@uni.sydney.edu.au)
Tianzou (David) Wang (tianzou.wang@Sydney.edu.au)

Induction

WHS, Assistance, Support and Policies

THE UNIVERSITY OF SYDNEY

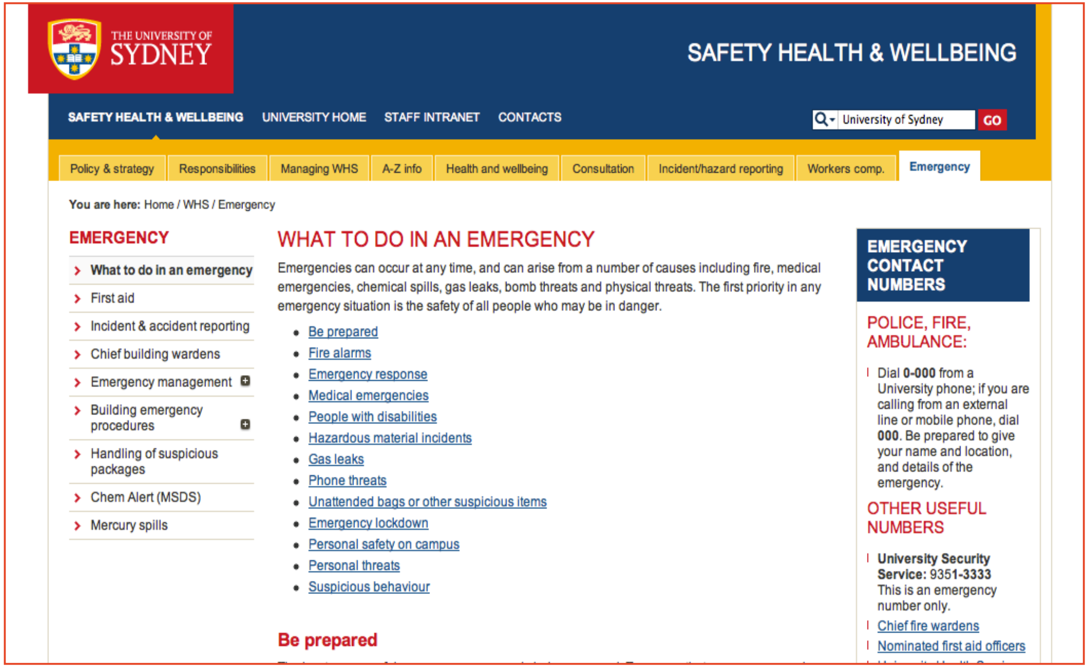


General Housekeeping – Use of Labs

- 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



You are here: Home / WHS / Emergency

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

Be prepared

EMERGENCY CONTACT NUMBERS

POLICE, FIRE, AMBULANCE:

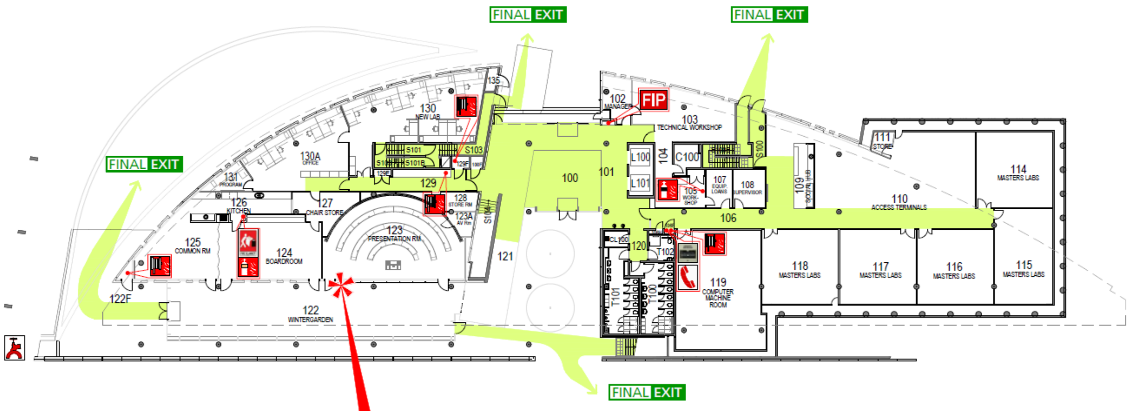
- Dial **0-000** from a University phone; if you are calling from an external line or mobile phone, dial **000**. Be prepared to give your name and location, and details of the emergency.

OTHER USEFUL NUMBERS

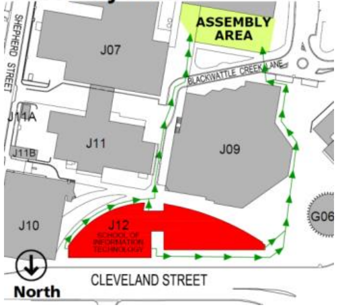
- **University Security Service:** 9351-3333 This is an emergency number only.
- Chief fire wardens
- Nominated first aid officers

EMERGENCIES

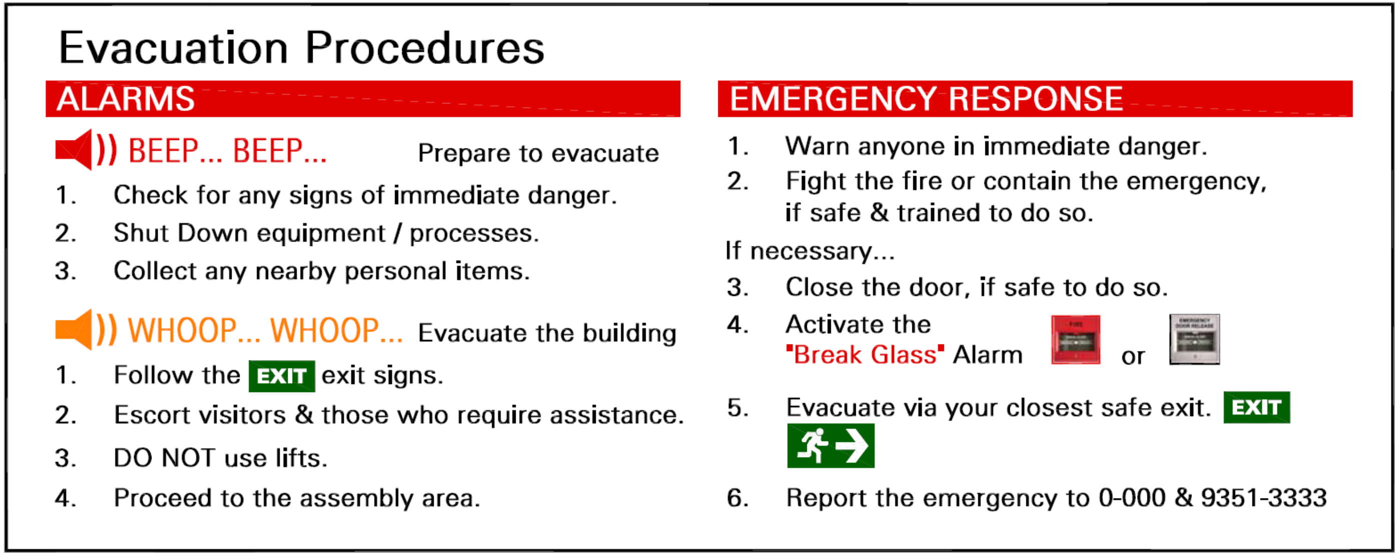
WHERE IS YOUR CLOSEST SAFE EXIT ?



Assembly Area



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.

WHOOP... WHOOP... 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

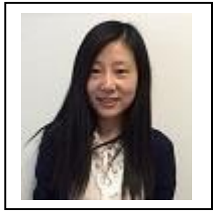
School of IT Safety Contacts

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

Do you have a disability?

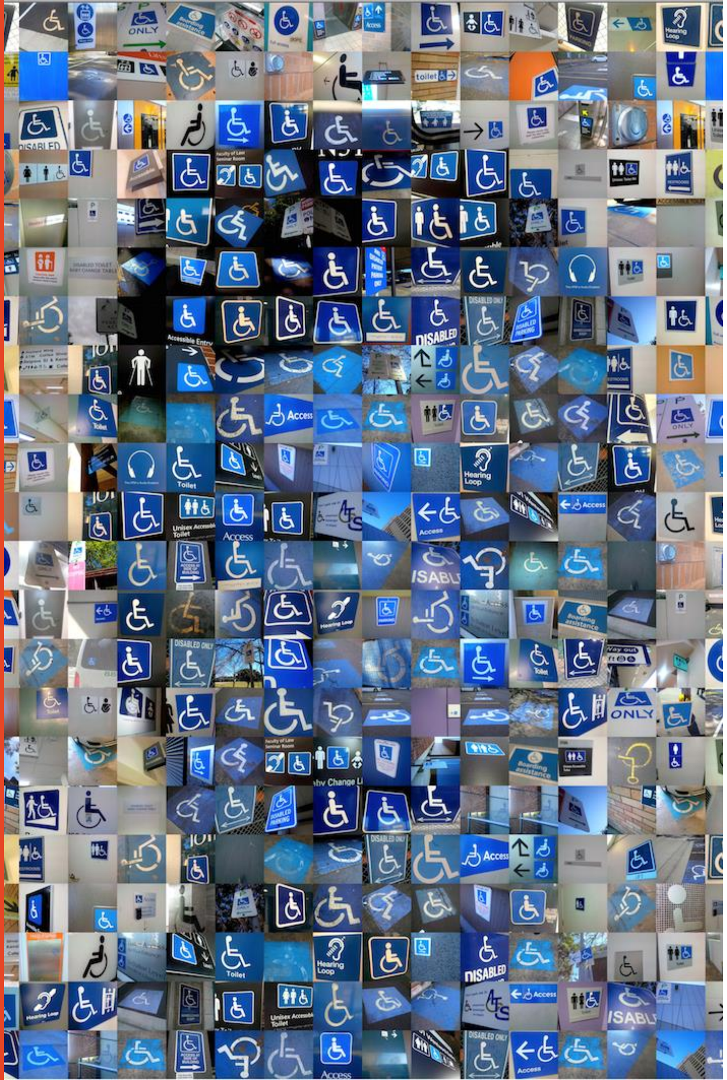
You may not think of yourself as having a ‘disability’ but the definition under the **Disability Discrimination Act (1992)** is broad and includes temporary or chronic medical conditions, physical or sensory disabilities, psychological conditions and learning disabilities.

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



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

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)

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

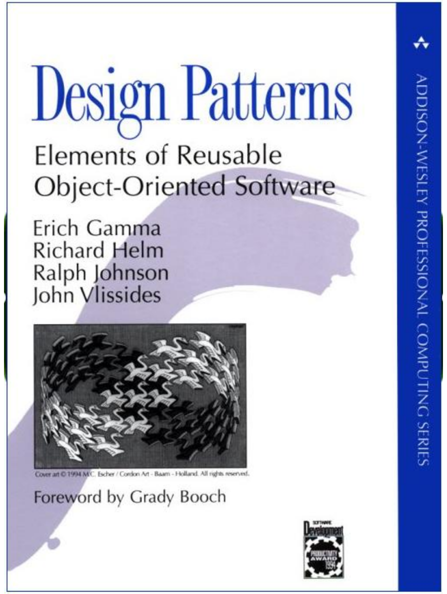
- This course has the following prerequisites:
  - INFO1113 OR INFO1103 OR INFO1105 OR INFO1905

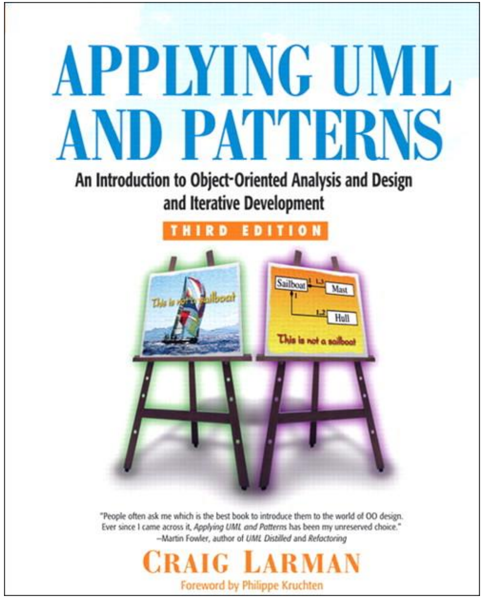
- This means that we expect students who enroll in this course to be:
  - Confident Java programmers
  - Familiar with data-structures

- Prohibitions
  - INFO3220 OR COMP9201

Main Resources

- We recommend the following textbooks


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


APPLYING UML AND PATTERNS
An Introduction to Object-Oriented Analysis and Design and Iterative Development
THIRD EDITION
CRAIG LARMAN
Foreword by Philippe Kruchten

Lab / Tutorial Work

- Labs are available on Canvas
- 3-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
- Respect your classmates
- Tutors will supervise your learning, provide you guidance
  - Not to debug your code, or solve the problems for you

Tools you will use

- Your coding will be in Java 9
- You will use Eclipse for writing your code
- You will use JavaFX for designing your GUIs
- You may use subversion or git or other version control systems
- Other?

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
Quizzes	Weekly (3-12)	Online Quizzes (Canvas)	10%
UML Modelling Assignment	Week 5	Individual submission on Canvas	5%
Design Patterns Assignment – Stage 1	Week 8	Individual submission on Canvas	10%
Design Patterns Assignment – Stage 2	Week 11	Submit on Canvas	20%
Exam	Exam period	Individual exam	50%

*Check latest updates of the unit outline on CUSP

Online Quiz

- Online Quizzes are multiple-choice quizzes, 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.
- Quiz covers recent tutorials and lectures
- When: Weekly during your lab class in week 3 to 12
- **Duration:** 10 minutes
- Marks: 10%

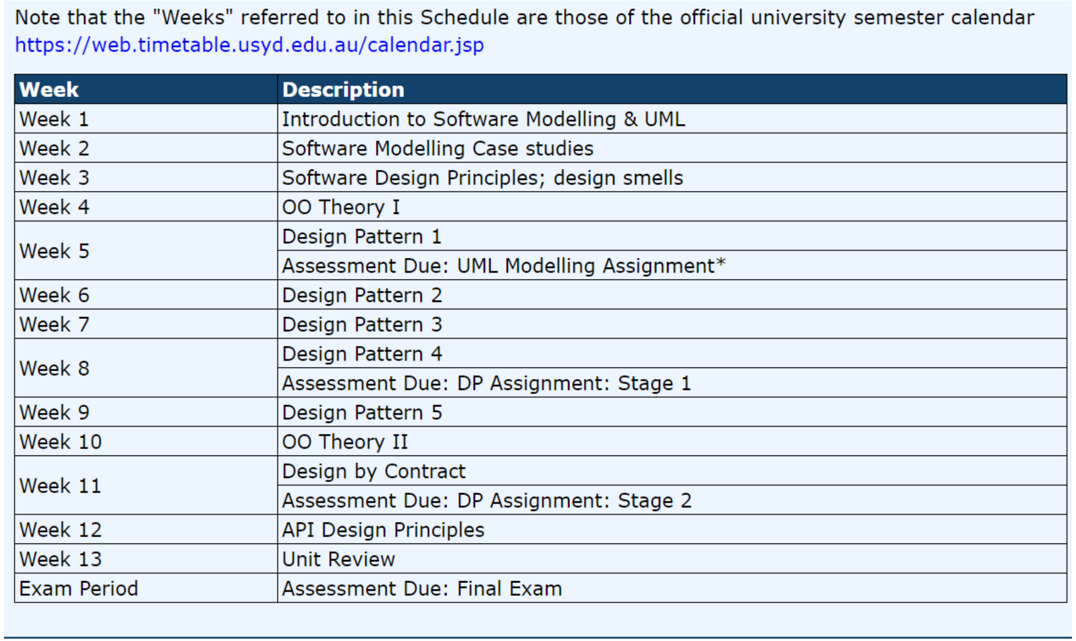
Assignment

- The Assignment consists of three stages:

  1. Design a computer game using UML diagrams
  2. Implement the computer game using Java with JavaFX; write an essay about the OO design of your design
  3. Extend somebody's else code; write essays about the OO design of the receiver code and your extensions

Topics Overview

- Always check on CUSP for the latest version*



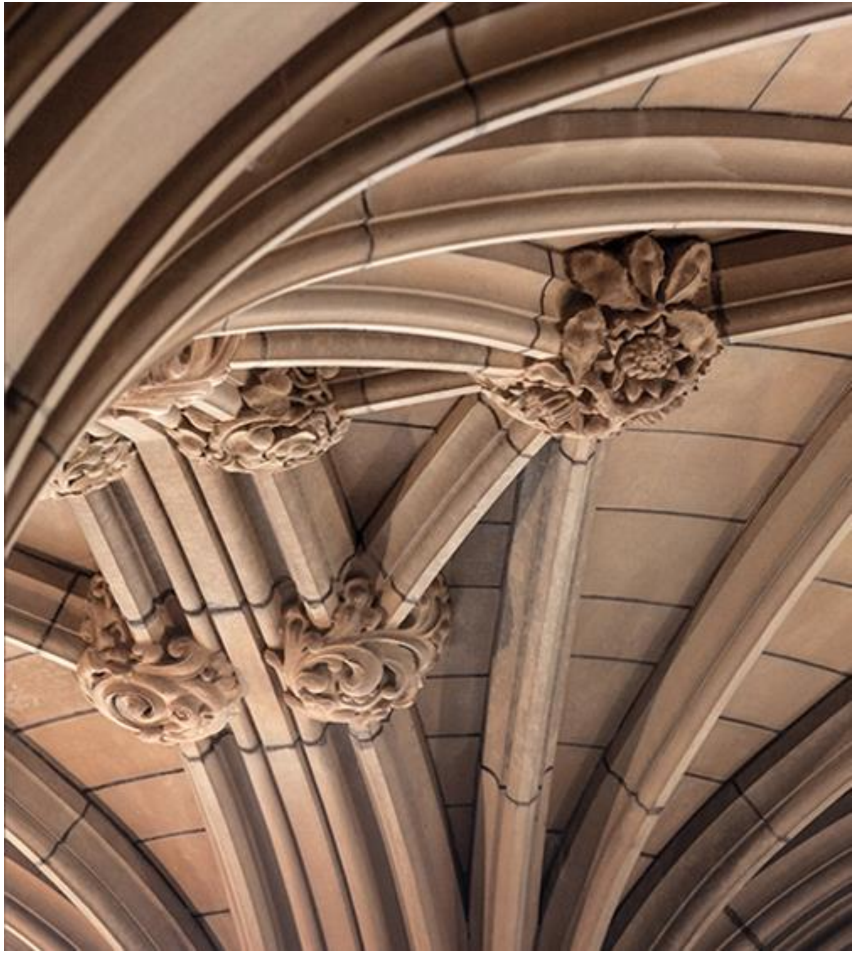
Note that the "Weeks" referred to in this Schedule are those of the official university semester calendar https://web.timetable.usyd.edu.au/calendar.jsp

Week	Description
Week 1	Introduction to Software Modelling & UML
Week 2	Software Modelling Case studies
Week 3	Software Design Principles; design smells
Week 4	OO Theory I
Week 5	Design Pattern 1
	Assessment Due: UML Modelling Assignment*
Week 6	Design Pattern 2
Week 7	Design Pattern 3
Week 8	Design Pattern 4
	Assessment Due: DP Assignment: Stage 1
Week 9	Design Pattern 5
Week 10	OO Theory II
Week 11	Design by Contract
	Assessment Due: DP Assignment: Stage 2
Week 12	API Design Principles
Week 13	Unit Review
Exam Period	Assessment Due: Final Exam

*SOFT2201: Software Construction and Design 1 (2018 - Semester 2)

*COMP9201: Software Construction and Design 1 (2018 - Semester 2)

Why Software Engineering?



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/year[1]
  - Total value added GDP in the US[2]: $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 respectively[3]
- Time to market
  - Deliver software on-time

[3] Chaos Report, Standish group Report, 2014

Software Failure - US Northeast Blackout[4]

What happened?

- Power outage of regions in US and Canada
- Power restored - took up to one week
- ~55 million people in 13 major cities
- Significant impact on people’s life, transport, communication, industry

Why did it happen?

- Bug in the alarm system at the control room of FirstEnergy corporation
- Operators did not re-distribute power after overloaded transmission lines
- A “race condition” in the energy management system software
  - affecting the order of operations in the system

[4] https://en.wikipedia.org/wiki/Northeast_blackout_of_2003

Software Failure - Ariane 5 Disaster[5]

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 Failure[6]

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://erichmusick.com/writings/technology/1992-london-ambulance-cad-failure.html

Software Project Failures

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	~$1bn (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 Engineering - No Silver Bullet[7]

No Silver Bullet - Essence and Accidents in Software Engineering

“There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity.” ! - Frederick P. Brooks

- **Essence:** difficulties inherent (or intrinsic) in the nature of SW
- **Accidents:** difficulties related to the production of software
- *Most techniques attack the accidents of software engineering*

[7] No Silver Bullet - Essence and Accident in Software Engineering - http://www.itu.dk/people/hesj/BSUP/artikler/no-silver-bullit.pdf

Software Engineering - Essence[7]

- **Complexity**
  - Many diverse software entities - interactions increase exponentially
  - Intrinsic complexity cannot be abstracted - aircraft software, air traffic control
- **Conformity**
  - Arbitrary changes from environment (people, systems) - no unifying principle
- **Changeability**
  - Changing a building model vs. a software
  - Stakeholders understanding of software changes
- **Invisibility**
  - Software is intangible (invisible)
  - Building model vs software models (UML - 13 diagram types)

What is Software Engineering?

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

What is the difference between SW Developers and SW Engineers?

Form groups of three and discuss for 5min

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 is 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

- Theories, methods, tools, techniques and approaches
  - Solve concrete SWEN problems
  - Increase productivity of the engineering process
  - Produce effective software
  - Produce efficient software
  - Control social and technical aspects in the development process
  - Manage complexity, changeability, invisibility and conformity

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 Engineering Fundamentals

- Software processes for managing and developing of SW systems
  - Waterfall vs. Incremental and agile software development
- Attributes of good software system
  - Maintainability, dependability and security, efficiency and acceptability
- Importance of Dependability and Performance
- Need for specifications and requirements
- Software reuse to save costs
  - Careful consideration – Ariane 5 reused software from Arian 4!

Software Engineering Body of Knowledge

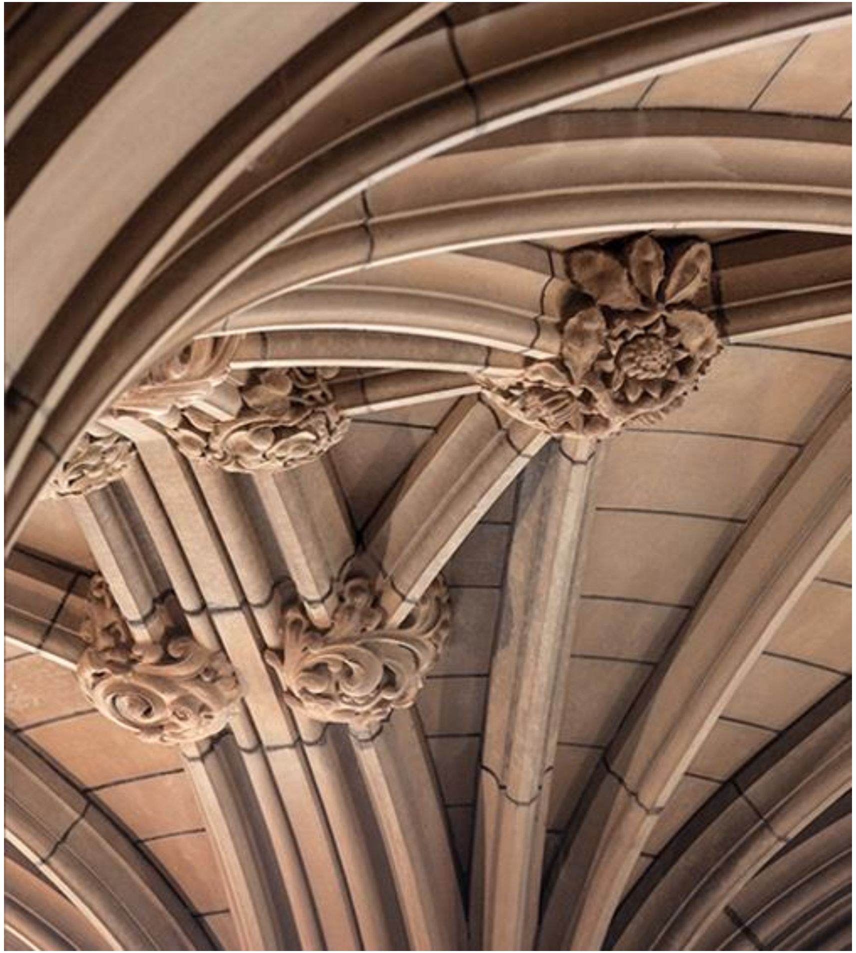
- Software Requirements
- **Software Design / Modelling**
- **Software Construction**
- Software Testing
- Software Maintenance
- Software Configuration Management
- Software Engineering Process
- Software Engineering Tools and Methods
- Software Quality

IEEE computer society

SWEBOK®

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

Software Design/Modelling & Construction



THE UNIVERSITY OF SYDNEY

Software Modelling

- The process of developing conceptual models of a system at different levels of abstraction
    - Fulfil the defined system requirements
    - Focus on important system details to deal with complexity
- Object-oriented design approach
    - Concepts in the domain problem are modelled as interacting objects
- Using graphical notations (e.g., UML) to capture different views or perspectives

Software Modelling – The Art of Abstraction

- Conceptual process to derive general rules and concepts from concrete information in a certain context

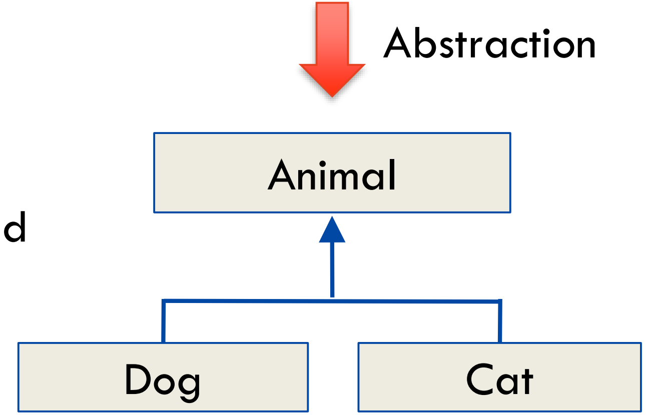
- Analysis and understanding of domain-specific problems
  - Decompose large problems into smaller understandable piece
  - Language required to break down complexity, i.e., find abstractions

- SW models with different levels of abstraction
  - Focus on certain details in each phase of the SW development process

- SW models with different views/perspectives
  - Time, structural, requirements, etc.

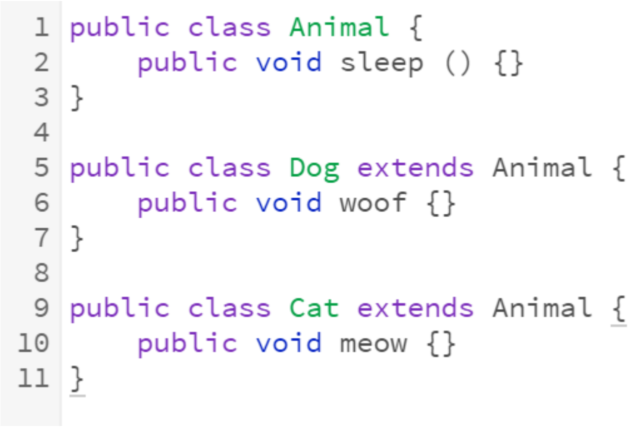
Software Abstraction – Example

Real world



Abstraction

Software world



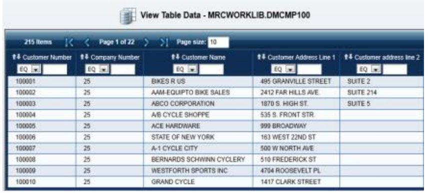
```java
public class Animal {
    public void sleep () {}
}

public class Dog extends Animal {
    public void woof {}
}

public class Cat extends Animal {
    public void meow {}
}
```

Cat http://s1.thingpic.com/images/ZP/L4FtpQYKNZzCXV8r34PWhqCF.jpeg
Dog https://s7d2.scene7.com/is/image/PetSmart/SV0401_CATEGORY_HERO-Dog-Dog-20160818?$SV0401$

Data Abstraction – Example

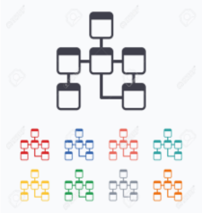


View Table Data - MRCWORKLIB.DMCMP100

215 Items | Page 1 of 22 | Page size: 10

Customer Number	Company Number	Customer Name	Customer Address Line 1	Customer address line 2
EQ	EQ	EQ	EQ	EQ
100001	25	BIKES R US	495 GRANVILLE STREET	SUITE 2
100002	25	AAM-EQUIPTO BIKE SALES	2412 FAR HILLS AVE	SUITE 214
100003	25	ABCO CORPORATION	1870 S. HIGH ST.	SUITE 5
100004	25	A/B CYCLE SHOPPE	535 S. FRONT STR.	
100005	25	ACE HARDWARE	999 BROADWAY	
100006	25	STATE OF NEW YORK	163 WEST 22ND ST	
100007	25	A-1 CYCLE CITY	500 W NORTH AVE	
100008	25	BERNARDS SCHWINN CYCLERY	510 FREDERICK ST	
100009	25	WESTFORTH SPORTS INC	4704 ROOSEVELT PL	
100010	25	GRAND CYCLE	1417 CLARK STREET	

View level



Logical level (conceptual data model)

Physical level (data model)

UML Principles

- Graphical notations to visually specify and document design artifacts software systems using OO concepts
  - Industry standard managed by Object Management Group (OMG)
  - Is not OO Analysis and Design (OOA&D) or method!
  - Is a language for OOA&D communicating visual models of the software being designed
  - Is not a skill, but how to design software with different level of abstractions
  - Many software diagraming tools, hand sketches are good too
- Combines techniques from data modeling (ER diagrams), business modeling (workflows), object and component modeling
- Capture system activities (jobs), system components and its interaction, software entities and its interaction, run-time behavior, external interfaces

UML Principles (Cont.)

- UML is not "Silver Bullet"
  - No tool/technique in software will make dramatic order-of-magnitude difference in productivity, reliability or simplicity
  - Design knowledge and thinking in objects is critical

- Three ways to apply UML
  - Sketching to explore part of the problem or solution space
  - Blueprint: detailed design diagrams for:
    - Reverse engineering to visualize and understand existing code
    - Forward engineering (code generation)
  - Programming language (Model Driven Engineering): executable specification of a system

- "Agile Modeling" emphasizes UML as a sketch
  - Not waterfall mindset

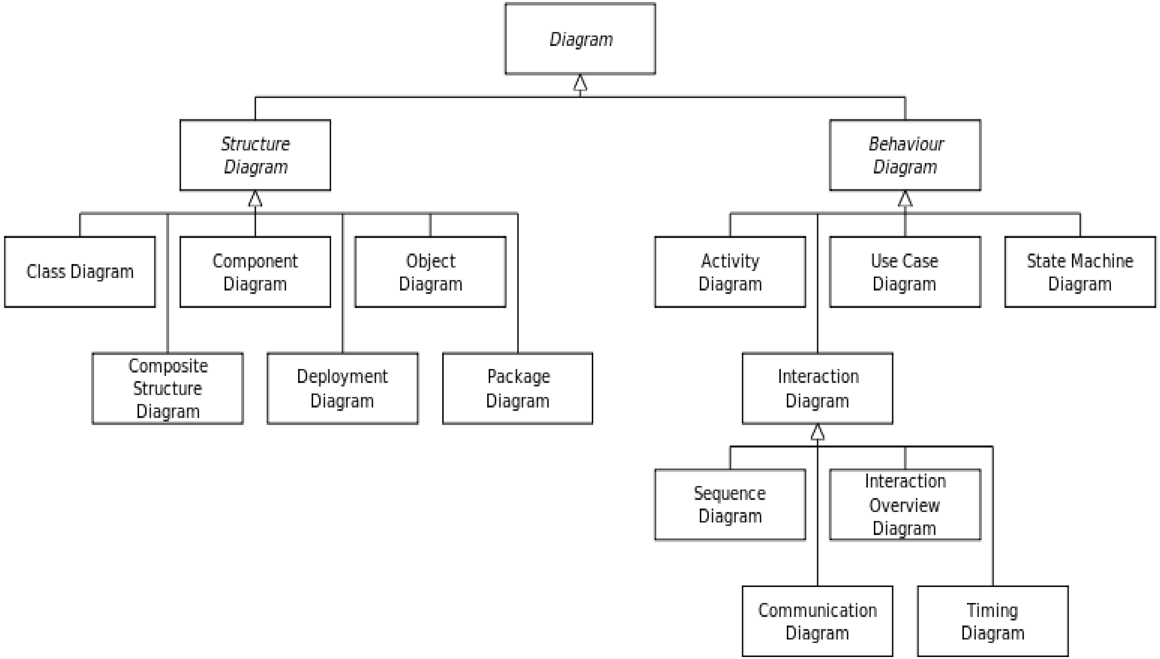
UML Diagrams

Structural (static) View

- Static structure of the system (objects, attributes, operations and relationships)

Behavioural (dynamic) View

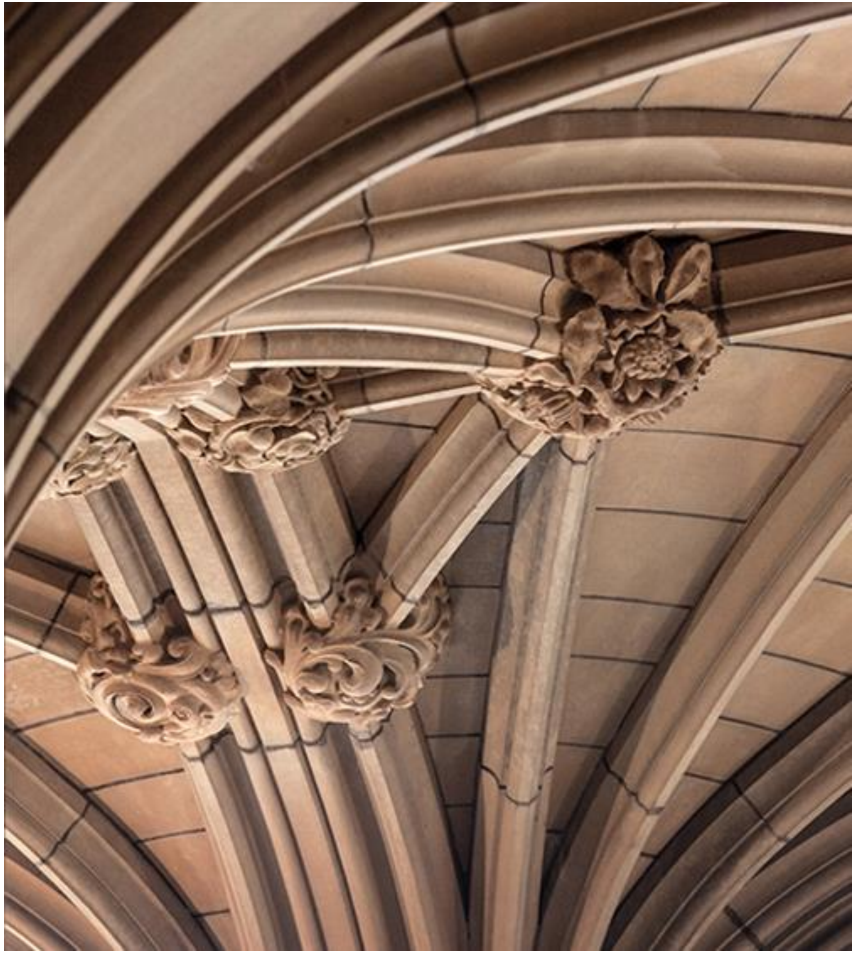
- Dynamic behavior of the system (collaboration among objects, and changes to the internal states of objects)
- Interaction (subset of dynamic view) - emphasizes flow of control and data



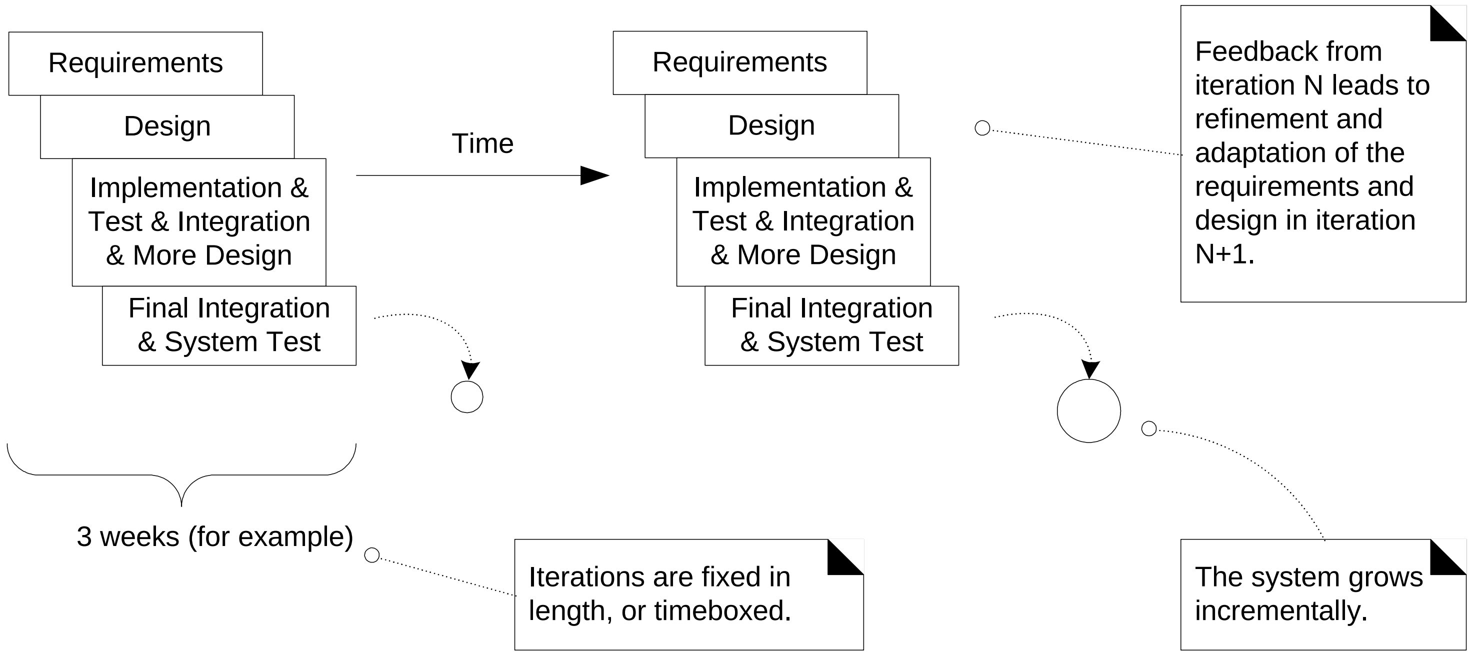
https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/UML

The Rational Unified Process

Iterative, Evolutionary and Agile



Iterative and Evolutionary Development


Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
Time
Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
3 weeks (for example)

Feedback from iteration N leads to refinement and adaptation of the requirements and design in iteration N+1.

Iterations are fixed in length, or timeboxed.

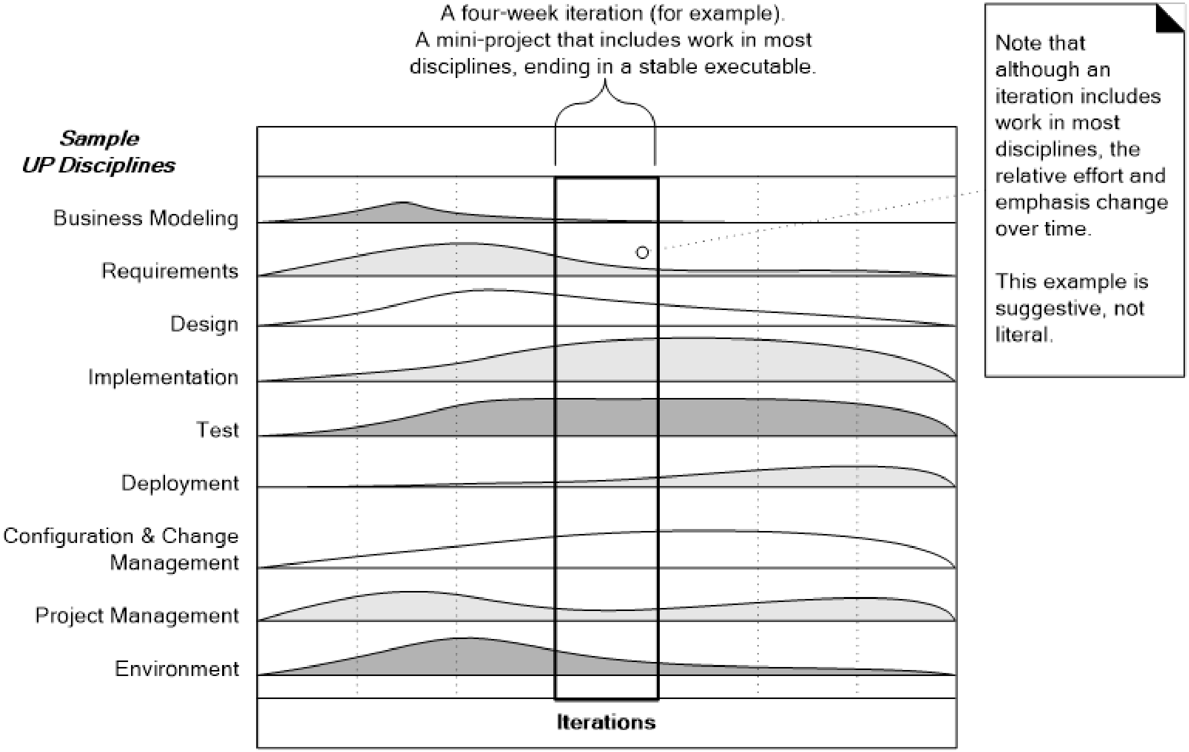
The system grows incrementally.

Rational Unified Process (UP)

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

- Iterative incremental development

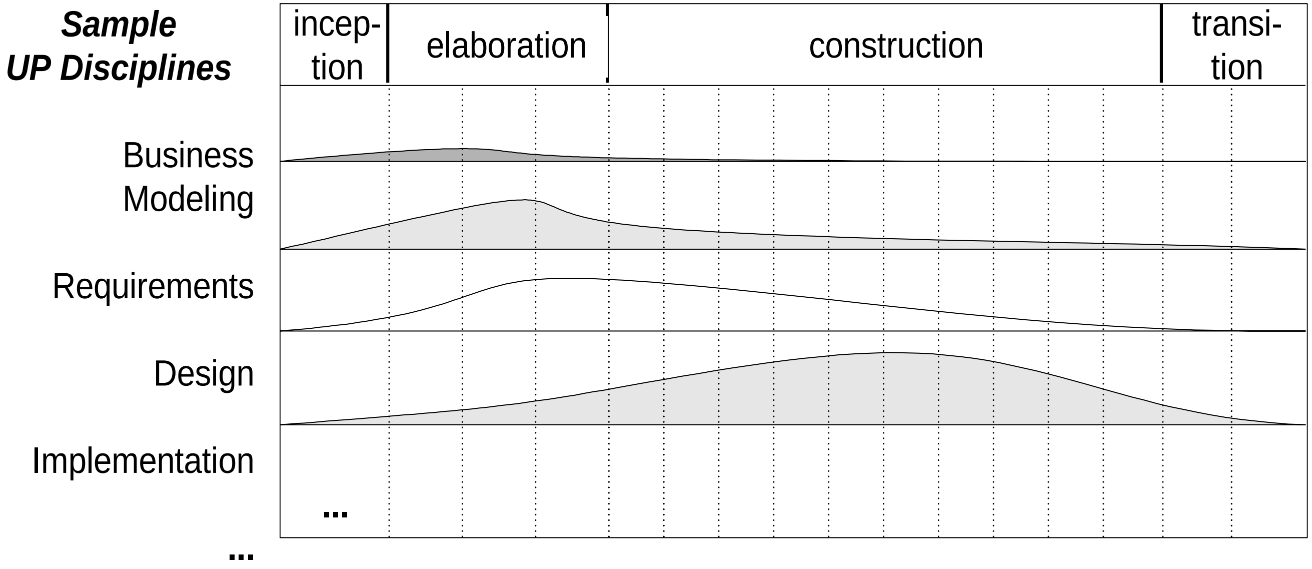
- Iterative evolutionary development


A four-week iteration (for example).
A mini-project that includes work in most disciplines, ending in a stable executable.
Sample UP Disciplines
Business Modeling
Requirements
Design
Implementation
Test
Deployment
Configuration & Change Management
Project Management
Environment
Iterations

Note that although an iteration includes work in most disciplines, the relative effort and emphasis change over time.

This example is suggestive, not literal.

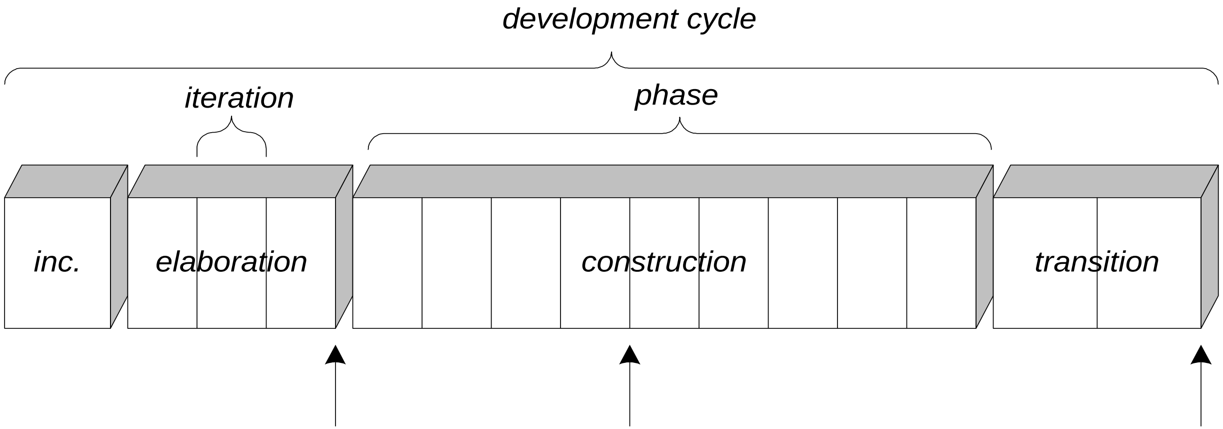
UP Phases and Disciplines



The relative effort in disciplines shifts across the phases.

This example is suggestive, not literal.

UP Phases and Disciplines



development cycle

iteration

phase

inc.

elaboration

construction

transition

milestone

An iteration end-point when some significant decision or evaluation occurs.

release

A stable executable subset of the final product. The end of each iteration is a minor release.

increment

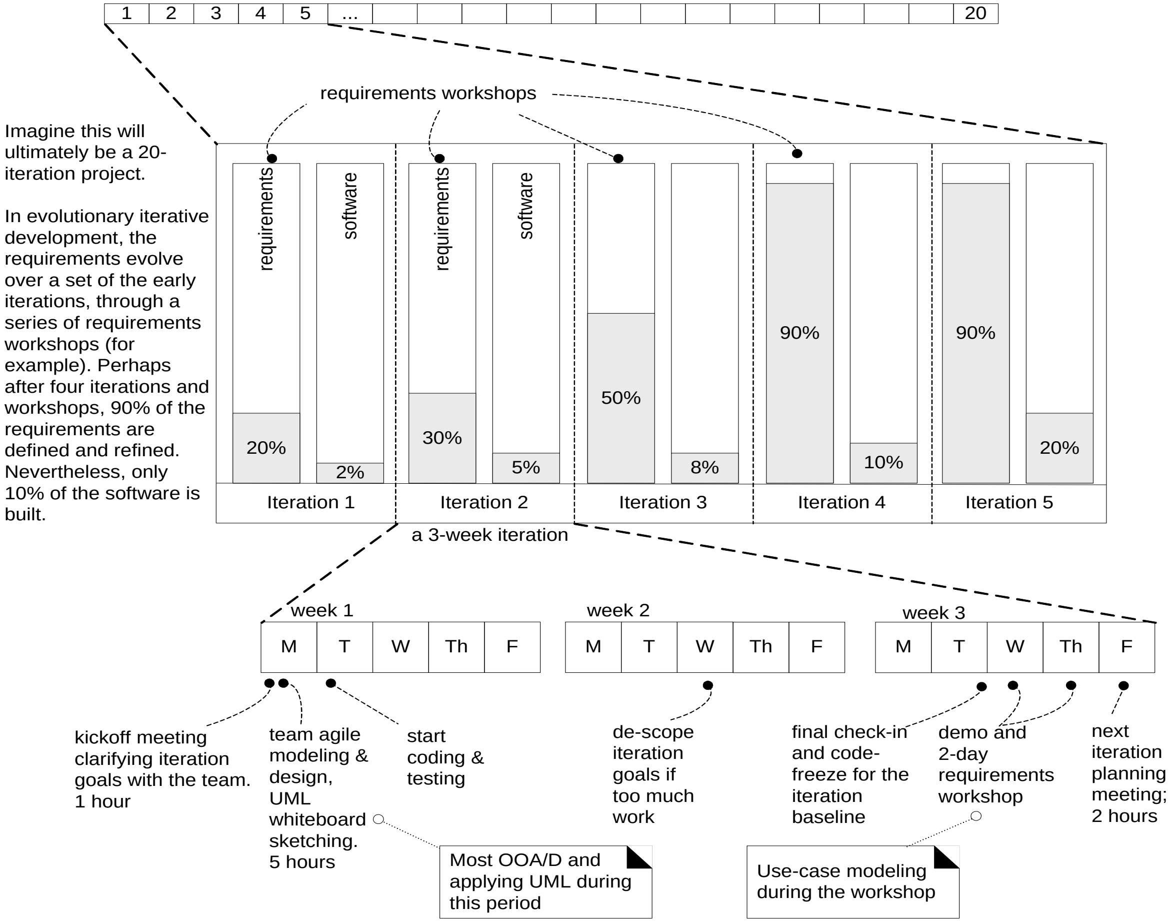
The difference (delta) between the releases of 2 subsequent iterations.

final production release

At this point, the system is released for production use.

UP - Example

- Iterative and evolutionary analysis and design
  - Firs five iterations of 20
  - 3-week iteration

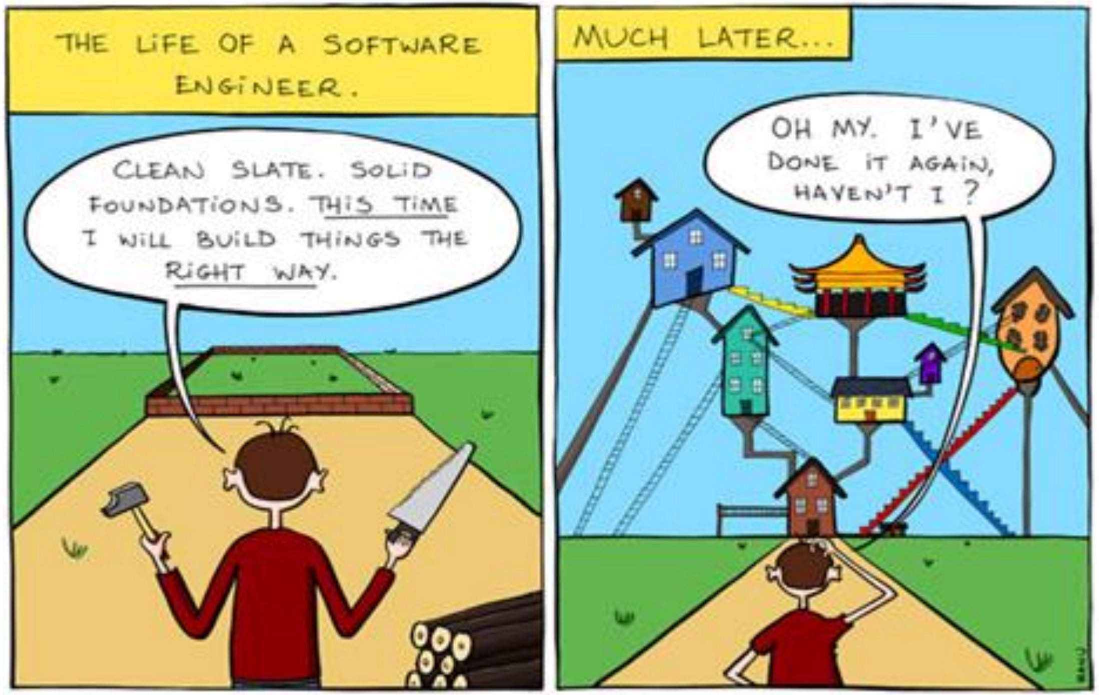


1
2
3
4
5
...
20
requirements workshops
Imagine this will ultimately be a 20-iteration project.
In evolutionary iterative development, the requirements evolve over a set of the early iterations, through a series of requirements workshops (for example). Perhaps after four iterations and workshops, 90% of the requirements are defined and refined. Nevertheless, only 10% of the software is built.
requirements
software
20%
2%
30%
5%
50%
8%
90%
10%
90%
20%
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
a 3-week iteration
week 1
week 2
week 3
M
T
W
Th
F
kickoff meeting clarifying iteration goals with the team. 1 hour
team agile modeling & design, UML whiteboard sketching. 5 hours
start coding & testing
de-scope iteration goals if too much work
final check-in and code-freeze for the iteration baseline
demo and 2-day requirements workshop
next iteration planning meeting; 2 hours
Most OOA/D and applying UML during this period
Use-case modeling during the workshop

Software Construction / Implementation

- Realization of design to produce a working software system
  - Meet customer requirements
- Design and implementation activities often interleaved
  - Agile development to accommodate for changes
- Object-Oriented design and Implementation model
  - Encapsulation
  - Abstraction
  - Reuse
  - Maintenance

Software Engineering


THE LIFE OF A SOFTWARE ENGINEER.
CLEAN SLATE. SOLID FOUNDATIONS. THIS TIME I WILL BUILD THINGS THE RIGHT WAY.
MUCH LATER...
OH MY. I'VE DONE IT AGAIN, HAVEN'T I ?

http://themetapicture.com/media/funny-software-engineer-comic.jpg

References

- Craig Larman. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

- Ian Sommerville. 2016. Software Engineering (10th ed.) Global Edition. Pearson, Essex England