MONASH University

MONASH
INFORMATION
TECHNOLOGY

Introduction to Unit, 2019 S1

FIT2099: Object-Oriented Design and Implementation

GROUP OF EIGHT AUSTRALIA

MONASH University

Census Fail

http://www.abc.net.au/news/2016-11-25/ibm-to-pay-over-$30m-in-compensation-for-census-fail/8057240

Why are we here?

MONASH University

Ultranet fail

https://www.theage.com.au/national/victoria/plug-pulled-on-schools-disastrous-ultranet-computer-system-20140419-36xse.html

MONASH University

Sync ‘n’ Slump

The 62-page report also revealed that the F-35 is temperamental when ground crew plug their Panasonic Toughbook diagnostic laptops into the aircraft and sync them: “In many instances, maintainers must attempt to synch several PMAs [portable maintenance aids – the laptops] with an aircraft before finding one that will successfully connect.”

https://www.theregister.co.uk/2017/01/12/f35_alis_software_delayed/

MONASH University

Malaysian Media Attack

"Ahead of Malaysia's elections on Sunday, independent online media say they are being targeted in Internet attacks which filter content and throttle access to websites, threatening to deprive voters of their main source of independent reporting."

http://www.reuters.com/article/us-malayasia-election-online-idUSBRE94302I20130504

Note: this is a software problem with political implications, but it is still also a software development and operations issue.

- How can we design systems that meet customer goals of being resistant to these kinds of attacks?

Why are we here?

MONASH University

Such examples are the rule, rather than the exception!

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

https://www.infoq.com/articles/standish-chaos-2015

Why are we here?

MONASH University

Software can and does stay in use for a ***long*** time

Examples of Legacy Investments and Systems

Agency	Investment or system	Description	Agency-reported age	Specific, defined plans for modernization or replacement
Department of the Treasury	Individual Master File	The authoritative data source for individual taxpayers where accounts are updated, taxes are assessed, and refunds are generated. This investment is written in assembly language code—a low-level computer code that is difficult to write and maintain—and operates on an IBM mainframe.	~56	No - The agency has general plans to replace this investment, but there is no firm date associated with the transition.

https://www.gao.gov/assets/680/677454.pdf

Why are we here?

MONASH University

- It's easy to make small programs that run once
  - And you can get away with many, many bad practices in doing so
- We want to build software that:
  - Involves more than 20 lines of code.
  - Has an acceptable number of bugs
  - Has fewer other quality problems
  - When there are bugs, we can fix them
  - When the requirements change, we can extend/modify the software to cope.
- All delivered within reasonable time and budget!

MONASH University

- There are many definitions of Software Quality to be found in the literature. Here's one from the IEEE:

> "the degree to which a software product meets established requirements; however, quality depends upon the degree to which those established requirements accurately represent stakeholder needs, wants, and expectations." --IEEE Standard for Software Quality Assurance Processes," in IEEE Std 730-2014

- In other words, in order to be high quality, software must not only meet the *stated* needs, it must also meet unstated needs

Aspects of Quality – “ilities”

MONASH University

- Some examples of aspects of quality that are often important for software:
  - Functionality
  - Reliability
  - Usability
  - Efficiency
  - Maintainability
  - Portability
- These example “ilities” are only some of the quality aspects that may be important?
  - Who can give some other examples?

Building software is hard!

MONASH University

- In the 1960s, a “software crisis” was recognized
- In 1968, NATO organized the first Software Engineering Conference

Picture credit: http://www.cs.cornell.edu/gries/banquets/iticse2002/iticse2002.html

“The basic problem is that certain classes of systems are placing demands on us which are beyond our capabilities and our theories and methods of design and production at this time. There are many areas where there is no such thing as a crisis — sort routines, payroll applications, for example. It is large systems that are encountering great difficulties. We should not expect the production of such systems to be easy.”

K. Kolence, in 1968 NATO Software Engineering Conference Report , http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF

Key skills in building quality software

MONASH University

- Design
  - making good decisions about how system is put together

Our key focus in FIT2099

- Quality Assurance
  - checking that artefacts (code and non-code) produced in the process are of satisfactory quality

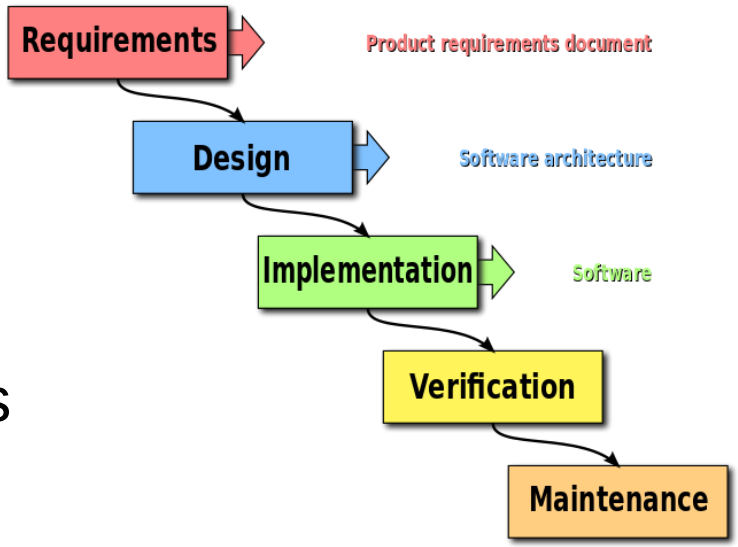
Touch on in 2099; main focus of FIT2107

- Management/Process
  - making these and other essential activities happen in a team, at the right time.

Touch on in 2099; main focus of FIT2101

What is software design?

- One (very bad) view: it's the production of a "design document" in the "design phase"
- Design actually the process of making decisions about how the software is to be implemented so as to have all the desired qualities
  - Of which functionality is only one
- Design is distinct from software requirements analysis



Ye Olde Waterfall Model

Why design?

MONASH University

- **ALL** software has a design
- However, not all software has a good design
- No conscious design usually leads to bad design
- Bad design leads to bad software
- Refactoring can improve design
- Not all bad design decisions can be easily refactored
- Whenever you make design decisions, make good ones!

MONASH University

- An OO program is conceptualized as a collection of *objects* that carry out the program's tasks by sending *messages* to each other
  - Objects are created using templates called *classes*
- Ideas developed in '60s and '70s.
  - Simula 67 – Ole-Juhan Dahl and Kristen Nygard
  - Smalltalk – Alan Kay and others at Xerox PARC
- Inspired by biological systems
- Popularized by C++, then Java, C#, etc.
  - Nearly all modern languages have capabilities or concepts originating in OO (Python, Javascript, Ruby, Scala, etc.)
- Pretty much all GUI systems were implemented using OO programing.

Object-oriented design

MONASH University

- OOP has proven good for constructing large systems
- But how does one identify the classes, objects, and messages needed for a system?
- Early 1990s: Booch, Jacobsen, Rumbaugh (the Three Amigos) and many others (e.g. Robert Martin) introduced
  - Notations for modeling object-oriented systems
    - Multiple competing notations were eventually replaced by UML (the *Unified* Modeling Language)
  - Methods for identifying the classes, objects, and messages required
  - Heuristics for evaluating a candidate (or implemented) design for "goodness"
    - Design principles, Design patterns, Design and Code "smells", etc.
    - Some heuristics from Structured Design were adapted for OO, e.g. coupling, cohesion, etc.

Why object-oriented design/programming?

MONASH University

- True, not all systems are implemented in an object-oriented manner
  - Even some written in OO languages!
- There are other design methods AND programming paradigms…BUT
  - Object-oriented design and OOP is still the most common in industry.
    - With java the overall most popular taking into account factors such as job ads, online tutorials, etc., according to 2017 surveys
      http://www.zdnet.com/article/which-programming-languages-are-most-popular-and-what-does-that-even-mean/
      https://stackify.com/popular-programming-languages-2018/
  - Most of the skills and design principles you learn doing OOD/OOP are transferable to other types of software development
- The concepts you learn are more important than the language you learn them in

Language Popularity Over Time

MONASH University

Very Long Term History

https://www.tiobe.com/tiobe-index/

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2018	2013	2008	2003	1998	1993	1988
Java	1	2	1	1	18	-	-
C	2	1	2	2	1	1	1
C++	3	4	3	3	2	2	5
Python	4	7	6	12	26	16	-
C#	5	5	7	9	-	-	-
Visual Basic .NET	6	13	-	-	-	-	-
JavaScript	7	9	8	7	21	-	-
PHP	8	6	4	5	-	-	-
Perl	9	8	5	4	3	11	-
Ruby	10	10	9	19	-	-	-
Objective-C	18	3	45	47	-	-	-
Ada	28	15	17	14	7	7	2
Lisp	31	12	14	13	6	4	3
Pascal	136	14	19	97	11	3	13

- You're going to have to learn new languages in future
  - some of which don't exist yet

Implementation

MONASH University

- Yes, you will be coding in this unit. The clue is in the name ☺
- The whole point of design is to make writing, modifying, and maintaining better code easier
- This unit is about the fundamental principles of good object-oriented design and implementation, illustrated and put into practice using Java
  - This unit is **not** "cowboy coding in Java"
- Not only is Java very popular in industry, it is a good language for learning key OO concepts and principles
  - And it is available on every platform

Where does this unit fit?

MONASH University

- After introductory programming
  - We will not be teaching “if… then”, for loops, etc. You are expected to know the basics already from the prerequisite units
- For SE/SD students
  - Alongside FIT2101 (Process & PM) and FIT2107 (QA)
  - Prepares you for FIT3077 (Architecture & Design) and third/fourth year project-based units
- For everyone
  - gives you a sense of the “what” and “why” of design, and how to put it into practice
- Prepares you for taking on bigger programming projects in final year/after graduation

Unit format

MONASH University

- Lectures (2 x 1 hour)
- Labs (1 x 2 hour)
  - Labs start WEEK 1, assessed labs start week 2
- Independent study
  - Set readings on Moodle. These are required and examinable

- You should expect to put in 12 hours work per unit per week, so that's another 8 hours outside the lectures and labs

Assessment

MONASH University

- 5 lab exercises (Weeks 2 through 6) – 2% each
- 3 Assignments. Done in pairs (same pair for each assignment)
  - Assignment 1 – 15%. Due end of Week 7
  - Assignment 2 – 15%. Due end of Week 9
  - Assignment 3 – 20%. Due end of Week 12

  Assignments involve designing and implementing extensions to an existing object-oriented system
- Exam – 40%. Open book
- Standard FIT hurdle requirement – minimum 40% overall in in-semester, 40% in exam, 50% overall to pass

Study resources

MONASH University

- There is no set textbook
- We will put online readings on Moodle
  - These are not optional. They are examinable
- If you need additional resources on Java programming, there are many, many textbooks and online resources
  - Post on the Moodle Discussion forum if you find one particularly helpful
- You need a working Java development environment to work on labs and assignments at home
  - We will use the (free) Eclipse IDE in labs http://www.eclipse.org
- You will use a git repository to manage all project data for the assignments
  - You will need a git client – Most modern IDEs have one integrated

Who are we?

MONASH University

- Murray Mount, Lecturer Clayton (Weeks 1-6)
- Robyn McNamara, Lecturer, Clayton (Weeks 7-12)
- Jasbir Dhaliwal, Lecturer, Malaysia
- Najam Nazar, Head Tutor (and Guest Lecturer) Clayton
- Chinnavit Chalidabhongse, Tutor, Clayton
- Deexita Goli, Tutor, Clayton
- Stephen McNamara, Tutor, Clayton
- Spike ??, Tutor, Clayton ← Mystery man ;-)
- Special Thanks to Robert Merkel and David Squire for much of the original lecture content and worked exercises.

Note: Photographs of some of the above may be found on Moodle..

Getting help

- In labs, ask your demonstrator
- Ask in the Moodle Discussion Forum (preferred)
  - Questions sent by email that are not of a personal nature, but about the unit content in general, will be redirected to Moodle
- Contact the head tutor by email
- Come to a consultation session

Roadmap for next few weeks

MONASH University

- First up – meet Java
- Next few weeks in lectures, using “code-along” and the “Fundamentals of Object Orientation in Java” document
  - Fundamental OO concepts, and how they are implemented in Java
  - Fundamental good coding and design principles and practices
  - Fundamental OO design principles
  - Refactoring to improve design
  - UML Basics
- You will:
  - Read the readings
  - Complete the labs
  - (Optional but recommended) Set up your development environment on your home computer