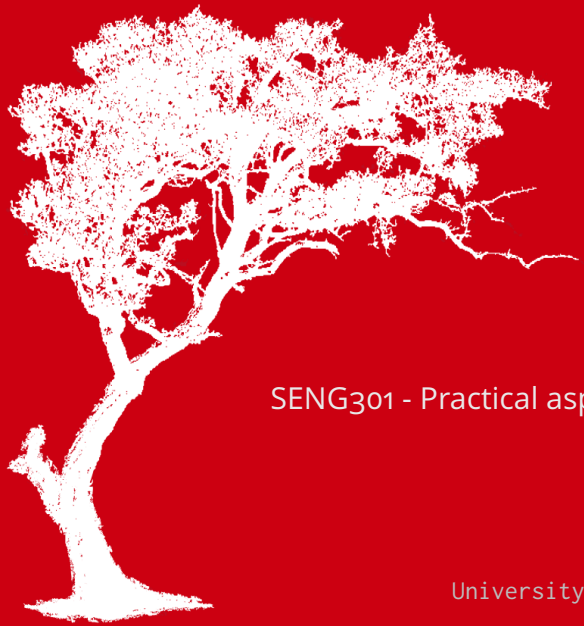


UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Fabian Gilson

Software Engineering II

SENG301 - Practical aspects and Software Engineering methods

Lecture 1 - Semester 1, 2020

University of Canterbury | Te Whare Wānanga o Waitaha
Computer Science and Software Engineering

Good judgment comes from experience, and experience comes from bad judgment.

– Frederick P. Brooks

Paraphrased from anonymous quote – orig. 1932

Practical details

Meet the team (I)

Fabian Gilson

fabian.gilson@canterbury.ac.nz

Erskine building, room 214

03.36.92.910

Moffat Mathews

moffat.mathews@canterbury.ac.nz

Erskine building, room 213

03.36.92.452

Meet the team (II)

Patrícia Andrade

patricia.inez@canterbury.ac.nz
Erskine building, room 324
03.36.92.453

Claudia Field

crf49@uclive.ac.nz

Matilda Porterfield

mpo74@uclive.ac.nz


Hand shaking

Practical details at a glance

Course timetable *(Wednesdays 12-1PM [E9] and Fridays 1-2PM [E5])*

- **term 1** 19 February → 5 April (7 weeks)
- **term 2** 29 April → 31 May (5 weeks)

Labs and tutorials *(Wednesdays 4-6PM [Lab2 or ER465] or Fridays 4-6PM [Lab2 or JE340])*

- **term 1** 19 February → 5 April
- **term 2** 29 April → 31 May

Assignments and final examination

- **assign.1** pair programming **[15%]**
- **assign.2** reflection report on weekly readings **[5%]**
- **assign.3** Design principles **[20%]**
- **final** knowledge and application of advanced software engineering **[60%]**

Lecture Schedule - *term 1*

19/02 Recap on software engineering methods
21/02 Agile Software Development
26/02 Planification to retrospection
28/02 Requirement analysis
04/03 Continuous integration
06/03 Continuous delivery
11/03 Testing and mocking
13/03 Resilience & reliability
18/03 Scrum team management
20/03 Software quality metrics
25/03 Software Architecture 101
27/03 User Experience 101

Labs and assignments schedule *(term 1)*

Practical labs

- **19-21/02** Java Persistence API 101
- **26-28/02** Scrum tutorial (part 1)
- **04-06/03** Scrum tutorial (part 2)
- **11-13/03** Unit and acceptance testing
- **18-20/03** Mocking and Stubing
- **25-27/03** Assignment 1 - group forming (compulsory)

Assignments

- **03/04** pair-programming (1 week no penalty drop-dead date) **[15%]**
- **28/04** reflection report **[5%]**

As the lecture goes, additional material will be discussed ($\rightarrow$ assignment 2)

Main Assignment *(for 3 Apr.)*

Pair programming, build on the case study used during the labs

- stories and technical knowledge

Assignment objectives **[15%]**

- propose the breakdown of specified stories into tasks
- practise pair programming
- implement the stories involving a database
  - design the system following **object-oriented** principles (cfr. SENG20X)
  - store into and retrieve data from a database (**ensure validity**)
  - write **unit** and user **acceptance** tests
  - your code must be **documented**, *i.e.* README, Javadoc, etc.

Final examination

Composed of two parts **[60%]**

- Software development methods **[30%]**
  - agile software development
  - software engineering practices
- Software design principles
  - See Moffat's requirements in term 2 **[30%]**

3-hour open book exam

Course material and textbooks

Lectures' slides designed to summarise content (available on *LEARN* and *echo360*)

Reference textbooks (for your own use beyond the course)

- Rosenberg and Stephens, *Use Case Driven Object Modeling with UML: Theory and Practice*, 2007
- Sommerville, *Software Engineering*, 2010
- Goldstein, *Scrum Shortcuts Without Cutting Corners: Agile Tactics, Tools, & Tips*, 2013

Papers also available in *"resources"* section on *LEARN*, *e.g*

- Beck, "Embracing change with extreme programming", 1999
- Boehm, "Get ready for agile methods, with care", 2002
- Cohn and Ford, "Introducing an agile process to an organization [software development]", 2003
- Angelov, Meesters, and Galster, "Architects in Scrum: What Challenges Do They Face?", 2016
- Object Management Group, *OMG Unified Modeling Language Specification, version 2.5.1*, 2017
- Murphy et al., "API Designers in the Field: Design Practices and Challenges for Creating Usable APIs", 2018

Official announcement

WANT TO BE PAID FOR YOUR NOTES?

The Disability Resource Service (DRS) is looking to buy high quality lecture notes from students enrolled at UC.

These notes will be used by students who experience difficulties taking notes for themselves for disability-related reasons, and DRS will pay $8 per lecture for them.

For most arts courses the notes need to be taken in MS Word. However, notes in science subjects where many formulas and mathematical symbols are used can be handwritten and scanned.

OF COURSE YOU DO!

STEP ONE:

Email drsnotes@canterbury.ac.nz as soon as possible with two samples of lecture notes you've taken.

STEP TWO:

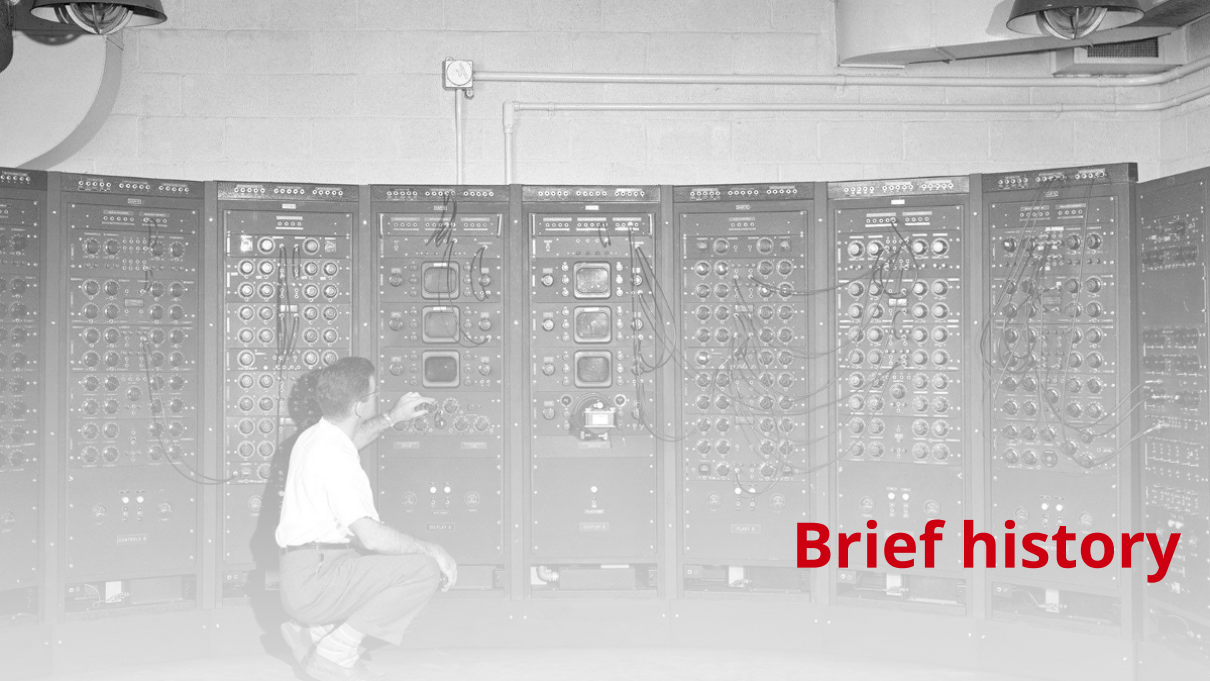
Complete the application form we'll email back to you.

STEP THREE:

Keep an eye on your UC email. If we like what we see and you're selected, we'll be in touch!

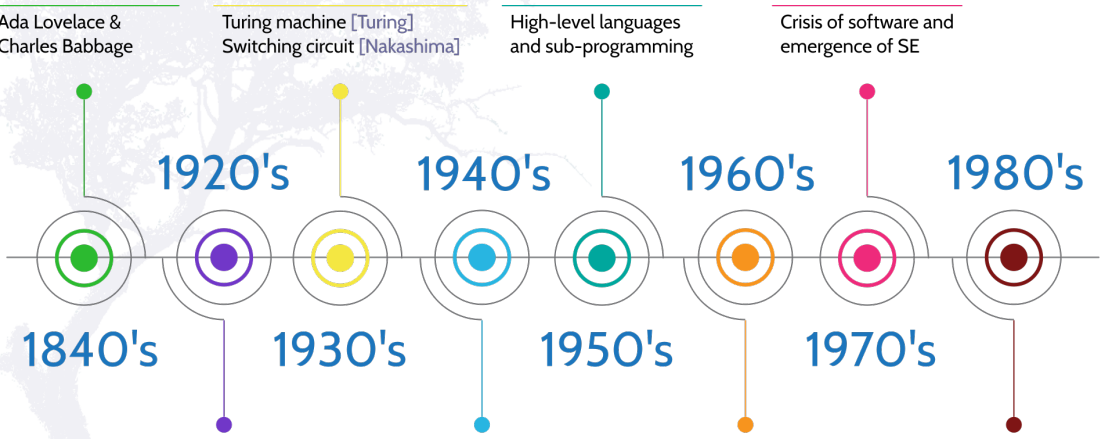
UC UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Disability Resource Service | University of Canterbury

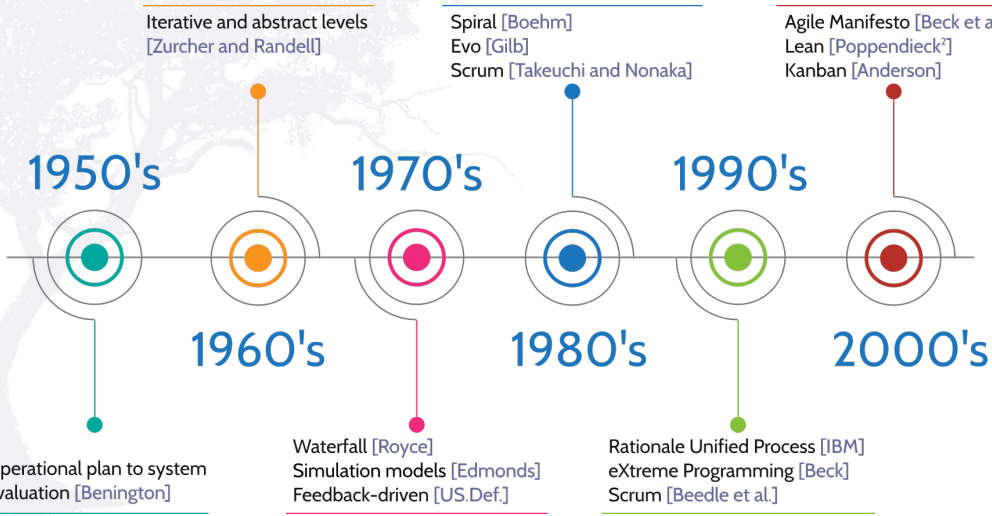


Brief history

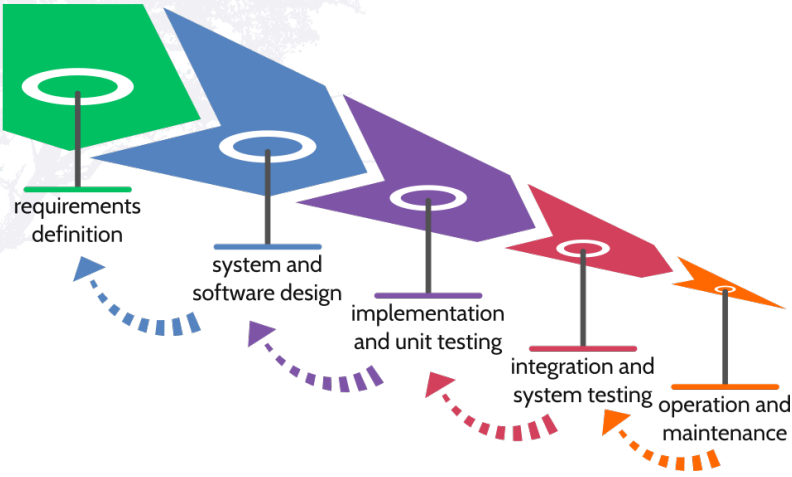
Software engineering practice



Software engineering methods



Waterfall (build it twice)



inspired from Royce, "Managing the Development of Large Software Systems", 1970

Spiral (no massive upfront analysis)

before each iteration, define the list of objectives, ie, what parts or functions will be developed

evaluate alternatives and mitigate the risks linked to the selected objectives

review and evaluate work done during this iteration release version or adapt requirements plan for next iteration

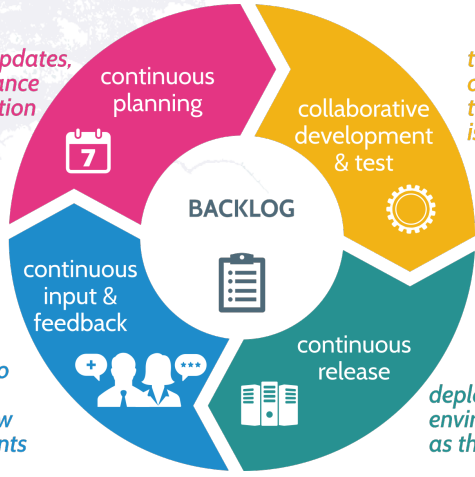
build a solution according to the identified risks, ie, design, write and test code or build a prototype,...

inspired from Boehm, "A Spiral Model of Software Development and Enhancement", 1986

Fully incremental (product vision only)

regular project status updates, monitoring of performance and continuous evaluation of pipeline of tasks

team assignments and organisation evolve from task to task and system is continuously tested



users are encouraged to provide feedback on system and suggest new features or improvements

deployment on production environment of new features as they are ready and tested

inspired from Schwaber and Beedle, *Agile Software Development with Scrum*, 2001

Let’s dig into what Agile is

Agile is a philosophy, not a method

What are the different implementations of Agile?

What are the working items in an Scrum project?

You can’t wait to get the first additional reading, right?

so come next lecture