Comp 47480: Contemporary Software Development

Mel Ó Cinnéide
School of Computer Science
University College Dublin

Some Basics

- Lecture schedule:
  - Tues 3-4, Thurs 4-5
- Practical slot
  - Tues 4-6
  - There won't be a practical every week
  - When there isn't a practical on Tuesday, I may use the practical slot for lectures (I will miss some lecture slots during the semester).
- Teaching Assistant: Jacek Wasilewski <jacek.wasilewski@ucdconnect.ie>
  - Contact Jacek with any issues you have regarding the practicals
- This module uses the CS mark->grade mapping here:
  https://www.cs.ucd.ie/Grading

Lecture Rules

No Recording Permitted!

Close All Laptops!

Hembrooke, H. and Gay, G., The laptop and the lecture: The effects of multitasking in learning environments, Journal of Computing in Higher Education, Volume 15, Issue 1, pp 46–64, September 2003.

Mueller, P. A. and Oppenheimer, D. M., The Pen Is Mightier Than the Keyboard: Advantages of Longhand Over Laptop Note Taking, Psychological Science, April, 2014.

Plagiarism & UCD Computer Science

- **Plagiarism is a serious academic offence**
  - [Student Code, section 6.2] or [UCD Registry Plagiarism Policy] or [CS Plagiarism policy and procedures]
- Our staff and demonstrators are **proactive** in looking for possible plagiarism in all submitted work
- Suspected plagiarism is reported to the CS Plagiarism subcommittee for investigation
  - Usually includes an interview with student(s) involved
  - 1st offence: **usually** 0 or NG in the affected components
  - 2nd offence: referred to the **University disciplinary committee**
- Student who enables plagiarism is equally responsible

http://www.ucd.ie/registry/academicsecretariat/docs/plagiarism_po.pdf

http://www.ucd.ie/registry/academicsecretariat/docs/student_code.pdf

http://libguides.ucd.ie/academicintegrity

Contacting Me / Questions

Ask me questions before, during or after lectures.

Questions emailed to me will generally be answered at the next lecture.

Questions related to the practicals (scheduling etc.) should be addressed to Jacek during the practical slot.

If I miss a lecture or practical…?

Important announcements may be made in lectures or practicals.

If you miss a lecture or practical, make sure to talk to someone who was there to make sure you don't miss anything.

Poor attendance correlates with poor grades!

Needless to say, if you miss lectures or practicals due to **extenuating circumstances**, contact me and I'll do what I can to accommodate you.

Questions?

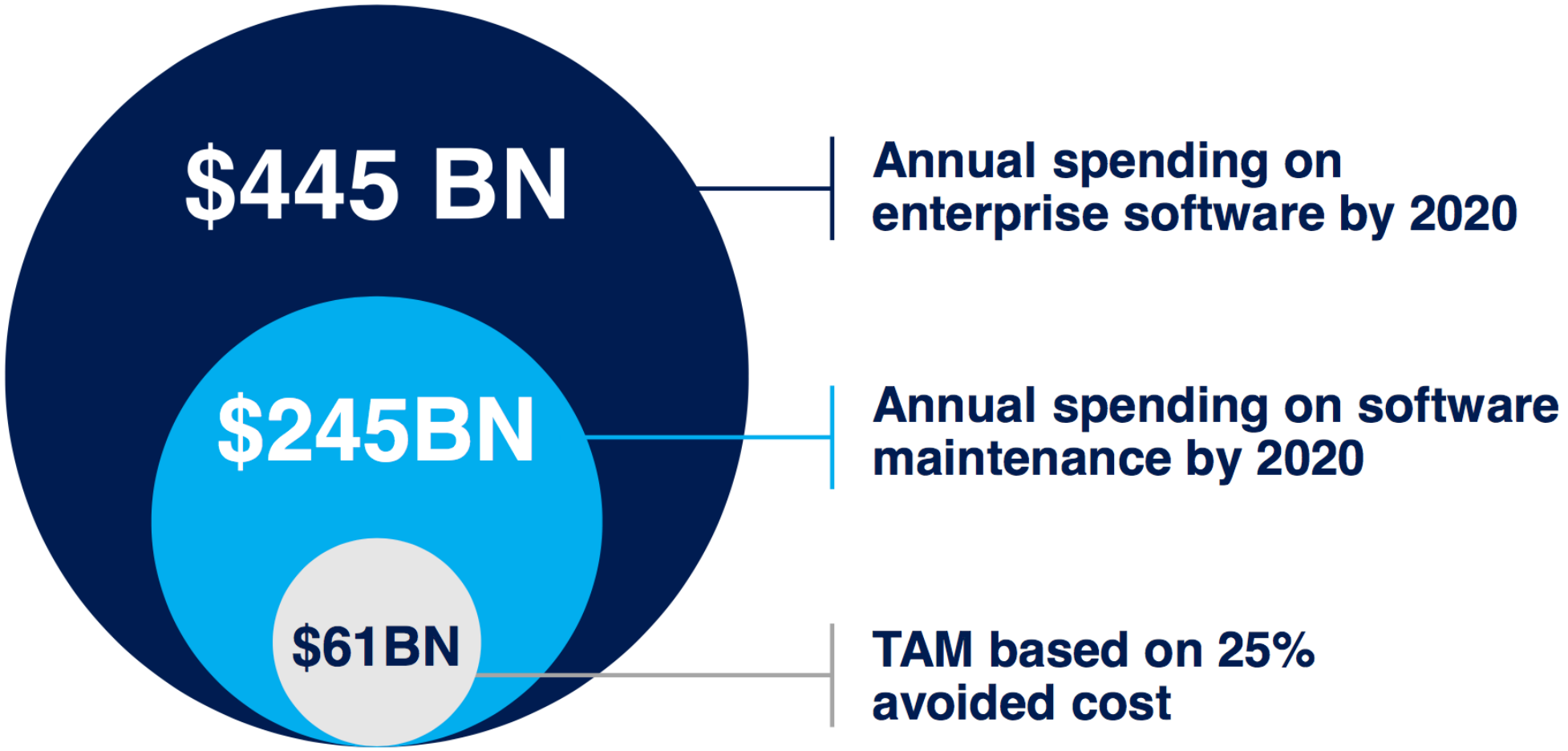
Just ask!

Course Overview

- We examine issues in Software Design and Development.
  - OOD + other stuff
- Essential Prerequisites
  - A working knowledge of object-oriented programming
  - Proficiency in Java
  - (For example, a knowledge of the material of Comp30070 is a sound basis for this module)
- Practical work
  - 6-8 laboratory sessions (practicals) in this module. Indicative topics:
    - Agile Methods
    - Test-Driven Development
    - OOD with UML
    - Code Smells
    - Refactoring
    - Software Design Principles
    - Design Patterns

Software Maintenance

MARKET SIZE



Sources: Gartner, http://www.gartner.com/newsroom/id/3672818; Accenture, https://spendtrends.accenture.com/wp-content/uploads/2015/05/Accenture-How-Software-Maintenance-Fees-are-Siphoning-IT-Budget-Procurement-BPO.pdf

Module Topics

- The module involves a number of topics
  - Software Methodology
  - Object-Oriented Design using UML
  - Software Testing
  - Design Principles
  - Code Smells and Metrics
  - Refactoring
  - Design Patterns

- We'll review each of these topics on the following slides...

Software Methodology

- The process of developing software has evolved enormously in the past half century (or even decade) and remains a fluid, complex area.

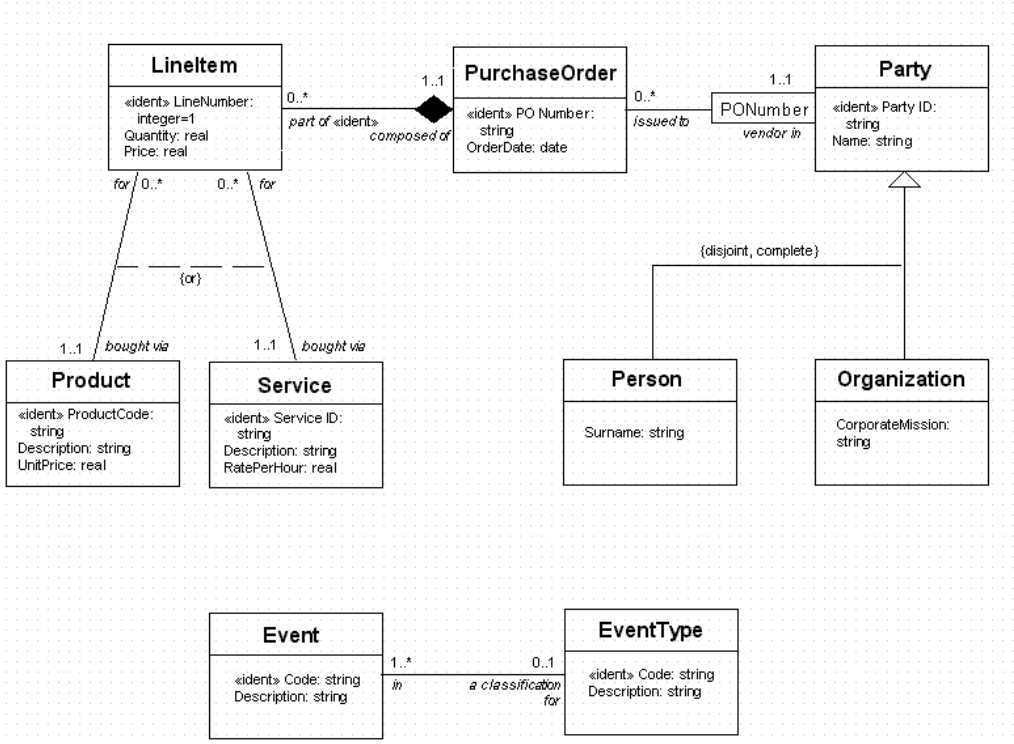
- We'll look at two development processes in more detail
  - Unified Process
  - Agile Processes

OOD and UML

- The standard notation for expressing an object-oriented design is **U**nified **M**odelling **L**anguage or **UML**.

- We'll see how various aspects of a system can be modelled using UML
  - Little emphasis on the notation itself; I assume this is familiar to you (or easy to pick up).

- One topic is to examine how a UML design can be used to drive the construction of the code.



Testing

- Of all software qualities, **correctness** is the most fundamental.
  - a *sine qua non* property
- The main tool for achieving and maintaining correctness is **testing**.
  - Unit testing
  - Integration testing
  - Acceptance testing
- We'll see that testing is not only related to correctness, but also to design quality improvement.

Design Principles

- What is good design?
- What is bad design?
- As software engineers, should we care?
  - Usually, yes!

- In this section we examine some of the fundamental principles that underpin software design
  - **S**ingle Responsibility Principle
  - **O**pen Closed Principle
  - **L**iskov Substitution Principle
  - **I**nterface Segregation Principle
  - **D**ependency Inversion
  - Law of Demeter
  - ...

- We'll that these design principles are not just notional elegance, they impact directly on vital qualities like **maintainability**.

Code Smells

- How can we tell if the design of our software has problems?
  - *Code Smells* are one approach
- A Code Smell is a sign in our software that all is not well
  - e.g. Long Method
- Smelly code will exhibit some these properties
  - Hard to understand
  - Hard to update
  - When updated, will lead to more smelly code
- In this section we'll look at a number of well-known code smells
  - Long Method
  - God class
  - Feature Envy
  - ...

Refactoring

- What do we do with smelly, unmaintainable code?
- **Refactoring** is the process of improving a program's design, without changing its behaviour.
  - E.g., a **long method** might be broken into several, smaller more cohesive methods.
- How can we be sure that program behaviour hasn't been changed?
  - We can use a refactoring tool that promises this (not 100% reliable!)
  - Use testing to ensure that behaviour is unchanged.
- In this section we'll look at a number of standard refactorings and see how they are used in practice:
  - Extract Method
  - Move Method
  - Replace Inheritance with Delegation
  - ...

Design Patterns

- Pattern definition:
  - "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.”
- Or in a software context:
  - A pattern as an encapsulation of an elegant design solution that can be used and reused in many different ways and contexts.
- Refactoring is often used to introduce a Design Pattern to a program.
- We'll examine a number of patterns from a standard catalogue, and do a design and implementation exercise related to patterns

‘How We Build Software’ Seminar Series

- This module incorporates a series of three seminars each entitled ‘How We Build Software’.
  - Takes place on certain Tuesdays, 3-4pm
  - Each seminar is delivered by a company involved in software development
  - Companies involved in previous years: IBM, Realex, Openet, Microsoft, Storyful, Workday, Demonware, OpenJaw, Dimension Data
- What you must do…
  - Attend each seminar
  - Ask questions
  - Write a report (this forms part of your practical work)

Primary Textbook

- There is no primary recommended text.
- The following slides have details of some of the books that this module is based on.
  - In each case, only parts of the text are covered
  - None of these is a required purchase
- There is a lot of information to be found on the web in these areas as well.

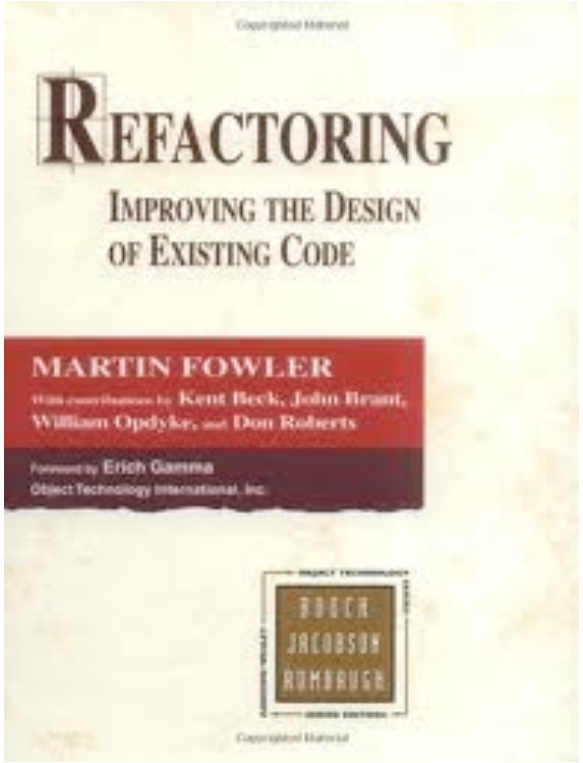
“Clean Code”

- One of the best texts that covers a lot of this module is:
  - **Robert Martin, *Clean Code*, Pearson, 2009.**
- This covers several relevant areas:
  - Testing
  - Refactoring
  - Code Smells
- This is essentially a practitioner's handbook for *software craftsmanship*.

Code Smells and Refactoring

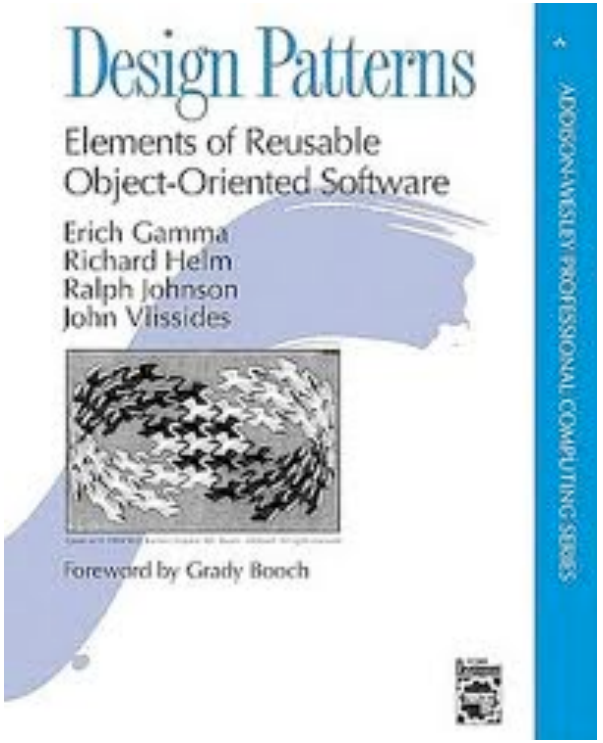
- The original text on this topic is:
  - **Martin Fowler, *Refactoring*, Addison-Wesley, 1999.**
  - UCD Library: 005.14 (1 copy)

- This book contains a catalogue of code smells and refactorings, along with advice on when to apply them and some detailed examples.



Design Patterns Text

- The best-known text in this area is:
  - **Erich Gamma *et al*, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.**
  - One of the best and best-selling books on software ever.
  - Available in Library:
    - Code: 005.12/Des
    - 3 copies in Short Loan Collection, 4 copies in General Collection.

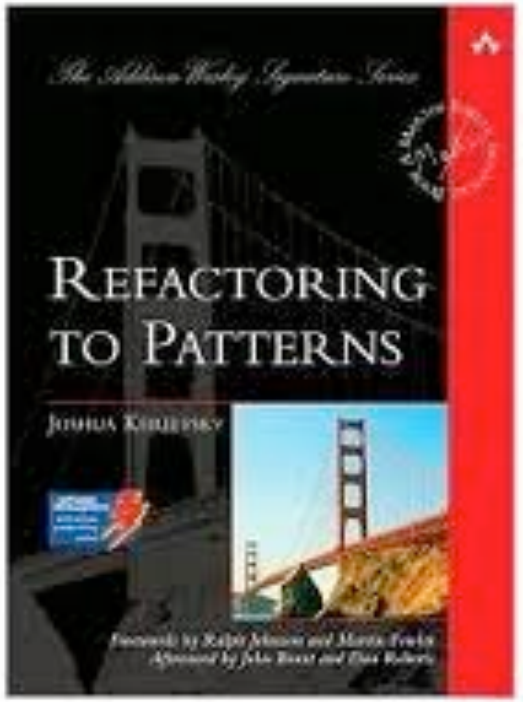


Another Design Patterns Textbook

- This is an easy-read, engaging design patterns book:
  - **Freeman et al, Head First Design Patterns, O'Reilly, 2004.**

Refactoring and Design Patterns

- See for example:
  - **Josh Kerievsky, *Refactoring to Patterns*, Pearson, 2005.**
  - UCD Library: 005.16 (1 copy)



Other Textbooks in the UCD Library

- Another UML text:
  - **Martin Fowler, *UML distilled : a brief guide to the standard object modeling language*, Addison-Wesley, 2000.**
  - Library: 005.117/FOW, 3 Gen, 3 copies in Short Loan Collection.
- Early sections on Software Quality are well worth reading in:
  - **Bertrand Meyer, *Object-Oriented Software Construction 2e*, Prentice-Hall, 1997..**
  - Library: 005.12/Mey 3 copies in Short Loan Collection.
- Easy reading on Extreme Programming:
  - **Kent Beck*, Extreme Programming Explained*, Addison-Wesley, 2000**.
  - Library: 005.1/Bec, 1 copy in General Collection.

Other Textbooks in Library

- Another patterns text:
  - **Craig Larman, *Applying UML and patterns: an introduction to object-oriented analysis and design*, Prentice Hall 1998.**
  - Library: 005.11/LAR, 1 copy in General Collection.

- The original work on patterns:
  - **Christopher Alexander, *A Timeless Way of Building*, Oxford University Press, 1979.**
  - Library: 720.1/ALE, 1 copy in General Collection (Architecture).

Finally

- The module web page will contain the lecture slides and other useful material:
  - csserver.ucd.ie/~meloc/47480/47480.htm
  - username: 47480
  - password: Tuesday

- The next section will be an introduction to the first topic of the course, **Software Development Methodology**.