Software Design and Modelling

Perdita Stevens

School of Informatics
University of Edinburgh

Plan

- What's this course about?
- What are you supposed to know already?

What's this course about?

How to:

- design simple object-oriented systems.
- create, read and modify UML diagrams documenting designs, both on paper and in an appropriate tool.
- determine whether a UML model and a body of Java code are consistent; if they are inconsistent, identify the inconsistencies precisely and propose remedies.
- evaluate and evolve object-oriented software designs, making use of common design patterns if appropriate.
- discuss the use of modelling and model-driven development tools in software development, e.g. why and how models of software can have varying degrees of formality, capabilities and limitations of the tools.

This course goes particularly well with Software Testing.

Elephant trap

At university, and in most summer jobs, you see small software systems and work with them over short timeframes.

Elephant trap

At university, and in most summer jobs, you see small software systems and work with them over short timeframes.

In that context, hacking works OK.

Elephant trap

At university, and in most summer jobs, you see small software systems and work with them over short timeframes.

In that context, hacking works OK.

But it does not work at scale!

Elephant trap

At university, and in most summer jobs, you see small software systems and work with them over short timeframes.

In that context, hacking works OK.

But it does not work at scale!

I will try to help you to understand why the techniques we learn in this course are worthwhile, but if you evaluate them against short small experiences, you may not get it.

Try to remember that real-world software systems can be many millions of LOC, many hundreds of person-years of effort, spread over many years, very complex.

Method

Learning to design well is hard.

Method

Learning to design well is hard.

Teaching someone to design well is impossible.

Method

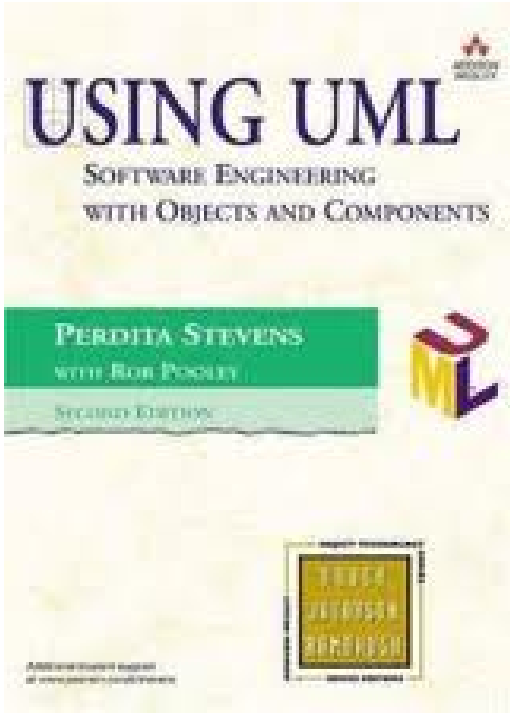
Learning to design well is hard.

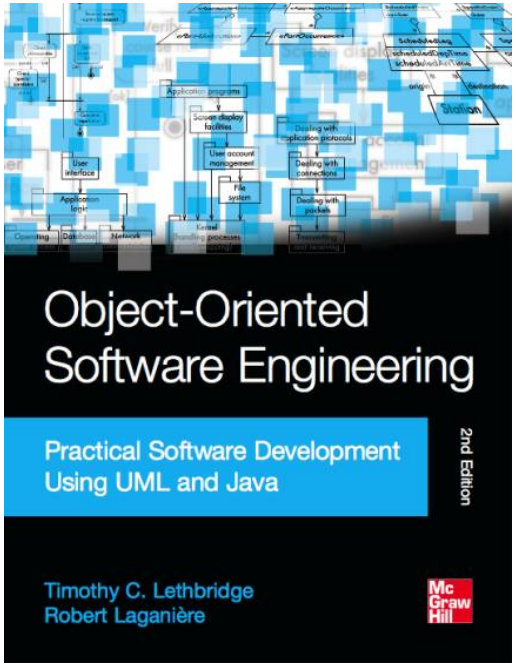
Teaching someone to design well is impossible.

But we can teach, e.g.

- the vocabulary of design criteria: what makes a design good?
- how to model designs so that they can be discussed
- how to learn from others' knowledge e.g. recorded as patterns.

Recommended books





Second-hand copies of UU are fine, but make sure they're second edition (for UML2).

Beyond exam success

80% of success is showing up.

80% of becoming a good software designer is caring and thinking about software design.

From now on, every time you read or write code, ask yourself: why is it designed this way? Could it be improved? How?

What are you supposed to know already?

1. How to program competently in Java (Inf1) – including understanding basic OO concepts.
2. What software engineering involves (Inf2) – including basic use of UML (class, sequence, use case diagrams only).

ASAP: please visit the course web site, join the Piazza class, and do the reading and video-watching that refreshes the pre-requisites.