UPPSALA
UNIVERSITET

# Course: Software Engineering

5-26 October 2018

The need for a disciplined approach to software development emerged in the mid to late sixties with the third generation of computers. Improvements in computing power led early developers to appreciate the difficulty in writing correct and efficient computer programs in the required time. A half century has passed and despite dramatic improvements in programming languages, algorithm development and techniques to manage projects, there is still a propensity for software projects to be late and over budget. The ubiquitous presence of computers and the vast size of modern systems means that there is a need to understand how to capture and manage this complexity.

“There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.” (Professor C.A.R. Hoare)

Language development through the use of information hiding, libraries of useful data structures and type checking has enabled far larger systems to be developed than anyone could ever have imagined. However, making use of these constructs and tools is not always well understood by programmers. This course will cover core aspects of modern object-oriented software engineering practices. This course teaches rigorous techniques for the specification, design, implementation and maintenance of software.

## Course background

Every country has its software failures, namely private or public funded projects that ended up being terminated despite huge investments. In Sweden a dental health service system, developed between 2007 and 2010, costing SEK 10 billion was declared not fit for purpose and discontinued. More recently a police case management system costing SEK 300 million functioned poorly and so was laid to rest. There are many more projects that are put into operation despite lingering difficulties and additional costs.

The problems more often than not stem from a poor understanding of what the system should do, a reliance on legacy code, an ever changing workforce and poor management. Agile methods and test-driven development mitigates some of these issues such that successful modern systems, such as Google or Facebook, consist of millions of lines of code and billions of lines of test data. However, they also rely on employing talented and imaginative software developers. This course goes behind the scenes to look at exactly what is required in order to develop robust and high-quality object-oriented software.

## Course content

At the end of the course, students will understand the concepts of object orientation and the relevance of these to code reusability, system extensibility and generic software development. The module teaches the fundamentals and principles of object-orientation, with an emphasis on the impact that the concept of an "object" has on practical programming. Students will be able to use class diagrams and logic to describe the models they develop. They will be able to apply important design techniques and will be able to use the language of patterns to find and record solutions to recurring problems of system architecture.

**Theme 1: Rigorous Design of Information Systems.**

We will look at how to translate a textual system description of a system into a class diagram and how to formally capture the behaviour of key entities using a formal specification language.

**Theme 2: Object Oriented Fundamentals**

This will cover the programming concepts of class, object, state, inheritance, interfaces, polymorphism, and generics. We will also introduce automatic testing that complements the more mathematical approach of the first theme and touch on refactoring.

**Theme 3: Advanced Object-Orientation**

The discussion will lead on to the difference between sub-typing and sub-classing using Liskov's substitutability principle. We will also argue the benefits of immutability and from here we will be able to explain the S.O.L.I.D. principles of object-oriented design.

**Theme 4: Design Patterns**

We will begin with history of patterns and their contemporary uses. We will then move on to discussing and implementing a variety of common patterns.

## Who should attend

The course is intended for programmers or technical managers who have a rudimentary understanding of object-orientation..

## Course method

The content of this course is based on material taught to part-time industrial students at Oxford University's Software Engineering program since 2002

## Value added

This course will allow the participants to familiarize themselves with concepts and become confident with terminology and practices that are used to create market leading and effective software.

**Utbildningsdagar i Uppsala:**
5, 12, 19, 26 Oktober 2018

**Undervisningsspråk:** Kursen ges på engelska

**Pris:** 39 000 kr exkl moms.
Alumn från Institutionen för informatik och media betalar 35 000 kr exkl moms.

**Sista anmälningsdag:** 2018-09-07

**Anmälan:**
www.uppdragsutbildning.uu.se/fortbildning-digitalteknik

**Kontakt**

Ansvarig lärare: Steve McKeever, steve.mckeever@im.uu.se
Generella frågor: Deqa Farah-Asbury deqa.farah-asbury@im.uu.se