Course Information and Syllabus

This course introduces a selection of theories, practices, and tools that, we believe, will enhance the student's ability in developing correct and high quality software. The view taken here is that of an engineer (programmer, software engineer, or software architect) and hence the focus of the course is primarily on the technical aspects of software development process.

Instructor

Yih-Kuen Tsay (蔡益坤), Room 1108, Management II, 3366-1189, `tsay@im.ntu.edu.tw`

Lectures

Thursday 9:10–12:10, Room 101, College of Management, Building I

Office Hours

Wednesday 1:30–2:30PM or by appointment

TA

Yi-Wen Chang (常怡文), 3366-1205, `r97725004@ntu.edu.tw`

Prerequisites

Object-Oriented Programming and Discrete Mathematics

Textbook

Class Notes and *Selected Readings*

Syllabus/Schedule

The goal of this course is to acquaint the students with some of the well-used methods and tools for practical software development as well as some fundamentals of software verification, so as to prepare them for a career in software development. After an introduction of the subject matters and a brief glimpse of formal logic and program correctness, we will study in great detail the UML, design patterns, and some fundamental elements of formal software verification and analysis.

- **Introduction** (.5 week: 09/17a)
  overview of software requirements, development process, design methods, and testing/verification tools
- **Formal Logic** (1.5 weeks: 09/17b, 09/24)
  propositional logic (*satisfiability, tautologies, deduction/proofs*), first-order logic (*validity, deduction/proofs, soundness, completeness*)
- **Program Correctness** (1 week: 10/01)
  axiomatic semantics of programs (*assertions, pre/post-conditions, invariants*), partial and total correctness

- **UML – Part I** (2 weeks: 10/08, 10/15)
  introduction, basics of modeling, overview of the UML, structural modeling (*class diagrams, classifiers, interfaces, packages, object diagrams*), behavioral modeling (*interactions, use case diagrams, interaction diagrams, activity diagrams*), architectural modeling (*components, collaborations, patterns, frameworks, component diagrams, deployment diagrams*)
- **Design Patterns** (Guest Instructors: Jeffrey CH Liu, Clement CW Su, and Jim CL Yu, IBM) (4 weeks: 10/22, 10/29, 11/05, 11/12)
  1. introduction (*use case, design document, development tool*), creational patterns (*abstract factory, builder, factory method, prototype, singleton*)
  2. behavioral patterns I (*chain of responsibility, command, interpreter, iterator, mediator, memento, observer, state*)
  3. behavioral patterns II (*strategy, template method, visitor*), structural patterns (*adapter, bridge, composite, decorator, facade, flyweight, proxy*)
  4. architecture review, design document wrap-up, other topics (*anti-patterns, advanced patterns, refactoring*)
- **Web Programming and Software Security** (1 week: 11/19)
  dynamic Web pages, client-server interactions, common security vulnerabilities, patterns, frameworks
- **UML – Part II** (2 weeks: 11/26, 12/3)
  advanced behavioral modeling (*events, state machines, processes and threads, statechart diagrams*), Object Constraint Language (OCL)
- **Software Modeling Tools** (1 week: 12/10)
  Alloy (software modeling, simulation, and checking)
- **Midterm** (**2009/12/17**)
- **Software Model Checking** (2 weeks: 12/24, 12/31)
  linear-time model checking (*Kripke structure, linear temporal logic, Büchi automata, automata-theoretic algorithms*), Spin (*Promela, never-claims*), JPF (Java Pathfinder)
- **Term Project Presentations** (**2010/01/07**)
- **Program Verification Tools** (1 week: 01/14)
  Spec#, JML (Java Modeling Language) tools (*Common JML Tools, ESC/Java2*)

Web Site

`http://www.im.ntu.edu.tw/~tsay/courses/sdm/`

Grading

Homework 20%, Midterm 40%, Term Project 40%.

References

[1] *Logic for Computer Science: Foundations of Automatic Theorem Proving*, J.H. Gallier, Harper & Row Publishers, 1985. (free!)

[2] *Logic in Computer Science: Modelling and Reasoning about Systems*, M. Huth and M. Ryan, Cambridge University Press, 2004.

[3] *The UML Resource Page*: `http://www.uml.org/`, OMG.

[4] *The Unified Modeling Language User Guide, 2nd Edition*, G. Booch, I. Jacobson, and J. Rumbaugh, Addison-Wesley, 2005.

[5] *Design Patterns: Elements of Reusable Object-Oriented Software*, E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Addison-Wesley, 1995.

[6] *Software Abstractions: Logic, Language, and Analysis*, D. Jackson, MIT Press, 2006.

[7] *The SPIN Model Checker: Primer and Reference Manual*, G.J. Holzman, Addison-Wesley, 2003.

[8] *Spin - Formal Verification Page*: `http://spinroot.com/`.

[9] *Temporal Verification of Reactive Systems: Safety*, Z. Manna and A. Pnueli, Springer-Verlag, 1995.

[10] *The Formal Methods Page*: `http://formalmethods.wikia.com/wiki/Formal_methods`, J. Bowen. (Note: this Web portal provides links to numerous formal methods and tools.)