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).

Instructors
Yih-Kuen Tsay (蔡益坤), Room 1108, Management II, 3366-1189, tsay@im.ntu.edu.tw
Bow-Yaw Wang (王柏堯), Room 1104, Management II, 3366-1194, bywang@im.ntu.edu.tw

Lectures
Thursday 9:10–12:10, Room 205, College of Management, Building II

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

TA
Ming-Hsien Tsai (蔡明憲), 3366-1205, minghsientsai@ntu.edu.tw

Prerequisites
Computer Programming and Discrete Mathematics

Textbook
Class Notes and Selected Readings

References
[10] Supplementary Readings
Course Web Site
http://www.im.ntu.edu.tw/~sdm/

Grading
Midterm 35%, Final 35%, Homework 10%, Term Project 20%.

Syllabus/Schedule
After an introduction of the subject matter and a brief glimpse of formal logic and program correctness, we will study in greater detail the UML, design patterns, and some fundamental elements of formal verification.

- **Introduction**: (.5 week: 09/21a)
  software development process, from requirements to design, design methods and tools, testing, verification.

- **Formal Logic and Program Correctness**: (2.5 weeks: 09/21b, 09/28, 10/05)
  propositional logic (satisfiability, tautologies, deduction/proofs), first-order logic (validity, deduction/proofs, soundness, completeness), axiomatic semantics of programs (assertions, pre/post-conditions, invariants), partial and total correctness, Java assertions.

- **UML – I**: (2 weeks: 10/12, 10/26)
  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 – I**: (3 weeks: 10/19, 11/2, 11/09)
  C++ overview, creational patterns (abstract factory, builder, factory method, prototype, singleton), structural patterns (adapter, bridge, composite, decorator, facade, flyweight, proxy)

- **Midterm**: (2006/11/16)

- **Design Patterns – II**: (3 weeks: 11/23, 11/30, 12/07)
  behavioral patterns (chain of responsibility, command, interpreter, iterator, mediator, memento, observer, state, strategy, template method, visitor), language features (assertions, exceptions, static versus dynamic typing)

- **UML – II**: (1 week: 12/14)
  advanced behavioral modeling (events, state machines, processes and threads, state-chart diagrams), object constraint language (OCL)

- **Automata Theory**: (1 week: 12/21)
  finite state automata, regular expressions

- **Temporal Logic and Model Checking**: (2 weeks: 12/28, 01/04)
  syntax and semantics of temporal logics (Kripke structure, linear and branching-time temporal logics, CTL, LTL), Spin (Promela), model checking (Büchi automata, automata-theoretic algorithm)

- **Deductive Methods**: (1 week: 01/11)
  refinement and program derivation.

- **Final**: (2007/01/18)