Introduction to Software Engineering Metrics

Description

This course covers the basic principles and concepts of software engineering; system requirements; secure programming in the large; modeling and testing; object oriented analysis and design using the UML; design patterns; frameworks and APIs; client-server architecture; user interface technology; and the analysis, design and programming of extensible software systems.

Sample labs and assignments:
- Evaluating the performance of various simple software designs.
- Adding features to an existing system.
- Testing a system to verify conformance to test cases.
- Building a GUI for an application.
- Numerous exercises building models in UML, particularly class diagrams and state machines.
- Developing and presenting a simple set of requirements (to be done as a team) for some innovative client server application of very small size.
- Implementing the above, using reusable technology to the greatest extent possible.

Additional teaching considerations:
- This course is a good starting point for exposing students to moderately sized existing systems. With such systems, they can learn and practice the essential skills of reading and understanding code written by others. Students should write code in the context of a particular domain, for example the biological, physical, mathematical or chemical sciences or even wider spectra such as game programming, business applications, and graphics and animation.
- It is suggested that a core subset of UML be taught, rather than trying to cover all features.
- It may be challenging for instructors to convey the nature of SE to students; however, this challenge may be addressed through strategies such as field trips to businesses and industries that utilize large software systems, guest lectures by developers and users of large software systems, and discussions about embedded systems in everyday life including ATMs, wireless devices, cell phones, various mobile devices, and computer games.

Minimum Contact Hours

42

Assessment Rubric

CS. 3. Practice the tenets of ethics and professional behavior promoted by computing societies; accept the professional responsibilities and liabilities associated with software development.

Emerging Standard
Studies the tenets of ethics and professional behavior promoted by international computing societies, such as ACM and IEEE-CS.

Developed Standard
Practices the tenets of ethics and professional behavior promoted by international computing societies, and recognizes the liabilities associated with software development.

Highly Developed Standard
Displays ethical and professional behavior associated with the responsibilities of software development.

**SE. 2. Construct a preliminary investigative report for a proposed system that includes scheduling and plans for mitigating potential risks.**

Emerging Standard
Constructs a project plan that identifies major milestones and a development schedule.

Developed Standard
Constructs a project plan that identifies major milestones and a development schedule that includes contingencies for recovering from schedule set-backs.

Highly Developed Standard
Constructs a project plan with revisions that includes options analysis and estimates risk associated with different options.

**CS. 17. Create effective, efficient and secure software, reflecting standard principles of software engineering and software assurance.**

Emerging Standard
Calculates the risks and liabilities of a computer-based solution using standard software assurance and engineering principles.

Developed Standard
Creates an effective, efficient and secure solution, utilizing principles of software assurance and software engineering.

Highly Developed Standard
Judges the safety and security of a software solution.

**SE. 3. Decompose complex systems using best practice object-oriented analysis and design tools and techniques.**

Emerging Standard
Completes a static analysis and design that identifies the classes/objects of a complex system.

Developed Standard
Using static design patterns, successfully identifies the classes/objects and specifying the relationships among them.

Highly Developed Standard
Successfully models the dynamic interactions among the objects and external interfaces in addition to a well-crafted static analysis and design.
SE. 7. Design and implement consistent and coherent user-centered interfaces that comply with UI standard practices.

Emerging Standard
Creates a structured, simple UI.

Developed Standard
Using static design patterns, successfully identifies the classes/objects and specifying the relationships among them Creates well-structured, consistent UI that provides all needed options and user feedback.

Highly Developed Standard
Creates a clear, consistent UI that is flexible enough to prevent user input errors.

SE. 4. Evaluate and test software system requirements that considers both validation and verification.

Emerging Standard
Describes the major phases of the testing process for a medium-sized team project.

Developed Standard
Develops a testing plan and schedule to individually test all requirements of a medium-sized team project.

Highly Developed Standard
Successfully implements a testing plan for a medium-sized team programming project.

CS. 33. Use standard analysis and design techniques to produce a team-developed, medium-sized, secure software application that is fully implemented and formally tested.

Emerging Standard
As part of a team, produces an executable, medium-sized software application that meets some program requirements and includes design documentation and some evidence of testing.

Developed Standard
As part of a team, produces a working, medium-sized software application on time that meets many program requirements including design and some test plan documentation.

Highly Developed Standard
As part of a team, successfully develops a medium-sized, secure software application on time that meets all program requirements including design and formal test plan documentation.