SE curriculum in CC2001 made by IEEE and ACM: Overview and Ideas for Our Work

Katerina Zdravkova
Institute of Informatics
E-mail: Keti@ii.edu.mk

What is Software Engineering?

- SE is the discipline concerned with the application of theory, knowledge, and practice for effectively and efficiently building software systems that satisfy the requirements of users and customers.
- It encompasses all phases of the life cycle of a software system.
- The life cycle includes requirement analysis and specification, design, construction, testing, and operation and maintenance

Suggested SE courses

- Advanced Software Development
- Software Engineering
- Software Design
- SE and Formal Specifications
- Empirical Software Engineering
- Software Process Improvement
- Component-Based Computing
- Programming Environment
- Safety-Critical Systems

Software Engineering (core lectures)

- Software design
- Using APIs
- Software tools and environments
- Software processes
- Software requirements and specifications
- Software validation
- Software evolution
- Software project management

Software Engineering (elective lectures)

- Component-based computing
- Formal methods
- Software reliability
- Specialized systems development

SE1. Software design (8 hours)

Topics:

- Fundamental design concepts and principles
- Design patterns
- Software architecture
- Structured design
- Object-oriented analysis and design
- Component-level design
- Design for reuse

SE1: Learning objectives / 1

1. Discuss the properties of good software design.
2. Compare and contrast object-oriented analysis and design with structured analysis and design.
3. Evaluate the quality of multiple software designs based on key design principles and concepts.
4. Select and apply appropriate design patterns in the construction of a software application.

SE1: Learning objectives / 2

5. Explain the value of application programming interfaces (APIs) in software development.
6. Use class browsers and related tools during the development of applications using APIs.
7. Design, implement, test, and debug programs that use large-scale API packages.

SE2. Using APIs (5 hours)

Topics:

- API programming
- Class browsers and related tools
- Programming by example
- Debugging in the API environment
- Introduction to component-based computing

SE2: Learning objectives

1. Explain the value of application programming interfaces (APIs) in software development.
2. Use class browsers and related tools during the development of applications using APIs.
3. Design, implement, test, and debug programs that use large-scale API packages.

SE3. Software tools and environments (3 hours)

Topics:

- Programming environments
- Requirements analysis and design modeling tools
- Testing tools
- Configuration management tools
- Tool integration mechanisms

SE3: Learning objectives

1. Select, with justification, an appropriate set of tools to support the development of a range of software products.
2. Analyze and evaluate a set of tools in a given area of software development (e.g., management, modeling, or testing).
3. Demonstrate the capability to use a range of software tools in support of the development of a software product of medium size.

SE4. Software processes (2 hours)

Topics:

- Software life-cycle and process models
- Process assessment models
- Software process metrics

SE4: Learning objectives / 1

1. Explain the software life cycle and its phases including the deliverables that are produced.
2. Select, with justification the software development models most appropriate for the development and maintenance of a diverse range of software products.
3. Explain the role of process maturity models.

SE4: Learning objectives / 2

4. Compare the traditional waterfall model to the incremental model, the object-oriented model, and other appropriate models.
5. For each of various software project scenarios, describe the project's place in the software life cycle, identify the particular tasks that should be performed next, and identify metrics appropriate to those tasks

SE5. Software requirements and specifications (4 hours)

Topics:

- Requirements elicitation
- Requirements analysis modeling techniques
- Functional and non-functional requirements
- Prototyping
- Basic concepts of formal specification techniques

SE5: Learning objectives / 1

1. Apply key elements and common methods for elicitation and analysis to produce a set of software requirements for a medium-sized software system.
2. Discuss the challenges of maintaining legacy software.
3. Use a common, non-formal method to model and specify (in the form of a requirements specification document) the requirements for a medium-size software system.

SE5: Learning objectives / 2

4. Conduct a review of a software requirements document using best practices to determine the quality of the document.
5. Translate into natural language a software requirements specification written in a commonly used formal specification language.

SE6. Software validation (3 hours)

Topics:

- Validation planning
- Testing fundamentals, including test plan creation and test case generation
- Black-box and white-box testing techniques
- Unit, integration, validation, and system testing
- Object-oriented testing
- Inspections

SE6: Learning objectives / 1

1. Distinguish between program validation and verification.
2. Describe the role that tools can play in the validation of software.
3. Distinguish between the different types and levels of testing (unit, integration, systems, and acceptance) for medium-size software products

SE6: Learning objectives / 2

4. Create, evaluate, and implement a test plan for a medium-size code segment.
5. Undertake, as part of a team activity, an inspection of a medium-size code segment.
6. Discuss the issues involving the testing of object-oriented software.

SE7. Software evolution (3 hours)

Topics:

- Software maintenance
- Characteristics of maintainable software
- Reengineering
- Legacy systems
- Software reuse

SE7: Learning objectives / 1

1. Identify the principal issues associated with software evolution and explain their impact on the software life cycle.
2. Discuss the challenges of maintaining legacy systems and the need for reverse engineering.
3. Outline the process of regression testing and its role in release management.
4. Estimate the impact of a change request to an existing product of medium size.

SE7: Learning objectives / 2

5. Develop a plan for re-engineering a medium-sized product in response to a change request.
6. Discuss the advantages and disadvantages of software reuse.
7. Exploit opportunities for software reuse in a given context.

SE8. Software project management (3 hours)

Topics:

- Team management
- Team organization and decision-making
- Project scheduling
- Software measurement and estimation techniques
- Risk analysis
- Software quality assurance
- Software configuration management
- Project management tools

SE8: Learning objectives

1. Demonstrate through involvement in a team project the central elements of team building and team management.
2. Prepare a project plan for a software project that includes estimates of size and effort, a schedule, resource allocation, configuration control, change management, and project risk identification and management.
3. Compare and contrast the different methods and techniques used to assure the quality of a software product.

SE9. Component-based computing

Topics:

- Fundamentals
- Basic techniques
- Applications (including the use of mobile components)
- Architecture of component-based systems
- Component-oriented design
- Event handling: detection, notification, and response
- Middleware

SE9: Learning objectives

1. Explain and apply recognized principles to the building of high-quality software components.
2. Discuss and select an architecture for a component-based system suitable for a given scenario.
3. Identify the kind of event handling implemented in one or more given APIs.
4. Explain the role of objects in middleware systems and the relationship with components.
5. Apply component-oriented approaches to the design of a range of software including those required for concurrency and transactions, reliable communication services, database interaction including services for remote query and database management, secure communication and access.

SE10. Component-based computing

Topics:

- Formal methods concepts
- Formal specification languages
- Executable and non-executable specifications
- Pre and post assertions
- Formal verification

SE10: Learning objectives

1. Apply formal verification techniques to software segments with low complexity.
2. Discuss the role of formal verification techniques in the context of software validation and testing.
3. Explain the potential benefits and drawbacks of using formal specification languages.
4. Create and evaluate pre- and post-assertions for a variety of situations ranging from simple through complex.
5. Using a common formal specification language, formulate the specification of a simple software system and demonstrate the benefits from a quality perspective.

SE11. Software reliability

Topics:

- Software reliability models
- Redundancy and fault tolerance
- Defect classification
- Probabilistic methods of analysis

SE11: Learning objectives

1. Demonstrate the ability to apply multiple methods to develop reliability estimates for a software system.
2. Identify and apply redundancy and fault tolerance for a medium-sized application.
3. Explain the problems that exist in achieving very high levels of reliability.
4. Identify methods that will lead to the realization of a software architecture that achieves a specified reliability level.

SE12. Specialized systems development

Topics:

- Real-time systems
- Client-server systems
- Distributed systems
- Parallel systems
- Web-based systems
- High-integrity systems

SE12: Learning objectives

1. Identify and discuss different specialized systems.
2. Discuss life cycle and software process issues in the context of software systems designed for a specialized context.
3. Select, with appropriate justification, approaches that will result in the efficient and effective development and maintenance of specialized software systems.
4. Given a specific context and a set of related professional issues, discuss how a software engineer involved in the development of specialized systems should respond to those issues.
5. Outline the central technical issues associated with the implementation of specialized systems development.

Comparison with the current 60+30+30 SE syllabus

- Part I: Introduction to Software Engineering (5 hours)
- Part II: Requirements engineering (analysis and definition) (19 hours)
- Part III: Software Design (5 hours)
- Part IV: Implementation and Testing (10 hours)
- Part V: Further problems (21 hours)

Part I in CC2001

- What is SE (2 lh): N
- Quality criteria for software products (1 lh): SE4/2
- Software process models (1 lh): SE4/1; SE5/4
- Basic concepts and software development documents (1 lh) SE1/1; SE2; SE3

Part II in CC2001 / 1

- Results of the “Analysis and Definition” phase (1 lh): SE3/2
- Cost estimation (2 lh): N
- Basic concepts of the function-oriented view (1 lh): SE5/5
- Basic concepts of data-oriented view (1 lh): SE5/5
- Basic concepts of rule-oriented view (1 lh): SE5/5

Part II in CC2001 / 2

- Structured analysis (1 lh): SE1/4
- Basic concepts of state-oriented view (1 lh): SE5/5
- Basic concepts of scenario-oriented view (1 lh): SE5/5
- Object-oriented analysis (6 lh): SE1/5
- Formal software specification and program verification (3 lh): N

Part III in CC2001

- Overview of design activities (2 lh): SE1/6; SE1/7
- Structured design (1 lh): SE1/4
- Object-oriented design (2 lh): SE1/5

Part IV in CC2001

- Implementation (2 lh): SE3/1; SE2
- Systematic testing (6 lh): SE3/3; SE6/1 – SE6/6
- Functional testing (2 lh): SE3/3

Part V in CC2001

- Software metrics (4 lh): SE4/3
- Maintenance (2 lh): SE7/1
- Reverse engineering (4 lh): SE7/3; SE7/5
- Quality of software development process and its standardization (3 lh): SE7/4
- Introduction to software ergonomics (4 lh): N
- Project management (4 lh): SE8

Uncovered CC2001 topics

- SE2: Using APIs [core] (5 hours)
- SE5/1: Requirements elicitation
- SE6: Software validation / Part IV
- SE9: Component-based computing (III)
- SE10: Formal methods (in parts)
- SE11: Software reliability (V 24)
- SE12: Specialized systems development

Ideas for Our Work

- Current syllabus
- the most of core CC2001 topics
- Small differences can be either neglected or modified during course evolution
- Uncovered CC2001 topics (SE2) could? replace some of the topics in part II
- Current course should be implemented!