COURSE DESCRIPTION

Dept., Number  CSC 131  Course Title  Computer Software Engineering
Semester hours  3  Course Coordinator  Ahmed Salem
URL (if any):  http://gaia.ecs.csus.edu/~salema/

Catalog Description

Principles of Software Engineering covering the software development life cycle, including software requirements engineering (elicitation, modeling, analysis and specification), software design, software implementation and testing. Main topics include various software development process models, method and techniques for specifying requirements, architectural and detailed design specification, prototyping, top-down and bottom-up software implementation and testing. Topics on project management, project documentation and the development of communication skills through written documentation and oral presentation. Prerequisite: At least a C- grade in CSC 130; however, it may be taken concurrently.

Textbook


References


Course Goals

The goals of the course are to:

1. Enable a student to understand the main characteristics of software.
2. Introduce the concept of a software development life cycle and examine some popular life-cycle models.
3. Discuss user roles in the software development process.
4. Describe the main technical activities associated with software engineering: requirements elicitation, modeling, analysis and specification, architectural and detailed design specification, implementation, testing and maintenance.
5. Teach a student a variety of techniques associated with the main activities of software engineering and provide practice in their use.
6. Explain the key characteristics of different kinds of software systems – for example, real-time, database-oriented, distributed, knowledge-based and safety-critical systems – and discuss the implications of these characteristics on the selection of development techniques.
7. Provide sufficient knowledge for a student to be able to choose development techniques, tools, and life-cycle models for a given project.
8. Give students an understanding of the importance of quality assurance, human factors, professional issues and project management in software development.
9. Give students an understanding of the importance of user involvement throughout the development process.
10. Illustrate the role of CASE tools in Software Engineering.
11. Demonstrate the need for and practice of effective communication skills, both oral and written.

**Prerequisites by Topic**

*Thorough understanding of:*
- Design and implementation of medium size programs using multi-level decomposition, data abstraction, and procedural abstraction.
- Programming style and program documentation concepts.
- Program development process, including the distinction between compiling, linking, and executing programs.
- Use of a symbolic debugger, testing and debugging techniques.
- Records/structures, classes, and objects, not including inheritance and polymorphism.
- One dimensional arrays of structured types.
- Sub-programs and their appropriate use, including understanding of forward declarations of subprograms and why they are needed.
- Parameter passing and its implications.
- Scope rules.
- Selected sorting and searching algorithms such as: insertion sort, and binary search.

*Basic understanding of:*
- Lifetime of variables.
- Dynamic memory allocation and use of pointers.
- Recursion as a problem solving technique, and its implementation in programming languages.
- Multi dimensional arrays.
- Representation of data structures, including simulated pointers (cursor representation).
- Evaluation of various alternatives to select appropriate data structures.
- Strings, their various representations, and operations.
- Linear list ADTs, their representation (contiguous/sequential, linked) and associated algorithms.
- Tools and techniques to allow generic data types, e.g. templates.

*Exposure to:*
- Big-O notation and ability to analyze simple algorithms.
- Exception handling.
Major Topics Covered in the Course

1. Introduction to course and topic.
2. Software development process models.
3. Software system engineering.
4. Requirements engineering.
5. Structured & Object-oriented analysis.
6. Use of Semi-formal modeling languages (e.g. UML).
7. Design (architectural and detailed).
8. Structured and object-oriented design.
9. Coding and integration.
10. Verification, validation, and testing.
11. Managing, planning and controlling a software project.
13. Documentation standards (e.g. IEEE).

Outcomes

Basic understanding of:
- The significance of the main characteristics of software product in its development.
- The application of the fundamental principles of software engineering process.
- The differences and benefits of various process models for software development.
- The techniques for modeling the system & software requirements and their analysis and the application of selected techniques & tools.
- Both the larger issues of project management and specific approaches to software development from specification and design, through to implementation, testing, and documentation.
- The managerial, engineering and technical activities of producing a software product.
- The knowledge and application of software development process model in a team environment.
- The key differences between the structured and object-oriented approaches for software analysis and design.

Exposure to:
- The use of the best practices associated with various aspects software engineering.
- An understanding of the significance and importance of current trends, ethical/performance issues in the software engineering field.

Laboratory Projects

1. Analyze a software system starting from a concept of operations document and delivering tested code.
2. Apply the concepts of a lifecycle process, software engineering development methods, and management, planning and control of the projects.
3. Students will be asked to produce code written in a specific language.

Estimated Curriculum Category Content (Semester hours)

<table>
<thead>
<tr>
<th>Area</th>
<th>Core</th>
<th>Advanced</th>
<th>Area</th>
<th>Core</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td></td>
<td></td>
<td>Data Structures</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Design</td>
<td>.5</td>
<td></td>
<td>Prog. Languages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comp. Arch.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Oral and Written Communications

Students are required to give one or two short presentations about their term project lessons learned throughout the development process. Students are also required to document the various software product artifacts (SRA, SDD, STP, etc.).

Social and Ethical Issues

The role of software engineering in developing software products and their use in academia and industry alike are addressed. Software quality and security issues are discussed. (Total: 3 hours.)

For more on social and academic issues students are encouraged to read the following referenced documents:

- [http://www.acm.org/serving/se/code.htm](http://www.acm.org/serving/se/code.htm).
- [http://www.acm.org/constitution/code.html](http://www.acm.org/constitution/code.html).

Theoretical Content

1. Lifecycle process (3 hours).
2. System engineering model (3 hours).
3. Classic management model (1 hour).

Problem Analysis

1. Software requirements elicitation and analysis (9 hours).
2. Structured analysis and design (3 hours).
3. Software design process (6 hours).
4. Object-Oriented Analysis and design (6 hours).
5. Unified Modeling Language-- UML (6 hours).
6. Software verification (3 hours).
7. Other software development methods (3 hours).
Solution Design

1. Software requirements elicitation and analysis (9 hours).
2. Structured analysis and design (3 hours).
3. Software design process (6 hours).
4. Object-Oriented Analysis and design (6 hours).
5. Unified Modeling Language-- UML (6 hours).
6. Software verification (3 hours).
7. Other software development methods (3 hours).