EECE 637 – Advanced Programming Practices (3 credits)

Catalog description:

This course is an advanced course on programming practices with a focus on verification. The course introduces programming tools and techniques that make individual engineers more effective and productive and help them develop quality code. Teams will work in Agile and eXtreme programming environments with a focus on design by contract. They will use formal specifications, design patterns and aspect oriented programming. Projects will use tools for code control, building, configuration, language recognition, dynamic documentation, fast prototyping, refinement, coverage, automated and manual debugging, and dynamic and static verification.

Areas: Software engineering, verification

Required or Elective:

Elective for CCE / ECE

Level: Third year, senior or graduate standing

Prerequisites:

By topic: Students are expected to have basic knowledge of data structures, and considerable programming experience. Software engineering is a plus.

Textbooks:

The course will use readings from the literature and will also cover material from the following two textbooks.

• The Practice of Programming. Brian Kernighan and Rob Pike. 1999, Addison-

Wesley.

• Essential Open Source Toolset. Andreas Zeller and Jens Krinke, 2005, John-Wiley.

Course objectives:

The objectives of this course are to give students:	*Correlates to Program Educational Objectives*
Knowledge and practice of the modern programming and software design paradigms.	**1,2,4**
Advanced knowledge and application of modern team work methodologies in software engineering and their relation to verification.	**1,3,4**
State of the art knowledge of formal programming tools.	**1,2,3**
Experience in building, writing and presenting publication quality systems and papers.	**2,3,4**

Topics

No.	**Subjects covered**	**75 Min. Lectures**
1	Origin of programming languages	2
2	Source control, maintenance, build, documentation	2
3	Programming paradigms – Agile, eXtreme, design by contract	2
4	Modern techniques – design patterns	2
5	Modern techniques – aspect oriented programming	2
6	Project Presentations	2

7	Language recognition	2
8	Formal tools	3
9	Software quality metrics	1
10	Prototyping, refinement, and refactoring	2
11	Coverage, manual and automated debugging	2
12	Project Presentations	2
13	Advanced topics in programming	2

Class/laboratory schedule

a) Two 75-minute lectures per week.

b) Use of computer lab or personal computer is needed for working on the projects.

Course outcomes:

At the end of the course students should be able to:	***Correlates to Program Outcomes***		
	H	M	L
1. Build industrial and commercial software systems from scratch	**a, c, e**	**n**	**g,m**
2. Formally specify systems and properties and check them	**a**	**m, n**	
3. Find and resolve bugs and problems in computing systems	**e** ,b	**a, m, n**	

4. Work in distributed teams and use lightweight productivity tools	**c, d**	g	k
5. Use common design patterns to solve computing problems and facilitate verification	**b,e**	**A**	**k**
6. Use Aspect oriented programming to design and augment functionality of new and existing computing systems and facilitate verification	**a,k**	**j, k**	**c, e**
7. Automate the system building and maintenance cycle	**a,j,k**		
8. Customize and augment existing programming tools and contribute to open source products	**c,h**	i	k
9. Write and present publication quality projects	**g**		
10. Evaluate and review software systems using objective quality metrics	**b,e**		N
11. Reuse and integrate existing solutions to minimize the cost of designing new products	**c,k**	**j**	**E**

Resources for the course:

Books, arctiles, publications, online material

Evaluation:

1. Class participation: 15 %
2. Exams: 25 %
3. Projects: 60 %

Students will work in teams to simulate several programming paradigms and use open source tools to build and submit three software projects. Projects will be software systems that address current problems and probably needed software development tools and will cover subjects discussed in class such as language recognition, documentation, formal specifications and coverage…Students will have one exam that will test their understanding of the concepts taught in the class. Good class projects should result in quality work as expected in conferences and workshops in the Field. Students may work on ideas of their own after consulting with the course instructor.

Professional component:

Engineering topics: 80%

General education: 10%

Mathematics and basic sciences: 10%

Computer usage:

Students pick their platforms and programming languages.

Person(s) who prepared this description and date of preparation:

Fadi Zaraket, Oct 2009