UNIVERSIDAD POLITÉCNICA DE MADRID
FACULTAD DE INFORMÁTICA
Campus de Montegancedo
Boadilla del Monte. 28660 Madrid

Software Design

Learning Guide – Information for Students

1. Description

Grade	Máster Universitario en Ingeniería de Software/ European Master on Software Engineering
Module	Software Development
Area	…
Subject	Software Design
Type	Compulsory
ECTS credits	4
Responsible department	Computer Languages and Systems and Software Engineering
Major/Section/	…

Academic year	2012/2013
Term	2nd term
Language	English
Web site	…

2. Faculty

NAME and SURNAME	OFFICE	email
Nelson Medinilla Martínez (Coord.)	5109	nelson@fi.upm.es

3. Prior knowledge required to take the subject

Passed subjects	• …
Other required learning outcomes	• …

4. Learning goals

SUBJECT-SPECIFIC COMPETENCES AND PROFICIENCY LEVEL		
Code	**Competence**	**Level**
SC12	To conceive and perform the design of software systems, assuring relevant quality attributes.	S

Proficiency level: knowledge (K), comprehension (C), application (A), and analysis and synthesis (S)

SUBJECT LEARNING OUTCOMES			
Code	**Learning outcome**	**Related competences**	**Profi-ciency level**
LR1	The student will be able to design a software system according to requirements, restrictions, quality standards, and developer criteria.	SC12	S
LR2	The student will be able to document each new design.	SC12	S
LR3	The student will be able to evaluate any software system design.	SC12	A

5. Subject assessment system

ACHIEVEMENT INDICATORS		
Ref	**Indicator**	**Related to LR**
I1	Design a small software system with client interaction. The system should work and fulfill the requirements, restrictions, quality standards and developer criteria.	LR1
I2	Make up a document, effective and efficient, about the software design.	LR2
I3	Evaluate her/his own design and any other software system design.	LR3

CONTINUOUS ASSESSMENT			
Brief description of assessable activities	**Time**	**Place**	**Weight in grade**
Present the design developed up to date	8th week	classroom	10%
Present and hand in the final design together its document.	16th week	classroom	90%
			Total: 100%

GRADING CRITERIA

The Software Design subject is practical, based on solving problems, especially, the development of small software systems. It is taught though seminar/workshop. The systems are developed in group. Each group works as software developer and client of another group.

The software system should satisfy their client and the criteria set out in I1 indicator. The evaluation of this system is the principal measure to grade the student respect to I1 indicator. The replays to oral questions complete the grade.

The I2 indicator on documentation will be graded according to the quality of the written document.

The I3 indicator on evaluation will be graded by the self-criticism, include in the written document, and previous evaluations of other designs.

Ninety percent of the grading will be done at the end of the term. This is because software development is evolutionary. Ten percent of the grading will be done at half way through the term by a formal presentation of the systems developed up to date.

The software system development is evolutionary for two reasons. The first one is technical: the systems are development in new or unknown contexts (from the student point of view). Moreover, this contexts are dynamic; i.e. its change their needs, frequently. The second reason is the learning process: The systems are modified as learning goes by. In short, the software development is evolutionary because the uncertainty presence.

6. Contents and learning activities

SPECIFIC CONTENTS		
Unit / Topic / Chapter	**Section**	**Related indicators**
Chapter 1: Software Engineering Two-dimensional Complexity	1.1 Software Engineering Complexity Concept Evolution. Software Engineering needs a Holistic Approach.	I1, I3
	1.2 Uncertainty as Tool.	I1, I3
	1.3 Relationships between software, design and process models in the uncertainty dimension. Evolutionary Approach.	I1, I3
Chapter 2: System Software Design Features	2.1 Software as Design. Review of software design concept.	I1, I3
	2.2 Software Design from the system point of view. Relationships between software and other kinds of systems. Software Design based on the System General Theory. System structures.	I1, I3
	2.3 Divide and Conquer as systems simplification tool.	I1, I3
	2.4 Ambiguity as powerful systems simplification tool.	I1, I3
	2.5 Information Hiding Principle. The Ambiguity or Indifferent (don’t care) Dependent Relationship.	I1, I3
	2.6 Design Simplification by decrease the quantity of information using abstractions, symmetries, monotonic structures, and others similar techniques.	I1, I3
	2.7 Influence of design structure on design properties. Allotropy.	I1, I3
Chapter 3: Object Oriented	3.1 Contrast between object and structured models.	I1, I3

Review	3.2 Ambiguity in: object, message, class and heritage. Substitution Liskov Principle.	I1, I3
	3.3 Evolutionary Design using Objects.	I1, I3
	3.4 Use Cases Technique Review.	I1, I3
Chapter 4: Design and Dominion Patterns	4.1 Theoretical foundations of patterns.	I1, I3
	4.2 Analysis of Design Patterns.	I1, I3
	4.3 Domain Patterns.	I1, I3
Chapter 5: Design Documentation	5.1 Documentation reasons.	I1, I2, I3
	5.2 Software Design Legibility using objects. Suitable use of abstractions and their structure.	I1, I2, I3
	5.3 Design Diagram Simplification using abstractions, patterns, symmetries and monotonic structures.	I1, I2, I3
	5.4 Evolutionary Documentation.	I1, I2, I3

7. Brief description of organizational modalities and teaching methods

TEACHING ORGANIZATION		
Scenario	**Organizational Modality**	**Purpose**
	Theory Classes	***Talk to students***
	Seminars/Workshops	***Construct knowledge through student interaction and activity***
	Practical Classes	***Show students what to do***
	Placements	***Round out student training in a professional setting***
	Personal Tutoring	***Give students personalized attention***
	Group Work	***Get students to learn from each other***
	Independent Work	***Develop self-learning ability***

POLITÉCNICA

UNIVERSIDAD POLITÉCNICA DE MADRID
FACULTAD DE INFORMÁTICA
Campus de Montegancedo
Boadilla del Monte. 28660 Madrid

TEACHING METHODS			
	Method	**Purpose**	
	Explanation/Lecture	***Transfer information and activate student cognitive processes***	Known as explanation, this teaching method involves the "*presentation of a logically structured topic with the aim of providing information organized according to criteria suited for the purpose*". This methodology, also known as *lecture*, mainly focuses on the verbal exposition by the teacher of contents on the subject under study. The term *master class* is often used to refer to a special type of lecture taught by a professor on special occasions
	Case Studies	***Learning by analyzing real or simulated case studies***	Intensive and exhaustive analysis of a real fact, problem or event for the purpose of understanding, interpreting or solving the problem, generating hypotheses, comparing data, thinking, learning or diagnosis and, sometimes, training in possible alternative problem-solving procedures.
	Exercises and Problem Solving	***Exercise, test and practice prior knowledge***	Situations where students are asked to develop the suitable or correct solutions by exercising routines, applying formulae or running algorithms, applying information processing procedures and interpreting the results. It is often used to supplement lectures.
	Problem-Based Learning (PBL)	***Develop active learning through problem solving***	Teaching and learning method whose starting point is a problem, designed by the teacher, that the student has to solve to develop a number of previously defined competences.
	Project-Oriented Learning (POL)	***Complete a problem-solving project applying acquired skills and knowledge***	Teaching and learning method where have a set time to develop a project to solve a problem or perform a task by planning, designing and completing a series of activities. The whole thing is based on developing and applying what they have learned and making effective use of resources.
	Cooperative Learning	***Develop active and meaningful learning through cooperation***	Interactive approach to the organization of classroom work where students are responsible for their own and their peers' learning as part of a co-responsibility strategy for achieving group goals and incentives. This is both one of a number of methods for use and an overall teaching approach, or philosophy.
	Learning Contract	***Develop independent learning***	An agreement between the teacher and student on the achievement of learning outcomes through an independent work proposal, supervised by the teacher, and to be accomplished within a set period. The essential points of a learning contract are that it is a written agreement, stating required work and reward, requiring personal involvement and having a time frame for accomplishment.

BRIEF DESCRIPTION OF THE ORGANIZATIONAL MODALITIES AND TEACHING METHODS	
THEORY CLASSES	Theoretical models used in the subject are analyzed.
PROBLEM-SOLVING CLASSES	…
PRACTICAL WORK	…
INDIVIDUAL WORK	Individual works come from group work.
GROUP WORK	Group work is the subject principal work way because software engineer works in groups and because group work is the main way of learning in this subject.
PERSONAL TUTORING	Personal Tutoring could be wanted in order to improve the learning process.

8. Teaching resources

TEACHING RESOURCES	
RECOMMENDED READING	Gamma Erich et al. "Design Patterns" Ed. Addison Wesley 1994
	Larman Craig "Applying UML and Patterns" Second Edition. Prentice Hall 2002.
	Parnas David "On the Criteria To Be Used in Decomposition Systems and Modules" Com. ACM Dec. 1972 Vol. 15 Nº 12 pp. 1053-1058.
WEB RESOURCES	Subject web site (http://)
	Subject Moodle site (http://)
EQUIPMENT	Laboratory
	Room 6202
	Group work room 6202

BRIEF DESCRIPTION OF THE ORGANIZATIONAL MODALITIES AND TEACHING METHODS	
THEORY CLASSES	Theoretical models used in the subject are analyzed.
PROBLEM-SOLVING CLASSES	…
PRACTICAL WORK	…
INDIVIDUAL WORK	Individual works come from group work.
GROUP WORK	Group work is the subject principal work way because software engineer works in groups and because group work is the main way of learning in this subject.
PERSONAL TUTORING	Personal Tutoring could be wanted in order to improve the learning process.

8. Teaching resources

TEACHING RESOURCES	
RECOMMENDED READING	Gamma Erich et al. “Design Patterns” Ed. Addison Wesley 1994
	Larman Craig “Applying UML and Patterns” Second Edition. Prentice Hall 2002.
	Parnas David “On the Criteria To Be Used in Decomposition Systems and Modules” Com. ACM Dec. 1972 Vol. 15 Nº 12 pp. 1053-1058.
WEB RESOURCES	Subject web site (http://)
	Subject Moodle site (http://)
EQUIPMENT	Laboratory
	Room 6202
	Group work room 6202

9. Subject schedule

Week	Classroom activities	Lab activities	Individual work	Group work	Assessment activities	Others
Week 1 (7 hours)	• Theory Class (Ch. 1) • Set up the groups. (2 hours)		• Individual work (2 hours)	• Make up the software system request. (3 hours)		
Week 2 (7 hours)	• Request and Negotiation about the System will be developed. (2 hours)		• Software Development (1 hour)	• Analysis of request and synthesis of possible solutions. Make an initial decision. Delivery work. (4 hours)		
Week 3 (7 hours)	• Theory Class (Ch. 2) (2 hours)		• Software Development (3 hours)	• Software Development (2 hours)		
Week 4 (7 hours)	• Software development in groups with client interaction. • Public discussion about design developed up to date. (Ch. 2, 3, 5) (2 hours)		• Software Development (3 hours)	• Software Development (2 hours)		

Week 5 (7 hours)	• Software development in groups with client interaction. • Public discussion about design developed up to date. (Ch. 2, 3, 5) (2 hours)		• Software Development (3 hours)	• Software Development (2 hours)		
Week 6 (8 hours)	• Software development in groups with client interaction. • Public discussion about design developed up to date. (Ch. 2, 3, 5) (2 hours)		• Software Development (3 hours)	• Software Development (3 hours)		
Week 7 (7 hours)	• Software development in groups with client interaction. • Public discussion about design developed up to date. (Ch. 2, 3, 5) (2 hours)		• Preparing the First Evaluation (2 hours)	• Preparing the First Evaluation (3 hours)		
Week 8 (7 hours)	• The assessment activity is in classroom.		• Preparing the First Evaluation (2 hours)	• Preparing the First Evaluation (3 hours)	• First Oral Evaluation (2 hours)	
Week 9 (7 hours)	• Public re-analysis of discussed design. (2 hours)		• Analysis of design. (1 hour)	• Analysis of design. As client, make up the request of changes to wanted system. (4 hours).		

Week 10 (7 hours)	• Request and Negotiation about the changes will be done. . (2 hours)		• Individual work. (1 hour)	• Analysis of changes wanted and synthesis of possible solutions. Make a decision. Delivery work. (4 hours)		
Week 11 (7 hours)	• Theory class (Ch. 4) (2 hours)		• Software development. (3 hours)	• Software development. (2 hours)		
Week 12 (7 hours)	• Software development in groups with client interaction. • Public discussion about design developed up to date. (Ch. 2, 3, 4, 5) (2 hours)		• Software development. (3 hours)	• Software development. (2 hours)		
Week 13 (7 hours)	• Software development in groups with client interaction. • Public discussion about design developed up to date. (Ch. 2, 3, 4, 5) (2 hours)		• Software development. (3 hours)	• Software development. (2 hours)		
Week 14 (7 hours)	• Software development in groups with client interaction. • Public discussion about design developed up to date. (Ch. 2, 3, 4, 5) (2 hours)		• Software development. (3 hours)	• Software development. (2 hours)		

Week 15 (7 hours)	• Software development in groups with client interaction. • Public discussion about design developed up to date. (Ch. 2, 3, 4, 5) (2 hours)		• Preparing the Final Evaluation (2 hours)	• Preparing the Final Evaluation (3 hours)		
Week 16 (7 hours)	• The assessment activity is in classroom.		• Preparing the Final Evaluation (2 hours)	• Preparing the Final Evaluation (3 hours)	• Final Evaluation. (2 hours)	

Note: Student workload specified for each activity in hours