



## 1. COURSE

CS292. Software Engineering II (Mandatory)

## 2. GENERAL INFORMATION

- 2.1 Course : CS292. Software Engineering II  
2.2 Semester : 6<sup>to</sup> Semestre.  
2.3 Credits : 4  
2.4 Horas : 2 HT; 4 HP;
- 2.5 Duration of the period : 16 weeks  
2.6 Type of course : Mandatory  
2.7 Learning modality : Blended  
2.8 Prerequisites : CS291. Software Engineering I. (5<sup>th</sup> Sem)  
CS291. Software Engineering I. (5<sup>th</sup> Sem)

## 3. PROFESSORS

Meetings after coordination with the professor

## 4. INTRODUCTION TO THE COURSE

The topics of this course extend the ideas of software design and development from the introduction sequence to programming to encompass the problems encountered in large-scale projects. It is a broader and more complete view of Software Engineering appreciated from a Project point of view.

## 5. GOALS

- Enable students to be part of and define software development teams facing real-world problems.
- familiarize the students with the process of administering a software project in such a way as to be able to create, improve and use tools and metrics that allow them to carry out the estimation and monitoring of a software project
- Create, evaluate and execute a test plan for medium-sized code segments, Distinguish between different types of tests, lay the foundation for creating, improve test procedures and tools for these purposes
- Select with justification an appropriate set of tools to support the development of a range of software products.
- Create, improve and use existing patterns for software maintenance. Disclose features and design patterns for software reuse.
- Identify and discuss different specialized systems, create, improve and use specialized standards for the design, implementation, maintenance and testing of specialized systems.

## 6. COMPETENCES

- 1) Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions. (**Usage**)
- 2) Design, implement and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (**Usage**)
- 3) Communicate effectively in a variety of professional contexts. (**Usage**)
- 6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (**Assessment**)

## 7. TOPICS

Unit 1: Tools and Environments (12)	
Competences Expected:	
Topics	Learning Outcomes
<ul style="list-style-type: none"> <li>• Software configuration management and version control</li> <li>• Release management</li> <li>• Requirements analysis and design modeling tools</li> <li>• Testing tools including static and dynamic analysis tools</li> <li>• Programming environments that automate parts of program construction processes (e.g., automated builds) <ul style="list-style-type: none"> <li>– Continuous integration</li> </ul> </li> <li>• Tool integration concepts and mechanisms</li> </ul>	<ul style="list-style-type: none"> <li>• Software configuration management and version control [Usage]</li> <li>• Release management [Usage]</li> <li>• Requirements analysis and design modeling tools [Usage]</li> <li>• Testing tools including static and dynamic analysis tools [Usage]</li> <li>• Programming environments that automate parts of program construction processes (e.g., automated builds) <ul style="list-style-type: none"> <li>– Continuous integration</li> </ul> [Usage] </li> <li>• Tool integration concepts and mechanisms [Usage]</li> </ul>
<b>Readings :</b> [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unit 2: Software Verification and Validation (12)	
Competences Expected:	
Topics	Learning Outcomes
<ul style="list-style-type: none"> <li>• Verification and validation concepts</li> <li>• Inspections, reviews, audits</li> <li>• Testing types, including human computer interface, usability, reliability, security, conformance to specification</li> <li>• Testing fundamentals <ul style="list-style-type: none"> <li>– Unit, integration, validation, and system testing</li> <li>– Test plan creation and test case generation</li> <li>– Black-box and white-box testing techniques</li> <li>– Regression testing and test automation</li> </ul> </li> <li>• Defect tracking</li> <li>• Limitations of testing in particular domains, such as parallel or safety-critical systems</li> <li>• Static approaches and dynamic approaches to verification</li> <li>• Test-driven development</li> <li>• Validation planning; documentation for validation</li> <li>• Object-oriented testing; systems testing</li> <li>• Verification and validation of non-code artifacts (documentation, help files, training materials)</li> <li>• Fault logging, fault tracking and technical support for such activities</li> <li>• Fault estimation and testing termination including defect seeding</li> </ul>	<ul style="list-style-type: none"> <li>• Distinguish between program validation and verification [Usage]</li> <li>• Describe the role that tools can play in the validation of software [Usage]</li> <li>• Undertake, as part of a team activity, an inspection of a medium-size code segment [Usage]</li> <li>• Describe and distinguish among the different types and levels of testing (unit, integration, systems, and acceptance) [Usage]</li> <li>• Describe techniques for identifying significant test cases for integration, regression and system testing [Usage]</li> <li>• Create and document a set of tests for a medium-size code segment [Usage]</li> <li>• Describe how to select good regression tests and automate them [Usage]</li> <li>• Use a defect tracking tool to manage software defects in a small software project [Usage]</li> <li>• Discuss the limitations of testing in a particular domain [Usage]</li> <li>• Evaluate a test suite for a medium-size code segment [Usage]</li> <li>• Compare static and dynamic approaches to verification [Usage]</li> <li>• Identify the fundamental principles of test-driven development methods and explain the role of automated testing in these methods [Usage]</li> <li>• Discuss the issues involving the testing of object-oriented software [Usage]</li> <li>• Describe techniques for the verification and validation of non-code artifacts [Usage]</li> <li>• Describe approaches for fault estimation [Usage]</li> <li>• Estimate the number of faults in a small software application based on fault density and fault seeding [Usage]</li> <li>• Conduct an inspection or review of software source code for a small or medium sized software project [Usage]</li> </ul>
<b>Readings :</b> [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

<b>Unit 3: Software Evolution (12)</b>	
<b>Competences Expected:</b>	
<b>Topics</b>	<b>Learning Outcomes</b>
<ul style="list-style-type: none"> <li>• Software development in the context of large, pre-existing code bases <ul style="list-style-type: none"> <li>– Software change</li> <li>– Concerns and concernlocation</li> <li>– Refactoring</li> </ul> </li> <li>• Software evolution</li> <li>• Characteristics of maintainable software</li> <li>• Reengineering systems</li> <li>• Software reuse <ul style="list-style-type: none"> <li>– Code segments</li> <li>– Libraries and frameworks</li> <li>– Components</li> <li>– Product lines</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Identify the principal issues associated with software evolution and explain their impact on the software lifecycle [Usage]</li> <li>• Estimate the impact of a change request to an existing product of medium size [Usage]</li> <li>• Use refactoring in the process of modifying a software component [Usage]</li> <li>• Discuss the challenges of evolving systems in a changing environment [Usage]</li> <li>• Outline the process of regression testing and its role in release management [Usage]</li> <li>• Discuss the advantages and disadvantages of different types of software reuse [Usage]</li> </ul>
<b>Readings :</b> [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unit 4: Software Project Management (12)	
Competences Expected:	
Topics	Learning Outcomes
<ul style="list-style-type: none"> <li>• Team participation <ul style="list-style-type: none"> <li>– Team processes including responsibilities for task, meeting structure, and work schedule</li> <li>– Roles and responsibilities in a software team</li> <li>– Team conflict resolution</li> <li>– Risks associated with virtual teams (communication, perception, structure)</li> </ul> </li> <li>• Effort estimation (at the personal level)</li> <li>• Risk <ul style="list-style-type: none"> <li>– The role of risk in the lifecycle</li> <li>– Risk categories including security, safety, market, financial, technology, people, quality, structure and process</li> </ul> </li> <li>• Team management <ul style="list-style-type: none"> <li>– Team organization and decision-making</li> <li>– Role identification and assignment</li> <li>– Individual and team performance assessment</li> </ul> </li> <li>• Project management <ul style="list-style-type: none"> <li>– Scheduling and tracking</li> <li>– Project management tools</li> <li>– Cost/benefit analysis</li> </ul> </li> <li>• Software measurement and estimation techniques</li> <li>• Software quality assurance and the role of measurements</li> <li>• Risk <ul style="list-style-type: none"> <li>– Risk identification and management</li> <li>– Risk analysis and evaluation</li> <li>– Risk tolerance (e.g., risk-adverse, risk-neutral, risk-seeking)</li> <li>– Risk planning</li> </ul> </li> <li>• System-wide approach to risk including hazards associated with tools</li> </ul>	<ul style="list-style-type: none"> <li>• Discuss common behaviors that contribute to the effective functioning of a team [Usage]</li> <li>• Create and follow an agenda for a team meeting [Usage]</li> <li>• Identify and justify necessary roles in a software development team [Usage]</li> <li>• Understand the sources, hazards, and potential benefits of team conflict [Usage]</li> <li>• Apply a conflict resolution strategy in a team setting [Usage]</li> <li>• Use an ad hoc method to estimate software development effort (eg, time) and compare to actual effort required [Usage]</li> <li>• List several examples of software risks [Usage]</li> <li>• Describe the impact of risk in a software development lifecycle [Usage]</li> <li>• Describe different categories of risk in software systems [Usage]</li> <li>• Demonstrate through involvement in a team project the central elements of team building and team management [Usage]</li> </ul>
<b>Readings :</b> [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

## 8. WORKPLAN

### 8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

### 8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students

to internalize the concepts.

### 8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

## 9. EVALUATION SYSTEM

\*\*\*\*\* EVALUATION MISSING \*\*\*\*\*

## 10. BASIC BIBLIOGRAPHY

- [Amb01] Vincenzo Ambriola. *Software Process Technology*. Springer, July 2001.
- [Blu92] Bruce I. Blum. *Software Engineering: A Holistic View*. 7th. Oxford University Press US, May 1992.
- [Con00] R. Conradi. *Software Process Technology*. Springer, Mar. 2000.
- [Key04] Jessica Keyes. *Software Configuration Management*. CRC Press, Feb. 2004.
- [Mon96] Carlo Montangero. *Software Process Technology*. Springer, Sept. 1996.
- [Oqu03] Flavio Oquendo. *Software Process Technology*. Springer, Sept. 2003.
- [Pre04] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. 6th. McGraw-Hill, Mar. 2004.
- [PS01] John W. Priest and Jose M. Sanchez. *Product Development and Design for Manufacturing*. Marcel Dekker, Jan. 2001.
- [Sch04] Stephen R Schach. *Object-Oriented and Classical Software Engineering*. McGraw-Hill, Jan. 2004.
- [WA02] Daniel R. Windle and L. Rene Abreo. *Software Requirements Using the Unified Process*. Prentice Hall, Aug. 2002.
- [WK00] Yingxu Wang and Graham King. *Software Engineering Processes: Principles and Applications*. CRC Press, Apr. 2000.