San Pablo Catholic University (UCSP)
Undergraduate Program in
Computer Science
SILABO

CS292. Software Engineering II (Mandatory)

2021-I

1. General information		
1.1 School	:	Ciencia de la Computación
1.2 Course	:	CS292. Software Engineering II
1.3 Semester	:	6^{to} Semestre.
1.4 Prerrequisites	:	CS291. Software Engineering I. (5^{th} Sem)
1.5 Type of course	:	Mandatory
1.6 Learning modality	:	Virtual
1.7 Horas	:	2 HT; 2 HP; 2 HL;
1.8 Credits	:	4

2. Professors

Lecturer

- Guillermo Enrique Calderón Ruiz <gcalderon@ucsp.edu.pe>
  - PhD in Ciencias de la Ingeniería, Pontificia Universidad Católica de Chile, Chile, 2011.
  - MSc in Ingeniería, Pontificia Universidad Católica de Chile, Chile, 2010.

3. Course foundation

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.

4. Summary

1. Tools and Environments 2. Software Verification and Validation 3. Software Evolution 4. Software Project Management

5. Generales 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. Contribution to Outcomes

This discipline contributes to the achievement of the following outcomes:

b) An ability to design and conduct experiments, as well as to analyze and interpret data. (**Usage**)

c) An ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability. (**Usage**)

f) An ability to communicate effectively. (**Usage**)

i) An ability to use the techniques, skills, and modern computing tools necessary for computing practice. (**Assessment**)

k) Apply the principles of development and design in the construction of software systems of variable complexity. (**Usage**)

7. Content

UNIT 1: Tools and Environments (12)

Competences: c,f,i

Content	**Generales Goals**

• Software configuration management and version control
• Release management
• Requierements analysis and desing modeling tools
• Testing tools including static and dynamic analysis tools
• Programming enviroments that automate parts of program construction pocesses (e.g., automated builds)
  – Continuous integration
• Tool integration concepts and mechanisms
	• Software configuration management and version control [Usage]
	• Release management [Usage]
	• Requierements analysis and desing modeling tools [Usage]
	• Testing tools including static and dynamic analysis tools [Usage]
	• Programming enviroments that automate parts of program construction pocesses (e.g., automated builds)
	  – Continuous integration
	[Usage]
	• Tool integration concepts and mechanisms [Usage]

Readings: Pressman (2004), Blum (1992), Schach (2004), Wang and King (2000), Keyes (2004), Windle and Abreo (2002), Priest and Sanchez (2001), Schach (2004), Montangero (1996), Ambriola (2001), Conradi (2000), Oquendo (2003)

UNIT 2: Software Verification and Validation (12)

Competences: c,f,i

Content	**Generales Goals**

• Verification and validation concepts
• Inspections, reviews, audits
• Testing types, including human computer interface, usability, reliability, security, conformance to specification
• Testing fundamentals
  – Unit, integration, validation, and system testing
  – Test plan creation and test case generation
  – Black-box and white-box testing techniques
  – Regression testing and test automation
• Defect tracking
• Limitations of testing in particular domains, such as parallel or safety-critical systems
• Static approaches and dynamic approaches to verification
• Test-driven development
• Validation planning; documentation for validation
• Object-oriented testing; systems testing
• Verification and validation of non-code artifacts (documentation, help files, training materials)
• Fault logging, fault tracking and technical support for such activities
• Fault estimation and testing termination including defect seeding
	• Distinguish between program validation and verification [Usage]
	• Describe the role that tools can play in the validation of software [Usage]
	• Undertake, as part of a team activity, an inspection of a medium-size code segment [Usage]
	• Describe and distinguish among the different types and levels of testing (unit, integration, systems, and acceptance) [Usage]
	• Describe techniques for identifying significant test cases for integration, regression and system testing [Usage]
	• Create and document a set of tests for a medium-size code segment [Usage]
	• Describe how to select good regression tests and automate them [Usage]
	• Use a defect tracking tool to manage software defects in a small software project [Usage]
	• Discuss the limitations of testing in a particular domain [Usage]
	• Evaluate a test suite for a medium-size code segment [Usage]
	• Compare static and dynamic approaches to verification [Usage]
	• Identify the fundamental principles of test-driven development methods and explain the role of automated testing in these methods [Usage]
	• Discuss the issues involving the testing of object-oriented software [Usage]
	• Describe techniques for the verification and validation of non-code artifacts [Usage]
	• Describe approaches for fault estimation [Usage]
	• Estimate the number of faults in a small software application based on fault density and fault seeding [Usage]
	• Conduct an inspection or review of software source code for a small or medium sized software project [Usage]

Readings: Pressman (2004), Blum (1992), Schach (2004), Wang and King (2000), Keyes (2004), Windle and Abreo (2002), Priest and Sanchez (2001), Schach (2004), Montangero (1996), Ambriola (2001), Conradi (2000), Oquendo (2003)

UNIT 3: Software Evolution (12)	
Competences: c,f,i	
Content	**Generales Goals**
• Software development in the context of large, pre-existing code bases – Software change – Concerns and concernlocation – Refactoring • Software evolution • Characteristics of maintainable software • Reengineering systems • Software reuse – Code segments – Libraries and frameworks – Components – Product lines	• Identify the principal issues associated with software evolution and explain their impact on the software lifecycle [Usage] • Estimate the impact of a change request to an existing product of medium size [Usage] • Use refactoring in the process of modifying a software component [Usage] • Discuss the challenges of evolving systems in a changing environment [Usage] • Outline the process of regression testing and its role in release management [Usage] • Discuss the advantages and disadvantages of different types of software reuse [Usage]
Readings: Pressman (2004), Blum (1992), Schach (2004), Wang and King (2000), Keyes (2004), Windle and Abreo (2002), Priest and Sanchez (2001), Schach (2004), Montangero (1996), Ambriola (2001), Conradi (2000), Oquendo (2003)	

UNIT 4: Software Project Management (12)

Competences: c,f,i

Content	**Generales Goals**
• Team participation – Team processes including responsabilities for task, meeting structure, and work schedule – Roles and responsabilities in a software team – Team conflict resolution – Risks associated with virtual teams (communication, perception, structure) • Effort estimation (at the personal level) • Risk – The role of risk in the lifecycle – Risk categories including security, safety, market, financial, technology, people, quality, structure and process • Team management – Team organization and decision-making – Role identification and assigment – Individual and team performance assessment • Project management – Scheduling and tracking – Project management tools – Cost/benefit analysis • Software measurement and estimation techniques • Software quality assurance and the role of measurements • Risk – Risk identification and management – Risk analysis and evaluation – Risk tolerance (e.g., risk-adverse, risk-neutral, risk-seeking) – Risk planning • System-wide approach to risk including hazards associated with tools	• Discuss common behaviors that contribute to the effective functioning of a team [Usage] • Create and follow an agenda for a team meeting [Usage] • Identify and justify necessary roles in a software development team [Usage] • Understand the sources, hazards, and potential benefits of team conflict [Usage] • Apply a conflict resolution strategy in a team setting [Usage] • Use an ad hoc method to estimate software development effort (eg, time) and compare to actual effort required [Usage] • List several examples of software risks [Usage] • Describe the impact of risk in a software development lifecycle [Usage] • Describe different categories of risk in software systems [Usage] • Demonstrate through involvement in a team project the central elements of team building and team management [Usage]

Readings: Pressman (2004), Blum (1992), Schach (2004), Wang and King (2000), Keyes (2004), Windle and Abreo (2002), Priest and Sanchez (2001), Schach (2004), Montangero (1996), Ambriola (2001), Conradi (2000), Oquendo (2003)

8. Methodology

El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.

El profesor del curso presentará demostraciones para fundamentar clases teóricas.

El profesor y los alumnos realizarán prácticas

Los alumnos deberán asistir a clase habiendo leído lo que el profesor va a presentar. De esta manera se facilitará la comprensión y los estudiantes estarán en mejores condiciones de hacer consultas en clase.

9. Assessment

Continuous Assessment 1 : 20 %

Partial Exam : 30 %

Continuous Assessment 2 : 20 %

Final exam : 30 %

References

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