Universidad Nacional Mayor de San Marcos
School of Computer Science
Syllabus of Course
Academic Period 2018-II

1. **Code and Name:** CS2902. Software Engineering II (Elective)
2. **Credits:** 4
3. **Hours of theory and Lab:** 2 HT; 4 HL; (15 weeks)
4. **Professor(s)**

   Meetings after coordination with the professor
5. **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.

6. **Information about the course**

   (a) **Brief description about 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.

   (b) **Prerrequisites:** CS2901. Software Engineering I. (5^{th} Sem)

   (c) **Type of Course:** Elective

   (d) **Modality:** Face to face

7. **Specific goals of the Course**

   - 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.

8. Contribution to Outcomes

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**)

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**)

9. Competences (IEEE)

C7. Being able to apply the software engineering principles and technologies to ensure that software implementations are robust, reliable, and appropriate for their intended audience.⇒ **Outcome c**

C11. Understanding of the concept of the lifecycle, including the significance of its phases (planning, development, deployment, and evolution).⇒ **Outcome c**

C12. Understanding the lifecycle implications for the development of all aspects of computer-related systems (including software, hardware, and human computer interface).⇒ **Outcome c,i**

C18. Ability to participate actively and as a member of a team. .⇒ **Outcome f**

CS1. Model and design computer-based systems in a way that demonstrates comprehension of the tradeoff involved in design choices.⇒ **Outcome c**

CS2. Identify and analyze criteria and specifications appropriate to specific problems, and plan strategies for their solution.⇒ **Outcome c**

CS4. Deploy appropriate theory, practices, and tools for the specification, design, implementation, and maintenance as well as the evaluation of computer-based systems.⇒ **Outcome c,i**

CS5. Specify, design, and implement computer-based systems.⇒ **Outcome c,i**

CS10. Deploy effectively the tools used for the construction and documentation of software, with particular emphasis on understanding the whole process involved in using computers to solve practical problems. This should include tools for software control including version control and configuration management.⇒ **Outcome i**

C7. Being able to apply the software engineering principles and technologies to ensure that software implementations are robust, reliable, and appropriate for their intended audience.⇒ **Outcome c**

C11. Understanding of the concept of the lifecycle, including the significance of its phases (planning, development, deployment, and evolution).⇒ **Outcome c**

C12. Understanding the lifecycle implications for the development of all aspects of computer-related systems (including software, hardware, and human computer interface).⇒ **Outcome c,i**

C18. Ability to participate actively and as a member of a team. .⇒ **Outcome f**

CS1. Model and design computer-based systems in a way that demonstrates comprehension of the tradeoff involved in design choices.⇒ **Outcome c**

CS2. Identify and analyze criteria and specifications appropriate to specific problems, and plan strategies for their solution.⇒ **Outcome c**

CS4. Deploy appropriate theory, practices, and tools for the specification, design, implementation, and maintenance as well as the evaluation of computer-based systems.⇒ **Outcome c,i**

CS5. Specify, design, and implement computer-based systems.⇒ **Outcome c,i**

CS10. Deploy effectively the tools used for the construction and documentation of software, with particular emphasis on understanding the whole process involved in using computers to solve practical problems. This should include tools for software control including version control and configuration management.⇒ **Outcome i**

10. List of topics

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

11. Methodology and Evaluation
Methodology:

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

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

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of the UTEC Online virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.
Evaluation System:

12. Content

Unit 1: Tools and Environments (12)	
Competences Expected: C1	
Learning Outcomes	**Topics**
• 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]	• 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
Readings : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unit 2: Software Verification and Validation (12)

Competences Expected: C20

Learning Outcomes	Topics
• 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]	• 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

Readings : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]

Unit 3: Software Evolution (12)

Competences Expected: C20

Learning Outcomes	Topics
• 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]	• 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

Readings : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]

Unit 4: Software Project Management (12)

Competences Expected: C24

Learning Outcomes	Topics
• 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]	• 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

Readings : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]