University de Piura (UDEP)
Sillabus 2022-I

1. COURSE

CS392. Tópicos en Ingeniería de Software (Elective)

2. GENERAL INFORMATION

2.1 Credits	:	4
2.2 Theory Hours	:	2 (Weekly)
2.3 Practice Hours	:	2 (Weekly)
2.4 Duration of the period	:	16 weeks
2.5 Type of course	:	Elective
2.6 Modality	:	Face to face
2.7 Prerrequisites	:	CS391. Software Engineering III. (7^{th} Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Software development requires the use of best development practices, IT project management, team management and efficient and rational use of quality assurance and portfolio management frameworks, these elements are part key and transversal for the success of the production process.

This course explores the design, selection, implementation and management of IT solutions in Organizations. The focus is on applications and infrastructure and their application in the business.

5. GOALS

- Understand a variety of frameworks for enterprise architecture analysis and decision making.
- Use techniques to evaluate and manage risk in the company's portfolio.
- Assess and plan the integration of emerging technologies.
- Understand the role and potential of IT to support business process management.
- Understand the different approaches to modeling and improving business processes.
- Describe and understand quality assurance models as a key framework for successful IT projects.
- Understand and apply the IT Governance framework as a key element in managing the Enterprise application portfolio.

6. COMPETENCES

Nooutcomes

Nospecificoutcomes

7. TOPICS

Unit 1: Software Design (18)

Competences Expected: c,d,i,j,m,o

Topics

- System design principles: levels of abstraction (architectural design and detailed design), separation of concerns, information hiding, coupling and cohesion , re-use of standard structures
- Design Paradigms such as structured design (top-down functional decomposition), object-oriented analysis and design, event driven design, component-level design, data-structured centered, aspect oriented, function oriented, service oriented
- Structural and behavioral models of software designs
- Design patterns
- Relationships between requirements and designs: transformation of models, design of contracts, invariants
- Software architecture concepts and standard architectures (e.g. client-server, n-layer, transform centered, pipes-and-filters)
- The use of component desing: component selection, design, adaptation and assembly of components, component and patterns, components and objects (for example, building a GUI using a standar widget set)
- Refactoring designs using design patterns
- Internal design qualities, and models for them: efficiency and performance, redundacy and fault tolerance, traceability of requeriments
- Measurement and analysis of design quality
- Tradeoffs between different aspects of quality
- Application frameworks
- Middleware: the object-oriented paradigm within middleware, object request brokers and marshalling, transaction processing monitors, workflow systems
- Principles of secure design and coding
  - Principle of least privilege
  - Principle of fail-safe defaults
  - Principle of psychological acceptability

Learning Outcomes

- Articulate design principles including separation of concerns, information hiding, coupling and cohesion, and encapsulation [Usage]
- Use a design paradigm to design a simple software system, and explain how system design principles have been applied in this design [Usage]
- Construct models of the design of a simple software system that are appropriate for the paradigm used to design it [Usage]
- Within the context of a single design paradigm, describe one or more design patterns that could be applicable to the design of a simple software system [Usage]
- For a simple system suitable for a given scenario, discuss and select an appropriate design paradigm [Usage]
- Create appropriate models for the structure and behavior of software products from their requirements specifications [Usage]
- Explain the relationships between the requirements for a software product and its design, using appropriate models [Usage]
- For the design of a simple software system within the context of a single design paradigm, describe the software architecture of that system [Usage]
- Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server [Usage]
- Investigate the impact of software architectures selection on the design of a simple system [Usage]
- Apply simple examples of patterns in a software design [Usage]
- Describe a form of refactoring and discuss when it may be applicable [Usage]
- Select suitable components for use in the design of a software product [Usage]
- Explain how suitable components might need to be adapted for use in the design of a software product [Usage]
- Design a contract for a typical small software component for use in a given system [Usage]
- Discuss and select appropriate software architecture for a simple system suitable for a given scenario [Usage]
- Apply models for internal and external qualities in designing software components to achieve an acceptable tradeoff between conflicting quality aspects [Us-

Unit 2: Software Project Management (14)	
Competences Expected: c,d,i,j,m,o	
Topics	**Learning Outcomes**
• 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 – The role of risk in the lifecycle – Risk categories including security, safety, market, financial, technology, people, quality, structure and process • 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] • Describe how the choice of process model affects team organizational structures and decision-making processes [Usage] • Create a team by identifying appropriate roles and assigning roles to team members [Usage] • Assess and provide feedback to teams and individuals on their performance in a team setting [Usage] • Using a particular software process, describe the aspects of a project that need to be planned and monitored, (eg, estimates of size and effort, a schedule, resource allocation, configuration control, change management, and project risk identification and management) [Usage] • Track the progress of some stage in a project using appropriate project metrics [Usage] • Compare simple software size and cost estimation techniques [Usage] • Use a project management tool to assist in the assignment and tracking of tasks in a software development project [Usage] • Describe the impact of risk tolerance on the software development process [Usage] • Identify risks and describe approaches to managing risk (avoidance, acceptance, transference, mitigation), and characterize the strengths and short-

Unit 3: (14)	
Competences Expected: c,d,i,j,m	
Topics	**Learning Outcomes**
• Administration of the service as a practice. • Service life cycle. • Definitions and generic concepts. • Models and key principles. • Processes. • Technology and architecture. • Competence and training.	• Use and apply ITIL correctly in the software process. [Usage]
Readings : [Som17], [PM15]	

Unit 4: (14)	
Competences Expected: c,d,i,j,m	
Topics	**Learning Outcomes**
• Fundamentals and Introduction. • Control and IT Governance Frameworks.	• Use and apply COBIT correctly in the software process. [Usage]
Readings : [Som17], [PM15]	

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. PLANNING

DATE	TIME	SESSION TYPE	PROFESSOR
See at EDU	See at EDU	See at EDU	See at EDU

10. EVALUATION SYSTEM

********* EVALUATION MISSING ********

11. BASIC BIBLIOGRAPHY

[PM15] Roger S. Pressman and Bruce Maxim. *Software Engineering: A Practitioner's Approach.* 8th. McGraw-Hill, Jan. 2015.

[Som17] Ian Sommerville. *Software Engineering.* 10th. Pearson, Mar. 2017.