Le MODULE
Conception Logicielle

Université Nice Sophia Antipolis | POLYTECH NICE-SOPHIA

Membre de UNIVERSITÉ CÔTE D'AZUR

Références

Les nombreuses références utilisées dans ce cours sont essentiellement données directement dans les slides.

Les principaux supports sont :

- O'Regan, G. (2017) *Concise Guide to Software Engineering: From Fundamentals to Application Methods*. Springer International Publishing (Undergraduate Topics in Computer Science). Available at: https://books.google.fr/books?id=D5dZswEACAAJ.

- Robillard, M. P. (2019) *Introduction to Software Design with Java*. Cham: Springer International Publishing. doi: 10.1007/978-3-030-24094-3.

Avant-propos

SOFTWARE ENGINEERING

Software Engineering

Software engineering is the application of a **systematic**, **disciplined**, **quantifiable** approach to the development, operation, and maintenance of software; that is, the application of engineering to software, and the ***study of such approaches***.

~2000 part. ~ 35% indus.

G. O'Regan, Concise Guide to Software Engineering, Undergraduate Topics in Computer Science

ftware engineering is not just

requires the **engi**

- to **plan and d**
  budget

its

Software Engineer

- The term “engineer” is applied only to people who have attained the necessary education and competence to be called engineers and who base their practice on classical engineering principles.
- The title places responsibilities on its holder to behave **professionally and ethically**.

Software Engineer

- They are responsible for **designing** and implementing high-quality and **reliable software** that is safe to use.
- They are also **accountable for their decisions** and actions and have a responsibility to object to decisions that violate professional standards.
- Engineers are required to behave professionally and ethically with their clients.
- Professional software engineers are required to follow **best practice** in software engineering and the defined software processes.

LE MODULE : CONCEPTION LOGICIELLE

Objectifs

- Savoir analyser et concevoir « objet »
  - Connaître l'essentiel de la notation UML
- Savoir produire des User Stories de valeur
- Savoir diriger son développement par des tests associés
- Connaître les patrons de conception les plus courants
  - savoir quand les appliquer et ne pas les appliquer
- Savoir appliquer des principes de programmation avancée
  - injection de dépendance
  - Architecture à composants

Table I.1. The 15 SWEBOK KAs	
Software Requirements	★☆☆
Software Design	★★☆
Software Construction	★★☆
Software Testing	★★☆
Software Maintenance	★☆☆
Software Configuration Management	☆☆☆
Software Engineering Management	★☆☆
Software Engineering Process	★☆☆
Software Engineering Models and Methods	
Software Quality	★☆☆
Software Engineering Professional Practice	
Software Engineering Economics	
Computing Foundations	★★☆
Mathematical Foundations	
Engineering Foundations	☆☆☆

expriment la part de cette dimension dans le module.

The Software Engineering Professional Practice knowledge area is concerned with the knowledge, skills, and attitudes that software engineers must possess to practice software engineering in a professional, responsible, and ethical manner.

Bourque, P. and Fairley, R. E. (eds) (2014) *{SWEBOK}: **Guide to the Software Engineering Body of Knowledge***. Version 3. Los Alamitos, CA: IEEE Computer Society. Available at: http://www.swebok.org/.

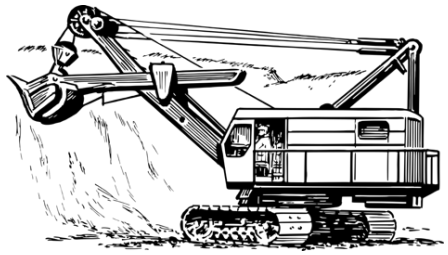
Mise en œuvre

- UNE étude de cas de bout en bout du module que l'on complexifie

SophiaTech Eats

SI4 - CASE STUDY FOR MODULE OBJECT ORIENTED CONCEPTION

  - des rendus et présentations réguliers
- Des technologies qui s'ajoutent
- En équipe projet dès le début
  - Avec des rôles attribués dans l'équipe



Tous responsables, tous impliqués

Software Developer (SD): Implements and codes the system components. Should have programming experience, knowledge of software development best practices, and collaborative skills.

Additionally, the following responsibilities are distributed:

Business Analyst and Product Owner (PO): Ensures that the functional objectives of the project are met and takes responsibility for gathering analysis and design documents produced by the team. She must always guide the team in alignment with the project goals.

Software Architect (SA): Ensures the overall design and structure of the system, considering its scalability and reliability. She consistently can explain the current architecture of the project and the choices made for future stages.

Quality Assurance Engineer (QA): Ensures the project's quality in terms of testing for functionality, performance, and code quality. She can explain the strengths and weaknesses of the project's quality throughout the project and outlining the upcoming steps.

Continuous Integration and Repository Manager (Ops): Oversees the continuous integration of code, manages the source code repository, tracks commits, and pull requests. Facilitates seamless coordination among developers and ensures that code is properly integrated and continuously tested. She should always be able to justify the state of the repository and the envisaged steps.

Steps in the software development process (IBM)

Gathering requirements to understand and document what is required by users and other stakeholders.

- *Deliverables: D1 & D2*

Choosing or building an architecture as the underlying structure within which the software will operate.

- *Deliverables: D1 & D2*

Developing a design around solutions to the problems presented by requirements, often involving process models and storyboards.

- *Deliverable: D2 & D4*

Building a model with a modeling tool that uses a modeling language like SysML or UML to conduct early validation, prototyping and simulation of the design.

- *Deliverables: D1 & D2*

Constructing code in the appropriate programming language. Involves peer and team review to eliminate problems early and produce quality software faster.

- *Deliverable: D4*

Testing with pre-planned scenarios as part of software design and coding — and ***conducting performance testing*** to simulate load testing on the application.

- *Deliverable: D5 & D4*

Managing configuration and defects to understand all the software artifacts (requirements, design, code, test) and build distinct versions of the software. Establish quality assurance priorities and release criteria to address and track defects.

- *Deliverable: D5 & D4*

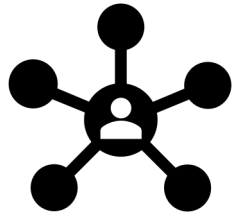
Week	Theme	Cours	TD	Rendu/Evaluation
12/9	Conception	1 - Introduction au module 2 - Introduction aux principaux diagrammes UML (2H)	Étude de cas TD1 : Travail sur les diagrammes UML	Constitution des groupes de TD
19/9	Conception	Diagrammes UML avancés & Analyse critique	Travail sur les diagrammes UML	D1. Spécifications initiales incluant Rendu des diagrammes UML initiaux + organisation de l'équipe (24/9)
26/9	Gestion de projet	Histoires utilisateurs et Estimations	Évaluation croisée (1h) Travail en équipe sur les histoires	E1. Auto-Evaluation des Diag. UML & Evaluation croisée (26/9)
3/10	Tests	Behavior Driven Development & Cucumber	Mise en place de l'environnement Définition de tests associés aux US	
10/10	GL	Rappels rapides des principes GRASP et SOLID & Injection de dépendances via PicoContainer	Définition des tests, renforcement de la modélisation, ...	
17/10	Gestion de projet	Bad Smells	Correction et avancements des codes	O1 : Évaluation par une soutenance orale
24/10	Contrôle		Correction et avancements des codes Introduction de nouvelles fonctionnalités	Contrôle pendant le cours
30/10	Vacances			
7/11	GL	Design Patterns	Intégration de patterns dans le code	
14/11	GL	Design Patterns	Intégration de patterns dans le code	
21/11	GL	Design Patterns	Intégration de patterns dans le code	
28/11	Architecture	Diagrammes à composants	Retour sur vos architectures	O2 : Évaluation par une soutenance orale
5/12	Gestion de Projet			
12/12	EXAMEN			
19/12			Finalisation du rendu	D2. Rendu terminal Document D4. Gestion du projet et codes D5. Évaluation de Qualité du projet

Contrôle des connaissances

★★☆

- Oraux : 20%
- Ecrits individuels : 40%
- Rendus du projet : 40%

Communication



- Diffusion des supports, sujets, etc. sur Moodle :
  - https://lms.univ-cotedazur.fr/2023/course/view.php?id=11330
- Canal de communication par défaut : **SLACK**
  - **#si4-conception**
- Question sur un des sujets de TD/projet : sur la chaine, pas de mp
  - Tout le monde profite de la réponse
  - Vous vous répondez entre vous

Formation des équipes

- 4 personnes par équipe (5 sinon, exceptionnellement)
  - Au moins deux étudiants du groupe de 2 provenances différentes en entrée de 3A
  - Possibilité d'être dans des TDs différents.
  - **Instructions sur Moodle**

➢ Liste des étudiants de l'équipe à positionner sur moodle
  - Discussion possible sur la chaîne slack #si4-conception

➢ Formation des équipes sur Github classroom, lien d'invitation dans moodle.

C’est parti !