COURSE SYLLABUS
CSC13106 – Software Architecture

1. GENERAL INFORMATION

Course name:	Software Architecture
Course name (in Vietnamese):	Kiến trúc phần mềm
Course ID:	CSC13106
Knowledge block:	
Number of credits:	4
Credit hours for theory:	45
Credit hours for practice:	30
Credit hours for self-study:	90
Prerequisite:	Object-Oriented Programming. Introduction to Software Engineering.
Prior-course:	Computer Networks. Database Management Systems. Software Requirements. Software Design. Software Testing. Web Application Development. Mobile Application Development. Game Development.
Instructors:	Ngo Huy Bien, PhD Office location: I82, 227 Nguyen Van Cu, Dist 5 HCMC. nhbien@fit.hcmus.edu.vn

2. COURSE DESCRIPTION

Software architecture plays a key role in modern software systems development. Good architectures not only ensure that a software system can be operated, maintained, modified, secured and scaled but also are medium for easier communication. Designing software architecture is challenging because it requires a large amount of domain-specific knowledge, experience, and creativity. A body of knowledge has been documented to help architects design and evaluate software architectures systematically.
This software architecture course introduces students to the fundamental tools and techniques used in software architecture design. The main goals include equipping students with the ability to plan, design, document, and evaluate software architectures. Key topics include architectural quality attributes, UML architecture diagrams, architecture views and documentation, Attribute-Driven Design, architectural styles and patterns, architecture design principles, architectural tradeoffs, architecture evaluation methods. This course also introduces domain-specific architectures, including SPA Framework Architecture, Domain-Driven Design, Clean Architecture, CQRS and Event Sourcing, Microservices Architecture, Docker/Kubernetes/Service Mesh, Event-Driven Architecture, Serverless Architecture, Lambda Architecture, and Application Frameworks. It helps students learn to select, evaluate, and use modern frameworks and platforms more efficiently.

3. COURSE GOALS

At the end of the course, students are able to

ID	Description	Program LOs
G1	Identify architectural quality attributes for a software system.	
G2	Design, document and communicate software architecture that meets architectural quality requirements.	
G3	Evaluate software architecture.	

G4	Use tools to design, communicate, and evaluate software architecture.	
G5	Explain the responsibilities and ethics of a software architect.	

4. COURSE OUTCOMES

CO	Description	I/T/U
G1.1	Elicit and document architectural requirements for complex software system.	T
G2.1	Explain common architecture styles and patterns.	T
G2.2	Design, refine, document and communicate software architectures.	T
G2.3	Engineer prototypes for modern architecture styles and patterns.	T, U
G3.1	Create software architecture evaluation reports.	T
G4.1	Use tools designing, documenting, communicating and evaluating software architecture (e.g. Lucidchart, Microsoft Visio, Draw.io, PlantUML.com, Structurizr.com).	T, U
G5.1	Explain the responsibilities and ethics of a software architect.	T

5. TEACHING PLAN

ID	Topic	Course outcomes	Teaching/Learning Activities	Assessments
1	Introduction to Software Architecture • Software Architecture	G1.1, G5.1	Lecturing. Q&A. Quiz.	A1, A3

	Concepts			
2	Quality Attributes • Architecturally Significant Requirements	G1.1, G5.1	Lecturing. Q&A. Quiz.	A1, A3
3	Documenting Software Architectures • UML Architecture Diagrams. • The 4+1 View Model. • The C4 Model. • Architecture Views.	G2.1, G2.2	Lecturing. Q&A. Quiz.	A1, A3
4	Architectural Styles Architectural Patterns	G2.1, G2.2, G2.3	Lecturing. Q&A. Quiz.	A1, A3
5	Software Architecture Design Methods • The Attribute-Driven Design (ADD) Method.	G2.1, G2.2, G2.3	Lecturing. Q&A. Quiz.	A1, A3
6	Software Architecture Evaluation • Design Principles. • The Architecture Tradeoff Analysis	G2.3, G3.1, G4.1	Lecturing. Q&A. Quiz.	A1, A3

	Method (ATAM). • Architecture Reconstruction.			
7	Microservices Containers Service Mesh	G2.2, G2.3, G4.1	Lecturing. Q&A. Lab.	A2, A3
8	SPA Framework Architecture Application Frameworks Domain-Driven Design The Clean Architecture Transactional Processing	G2.2, G2.3, G4.1	Lecturing. Q&A. Lab.	A2, A3
9	CQRS and Event Sourcing	G2.2, G2.3, G4.1	Lecturing. Q&A. Lab.	A2, A3
10	Message Brokers Event Streaming Event-Driven Architecture Serverless Architecture Lambda Architecture Software Integration	G2.2, G2.3, G4.1	Lecturing. Q&A. Lab.	A2, A3
11	Review	All goals	Lecturing. Q&A. Quiz.	A3

For the practical laboratory work, there are 10 weeks which cover similar topics as it goes in the theory class. Each week, teaching assistants will explain and demonstrate key ideas on the corresponding topic and ask students to do their lab exercises either on computer in the lab or at home. All the lab work submitted will be graded. There would be a final exam for lab work.

6. ASSESSMENTS

ID	Topic	Description	Course outcomes	Ratio (%)
A1	Quizzes	Quiz #1, Quiz #2, Quiz #3, Quiz #4, Quiz #5, Quiz #6	G1.1, G1.2, G2.1, G2.2, G3.1, G5.1	20%
A2	Labs	Lab #1, Lab #2, Lab #3, Lab #4	G1.2, G2.2, G2.3, G3.1, G4.1	40%
A3	Oral Exam	Lab results, lecture content.	All goals	40%

7. RESOURCES

References

- Len Bass, Paul Clements and Rick Kazman (2021). Software Architecture in Practice. Addison-Wesley. ISBN: 978-0136886099.
- Robert C. Martin (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. Pearson Education. ISBN: 978-0134494166.
- Vlad Khononov (2021). Learning Domain-Driven Design. O'Reilly Media. ISBN: 978-1098100131.
- Ethan Garofolo (2020). Practical Microservices: Build Event-Driven Architectures with Event Sourcing and CQRS. Pragmatic Bookshelf. ISBN: 978-1680506457.

Others

- David C. Hay (2002). Requirements Analysis: From Business Views to Architecture. Prentice Hall PTR.
- Paul Clements et al. (2010). Documenting Software Architectures Views and Beyond. Pearson.

- Humberto Cervantes and Rick Kazman (2016). Designing Software Architectures: A Practical Approach. Addison-Wesley Professional.
- Nick Rozanski and Eoin Woods (2012). Software Systems Architecture - Working with Stakeholders Using Viewpoints and Perspectives. Addison-Wesley Professional.
- Simon Brown (2019). Software Architecture for Developers - Volume 2 - Visualise, Document and Explore Your Software Architecture. Leanpub.
- Grady Booch et al. (2005). The Unified Modeling Language User Guide. Addison Wesley Professional.

- David Garlan and Mary Shaw (1994). An Introduction to Software Architecture.
- Frank Buschmann et al. (1996). Pattern-Oriented Software Architecture: A System of Patterns. John Wiley & Sons Ltd.
- Frank Buschmann et al. (2007). Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing. John Wiley & Sons Ltd.
- Robert C. Martin (2003). Agile Software Development: Principles, Patterns, and Practices. Pearson.
- Erich Gamma et al (1994). Design Patterns Elements of Reusable Object Oriented Software. Addison-Wesley Professional.
- Deepak Alur, Dan Malks and John Crupi (2003). Core J2EE Patterns: Best Practices And Design Strategies. Prentice Hall PTR.
- Martin Fowler et al. (2002). Patterns of Enterprise Application Architecture. Addison Wesley.
- Philip A. Bernstein and Eric Newcomer (2009). Principles of Transaction Processing. Second Edition. Morgan Kaufmann.
- Dino Esposito and Andrea Saltarello (2014). Microsoft .NET: Architecting Applications for the Enterprise. Microsoft Press.
- Emmit Scott (2015). SPA Design and Architecture. Manning Publications Co.

- Chris Richardson (2019). Microservices Patterns: With Examples in Java. Manning Publications.
- Sam Newman (2021). Building Microservices - Designing Fine-Grained Systems. O'Reilly Media.

- Sam Newman (2019). Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith. O'Reilly Media.

- Eric Evans (2003). Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison Wesley.
- Vaughn Vernon (2013). Implementing Domain-Driven Design. Addison-Wesley Professional.

- Martin Kleppmann (2016). Making Sense of Stream Processing. O'Reilly Media.
- Nathan Marz and James Warren (2015). Big Data: Principles and Best Practices of Scalable Realtime Data Systems. Manning Publications.

8. GENERAL REGULATIONS & POLICIES

- All students are responsible for reading and following strictly the regulations and policies of the school and university.
- Students who are absent for more than 3 theory sessions are not allowed to take the exams.
- Students who are absent during a test or lab session will receive a grade of 0 for the test or the lab. The exception will only apply to students who have a medical certificate from a hospital or evidence of a family emergency.
- For any kind of cheating and plagiarism, students will be graded 0 for the course. The incident is then submitted to the school and university for further review.
- Students are encouraged to form study groups to discuss on the topics. However, individual work must be done and submitted on your own.