Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматизації
проєктування енергетичних
процесів і систем

Syllabus

Software Engineering Components

Software Architecture

Catalog Description

Higher education level	***First (Undergraduate)***
Knowledge field	*Information Technologies*
Profession	*121 Software Engineering*
Curriculum	Software engineering of intelligent cyber-physical systems and web technologies
Course status	*Mandatory*
Form of training	*Full-time*
Grade, term	*Third grade, fall semester*
Credits (hours)	*5 credits / 150 hours (36 hours of lectures, 36 hours of practice, 78 hours of individual assignments)*
Term control	*Exam, modular test*
Schedule	http://rozklad.kpi.ua/
Teaching language	*Ukrainian/English*
Instructors	Lecturer: *Ph.D. Smakovskyi Denys, d.s.smakovskiy@gmail.com* Laboratory work: *Smakovskyi Denys, Olha Bespala*
Розміщення курсу	https://drive.google.com/drive/folders/1kGiiJRun5cKBE0vO7oH9Lyn0emCmbW-1?usp=sharing

Course Program

1. Course description, aim, subject, and expected outcomes

The discipline "Components of software engineering" is one of the mandatory disciplines of the training cycle. The module "Software Architecture" is devoted to the structural principles of software development and the organization of the software development process for a long time. Much attention is paid to the patterns of software development and the mechanism of dependency injection to ensure the development process through testing.

The purpose of the discipline is to acquaint students with modern principles and patterns of software to achieve optimal organization of software systems.

The subject of the discipline is the concepts and principles of software construction.

*Task. As a result of studying the discipline, students should form the following **professional competencies**:*

PC 2 - Ability to participate in software design, including modeling (formal description) of its structure, behavior, and functioning processes;

PC 3 - Ability to develop architectures, modules, and components of software systems;

PC 5 - Ability to follow specifications, standards, rules, and recommendations in the professional field in the implementation of life cycle processes;

PC 7 - Knowledge of information data models, the ability to create software for storage, retrieval and processing of data;

FC 10 - Ability to accumulate, process and systematize professional knowledge on the creation of testing and maintenance of software and recognition of the importance of lifelong learning;

FC 11 - Ability to implement phases and iterations of the software life cycle of information technology systems based on appropriate models and approaches to software development;

FC 12 - Ability to carry out the system integration process, apply change management standards and procedures to maintain the integrity, overall functionality and reliability of the software;

FC 13 - Ability to reasonably select and master tools for software development and maintenance;

Після засвоєння навчальної дисципліни студенти мають продемонструвати такі **програмні результати навчання:**

After mastering the discipline, students must demonstrate the following program learning outcomes:

PRN 1 - Analyze, purposefully search for and select the necessary information and reference resources and knowledge to solve professional problems, taking into account modern advances in science and technology.

PRN 2 - Know the code of professional ethics, understand the social significance and cultural aspects of software engineering and adhere to them in professional activities.

PRN 3 - Know the basic processes, phases, and iterations of the software life cycle.

PRN 4 - Know and apply professional standards and other regulations in the field of software engineering.

PRN 6 - Ability to select and use the appropriate task methodology for creating software.

PRN 7 - Know and apply in practice the fundamental concepts, paradigms, and basic principles of operation of language, tools, and computing software engineering

PRN 8 - Be able to develop a human-machine interface

PRN 9 - Know and be able to use methods and tools for collecting, formulating, and analyzing software requirements.

PRN 10 - Conduct a pre-project survey of the subject area, systematic analysis of the design object.

PRN 11 - Select source data for design, guided by formal methods of requirements description and modeling.

PRN 13 - Know and apply methods of algorithm development, software design, and data and knowledge structures.

PRN 14 - Apply in practice the tools of domain analysis, design, testing, visualization, measurement and documentation of software.

PRN 15 - Motivated to choose programming languages and development technologies to solve problems of software creation and maintenance

PRN 16 - Have the skills of team development, approval, design, and release of all types of software documentation.

PRN 17 - Be able to apply methods of component software development

PRN 18 - Know and be able to apply information technology for data processing, storage, and transmission.

PRN 19 - Know and be able to apply methods of software verification and validation.

PRN 20 - Know the approaches to assessing and ensuring the quality of software

2. Course prerequisites (Where the course fits into our curriculum)

Prerequisites of the discipline. *Students must have the basics of programming, algorithms and data structures, databases.*

Post-requisites of the discipline. *The acquired knowledge in the study of the discipline forms the basic knowledge for the study of the following disciplines: "Software Security" and "Building scalable data processing systems in real-time", as well as undergraduate practice and diploma design.*

3. Course contents

The discipline consists of four credit modules:

“Software Engineering Components-1. Introduction to Software Engineering ”,“ Software Engineering Components - 2. Software Modeling. Software Requirements Analysis. ”,“ **Software Engineering Components - 3. Software Architecture** ”,“ Software Engineering Components 4. Software Quality and Testing ”. This syllabus describes the “**Software Architecture”** module.

Topic 1. Overview of the capabilities of modern programming languages for building systems with an architecture that supports stable modification.

1.1. Reflection technology for the Java programming language. Reflection as a basis for building modern frameworks. Proxy as a mechanism for introducing additional components into existing code.

1.2. Test Driven Development using JUnit.

1.3. Basics of MVC pattern. Dependence substitution testing.

Topic 2. Patterns for building components of software systems that support modification.

2.1. Review. GoF patterns. Structural, creational, and behavioral patterns

2.2. GRASP Pattern Review: Creator, Information Expert, Low Coupling, Controller, High Cohesion, Polymorphism, Pure Fabrication, Indirection, Protected Variations.

2.3. Examples of using GRASP patterns.

2.4. Database architecture and software for their design and reverse engineering.

2.5. SOLID object-oriented programming principles: Single responsibility principle, open-closed principle, Liskov substitution principle, interface segregation principle, dependency inversion principle.

Topic 3. Modern frameworks as a further development of design patterns

3.1. Overview of the Spring framework. Application architecture on Spring.

3.2. XML configuration and concept of beans in Spring.

3.3. Injecting dependencies into Spring.

3.4. Bean settings. Properties.

3.5. Configuration on bean annotations. Bean life cycle callbacks. BeanPostProcessor

3.6. Access to databases using Spring Data, transactions in Databases.

3.7. Aspect-oriented programming on Spring.

Topic 4. REST architectural principle. Spring MVC framework for Web applications development.

4.1. REST architectural principle. Principles of application of methods GET, POST, PUT, DELETE, PATCH.

4.2. Use the Spring MVC framework based on Spring Boot to build REST-based applications.

4.3. Use the Spring MVC framework based on Spring Boot to build applications with Web page templates.

4. Course textbooks and materials

Required reading:

1. Мартін Р. Чиста Архітектура. Мистецтво розробки програмного забезпечення. - Харків : Ранок, 2019. 368 с.

2. Фрімен Е., Робсон Е. Head First. Патерни проектування. - Харків : Фабула, 2020. - 672 с.

3. Брауде Э. Технология разработки программного обеспечения – СПб.: Питер, 2004. – 655 с.

4. Пышкин Е. В. Основные концепции и механизмы объектно-ориентированного программирования – СПб.: БХВ-Петербург, 2005. – 640 с.

5. Соммервилл И. Инженерия программного обеспечения – М. : Вильямс, 2002. – 624 с.

6. Константайн Л. Разработка программного обеспечения – СПб.: Питер, 2004. – 592 с.

Optional reading:

7. Хамбл, Д. Непрерывное развертывание ПО: автоматизация процессов сборки, тестирования и внедрения новых версий программ / Джез Хамбл. – М.:Издательский дом «Вильямс», 2011. – 436 с.

8. Грэхем И. Объектно-ориентированные методы. Принципы и практика – М.: Вильямс, 2004. – 880 с.

9. Рамбо Дж., Якобсон А., Буч. Г. UML: специальный справочник. – СПб.: Питер, 2002. – 656с.

10. Буч Г. Язык UML. Руководство пользователя – М.: ДМК Пресс; СПб.: Питер, 2004. – 432 с.

11. JUnit 5 User Guide [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://junit.org/junit5/docs/current/user-guide/ (дата звернення 26.03.2021) – Назва з екрана.

12. Getting Started Guides [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://spring.io/guides#getting-started-guides (дата звернення 26.03.2021) – Назва з екрана.

13. Building REST services with Spring [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://spring.io/guides/tutorials/rest/ (дата звернення 26.03.2021) – Назва з екрана.

Educational Content

5. Pedagogical advice

Lectures

Topic 1. Overview of the capabilities of modern programming languages for building systems with an architecture that supports stable modification.

Lecture 1. Reflection technology for the Java programming language. Reflection as a basis for building modern frameworks. Proxy as a mechanism for introducing additional components into existing code.

Lecture 2. Test-Driven Development with JUnit.

Lecture 3. Basics of MVC pattern. Dependence substitution testing.

Topic 2. Patterns for building components of software systems that support modification.

Lecture 4. Review. GoF patterns. Structural, creational, and behavioral patterns

Lecture 5. GRASP Pattern Review: Creator, Information Expert, Low Coupling, Controller, High Cohesion, Polymorphism, Pure Fabrication, Indirection, Protected Variations.

Lecture 6. Examples of using GRASP patterns.

Lecture 7. Database architecture and software for their design and reverse engineering.

Lecture 8. SOLID object-oriented programming principles: Single responsibility principle, open-closed principle, Liskov substitution principle, interface segregation principle, dependency inversion principle.

Topic 3. Modern frameworks as a further development of design patterns

Lecture 9. Overview of the Spring framework. Application architecture on Spring.

Lecture 10. XML configuration and concept of beans in Spring.

Lecture 11. Injecting dependencies into Spring.

Lecture 12. Bean settings. Properties.

Lecture 13. Configuration on bean annotations. Bean life cycle callbacks. BeanPostProcessor

Lecture 14. Access to databases using Spring Data, transactions in Databases.

Lecture 15. Aspect-oriented programming on Spring.

Topic 4. REST architectural principle. Spring MVC framework for Web applications development.

Lecture 16. REST architectural principle. Principles of application of methods GET, POST, PUT, DELETE, PATCH.

Lecture 17. Use the Spring MVC framework based on Spring Boot to build REST-based applications.

Lecture 18. Use the Spring MVC framework based on Spring Boot to build applications with Web page templates.

Computer Labs

Topic 1. Overview of the capabilities of modern programming languages for building systems with an architecture that supports stable modification.

1. Basics of working with Reflection and proxying calls to the object.
2. Development of program components suitable for modular testing and the basis of development through testing.

Topic 2. Patterns for building components of software systems that support modification.

3. Fundamentals of the Model-View-Controller design pattern and testing through dependency substitution.
4. GoF patterns in applications with MVC architecture.
5. Design of database architecture with modeling and reverse engineering tools.

Topic 3. The use of modern frameworks as a further development of design patterns

6. Використання ін’єкції залежності за допомогою фреймворку Spring у додатках з MVC архітектурою.

Topic 4. REST architectural principle. Spring MVC framework for Web applications development

7. Використання фреймворку Spring MVC для розробки Веб-додатків.

6. Individual Assignments

Students' independent work is divided into 18 weeks. It includes doing a computer workshop and studying the following sources.

Topic 1. Overview of the capabilities of modern programming languages for building systems with an architecture that supports stable modification.

1. [1], c. 253-256. Test Boundaries.
2. [11], Getting Started
3. [11], Writing Tests

Topic 2. Patterns for building components of software systems that support modification.

4. [1], c. 221-226. Presenters and modest objects.

5. [2], c. 38-72. Design patterns introduction.

6. [2], c. 73 - 306. Decorator, Abstract Factory, Singleton, Command, Adapter Facade patterns.

7. [2], c. 310 - 521.Template Method Pattern. Iterator and Composer patterns State & Proxy patterns.

8. [2], c. 526 - 568. Складні Патерни. Патерни для кращого життя. Інші патерни.

Topic 3. The use of modern frameworks as a further development of design patterns

9. [12], Building Java Projects with Maven

10. [12], Accessing Relational Data using JDBC with Spring

11. [12],Validating Form Input

12. [12], Accessing Data with JPA

13. [12],Creating Asynchronous Methods

Topic 4. REST architectural principle. Spring MVC framework for Web applications development

14.[13], Building REST services with Spring

15. [12], Consuming a RESTful Web Service

16. [12],]Accessing JPA Data with REST

17. [12], Building a RESTful Web Service

18. [12],Securing a Web Application

Course Rules and Assessment Policy

7. Course study rules

In terms of skills and competencies, it is crucial for students to complete a computer workshop. Students receive a semester rating for the defense of laboratory work. The maximum points that can be obtained for the performance and defense of laboratory work are given in the table:

#	Lab	Term	Score
1	Basics of working with Reflection and proxying calls to the object.	2nd week	10
2	Development of program components suitable for modular testing and the basis of development through testing.	4th week	5
3	Fundamentals of the Model-View-Controller design pattern and testing through dependency substitution.	5th week	10
4	GoF Patterns	7th week	10
5	DB architecture	9th week	5
6	Dependency Injection	12th week	10
7	Web Project (REST + Web MVC)	18th week	10
	Total		60

If the laboratory work is completed in a period later than the period given in the table, the maximum score is reduced by 20%. A prerequisite for admission to the exam is the completion of all laboratory work.

The total grade consists of: 1) laboratories (programming assignments) 60%, 2) final exam 40%.

Presently there are three programming assignments, each worth up to 20% of the total grade. Students have to submit correctly fulfilled assignments during the specified term to obtain the full score for it, otherwise, 20% penalty points are applied. Other penalty points can be applied for the mistakes or incompleted subtasks at the lab but not more than 40% of the total score for laboratory work.

8. Assessment policy

How students are assessed: modular test, programming assignments

Calendar control: conducted twice a term to monitor the current state of compliance with the requirements of the syllabus.
Term assessment: final exam
Admission condition of term assessment: all programming assignment submission, start score not less than 30 points.
Exam scores map to the course grade according to the table:

Score	*Grade*
100-95	*Excellent*
94-85	*Very Good*
84-75	*Good*
74-65	*Satisfactory*
64-60	*Sufficient*
Less than 60	*Unsatisfactory*
Conditions for exam admission not met	*Not allowed*

9. Additional topics

Exam questions (see appendix).

Syllabus:

Developed by DAPPS Department Associate Professor, Ph.D. Denys Smakovskyi

Approved by Design Automation of Power Processes and Systems Department (minutes #16 on June 18, 2021)

Endorsed by Methodical Commission of Heat Power Faculty (minutes #11 on June 24, 2021)

Appendix.

Exam Questions.

1. Reflection as a mechanism for creating frameworks

2. Test-Driven Development as the main approach to modern software development
3. * Unit family frameworks. Testing of basic and erroneous scenarios
4. Substitution of dependencies at testing with Mock-frameworks. Setting the mocks, checking the interaction.
5. Continuous Integration/Deployment/Delivery
6. GoF patterns. Pattern Types. Pattern Elements.
7. GoF pattern Abstract Factory
8. GoF pattern Adapter
9. GoF pattern Bridge
10. GoF pattern Builder
11. GoF pattern Chain of Responsibility
12. GoF pattern Command
13. GoF pattern Composite
14. GoF pattern Decorator
15. GoF pattern Facade
16. GoF pattern Factory Method
17. GoF pattern Flyweight
18. GoF pattern Interpreter
19. GoF pattern Iterator
20. GoF pattern Mediator
21. GoF pattern Memento
22. GoF pattern Observer
23. GoF pattern Prototype
24. GoF pattern Proxy
25. GoF pattern Singleton
26. GoF pattern State
27. GoF pattern Strategy
28. GoF pattern Template Method
29. GoF pattern Visitor
30. Grasp Patterns. Responsibility□Driven Design
31. Grasp Pattern. Creator
32. Grasp Pattern. Information Expert (or just Expert)
33. Grasp Pattern Low Coupling
34. Grasp Pattern Controller
35. Grasp Pattern High Cohesion
36. Grasp Pattern Polymorphism
37. Grasp Pattern Pure Fabrication
38. Grasp Pattern Indirection
39. Grasp Pattern Protected Variations
40. Domain-Driven Design. Databases modeling.
41. The CAP Theorem
42. Reliability of distributed systems
43. Scalability of distributed systems
44. Event-Driven Architectures
45. Event Sourcing
46. CQRS
47. SOLID. Single responsibility principle
48. SOLID. Open–closed principle
49. SOLID. Liskov substitution principle
50. SOLID. Interface segregation principle
51. SOLID. Dependency inversion principle
52. Dependency injection

53. Inversion of Control
54. Spring IoC Container. Application context
55. Spring beans. Bean Scopes. Lifecycles
56. Aspect-oriented programming. Spring AOP.