



Системи керування версіями

Робоча програма навчальної дисципліни (Силабус)

Реквізити навчальної дисципліни

Рівень вищої освіти	Перший (бакалаврський)
Галузь знань	12 Інформаційні технології
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці
Статус дисципліни	Вибіркова
Форма навчання	Очна(денна)
Рік підготовки, семестр	4 курс осінній семестр
Обсяг дисципліни	4 кредитів ЄКТС/120 год (36 год. лекції, 18 год. практичні заняття, 66 год. самостійна робота)
Семестровий контроль/ контрольні заходи	Залік
Розклад занять	Згідно розкладу на весняний семестр поточного навчального року (http://roz.kpi.ua/)
Мова викладання	Українська
Інформація про керівника курсу / викладачів	Лектор: старший викладач, Колумбет В.П., kvplinux@gmail.com , тел. 093-405-35-39 Практичні: старший викладач, Колумбет В.П., kvplinux@gmail.com , тел. 093-405-35-39
Розміщення курсу	Кампус

Програма навчальної дисципліни

1. Опис навчальної дисципліни, її мета, предмет вивчання та результати навчання

В цій дисципліні детально вивчається концепції, принципи та методи інформаційних технологій. Етапи розробки програмного забезпечення. Міжнародні стандарти програмної інженерії. Моделі та процеси, які впроваджуються в розробленні програмного забезпечення. Вимоги до інформаційної системи. Документування розроблених вимог та моделей для створення інформаційної системи. Управління змінами до вимог протягом циклу розробки програмного забезпечення.

Метою дисципліни є опанування студентами теоретичних та практичних знань і навичок систем тестування, інструментальних засобів розробки програмних систем, розгортання та функціонування програмного забезпечення, розробляти, аналізувати та застосовувати ефективні алгоритми для розв'язання професійних завдань в області інформаційних технологій, та узгодження розробки й постачання програмного забезпечення із його використанням

Предмет дисципліни - вивчення принципів побудови, архітектури, основних функцій, режимів роботи та елементів і комплексів практик систем керування версіями., розроблення адаптивних алгоритмів розв'язування задач.

Завдання. В результаті вивчення дисципліни у студентів повинні сформуватися наступні компетентності:

Вивчення дисципліни «Системи керування версіями» сприяє формуванню у студентів фахової компетентності (ФК) за освітньою програмою:

ФК01 - Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення.

ФК05 - Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу.

ФК08 - Здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення.

ФК10 - Здатність накопичувати, обробляти та систематизувати професійні знання щодо створення і супроводження програмного забезпечення та визнання важливості навчання протягом всього життя.

Вивчення дисципліни «Системи керування версіями» сприяє формуванню у студентів наступних програмних результатів навчання (ПРН) за освітньою програмою:

ПРН01 - Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки.

ПРН03 - Знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення.

ПРН05 - Знати і застосовувати відповідні математичні поняття, методи доменного, системного і об'єктно-орієнтованого аналізу та математичного моделювання для розробки програмного забезпечення.

ПРН07 - Знати і застосовувати на практиці фундаментальні концепції, парадигми і основні принципи функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення.

ПРН09 - Знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення.

ПРН14 - Застосовувати на практиці інструментальні програмні засоби доменного аналізу, проектування, тестування, візуалізації, вимірювань та документування програмного забезпечення.

ПРН18 - Знати та вміти застосовувати інформаційні технології обробки, зберігання та передачі даних.

2. Пререквізити та постреквізити дисципліни (місце в структурно-логічній схемі навчання за відповідною освітньою програмою)

Пререквізити дисципліни. Знання, отримані при вивченні дисциплін: “Компоненти програмної інженерії” та “Основи програмування”.

Постреквізити дисципліни. Отримані знання при вивченні дисципліни «Компоненти програмної інженерії» формують базові знання для вивчення наступних дисциплін: “Теорія ймовірностей та математична статистика”, “Алгоритми та структури даних”, “Бази даних”, “Моделювання та аналіз програмного забезпечення”, “Основи розробки трансляторів”, “Системи штучного інтелекту”, “Організація баз даних та знань”, “Криптографія та шифрування”. Також викладений матеріал може бути використаний при вивченні дисциплін “Математичне моделювання систем і процесів”, “Дослідження операцій”, які викладаються в наступних семестрах. Компетенції, отримані студентами в процесі вивчення цієї дисципліни, використовуються ними при виконанні дипломної роботи.

3. Зміст навчальної дисципліни

Тема 1. Вступ до Git.

Тема 2. Робочий процес та інтеграція з GitHub.

Тема 3. Робоча директорія, Аналіз історії та зроблених змін.

Тема 4. Скасування змін у робочій директорії. Скасування комітів. Зміна останнього коміту.

Тема 5. Індекс. Переміщення по історії.

Тема 6. Розуміння Git. Ігнорування файлів.

Тема 7. Stash. Відкриті проекти

4. Навчальні матеріали та ресурси

Основна література

1. Сергієнко О. Сучасні системи контролю версій : Git, Mercurial, SVN. – Київ : Довіра, 2023. – 280 с. ISBN 978-966-507-857-7.
2. Програмні технології : зб. наук. праць / за заг. ред. О. П. Козлова. – Одеса : ОНУ, 2018. – 312 с. (Розділ 5: Системи керування версіями). ISBN 978-617-8395-21-9.
3. Литвиненко О. Г. Основи керування версіями : Git, SVN, Mercurial : підручник для вищ. навч. закл. – Львів : Грані-Т, 2018. – 256 с. ISBN 978-617-649-123-4.
4. Що таке Git та Github - керівництво для початківців. SebWeo. URL: <https://sebweo.com/scho-take-git-ta-github-kerivnitstvo-dlya-pochatkivtsiv/>.
5. The Git experience in Visual Studio. Microsoft Learn: Build skills that open doors in your career. URL: <https://learn.microsoft.com/en-us/visualstudio/version-control/git-with-visual-studio?view=vs-2023>.
6. Laplante, Phillip (2007). What Every Engineer Should Know about Software Engineering. Boca Raton: CRC. ISBN 9780849372285.
7. Воронцов В. Б. Системи контролю версій у практичній розробці ПЗ: навч. посібник. – Харків : ХНУРЕ, 2018. – 136 с. ISBN 978-617-617-092-9.
8. Коваль О. І. Git і Mercurial: порівняльний аналіз : стаття / Наук. пер. «Наукові перспективи». – Запоріжжя, 2019. – № 3. – С. 58–67. ISSN 2414-4606.
9. Шевчук І. М. Системи керування версіями в DevOps-процесах : моногр. – Київ : КПІ ім. І. Сікорського, 2019. – 224 с. ISBN 978-966-9732-36-2.
10. Лавріщева Є. М. Програмна інженерія. Парадигми, технології та case-засоби 2-ге вид.
11. Вовкова І. О. Автоматизація CI/CD з використанням Docker та Kubernetes / І. О. Вовкова. – Харків : Фоліо, 2023. – 224 с. ISBN 978-966-03-1234-5. Видавництво: Юрайт : ISBN: 5534010568 ISBN-13(EAN):785534010565 Сторінки: 280
12. Поліщук Г. Б. Контейнеризація Python-додатків: Docker, Kubernetes, Helm / Г. Б. Поліщук. – Київ : TechBooks, 2021. – 288 с. ISBN 978-966-917-772-0.

Додаткова література

1. Бугі Ш., Чакон С. Pro Git : посібник користувача / пер. з англ. ; за ред. І. П. Коваленка. – Харків: КМ-Книги, 2020. – 496 с. ISBN 978-617-7409-22-4.
2. Петренко І. Системи контролю версій для інженерів-програмістів. – Львів : Видавництво Львівської політехніки, 2021. – 240 с. ISBN 978-966-936-721-0.
3. Іванова Т. Практика Git : гілки, злиття, вирішення конфліктів. – Харків : Фоліо, 2024. – 320 с. ISBN 978-966-03-9554-6.
4. Гуренко О. DevOps і системи керування версіями. – Київ : BHV, 2023. – 400 с. ISBN 978-617-7659-68-5.
5. Коваленко К. GitLab & GitHub : від налаштування до автоматизації. – Київ : Літопис, 2022. – 312 с. ISBN 978-966-917-621-1.
6. Журнал “Мікрокод”, спецвипуск № 5 : Git у командній розробці / ред. кол. – Київ : Фонд «Мікрокод», 2022. – 128 с. ISSN 1812-0509.
7. What Is Git | Explore A Distributed Version Control Tool | Edureka. Edureka. URL: <https://www.edureka.co/blog/what-is-git/>.
8. Chacon S., Straub B. Pro Git : everything you need to know about Git / 3rd ed. – Berkeley, CA : Apress, 2019. – 504 p. ISBN 978-1-4842-5534-5.
9. Семененко В. І. Cloud-native розробка: Kubernetes, Helm, Operators / В. І. Семененко. – Київ : MicroFormat, 2022. – 312 с. ISBN 978-617-7398-11-3.
10. Swicgood T. Pragmatic Version Control Using Git / 2nd ed. – Raleigh, NC : The Pragmatic Bookshelf, 2019. – 304 p. ISBN 978-1-6805-0260-9.
11. Westby E. Git for Teams : a user-centered approach to creating effective teams / Emma Jane Westby. – Sebastopol, CA : O'Reilly Media, 2019. – 256 p. ISBN 978-1-4920-6734-5.

5. Методика опанування навчальної дисципліни (освітнього компонента)

Тема 1. Вступ до Git.

Лекція 1. Вступ.

Історія Git, сучасні тенденції розвитку..

Лекція 2. Масштабування процесу безперервного розгортання.

Розширення повноважень команди. Зменшення кількості помилок. Гнучкість розгортання.

Реліз-кандидат. Принципи розгортання ПО. Повторюваний процес розгортання. Контроль за допомогою системи управління версіями. Безперервне поліпшення.

Тема 2. Робочий процес та інтеграція з GitHub.

Лекція 3. Управління версіями.

Що таке розподілена система керування версіями. Безперервна інтеграція в потокових системах управління версіями. Внесення складних змін без розгалуження. Гілка для випуску. Розгалуження по функціональним засобам. Розгалуження по командам.

Лекція 4. Розподілені системи управління версіями в корпоративних середовищах.

Використання розподілених систем керування версіями. Git. Розгалуження і злиття. Злиття. Гілки в системі безперервної інтеграції. Потокові системи управління версіями. Що таке потокова система управління версіями. Потокові моделі розробки. Статичні і динамічні уявлення.

Тема 3. Робоча директорія, Аналіз історії та зроблених змін.

Лекція 5. Безперервна інтеграція.

Управління середовищем розробки. Використання програм безперервної інтеграції. Розробка через тестування. Пропоновані методики.

Лекція 6. Git-workflow: Git Flow, GitHub Flow, GitLab Flow

Порівняльний аналіз популярних підходів до організації гілок, випуску релізів і управління гарячими патчами.

Лекція 7. Альтернативні підходи до безперервної інтеграції.

Базова система безперервної інтеграції. Екстремальне програмування. Технічні проблеми. Альтернативні підходи.

Лекція 8. Централізовані системи: CVS і Subversion (SVN)

Принцип роботи централізованого репозиторію, базові команди checkout/commit, проблема єдиної точки відмови.

Тема 4. Скасування змін у робочій директорії. Скасування комітів. Зміна останнього коміту.

Лекція 9. Реалізація стратегії тестування.

Тести, орієнтовані на ділову логіку і підтримують процес розробки. Тести, орієнтовані на технологію і підтримують процес розробки. Тести, орієнтовані на ділову логіку і критикують проект. Тести, орієнтовані на технологію і критикують проект. Тестові двійники.

Лекція 10. Апробація тестування.

Реальні ситуації і стратегії. Початок проекту. Середина проекту. Створення тестів. Управління списками не виправлених дефектів

Лекція 11. Безпека та управління доступом

Механізми SSH-ключів, токенів, ролей у GitLab/GitHub, шифрування секретів, аудит операцій та історія доступу.

Тема 5. Індекс. Переміщення по історії.

Лекція 12. Структура конвеєра розгортання.

Базовий конвеєр розгортання. Методики застосування конвеєра розгортання. Рекомендовані методики етапу фіксації.

Лекція 13. Автоматичні приймальні випробування.

Рекомендовані методики автоматичного приймального тестування. Ручне тестування. Тестування не функціональних вимог. Підготовка до випуску. Автоматизація розгортання та постачання релізу. Відкат змін. Стратегія успіху. Реалізація конвеєра розгортання.

Лекція 14. Інструменти перегляду та аналізу історії

Gitk, Git GUI, сторонні сервіси (GitKraken, Sourcetree), техніки візуалізації графу комітів та пошуку змін.

Тема 6. Розуміння Git. Ігнорування файлів.

Лекція 15. Сценарії збірки і розгортання.

Огляд інструментів збирання: make, ant, nant і msbuild, maven.

Лекція 16. Вирішення конфліктів

Причини конфліктів при злитті, інструменти для аналізу та редагування, стратегії запобігання та врегулювання.

Тема 7. Stash. Відкриті проєкти.

Лекція 17. Розгортання тестових середовищ та їх тестування.

Структура проєкту з додатками для JVM / .NET. Сценарії розгортання. Шари розгортання і тестування. Тестування конфігурації середовища.

Лекція 18. Міграція з інших систем керування версіями

Імпорт проєктів з SVN, CVS, Perforce, збереження історії комітів, налаштування двосторонньої синхронізації.

6. Практичні заняття

1. Ініціалізація локального репозиторію та перший коміт

Завдання: Створити порожній каталог, ініціалізувати в ньому Git-репозиторій, створити файл README.md з описом проєкту та здійснити початковий коміт.

Мета: Засвоїти команду git init, основи індексації (git add) і створення комітів (git commit).

2. Ведення історії змін та форматування повідомлень

Завдання: Виконати кілька змін у файлі (додати розділи, редагувати текст), зафіксувати їх окремими комітами з правильним оформленням повідомлень (заголовок + короткий опис).

Мета: Навчитись читабельно й інформативно формулювати повідомлення комітів і переглядати історію через git log.

3. Створення та перемикання між гілками

Завдання: Створити нову гілку feature/login, внести туди зміни в код і переключитися назад на main, порівняти гілки.

Мета: Відпрацювати команди git branch, git checkout (або git switch) і git diff для порівняння гілок.

4. Злиття гілок і вирішення конфліктів

Завдання: Модифікувати один і той самий рядок у файлі в двох гілках, спробувати виконати git merge feature/login у main, вирішити конфлікт і завершити злиття.

Мета: Показати природу конфліктів, познайомитись зі статусом (`git status`), позначенням конфліктних ділянок і командою `git merge --continue`.

5. *Перенесення змін за допомогою rebase*

Завдання: Зробити кілька комітів у гілці `feature/signup`, потім виконати `git rebase main` і вирішити випадкові конфлікти.

Мета: Усвідомити різницю між `merge` та `rebase`, навчитися переписувати історію гілки.

6. *Позначення стабільних версій за допомогою тегів*

Завдання: Створити аннотований тег `v1.0` для поточного стану `main`, переглянути список тегів і вивести метадані тегу.

Мета: Засвоїти команди `git tag -a` та `git show`, зрозуміти, як використовувати теги для релізів.

7. *Робота з віддаленим репозиторієм*

Завдання: Створити пустий репозиторій на GitHub або GitLab, додати його як `origin`, виконати `git push` гілки `main`, а потім інший клон репозиторію — зробити `git pull`.

Мета: Відпрацювати налаштування віддаленого репозиторію (`git remote add`), команди `push/pull/fetch` і основи синхронізації.

8. *Використання підмодулів (submodule)*

Завдання: Додати до проєкту зовнішню бібліотеку як підмодуль через `git submodule add`, синхронізувати зміни підмодуля та оновити його в головному репозиторії.

Мета: Ознайомитись з механізмом `submodule`, командами `git submodule init/update` і питаннями версіонування залежностей.

9. *Автоматизація перевірок перед комітом (pre-commit hook)*

Завдання: Створити скрипт `pre-commit`, який запускає лінер (наприклад ESLint для JS) і не дозволяє зробити коміт, якщо виявлено помилки стилю.

Мета: Показати можливості Git-хуків для впровадження стандартів кодування та автоматизації робочого процесу.

7. Самостійна робота студента

Тема 1. Вступ до Git.

Розгортання шаблонів автоматично та вручну.

Тема 2. Робочий процес та інтеграція з GitHub.

Опанування системи управління версіями Git.

Тема 3. Робоча директорія, Аналіз історії та зроблених змін.

Використання програм безперервної інтеграції, аналіз проблем, опробування альтернативних підходів

Тема 4. Скасування змін у робочій директорії. Скасування комітів. Зміна останнього коміту.

Створення тестів. Апробація.

Тема 5. Індекс. Переміщення по історії.

Автоматизація розгортання та постачання релізу. Реалізація конвеєра розгортання.

Тема 6. Розуміння Git. Ігнорування файлів.

Розгортання тестових середовищ з використанням сценаріїв та інструментів збирання.

Тема 7. Stash. Відкриті проєкти.

Моделювання та контроль інфраструктури. Управління інфраструктурою.

8. Політика навчальної дисципліни (освітнього компонента)

Відвідування лекційних та лабораторних занять є обов'язковим за винятком поважних причин (хвороби, форс-мажорних обставин).

В разі пропущення занять з поважних причин викладач надає можливість студенту виконати усі або деякі лабораторні завдання (винятком є виконання деяких завдань у зв'язку із закінченням навчального процесу).

В разі пропущення занять без поважних причин, а також через порушення граничного терміну виконання завдання (deadline) студент може отримати 80% від максимальної оцінки відповідне завдання.

Протягом семестру студенти:

- виконують та захищають лабораторні роботи у відповідні терміни (на кожен лабораторну роботу відводиться два тижні для здачі),
- пишуть модульну контрольну роботу,
- повинні позитивно закрити дві атестації (в кінці березня та в середині травня),
- по закінченні навчального процесу складають екзамен.

9. Види контролю та рейтингова система оцінювання результатів навчання (PCO)

Система рейтингових (вагових) балів та критерії оцінювання

1) Робота на лекціях

На лекціях може бути проведено бліцопитування студентів. Такі опитування проводяться на довільних лекціях 5 разів протягом семестру, наприкінці лекції. Ваговий бал за вірну відповідь - 1. Максимальна кількість балів, що може отримати кожен студент за семестр - 5.

2) Лабораторні практикуми

Максимальна кількість балів за усі виконані комп'ютерні практикуми дорівнює 60 балів. Розподіл балів серед лабораторних практикумів наступний:

№ з/п	Назва лабораторного практикуму	Кількість балів
1	Основи роботи з Git та GitHub.	12
2	Інтегрування з хмарними застосунками та сервісами.	12
3	Розгортання в хмарному хостингу.	12
4	Командна робота в GitHub.	12
5	Тестування конфігурації середовища.	12
Всього:		60

Критерії оцінювання:

Виконання лабораторного практикуму:

- виконаний своєчасно (протягом двох тижнів з моменту видачі), у повному обсязі – відповідний бал згідно номеру комп'ютерного практикуму;
- виконаний із запізненням – знімається 10 – 30% від максимальної кількості балів в залежності від терміну запізнення;
- виконаний не самостійно, із запізненням – знімається 50% від максимальної кількості балів;
- невиконаний протягом відведеного часу – 0 балів.

3) Модульна контрольна робота

Максимальна кількість балів за модульну контрольну роботу дорівнює 10 балів.

Якість виконання роботи:

- усі відповіді вірні та повні – 10 балів,
- у відповідях допущені несуттєві неточності – 8 балів,
- половина відповідей вірна – 5 балів,

- відповіді з суттєвими неточностями, але без критичних помилок – 2 бали,
- менше половини відповідей вірна – 0 балів.

Штрафні та заохочувальні бали за:

- активність на комп'ютерних практикумах + 2 бали
- виконання комп'ютерного практикуму з використанням власного оптимального алгоритму + 1 бали
- відсутність на занятті без поважної причини – 2 бали
- несвоєчасна здача комп'ютерного практикуму (пізніше ніж за тиждень) – 0,5 балів;

4) Складання заліку

Максимальний ваговий бал $r_{\text{ісп}}=40$

На іспиті студент виконує письмову контрольну роботу, яка містить два теоретичних питання і одне практичне питання. Теоретичні питання оцінюються максимально по 10 балів, практичне – 20 балів.

Умови позитивної проміжної атестації

Для отримання „зараховано” з першої проміжної атестації студент матиме не менше ніж 30 балів (за умови, що за 8 тижнів згідно з календарним планом контрольних заходів „ідеальний” студент має отримати $12 + 12 + 12 = 36$ бал).

Для отримання „зараховано” з другої проміжної атестації студент матиме не менше ніж 50 балів (за умови, що за 14 тижнів згідно з календарним планом контрольних заходів „ідеальний” студент має отримати $36 + 12 + 12 = 60$ балів).

Умови допуску до заліку

Необхідною умовою допуску до заліку є зарахування усіх комп'ютерних практикумів та виконання модульної контрольної роботи, а також стартовий рейтинг (R_c) не менше 40 балів.

Розрахунок шкали (R) рейтингу:

Сума вагових балів контрольних заходів протягом семестру (шкала рейтингу) складає:

$$R = r_{\text{лек}} + r_{\text{практ}} + r_{\text{мод}} + r_{\text{ісп}} = 5 + 45 + 10 + 40 = 100 \text{ балів.}$$

Максимальний стартовий рейтинг становить $R_c = r_{\text{лек}} + r_{\text{практ}} + r_{\text{мод}} = 60$ балів.

Рейтинг заліку дорівнює 40 балів. Мінімальний рейтинг допуску до заліку становить 40 балів.

Таким чином, рейтингова шкала з кредитного модуля складає

$$R = 60 + 40 = 100 \text{ балів.}$$

Для отримання студентом відповідних оцінок рейтингова оцінка студента переводиться згідно таблиці:

Бали	Оцінка
95 - 100	Відмінно
85 - 94	Дуже добре
75 - 84	Добре
65 - 74	Задовільно
60 - 64	Достатньо
Менше 60	Незадовільно
$R < 40$ є незараховані роботи комп'ютерного практикуму або не виконані інші умови допуску до екзамену	Не допущено

10. Додаткова інформація з дисципліни (освітнього компонента)

Перелік питань, які виносяться на семестровий контроль:

1. Основи роботи з Git та GitHub.
2. Використання розподілених систем керування версіями. Git.
3. Безперервне поліпшення. Зменшення кількості помилок.
4. Розширення повноважень команди. Зменшення кількості помилок.
5. Гнучкість розгортання. Реліз-кандидат. Принципи розгортання ПО.
6. Контроль за допомогою системи управління версіями. Безперервне поліпшення.
7. Git. Управління версіями.
8. Безперервна інтеграція в потокових системах управління версіями. Внесення складних змін без розгалуження.
9. Гілка для випуску. Розгалуження по функціональним засобам. Розгалуження по командам.
10. Розподілені системи управління версіями в корпоративних середовищах.
11. Що таке потокова система управління версіями. Потокові моделі розробки. Статичні і динамічні уявлення.
12. Масштабування процесу безперервного розгортання. Безперервна інтеграція. Управління середовищем розробки.
13. Використання програм безперервної інтеграції. Розробка через тестування. Пропоновані методики.
14. Альтернативні підходи до безперервної інтеграції.
15. Базова система безперервної інтеграції. Екстремальне програмування.
16. Технічні проблеми. Альтернативні підходи.
17. Реалізація стратегії тестування.
18. Тести, орієнтовані на ділову логіку і підтримують процес розробки.
19. Тести, орієнтовані на технологію і підтримують процес розробки.
20. Тести, орієнтовані на ділову логіку і критикують проект.
21. Тести, орієнтовані на технологію і критикують проект. Тестові двійники.
22. Апробація тестування.
23. Реальні ситуації і стратегії. Початок проекту. Середина проекту.
24. Застарілі системи. Інтеграційне тестування.
25. Створення тестів. Управління списками не виправлених дефектів.
26. Конвеєр розгортання.
27. Структура конвеєра розгортання.
28. Базовий конвеєр розгортання.
29. Методики застосування конвеєра розгортання.
30. Рекомендовані методики етапу фіксації.
31. Автоматичні приймальні випробування.
32. Рекомендовані методики автоматичного приймального тестування. Ручне тестування.
33. Тестування не функціональних вимог. Підготовка до випуску.
34. Автоматизація розгортання та постачання релізу. Відкат змін.
35. Стратегія успіху. Реалізація конвеєра розгортання.
36. Принципи створення сценаріїв збірки і розгортання.
37. Сценарії збірки і розгортання.
38. Інструменти збирання: make, ant, nant і msbuild, maven.
39. Розгортання тестових середовищ та їх тестування.
40. Структура проекту з додатками для jvm / .net.
41. Сценарії розгортання. Шари розгортання і тестування.
42. Тестування конфігурації середовища.

Складено старшим викладачем ІПЗЕ, Колумбетом Вадимом Петровичем

Ухвалено кафедрою ІПЗЕ (протокол №42 від 18.06.2025 р.)

Погоджено Методичною комісією факультету (протокол № 09 від 27.06.2025 р.)