

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ
КАФЕДРА ІНТЕЛЕКТУАЛЬНИХ ПРОГРАМНИХ СИСТЕМ**

«ЗАТВЕРДЖУЮ»

Заступник декана
з навчальної роботи

_____ Кашпур О.Ф.

«___» _____ 2019 року

**РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
МЕТОДИ ТЕСТУВАННЯ ТА ОЦІНКИ
НАДІЙНОСТІ ПРОГРАМНИХ СИСТЕМ
для студентів**

галузь знань	12 Інформаційні технології
спеціальність	121 Інженерія програмного забезпечення
освітній рівень	магістр
освітня програма	Програмне забезпечення систем
вид дисципліни	вибіркова

Форма навчання	денна
Навчальний рік	2021/2022
Семестр	3
Кількість кредитів ECTS	4
Мова викладання, навчання та оцінювання	українська
Форма заключного контролю	іспит

Викладач: **к. ф.-м. н., доцент Слабоспицька О.О.** (лекції, лабораторні заняття).

Пролонговано: на 20__/20__ н. р. _____ (_____) «__» 20__ р.

на 20__/20__ н. р. _____ (_____) «__» 20__ р.

Розробник: Слабоспицька Ольга Олександрівна, к. ф.-м. н., с. н. с., доцент кафедри інтелектуальних програмних систем.

ЗАТВЕРДЖЕНО

Завідувач кафедри інтелектуальних програмних систем

_____ О.І. Провотар

Протокол № __ від «__» _____ 2019 р.

Схвалено науково-методичною комісією факультету комп'ютерних наук та кібернетики

Протокол від «__» _____ 2019 року №__

Голова науково-методичної комісії _____ Л.Л. Омельчук

«__» _____ 2019 року

Затверджено вченою радою факультету комп'ютерних наук та кібернетики

Протокол від «__» _____ 2019 року №__

Голова вченої ради факультету _____ А.В. Анісімов

1. Мета дисципліни «Методи тестування та оцінки надійності програмних систем» – засвоєння студентами базових знань щодо основних понять в галузі тестування програмних систем та оцінки їх надійності, опанування сучасних методів статичного й динамічного тестування та задач, моделей і методів інженерії надійності, технологічних засад розгортання ресурсно-ефективних процесів тестування й забезпечення надійності програмних систем в їх життєвому циклі – для успішної діяльності в складі груп тестування, гарантування якості, вдосконалення процесу розроблення програмних продуктів.

2. Попередні вимоги до опанування або вибору навчальної дисципліни. Для успішного вивчення дисципліни «Методи тестування та оцінки надійності програмних систем» студенти повинні:

1. Знати:

1. Основні моделі життєвого циклу програмних систем;
2. Основні підходи та базові шаблони проектування програмних систем;
3. Основні концепції процедурного та об'єктно-орієнтованого програмування;
4. Програмні, організаційні й технологічні засоби забезпечення конкурентоздатної якості програмних систем;
5. Базові методи математичної статистики.

2. Вміти:

1. Вилучати, балансувати та документувати вимоги до програмної системи;
2. Безпечно працювати з програмним кодом у системах контролю версій;
3. Обчислювати стандартні статистичні характеристики вибірки даних (середнє, медіану, середньоквадратичне й стандартне відхилення, квартилі);
4. Застосовувати критерії згоди для перевірки статистичних гіпотез.

3. Володіти:

1. Базовими навичками застосування поширених інтегрованих середовищ розробки програмного забезпечення, зокрема Microsoft Visual Studio, J2EE;
2. Базовими навичками застосування систем контролю версій, зокрема Git.

4. Успішно опанувати курси освітньо-наукової програми «Програмне забезпечення систем»:

1. Теоретичні основи та методи розробки інформаційних систем;
 2. Методи забезпечення якості програмних систем,
- а також курси з дискретної математики, програмної інженерії та розроблення інформаційних систем, які викладаються в циклі професійної підготовки фахівців освітньо-кваліфікаційного рівня «бакалавр».

3. Анотація навчальної дисципліни. Навчальна дисципліна «Методи тестування та оцінки надійності програмних систем» є складовою освітньо-наукової програми підготовки фахівців за другим (магістерським) рівнем вищої освіти у галузі знань 12 Інформаційні технології за спеціальністю 121 Інженерія програмного забезпечення в рамках освітньо-наукової програми «Програмне забезпечення систем».

Дана дисципліна належить до переліку № 1 дисциплін вільного вибору студента. Викладається у **3 семестрі в обсязі – 120 год. (4 кредити ECTS)**, зокрема: лекції – 28 год., лабораторні – 10 год., консультації – 2 год., самостійна робота – 80 год. У курсі передбачено 2 змістовні частини, 2 контрольні роботи та 5 лабораторних робіт. Завершується дисципліна – **іспитом.**

В результаті вивчення навчальної дисципліни студенти повинні:

Знати:

1. Методи статичного (індивідуального й колективного) та динамічного (модульного, інтеграційного, функцій, системного, приймального) тестування локальних і Веб-

застосунків за наявності й відсутності доступу до тестованого коду та його специфікації у планових, ітеративних і гнучких програмних проектах.

2. Підходи та засоби автоматизації тестування коду різних рівнів.
3. Моделі ефективних процесів тестування й забезпечення надійності та ступенів їх зрілості, а також підходи до розгортання цих процесів згідно з моделями.
4. Задачі та методи встановлення, прогнозування й оцінювання показників надійності програмних систем без даних їх тестування та за їх наявності.
5. Методи побудови стратегії демонстраційного тестування надійності та прогнозування оптимального моменту випуску програмної системи.
6. Основні вбудовані засоби та автономні каркаси автоматизованого тестування, інструментальні засоби визначення надійності програмних систем.

Вміти:

1. Планувати, виконувати й документувати статичне тестування в плановому і гнучкому програмному проекті.
2. Планувати, проектувати, виконувати й документувати модульні, інтеграційні, системні, регресійні, приймальні тести та тести функцій у плановому і гнучкому програмному проекті з ефективним застосуванням засобів автоматизації тестування.
3. Автоматизовано керувати життєвим циклом виявлених дефектів.
4. Обґрунтовано вибирати модель процесу тестування в організації-розробнику, розгортати і вдосконалювати його згідно з нею, визначати необхідний рівень надійності програмної системи й відстежувати його досягнення.
5. Визначати рівень надійності програмної системи, прийнятний для всіх зацікавлених сторін та відстежувати й перевіряти його досягнення.

4. Завдання (навчальні цілі). Основними завданнями дисципліни «Методи тестування та оцінки надійності програмних систем» є набуття знань, умінь та навичок (компетентностей) на рівні новітніх досягнень в області тестування та оцінки надійності програмних систем відповідно до освітньої кваліфікації магістр з інженерії програмного забезпечення. Зокрема, розвивати:

- Здатність мотивувати людей та рухатися до спільної мети, працювати в команді співробітників (ЗК04).
- Здатність спілкуватися з представниками інших професійних груп різного рівня (з експертами з інших галузей знань/видів економічної діяльності) (ЗК05).
- Здатність удосконалювати свої навички на основі аналізу попереднього досвіду (ЗК06).
- Здатність приймати обґрунтовані рішення (ЗК08).
- Здатність аналізувати предметні області, формувати, аналізувати та моделювати вимоги до програмного забезпечення (СК01).
- Здатність оцінювати ступінь обґрунтованості застосування специфікацій, стандартів, правил і рекомендацій в професійній галузі та дотримуватися їх при реалізації процесів життєвого циклу програмного забезпечення (СК05).

5. Результати навчання за дисципліною.

Результат навчання (1. знати; 2. вміти; 3. комунікація; 4. автономність та відповідальність)		Форми (та/або методи і технології) викладання і навчання	Методи оцінювання та пороговий критерій оцінювання (за необхідності)	Відсоток у підсумковій оцінці з дисципліни
Код	Результат навчання			
PH1.1	Знати об'єкти та техніки статичного тестування.	Лекції, самостійна робота.	Контрольна робота №1, іспит.	5%
PH1.2	Знати рівні й види тестування коду та відповідні їм методи проектування, виконання й документування тестів разом з виявленими дефектами для локальних і веб-застосунків.	Лекції, лабораторні заняття, самостійна робота.	Контрольна робота №1, іспит.	20%
PH1.3	Знати умови та методи застосування димового, регресійного й дослідницького тестування.	Лекції, лабораторні заняття, самостійна робота.	Контрольна робота №1, іспит.	5%
PH1.4	Знати основні моделі процесу тестування та його зрілості й умови їх доцільного застосування в організації-розробнику.	Лекції, самостійна робота.	Контрольна робота №1, іспит.	5%
PH1.5	Знати основні задачі інженерії надійності та методи їх розв'язання при конструюванні програмних систем.	Лекції, самостійна робота.	Контрольна робота №2, іспит.	15%
PH1.6	Знати основні моделі процесу інженерії надійності та його зрілості й умови їх доцільного застосування в організації-розробнику.	Лекції, самостійна робота.	Контрольна робота №2, іспит.	5%
PH2.1	Вміти автоматизовано розробляти, виконувати й документувати набори статичних і динамічних тестів для виявлення й аналізу дефектів згідно з запитаним рівнем якості програмної системи.	Лабораторні заняття, самостійна робота.	Захист лабораторних робіт, іспит.	15%
PH2.2	Вміти прогнозувати очікуваний та оцінювати досягнутий рівень якості програмної системи.	Лекції, самостійна робота.	Контрольні роботи №1, №2, іспит.	5%

РН2.3	Вміти визначати доцільні моделі процесів тестування та інженерії надійності та заходи з їх удосконалення.	Лекції, самостійна робота.	Контрольні роботи №1, №2, іспит.	5%
РН3.1	Зрозуміло формулювати проблеми вивчення теоретичного матеріалу й виконання лабораторних робіт у питаннях до викладача й колег та обґрунтовано відповідати на такі питання з їх боку.	Лекції, лабораторні заняття, години консультацій.	Захист лабораторних робіт.	5%
РН3.2	Послідовно й зрозуміло обґрунтовувати власні рішення в лабораторних роботах.	Лабораторні заняття.	Захист лабораторних робіт.	5%
РН4.1	Самостійно виконувати тестування коду, документувати дефекти й оцінювати його досягнуту якість.	Лабораторні заняття.	Захист лабораторних робіт.	5%
РН4.2	Самостійно аналізувати теоретичний матеріал і практичний досвід для обґрунтування лабораторних і контрольних робіт.	Лабораторні заняття, опрацювання рекомендованих інформаційних джерел.	Контрольні роботи №1, №2, захист лабораторних робіт.	5%

6. Співвідношення результатів навчання дисципліни із програмними результатами навчання.

Програмні результати навчання	Результати навчання дисципліни												
	РН1.1	РН1.2	РН1.3	РН1.4	РН1.5	РН1.6	РН2.1	РН2.2	РН2.3	РН3.1	РН3.2	РН4.1	РН4.2
ПРН02. Обґрунтовувати вибір методів формування вимог до програмної системи, розробляти, аналізувати та систематизувати вимоги.	+	+			+		+	+					
ПРН04. Оцінювати і обирати методи і моделі розробки, впровадження, експлуатації програмних засобів та управління ними на всіх етапах життєвого циклу.				+		+			+		+		
ПРН05. Розробляти і оцінювати стратегії проектування програмних засобів; обґрунтовувати, аналізувати і оцінювати	+	+	+				+	+	+	+	+	+	+

прийняті проектні рішення з точки зору якості кінцевого програмного продукту.																				
ПРН06. Аналізувати, оцінювати і обирати методи, сучасні програмно-апаратні інструментальні та обчислювальні засоби, технології, алгоритмічні та програмні рішення для ефективного виконання конкретних виробничих задач з програмної інженерії.																				
ПРН08. Проводити аналітичне дослідження параметрів функціонування програмних систем для їх валідації та верифікації, а також проводити аналіз обраних методів, засобів автоматизованого проектування та реалізації програмного забезпечення.																				
ПРН09. Знати і застосовувати сучасні професійні стандарти і інші нормативно-правові документи з інженерії програмного забезпечення.																				
ПРН10. Вміти приймати організаційно-управлінські рішення в умовах невизначеності.																				

7. Схема формування оцінки.

7.1 Форми оцінювання студентів.

Семестрове оцінювання:

1. Контрольна робота 1: РН1.1, РН1.2, РН1.3, РН1.4, РН2.2, РН2.3, РН4.2 – **20 балів/12 балів.**
2. Контрольна робота 2: РН1.5, РН1.6, РН2.2, РН2.3, РН4.2 – **20 балів/12 балів.**
3. Лабораторна робота 1: РН2.1, РН3.1, РН3.2, РН4.1, РН4.2 – **4 бали/2 бали.**
4. Лабораторна робота 2: РН2.1, РН3.1, РН3.2, РН4.1, РН4.2 – **5 балів/3 бали.**
5. Лабораторна робота 3: РН2.1, РН3.1, РН3.2, РН4.1, РН4.2 – **5 балів/3 бали.**
6. Лабораторна робота 4: РН2.1, РН3.1, РН3.2, РН4.2 – **3 бали/2 бали.**
7. Лабораторна робота 5: РН2.2, РН3.1, РН3.2, РН4.1, РН4.2 – **3 бали/2 бали.**

Підсумкове оцінювання (у формі іспиту):

- Максимальна кількість балів які можуть бути отримані студентом: 40 балів.
- Результати навчання, які будуть оцінюватись: РН1.1, РН1.2, РН1.3, РН1.4, РН1.5, РН1.6, РН2.1, РН2.2, РН2.3.
- Форма проведення і види завдань: письмова робота.
- Види завдань: 4 письмових завдання (2 тестові завдання, аналітичне питання та задача з інженерії надійності).
- Для отримання загальної позитивної оцінки з дисципліни оцінка за іспит має бути не меншою, ніж 24 бали.
- Студенти не допускаються до іспиту, якщо протягом семестру вони набрали менше ніж 36 балів.

- Студент не допускається до іспиту, якщо протягом семестру він не виконав і не здав 100 % лабораторних робіт, передбачених планом.

Критерії оцінювання на іспиті.

Завдання	Тема завдання	Максимальний відсоток від 40 балів	Всього відсотків
Завдання 1	Тестове питання за теоретичним матеріалом курсу.	10%	10%
Завдання 2	Аналітичне питання, що потребує зіставлення теоретичного матеріалу курсу, лабораторних робіт, власного досвіду.	25%	25%
Завдання 3	Теоретичне питання.	27.5%	27.5%
Завдання 4	Задача з інженерії надійності.	37.5%	37.5%
			100%

Питання для підготовки до іспиту.

Тестові питання з тестування:

- Для інтеграційного тестування системи взаємодіючих Веб-сервісів доцільне:
 - низхідне тестування;
 - метод «великого вибуху»;
 - висхідне тестування;
 - метод модифікованого сендвіча.
- В індустріальній технології надійного тестування UniTesK для подання вимог застосовують:
 - мову RAISE;
 - мову VDM;
 - формалізм скінченого автомату;
 - формалізм програмного контракту.
- Коли тестувальник може тільки запустити тестований код на виконання (текст коду і його специфікація недоступні), застосовним є:
 - дослідницьке тестування;
 - стохастичне тестування;
 - метод функціональних діаграм;
 - статистичне тестування.
- Нехай функція вводить дату в форматі ДД.ММ.РРРР і надає повідомлення, чи є рік РРРР високосним. Тестові дані 31.06.2012 і 29.02.1900 належать до:
 - одного припустимого класу еквівалентності;
 - одного неприпустимого класу еквівалентності;
 - різних припустимих класів еквівалентності;
 - різних неприпустимих класів еквівалентності.
- Автономного тестування модулів складної програмної системи можна уникнути за допомогою стратегії:
 - димового тестування;
 - висхідного тестування;

- c. модифікованого сандвіча;
 - d. тестування з постійною інтеграцією.
7. Можливі причини появи дефектів у продукті можна впорядкувати за важливістю, користуючись:
- a. контрольними картами;
 - b. діаграмою Парето;
 - c. діаграмою Ісікави;
 - d. діаграмою розсіювання.
8. Якщо вартість виконання тестів невисока, а вартість пропуску помилки – критично висока, для регресійного тестування слід застосувати:
- a. випадкові методи;
 - b. безпечні методи;
 - c. повторний запуск усіх тестів;
 - d. методи мінімізації.
9. Найпотужнішим критерієм структурного тестування є критерій:
- a. комбінацій умов;
 - b. умов і гілок;
 - c. умов;
 - d. MC/DC.
10. Зіставлення рівням зрілості процесу тестування очікуваних переваг від його вдосконалення передбачено в моделі:
- a. TIM;
 - b. MMAST;
 - c. TPI Next;
 - d. MTPF.
11. Для функціонального тестування програми з великою кількістю взаємозалежних параметрів найбільше придатна техніка:
- a. покриття гілок та умов;
 - b. розбиття вхідного простору на категорії;
 - c. аналізу класів еквівалентності;
 - d. аналізу граничних значень.
12. Для вдосконалення хаотичного процесу тестування найбільш придатна модель зрілості:
- a. ISO/IEC/IEEE 29119-2;
 - b. MTPF;
 - c. Радулеску;
 - d. TPI Next.
13. Якщо наявний тільки документ специфікації вимог природною мовою, для їх тестування можна використати:
- a. методологію ATLM;
 - b. обернене семантичне трасування;
 - c. формальну інспекцію);
 - d. цілеспрямовану перевірку.
14. Нехай програма друкує повідомлення щодо виду трикутника, довжини сторін якого вводяться з клавіатури: чи він є нерівностороннім, рівнобедреним або ж рівностороннім. До складу одного з неприпустимих класів еквівалентності має входити:
- a. трійка (2, 3, 4)
 - b. трійка (1, 2, 3);

- c. жодна з трійок а),б);
 - d. обидві трійки а),б).
15. Формальну інспекцію ранніх робочих продуктів програмної системи можна підтримати за допомогою засобу:
- a. Selenium;
 - b. FxCop;
 - c. Collaborator;
 - d. Mantis.
16. У загальному випадку, для інтеграційного тестування великих і складних програмних систем рекомендується:
- a. низхідне тестування;
 - b. метод сандвіча;
 - c. висхідне тестування;
 - d. метод модифікованого сандвіча.
17. Спільним обмеженням методів аналізу граничних значень та еквівалентного розбиття є непридатність:
- a. для модульного тестування;
 - b. для тестування функцій інтерфейсу користувача;
 - c. для перевірки великої кількості комбінацій значень вхідних параметрів;
 - d. для функціонального тестування.
18. І рекомендації, і засоби вдосконалення процесу тестування передбачено в моделі його зрілості:
- a. ТММ;
 - b. ТРІ;
 - c. МТРФ;
 - d. ММАСТ.
19. Для функціонального тестування програмних систем з великою кількістю незалежних вхідних параметрів, множини значень яких є скінченними, доцільна техніка:
- a. розбиття вхідного простору на категорії;
 - b. попарного тестування;
 - c. функціональних специфікацій;
 - d. еквівалентного розбиття.
20. Для виявлення тих кількісних характеристик тестованого коду, що впливають на кількість виявлених помилок у ньому, можна застосувати:
- a. діаграму Парето;
 - b. діаграму розкиду;
 - c. контрольні карти;
 - d. діаграму Ісікави.
21. Основним поняттям дослідницького тестування за Дж. Віттакером є:
- a. модель стратегії тестування;
 - b. тур тестування;
 - c. сценарій тестування;
 - d. модель процесу тестування.

Тестові питання з інженерії надійності:

1. Коли відомий тільки текст коду програми, але його запуск неможливий, очікувану кількість помилок у ній можна визначити за допомогою моделі:
- a. Малайя-Дентона;

- b. Холстеда;
 - c. NASA GSFC;
 - d. COQUALMO.
2. Для побудови блок-схеми надійності ПС слід застосувати:
- a. техніку SFTA;
 - b. модель Муси часу виконання;
 - c. техніку SFMEA;
 - d. техніку ЕТА.
3. Якщо дані тестування демонструють спадну інтенсивність відмов як функцію часу тестування, без використання статистичних обчислень можна однозначно встановити невідповідність цим даним моделі:
- a. Желінські-Моранди;
 - b. Шнейдевінда;
 - c. Літлвуда-Веррала;
 - d. Муси.
4. За відсутності даних тестування, прогнозу оцінку очікуваної кількості виявлених дефектів у коді на момент початку його приймального тестування можна отримати за допомогою моделі:
- a. Релея;
 - b. СОСUALMO (Боема-Чулані);
 - c. Гефні-Девіса;
 - d. Rome Laboratory.
5. Нехай відмова програмної системи відбувається тільки тоді, коли одночасно відмовляють усі три її компонента, і всі вони одночасно й незалежно виконуються впродовж часу виконання ПС з однаковою інтенсивністю відмов. Імовірність безвідмовного функціонування всієї програмної системи впродовж певного часу:
- a. менша імовірності безвідмовного функціонування компонента;
 - b. дорівнює їй;
 - c. перевищує її;
 - d. співвідношення залежить від додаткових чинників.
6. За відсутності даних тестування, прогнозу оцінку очікуваної кількості виявлених дефектів у коді на момент початку його модульного тестування можна отримати за допомогою моделі:
- a. Малая-Дентона;
 - b. СОСUALMO (Боема-Чулані);
 - c. прогновної статистичної моделі Муси;
 - d. Rome Laboratory.
7. Нехай при тестуванні програмної системи спостережено 5 відмов, між якими вона працювала безвідмовно X_i годин операційного часу: $X_1= 8$; $X_2= 5$; $X_3 =3$; $X_4 =1$; $X_5=2$. Цим даним неадекватна модель:
- a. Муси часу виконання;
 - b. Шика-Волвертона;
 - c. Желінські-Моранди;
 - d. усі моделі.
- Поясніть відповідь.
8. Якщо відомий тільки тип створюваної програмної системи, для прогнозування її надійності можна використати модель:
- a. Малайя-Дентона;
 - b. Rome Laboratory;

- c. Боема-Чулані;
 - d. Холстеда.
9. Нехай відмова будь-якого з чотирьох компонентів ПС призводить до її відмови, і всі вони одночасно виконуються впродовж часу виконання ПС з однаковою інтенсивністю відмов. Імовірність відмови всієї ПС протягом певного часу:
- a. менша імовірності відмови компонента;
 - b. дорівнює їй;
 - c. перевищує її;
 - d. співвідношення залежить від додаткових чинників.
10. Якщо доступний тільки текст коду програмної системи, для прогнозування її надійності можна використати модель:
- a. COQUALMO;
 - b. відносної складності Мунсона-Кошгофтаара;
 - c. NASA GSFC;
 - d. Rome Laboratory.
11. Нехай відмова будь-якого з трьох компонентів ПС призводить до її відмови, і всі вони одночасно виконуються впродовж часу виконання ПС з однаковою інтенсивністю відмов. Інтенсивність відмов всієї ПС:
- a. менша інтенсивності відмов компонента;
 - b. дорівнює їй;
 - c. перевищує її;
 - d. співвідношення залежить від додаткових чинників.
- Обґрунтуйте відповідь.
12. Вплив методології виявлення помилок у коді на кількість залишкових помилок враховується у моделі:
- a. Rome Laboratory;
 - b. Малайя-Дентона;
 - c. COQUALMO (Боема-Чулані);
 - d. OQUAMO.
13. Якщо тестування програмної системи виконується поетапно – на кожному етапі запускається кілька тестів, а всі виявлені помилки усуваються тільки після закінчення етапу, – для прогнозування імовірності її відмови упродовж певного часу можна використати модель:
- a. Гефні-Девіса;
 - b. Желінські-Моранди;
 - c. Шумана;
 - d. La Padula.
14. Для прогнозування надійності ПС, тестування якої виконується методом підсівання помилок, можна використати модель:
- a. Шика-Волвертона;
 - b. Желінські-Моранди;
 - c. Бейзіна;
 - d. La Padula.
15. Якщо дані тестування – значення інтервалів часу між відмовами – демонструють немонотонну інтенсивність відмов в залежності від часу, без використання критеріїв згоди можна однозначно встановити невідповідність їм моделі:
- a. Вейбулла;
 - b. статистичної прогнозної моделі Муси;
 - c. Літлвуда-Веррала;

- d. Шнейдевінда.
16. Неявний функціональний профіль можна використовувати тільки за умови:
- статистичної незалежності ключових вхідних змінних;
 - значущої кореляції цих змінних;
 - визначеності операційного профілю;
 - виконання в організації-розробнику програми забезпечення надійності.
17. Кількість залишкових помилок у програмі, тестованій паралельно кількома групами тестерів, можна оцінити за допомогою моделі:
- Бейзіна;
 - простої евристичної моделі;
 - Шумана;
 - геометричної моделі Моранди.
18. Якщо відомий тільки тип створюваної програмної системи та рівень зрілості середовища розробки, для прогнозування інтенсивності відмов на момент початку системного тестування можна використати модель:
- прогнозу статистичну модель Муси;
 - Rome Laboratory;
 - Боєма-Чулані;
 - Холстеда.
19. Якщо вірогідне внесення в код нових помилок під час усунення виявлених, для прогнозування надійності слід використати модель:
- Літлвуда-Веррала;
 - Rome Laboratory;
 - прогнозу статистичну модель Муси;
 - Шнейдевінда.
20. Нехай під час статистичного тестування у $n_1=4$ тестах з $n=10$ спостережено відмову. Найбільша кількість подальших тестів, які разом будуть успішними з імовірністю, не меншою $p=0.3$, дорівнює:
- 1;
 - 2;
 - 3;
 - 4.

Аналітичні питання.

- Чи можна прогнозувати надійність програмної системи, коли відома тільки специфікація вимог до неї? Якщо так, опишіть відповідні моделі.
- За допомогою яких технік можна прогнозувати інтенсивність відмов компонентної програмної системи на момент початку її системного тестування? Яких вхідних даних потребують ці техніки?
- Чи існують умови, за яких методи мінімізації набору регресійних тестів доцільніші, ніж методи їх випадкового вибору? Якщо так, що це за умови? За їх виконання, які характеристики методів мінімізації набору кращі, ніж випадкових методів?
- Нехай дані системного тестування програмної системи описуються базовою моделлю Муси часу виконання. Яка частка можливої кількості відмов має бути спостережена, щоб інтенсивність відмов на момент початку тестування зменшилась в чотири рази? Яким чином і за допомогою яких додаткових даних можна оцінити час тестування, необхідний для такого зменшення?

5. Чи є моделі прогнозування надійності, придатні для використання на всіх стадіях програмного проекту? Якщо так, наведіть їх перелік. Чим вони розрізняються між собою?
6. Які моделі можна використати для формування профілю зрілості процесу тестування? За яких умов застосування кожної з них найбільш доцільне?
7. Нехай дані тестування програмної системи задовільно описуються базовою моделлю Муси з параметром $\beta_0 = 8$ та інтенсивністю відмов на момент початку системного тестування $\lambda_0=0.01$. Чи можливо досягнути зменшення поточної інтенсивності відмов у чотири рази? Якщо так, скільки додаткового часу тестування для цього потрібно?
8. Чи можна одночасно прогнозувати щільність виявлених і невиявлених дефектів у робочих продуктах програмної системи? Якщо так, опишіть відповідні моделі.
9. Які моделі можна використати для прогнозування надійності програмної системи, якщо її тестування виконується поетапно – на кожному етапі запускається кілька тестів, а всі виявлені помилки усуваються тільки після закінчення етапу? Які показники надійності при цьому прогноуються?
10. На підставі яких критеріїв можна встановити готовність програмної системи до випуску в аспекті її надійності?
11. Яким чином можна прогнозувати оптимальний момент випуску програмного продукту з точки зору підтвердження його надійності?
12. Які стратегії тестування і на яких рівнях потребують одночасного застосування драйверів і заглушок?
13. Які види тестування і на яких рівнях слід виконувати для Веб-застосунку?
14. Які критерії структурного тестування є найдоцільнішими за співвідношенням потужності та трудомісткості застосування? У чому полягає кожний з них?
15. Якими техніками можна виявити фрагменти “надлишкового” коду ПС, що не підтримують жодну з вимог до неї? Які умови застосування кожної з них та інструментальні засоби підтримки?
16. За допомогою яких технік можна паралельно тестувати специфікацію вимог і проект архітектури об’єктно-орієнтованої програмної системи?
17. За допомогою яких інструментів аналізу якості можна сформулювати список причин незадовільної кількості помилок у програмних продуктах?
18. У чому полягають і якими техніками опрацьовуються обмеження формальної інспекції? Які умови застосування цих технік та інструментальні засоби підтримки?
19. За допомогою яких технік можна тестувати вимоги до програмної системи? Які вхідні дані необхідні і які інструментальні засоби застосовні для кожної з них?
20. Чи можливе тестування коду ПС у разі неможливості його запуску? Якщо так, які техніки застосовні, які вхідні дані вони використовують і якими інструментальними засобами підтримуються?

Теоретичні питання

1. Призначення, склад, сутність, подібності й розбіжності технік функціонального тестування.
2. Види операційних профілів та техніки їх формування й застосування.

3. Призначення, склад, сутність, вирази, подібності й розбіжності статичних моделей прогнозування/оцінювання надійності програмних систем.
4. Призначення, склад, сутність, доцільні умови застосування, подібності й розбіжності моделей зрілості процесу тестування програмних систем.
5. Призначення, сутність, умови застосування, подібності й розбіжності методології ATLM та технології UniTESK (ІСП РАН) як підходів до автоматизації тестування в організації-розробнику.
6. Цільові показники, моделі та процедури розподілу вимог до надійності складної програмної системи.
7. Призначення, сутність, умови застосування, подібності й розбіжності моделей процесу тестування.
8. Призначення та методи регресійного тестування. Умови доцільності застосування методів.
9. Сутність, подібності й розбіжності модульного та інтеграційного тестування застосунків: призначення, об'єкти, пошукувані дефекти, стратегії та середовище.
10. Цілі, об'єкти, пошукувані дефекти та види тестів для системного тестування програмних систем
11. Призначення, склад, сутність, умови застосування, подібності й розбіжності технік тестування на підставі очікуваного використання та спрямованого пошуку помилок
12. Призначення, сутність, подібності й розбіжності класифікацій моделей пізнього прогнозування надійності А. Гойля та Дж. Муси. Приклади моделей у класах.
13. Призначення, склад, сутність, подібності й розбіжності регресійних та вимірювальних моделей оцінювання/прогнозування надійності програмних систем
14. Призначення, склад, сутність, умови застосування, подібності й розбіжності динамічних моделей прогнозування надійності дискретного часу.
15. Цільові показники, моделі та процедури моделювання надійності програмної системи у залежності від надійності її компонентів.
16. Склад, призначення, подібності й розбіжності моделей процесу забезпечення надійності програмного забезпечення.
17. Призначення, склад, сутність, подібності й розбіжності статистичної прогнозної моделі Муси та емпіричних моделей оцінювання/прогнозування надійності програмних систем.
18. Призначення, склад, сутність, умови застосування, базові припущення та аналітичні подання основних динамічних моделей прогнозування надійності неперервного часу.
19. Цілі, задачі та підходи інженерії надійності на етапах життєвого циклу програмних систем
20. Призначення, сутність, тип, умови застосування моделі зрілості процесу забезпечення надійності.

Задачі з інженерії надійності.

1. Нехай вхідні дані для застосунку можна розбити на три неперетинні області, вибори з яких є незалежними та рівноімовірними. Виконано 4 тести на даних з однієї області, 3 – на даних з другої області та 1 – на даних з третьої області, причому відмови

спостережено для 3, 1 і 1 тестів відповідно. Обчислити мінімальну кількість тестів, імовірність безвідмовного виконання хоча б одного з яких під час їх послідовного запуску не менша $p = 0.5$

2. Нехай дані тестування програмної системи задовільно описуються базовою моделлю Муси часу виконання з параметрами $\beta_0 = 20$ і $\beta_1 = 0.005$, і на момент $t = 200$ процесорного часу програмна система працює. Обчислити інтенсивність відмов програмної системи в середній момент другої після t відмови.
3. За допомогою яких програмних засобів можна обчислити параметри моделі Муси?
4. Нехай дані тестування програмної системи задовільно описуються моделлю Муси часу виконання з параметрами $\beta_0 = 6$, $\beta_1 = 0.001$. Обчислити імовірність, що за першу третину часу тестування програмної системи буде спостережено більше половини її відмов.
5. За допомогою яких програмних засобів можна обчислити параметри моделі Муси?
6. Нехай дані тестування програмної системи задовільно описуються моделлю Желінськи-Моранди з параметрами $\beta_0 = 12$, $\beta_1 = 0.002$. Обчислити імовірність, що на момент, коли вичерпано третину середнього часу її тестування, буде спостережено п'ять відмову.
7. За допомогою яких програмних засобів можна обчислити параметри моделі?
8. Нехай дані тестування програмної системи описуються моделлю Желінськи-Моранди з параметрами $\beta_0 = 20$; $\beta_1 = 0.005$. Обчислити оцінку середнього часу тестування (годин операційного часу), необхідного для досягнення цільової інтенсивності відмов $\lambda = 0.0001$, якщо після другої відмови програмна система безвідмовно працює $t = 10$ годин.
9. За допомогою яких програмних засобів можна обчислити параметри моделі?
10. Нехай до програми внесено 15 штучних помилок, з яких виявлено 10 та ще 10 власних помилок. Обчислити оцінку N кількості помилок у програмі та імовірність, що у програмі більше $N + 2$ помилок.
11. До якої групи належить відповідна модель оцінювання надійності?
12. Нехай дані тестування програмної системи задовільно описуються базовою моделлю Муси з параметрами $\beta_0 = 8$, $\beta_1 = 0.001$. Обчислити час, необхідний для спостереження менше чверті можливих відмов з імовірністю $p = 0.3$.
13. За допомогою яких програмних засобів можна обчислити параметри моделі Муси?
14. Нехай дані тестування програмної системи задовільно описуються базовою моделлю Муси з параметрами $\beta_0 = 10$, $\beta_1 = 0.04$. Обчислити оцінку середнього часу t після другої відмови, необхідного для досягнення середньої кількості залишкових помилок $\mu(t) = 1$.
15. Нехай дані тестування програми задовільно описуються базовою моделлю Муси з параметрами $\beta_0 = 4$, $\beta_1 = 0.001$. Обчислити середню залишкову кількість помилок у момент, коли до кінця тестування в середньому залишилася третина його терміну.
16. Нехай дані тестування програмної системи задовільно описуються базовою моделлю Муси з параметрами $\beta_0 = 10$, $\beta_1 = 0.03$. Обчислити середній момент t (годин процесорного часу), до якого з імовірністю $p = 0.5$ не станеться відмови, якщо після останньої відмови в момент $t_0 = 50$ програмна система безвідмовно працює $t^* = 20$ годин.

17. Нехай до програми внесено 10 штучних помилок, з яких виявлено 7 та ще одну власну помилку. Обчислити оцінку N кількості помилок у програмі та імовірність, що у програмі не менше 5 помилок.
18. До якої групи належить відповідна модель оцінювання надійності?
19. Нехай дані тестування програмної системи описуються моделлю Желінські-Моранди з параметрами $\beta_0=8$, $\beta_1=0.005$. Обчислити час тестування, необхідний для спостереження хоча б однієї її відмови з імовірністю $p=0.5$.
20. Нехай до програми внесено 8 штучних помилок, з яких виявлено 4 та ще 2 власні помилки. Обчислити оцінку N кількості помилок у програмі та імовірність, що їх насправді більше $N+2$.
21. До якої групи належить відповідна модель оцінювання надійності?
22. Нехай до програми внесено 6 штучних помилок, які виявлено всі та ще 4 власні помилки. Обчислити оцінку N кількості помилок у програмі та імовірність, що їх насправді не більше $N+1$.
23. До якої групи належить відповідна модель оцінювання надійності?
24. Нехай досвід процесу розроблення програмних систем певного типу свідчить, що їм властиві помилки введення-виведення, доступу до бази даних та логічні помилки, які усуваються з однаковою імовірністю.
25. Під час 6 тестів програмної системи спостережено 2 успішні тести, дві відмови через логічні помилки та по одній відмові через помилку доступу до бази даних і помилку введення-виведення.
26. Обчислити за цих умов найбільшу кількість додаткових тестів, які всі продемонструють відмову з імовірністю, не меншою $p=0.4$.
27. До якої групи належить відповідна модель оцінювання надійності?
28. Нехай з $n=5$ тестів програмної системи в $n^+=2$ не спостережено відмов, у $n_1=1$ виявлено помилки 1-го типу, у $n_2=2$ виявлено помилки другого типу. Імовірність появи помилок другого типу вдвічі вища імовірності появи помилок першого типу.
29. Обчислити імовірність відмови хоча б одного з трьох наступних тестів за цих умов.
30. До якої групи належить відповідна модель оцінювання надійності?
31. Нехай вхідні дані програми можна розбити на три неперетинні області, причому вибори з двох перших рівно імовірні, а з третьої – вдвічі імовірніший. Виконано по 3 тести на даних кожної з цих областей, причому в кожній трійці один з тестів не призвів до відмови. Обчислити імовірність відмови хоча б в одному з трьох наступних запусків програми.
32. До якої групи належить відповідна модель оцінювання надійності?
33. Нехай вхідні дані програми можна розбити на дві неперетинні області, причому вибори з першої втричі імовірніші, ніж з другої. Виконано 2 і 4 прогони на даних з цих областей, по одному з яких демонстрували відмову. Обчислити мінімальну кількість прогонів, імовірність відмови хоча б одному з яких буде не меншою $p=0.5$.
34. Нехай дані тестування програмної системи описуються моделлю Дж. Муси з параметрами $\beta_0=10$, $\beta_1=0.005$, і після її останньої відмови в момент $t_b=400$ (годин процесорного часу) ПС безвідмовно працює ще $t_0=25$ годин. Обчислити мінімальний час t^* , впродовж якого програмна система безвідмовно працюватиме з імовірністю $p=0.5$.

7.2 Організація оцінювання.

Терміни проведення форм оцінювання:

1. Контрольна робота 1: до 7 тижня семестру.
2. Контрольна робота 2: до 14 тижня семестру.
3. Лабораторні роботи 1-3: до 7 тижня семестру.
4. Лабораторні роботи 4, 5: до 14 тижня семестру.

Студенти мають право на одне перескладання кожної контрольної роботи у визначений викладачем термін.

У випадку відсутності студентів з поважних причин відпрацювання та перескладання контрольних робіт здійснюються у відповідності до «Положення про порядок оцінювання знань студентів при кредитно-модульній системі організації навчального процесу» від 1 жовтня 2010 року.

Студенти мають право захищати лабораторні роботи протягом усього навчального семестру.

7.3 Шкала відповідності оцінок.

Відмінно / Excellent	90-100
Добре / Good	75-89
Задовільно / Satisfactory	60-74
Незадовільно / Fail	0-59

8. Структура навчальної дисципліни. Тематичний план лекцій і лабораторних занять.

№ лекції	Назва лекції	Кількість годин		
		Лекції	Лабораторні заняття	Самостійна робота
Частина 1. Цілі та методи тестування програмних систем.				
1	Тема 1. Основні поняття в галузі тестування програмних систем.	2		4
2	Тема 2. Методи статичного тестування в процесі конструювання програмних систем. Формальна інспекція.	2		7
3	Тема 3. Базові методи динамічного тестування коду.	2	2	4
4	Тема 4. Стратегії модульного, інтеграційного та системного тестування програмних систем	2	4	4
5	Тема 5. Стратегії тестування інтерфейсу користувача та веб-застосунку.	4	4	6
6	Тема 6. Призначення та методи регресійного й дослідницького тестування. Автоматизоване тестування.	4		6
7	Тема 7. Основні моделі процесу тестування та його зрілості.	2		7

Контрольна робота 1				2
Всього по частині 1		18	10	40
Частина 2. Задачі й методи оцінювання надійності програмних систем.				
8	Тема 8. Інженерія надійності програмних систем: цілі, основні поняття, задачі.	2		4
9	Тема 9. Моделювання показників надійності та їх розподіл по модулях програмної системи.	2		8
10	Тема 10. Моделі раннього прогнозування надійності програмних систем – без даних про відмови.	2		8
11	Тема 11. Моделі визначення надійності програмної системи за даними про її відмови.	2		10
12	Тема 12. Планування демонстраційного тестування надійності. Моделі процесу інженерії надійності.	2		8
Контрольна робота 2				2
Всього по частині 2		10		40
Консультація			2	
ВСЬОГО		28	10	80

Загальний обсяг – **120** год, в тому числі:

Лекції – **28** год.

Лабораторні заняття – **10** год.

Консультації – **2** год.

Самостійна робота – **80** год.

Лабораторні роботи:

Лабораторна робота 1: Розроблення, відлагодження та модульне тестування самим розробником консольної програми обчислення значення заданої тригонометричної або гіперболічної функції ($\sinh(x)$ тощо) на підставі її розкладу в ступеневий ряд Маклорена та збереження результатів обчислень у файлі.

Лабораторна робота 2: Тестування функцій і системне тестування заданого застосунку-форми.

Лабораторна робота 3: Виконання та документування дослідницького тестування заданих динамічних сайтів на підставі накопичуваного досвіду з використанням on-line ресурсів.

Лабораторна робота 4: Документування дефектів, виявлених у лаб. роботах №1-3, у баг-трекері Mantis.

Лабораторна робота 5: Обчислення якості застосунків, тестованих у лаб. роботах №1-3, на підставі результатів тестування, документованих у лаб. роботі №4.

9. Рекомендовані джерела.

Основні:

1. Куликов С.С. Тестирование программного обеспечения. Базовый курс: практ. пособие. 2-е изд. / С. С. Куликов. – Минск: Четыре четверти, 2018. – 300 с.
2. Казарин О.В. Надежность и безопасность программного обеспечения: учеб. пособие для бакалавриата и магистратуры / О. В. Казарин, И. Б. Шубинский. – М.: Издательство Юрайт, 2018. – 342 с.
3. Майерс Г. Искусство тестирования программ / ГМайерс., Т Баджетт, К.Сандлер – «ДИАЛЕКТИКА», 2012, 3-е изд. – 272 с.
4. Сеницын С.В. Верификация программного обеспечения. Уч. пособие / С.В. Сеницын, Н.Ю. Налютин – Бином, 2008. – 368 с.
5. Котляров В.П. Основы тестирования программного обеспечения. Учебное пособие / В.П.Котляров – М.: Национальный Открытый Университет «Интуит», 2016 – 349 с.
6. Черников Б.В. Оценка качества программного обеспечения. Практикум / Б.В. Черников, Б.Е. Поклонов – М.: ИД «ФОРУМ»: ИНФРА-М, 2012. – 400 с.
7. Кирьянчиков В.А. Конспект лекций по дисциплине «Качество и надежность программного обеспечения». / В.А. Кирьянчиков, Э.А. Опалева – Санкт-Петербург: Изд-во СПбГЭУ «ЛЭТИ», 2002. – 86 с.
8. Бобало Ю.Я. Математичні моделі та методи аналізу надійності радіоелектронних, електротехнічних та програмних систем / Ю.Я. Бобало та ін. – Львів, Видавництво Львівської політехніки, 2013. – 300 с.
9. Lakey P.V, Neufelder A. System and software reliability assurance notebook. Rome Laboratory Report – Rome NY: Griffits Air Force Base. – 1997. – 186 p.

Додаткові:

1. Мюллер Т. Программа обучения. Сертифицированный тестировщик Базового уровня. – Т. Мюллер, Д. Фрайденберг ISTQB, 2011. – 101 с.
2. Плаксин М.А. Тестирование и отладка программ для профессионалов будущих и настоящих [Электронный ресурс]. – М.: БИНОМ. Лаборатория знаний, 2015. – 170 с.
3. Кент Бек. Экстремальное программирование: разработка через тестирование. / К. Бек; пер. П. Анджан. – СПб.: Питер, 2017. – 224 с.
4. Криспин Л. Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд / Л. Криспин, Дж. Грегори – Вильямс, 2010. – 464 с.
5. Copeland L. A practitioner Guide to Software Test Design / L. Copeland – STQE Publishing. 2004 – 238 p.
6. Spillner A. H. Software Testing Foundations: A Study Guide for the Certified Tester Exam / A. Spillner, T. Linz, H. Schaefer – 4th ed., Rocky Nook, 2014 – 305 p.
7. Whittaker J.A. Exploratory Software Testing / J.A. Whittaker – Addison-Wesley, 2010. – 253 p.
8. Фолк Д. Тестирование программного обеспечения / Д. Фолк, Е. Нгуен, С. Канер – Диасофт, 2003 – 400 с.
9. Павлов В.Л. Метод обратной семантической трассировки для контроля качества в гибкой разработке программных проектов / В.Л. Павлов и др. // Проблеми програмування. 2008. № 2-3. Спец. випуск – С. 211-218

10. Кулямин В.В. Перспективы интеграции методов верификации программного обеспечения. [Электронный ресурс]. – Режим доступа: <http://citforum.ru/SE/testing/integration/>.
11. Макгрегор Дж. Тестирование объектно-ориентированного программного обеспечения. Практическое пособие. / Дж. Макгрегор, Д. Сайкс – 2002 – 432 с
12. Карпов А. Как статический анализ дополняет TDD [Электронный ресурс]. – Режим доступа: <http://www.viva64.com/ru/a/0080/>
13. Гэртнер М. ATDD – разработка программного обеспечения через приемочные тесты / М. Гэртнер – М.: ДМК Пресс, 2013. – 232 с.
14. Stephens M. Design Driven Testing: Test Smarter, Not Harder / M. Stephens, D. Rosenberg – 2010 – 365 p.
15. Pairwise testing. Сборник/коллекция ссылок на описание и обсуждение метода. [Электронный ресурс]. – Режим доступа: <http://softwaretesting.ru/forum/index.php?/topic/22360/>.
16. Grindal M. Combination Testing Strategies – A Survey / M. Grindal, J. Offutt, S.F. Andler // Softw. Test. Verif. Reliab. – 2005. – N 15. – P. 167–199.
17. Месарош Дж. Шаблоны тестирования xUnit. Рефакторинг кода тестов / Дж. Месарош. Пер. с англ. – М.: Вильямс, 2009. – 832 с.
18. Офіційний сайт проекту TestDriven.NET. [Электронный ресурс]. – Режим доступа: <http://www.testdriven.net/>.
19. Офіційний сайт проекту EasyMock. [Электронный ресурс]. – Режим доступа: <http://www.easymock.org/>.
20. Офіційний сайт проекту Rhino.Mocks [Электронный ресурс]. – Режим доступа: <http://www.ayende.com/projects/rhino-mocks.aspx>.
21. Дудников Д. Тестируйте не числом, а умением. [Электронный ресурс]. – Режим доступа: <http://software-testing.ru/forum/index.php?/topic/17363/>.
22. Офіційний сайт групи UniTesK Lab Інституту системного програмування РАН – [Электронный ресурс]. – Режим доступа: <http://www.unitesk.ru/>.
23. Swinkels R. A comparison of TMM and other Test Process Improvement Models. Draft. / R. Swinkels // Frits Philips Institute, 2000. – 53 P. [Электронный ресурс]. – Режим доступа: <http://www.bruegge.informatik.tu-muenchen.de/static/contribute/Lehrstuhl/documents/12-4-1-FPdef.pdf>.
24. Koomen T. TMap® Next for result-driven testing / T. Koomen et al. – Sogeti, 2007 – 106 p.
25. Реєстр каркасів тестування для різних мов програмування. [Электронный ресурс]. – Режим доступа: http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks.
26. Баранцев А. Кроссбраузерное тестирование / А.Баранцев // [Электронный ресурс]. – Режим доступа: <http://software-testing.ru/forum/index.php?/topic/21974/>.
27. Реєстр засобів статичного аналізу коду. [Электронный ресурс]. – Режим доступа: http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis.
28. Повышение качества при использовании средств диагностики Visual Studio. [Электронный ресурс]. – Режим доступа: <http://msdn.microsoft.com/ru-ru/library/dd264943.aspx>
29. Lyu M.R. Handbook of Software Reliability Engineering / M.R. Lyu – Mc-Graw-Hill/IEEE, 1996. – 850 p.

30. Монахов Ю.М. Функциональная устойчивость информационных систем. Часть 1. Надежность программного обеспечения. Уч. пособие / Ю.М. Монахов – Владимир, изд-во Владимирского ГУ им. А.Г. и Н.Г. Столетовых – 2011. – 60 с.
31. Смагин В.А. Основы теории надёжности программного обеспечения: учеб. для вузов / В.А. Смагин, А.Н. Дорохов – БГТУ «ВОЕНМЕХ». – СПб.: 2009. – 303 с.
32. Електронна бібліотека з надійності фірми ReliaSoft (ReliaWiki). [Електронний ресурс]. – Режим доступу: <http://reliawiki.com/index.php/>.
33. Азовцев В.В. Программирование и основы алгоритмизации. Исследование методов оценки и повышения надежности программного обеспечения. Пояснительная записка к курсовому проекту по курсу «Технология разработки программного обеспечения». [Електронний ресурс]. – Режим доступу: <http://www.azovikdip.ru/>.
34. Маєвський Д.А. Анализ моделей надежности гарантоспособного программного обеспечения / Д.А. Маєвський, С.А. Яремчук // Електромашинобудування та електрообладнання. – 2010. – Вип. 76 – С.68–79.
35. Musa J.D. Software Reliability Engineering: More Reliable Software Faster and Cheaper. / J.D. Musa – Second Ed., 2004. – 235 p.
36. Goel A.L. Software Reliability Models: Assumptions, Limitations and Applicability // IEEE Trans. on Software Engineering / A.L. Goel // V. SE-11. – N12. – 1985. – P. 1411-1423.