**Лекція №3. Інфраструктура перевірки правильності програмних систем**

Під *інфраструктурою* перевірки правильності (доведення, верифікації і тестування) програмних систем розуміють інтегрований набір загальнодоступних технічних, технологічних і методологічних ресурсів, що знаходяться у розпорядженні команди розробників, верифікаторів і тестувальників, які виконують роботи з розроблення правильної системи за договорами із організаціями-замовниками.

В лекції №2 ми розглядали частково розглядали ці питання, зокрема, стратегічну мету тестування, об'єкти процесу тестування, дії команди розробників, верифікаторів і тестувальників, класифікацію помилок (помилка, дефект, відмова, які розподіляють на класи: 1) логічні і функціональні помилки; 2) помилки обчислень і часу виконання; 3) помилки вводу-виводу і маніпулювання даними; 4) помилки інтерфейсів; 5) помилки обсягу даних і ін.).

Наведені типи помилок розподіляються за процесами ЖЦ і їм відповідають такі джерела їхнього виникнення:

– ненавмисне відхилення розробників від робочих стандартів або планів реалізації;

– специфікації функціональних і інтерфейсних вимог виконані без дотримання стандартів розробки, що призводить до порушення функціонування програм;

– організації процесу розробки – недосконале або недостатнє управління керівником проекту ресурсами (людськими, технічними, програмними і т.д.) і питаннями тестування й інтеграції елементів проекту.

За рекомендацією стандарту ISO/IEC–12207 процес тестування кожного процесу ЖЦ і типи помилок, що виявляються під час, можна визначити так.

*Процес розробки вимог*. При визначенні вихідної концепції системи і вихідних вимог до системи виникають помилки аналітиків при специфікації вищого рівня системи і побудові концептуальної моделі предметної області.

Характерними помилками цього процесу є:

– неадекватність специфікації вимогам кінцевих користувачів;

– некоректність специфікації взаємодії ПС із середовищем функціонування або з користувачами;

– невідповідність вимог замовника окремим і загальним властивостям ПС;

– некоректність опису функціональних характеристик;

– незабезпеченість інструментальними засобами всіх аспектів реалізації вимог замовника й ін.

*Процес проектування.* Помилки при проектуванні компонентів можуть бути наслідком недоліків в описі алгоритмів, логіки керування, структур даних, інтерфейсів, логіки моделювання потоків даних, форматів вводу-виводу та ін. В основі цих помилок лежать дефекти специфікацій заданих аналітиками і недоробки проектувальників. До них належать помилки, пов'язані з:

– погодженістю інтерфейсу користувача із середовищем;

– описом функцій (неадекватність цілей і задач компонентів, що виявляються при перевірці комплексу компонентів);

– визначенням процесу обробки інформації і взаємодії між процесами (результат некоректного визначення взаємозв'язків компонентів і процесів);

– некоректним завданням даних і їхніх структур при описі окремих компонентів і ПС у цілому;

– некоректним описом алгоритмів модулів;

– визначенням умов виникнення можливих помилок у програмі;

– порушенням прийнятих для проекту стандартів і технологій.

*Процес кодування.* На даному процесі виникають помилки, що є результатом дефектів проектування, помилок програмістів і менеджерів у процесі розроблення і налагодження системи. Причиною помилок є:

– безконтрольність значень вхідних параметрів, індексів масивів, параметрів циклів, вихідних результатів та ін.;

– неправильна обробка нерегулярних ситуацій при аналізі кодів повернення від викликуваних підпрограм, функцій і ін.;

– порушення стандартів кодування (погані коментарі, нераціональне виділення модулів і компонентів та ін.);

– використання одного імені для позначення різних об'єктів або різних імен одного об'єкта, погана мнемоніка імен;

– непогоджене внесення змін у програму різними розробниками та ін.

*Процес тестування****.*** На цьому процесі помилки допускаються програмістами і тестувальниками при виконанні технології збирання і тестування, вибору тестових наборів і сценаріїв тестування та ін. Відмови в програмному забезпеченні, викликані такого роду помилками, повинні виявлятися, усуватися і не впливають на статистику помилок компонентів і на програмне забезпечення в цілому.

*Процес супроводу.* На процесі супроводу виявляються помилки, причиною яких є недоробки і дефекти експлуатаційної документації, недостатні показники кодифікованості й легкості читання, а також некомпетентність осіб, відповідальних за супровід і/або удосконалення ПС. Залежно від сутності внесених змін на цьому процесі можуть виникати практично будь-які помилки, аналогічні раніше перерахованим помилкам на попередніх процесах.

**Джерела помилок.**

Помилки можуть бути виникнути в процесі розроблення проекту, компонентів, коду і документації. Як правило, вони виявляються при виконанні або супроводі програмного забезпечення в найбільш несподіваних і різних її точках.

Причиною появи помилок є – нерозуміння вимог замовника; неточна специфікація вимог у документах проекту та ін. Це приводить до того, що реалізуються деякі функції системи, що будуть працювати не так, як пропонує замовник. У зв'язку з цим проводиться спільне обговорення замовником і розробником деяких деталей вимог для їхнього уточнення.

Команда розробників системи може також змінити мову опису системи.

Деякі помилки можуть бути не виявлені (наприклад, неправильно задані індекси або значення змінних цих операторів).

**Визначення тесту.**

Для перевірки правильності програм спеціально розробляються тести і тестові дані. Під *тестом* розуміється деяка програма, призначена для перевірки працездатності іншої програми і виявлення в ній помилкових ситуацій. Тестову перевірку можна провести також шляхом введення в програму, які перевіряється, операторів, які будуть сигналізувати про хід її виконання й отримання результатів.

*Тестові дані* слугують для перевірки роботи системи і складаються різними способами: генератором тестових даних, проектною групою на основі документів або наявних файлів, користувачем з специфікаціях вимог та ін. Дуже часто розробляються спеціальні форми вхідних документів, у яких відображається процес виконання програми за допомогою тестових даних.

Створюються тести, що перевіряють:

– повноту функцій;

– погодженість інтерфейсів;

– коректність виконання функцій і правильність функціонування системи в заданих умовах;

– надійність виконання системи;

– захист від збоїв апаратури і не виявлених помилок та ін.

Тестові дані готуються як для перевірки окремих програмних елементів, так і для груп програм або комплексів на різних стадіях процесу розроблення.

Багато типів тестів готуються замовником для перевірки роботи програмної системи. Структура і зміст тестів залежать від виду елемента тестування, яким може бути модуль, компонент, група компонентів, підсистема або система. Деякі тести залежать від мети і необхідності знати: чи працює система відповідно до її проекту, чи задоволені вимоги і чи бере участь замовник у перевірці роботи тестів тощо.

Залежно від задач, що ставляться перед тестуванням програм, складаються тести перевірки проміжних результатів проектування елементів на процесах ЖЦ, а також створюються тести іспитів остаточного коду системи.

**Тестування інтегрованої системи.**

Тести для перевірки окремих елементів системи і тести інтегрованої системи мають загальні і відмінні риси.

***Приклад.*** *Схема інтеграції компонентів*.

На першому рівні схеми знаходяться, наприклад, компоненти А, B, D, на другому рівні – E, C, G. Вони пов'язані між собою інтерфейсом (рис.1).

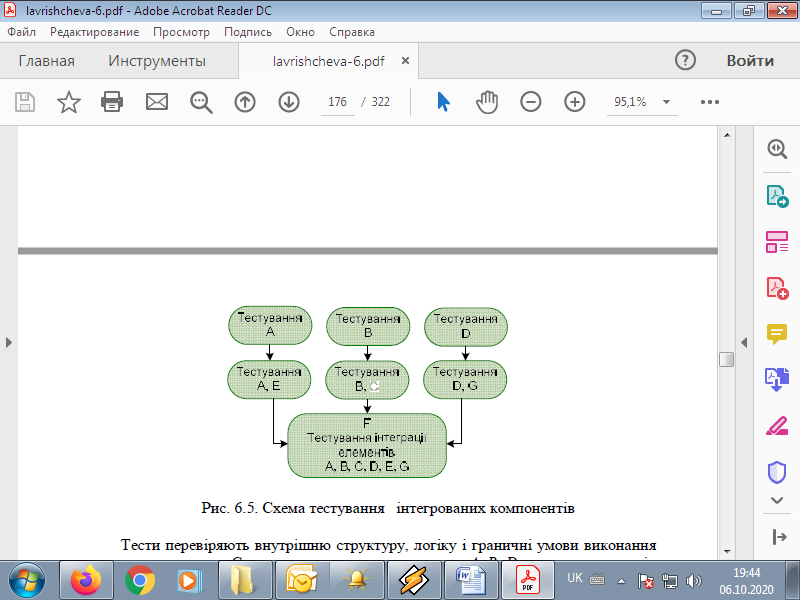


Рис. 1. Схема тестування інтегрованих компонентів

Кожен компонент схеми тестується окремо від інших компонентів тестами, що містять у собі набори даних і сценаріїв, складені відповідно до їхніх типів і функцій, специфікованих у вимогах до системи. Тестування проводиться в контрольному операційному середовищі на заданій безлічі тестових даних і операцій, розроблених з ними.

Тести перевіряють внутрішню структуру, логіку і граничні умови виконання кожного компонента. Спочатку тестуються компоненти *А, B*, *D* незалежно один від одного і кожний з окремим тестом. Після їхньої перевірки виконується перевірка інтерфейсів для зв'язку компонентів другого рівня: *Аº*→º*E*, *B* → *C*, *D* → *G,* а потім вже компоненти *Е*, C, *G*.

Компоненти й інтерфейси інтегруються і утворюють компонент *F,* він перевіряється на правильність інтеграції і функцій.

При тестуванні можуть виникати помилки. Вони, зазвичай, – результат неправильного завдання параметрів в операторах виклику або помилок в алгоритмі обчислення процедур або функцій. Помилки, що виникають у в зв'язках, усуваються, а потім повторно перевіряється зв'язок з компонентом *F* у вигляді трійки: компонент – інтерфейс – компонент.

Наступний крок тестування комплексної системи – перевірка функціонування системи за допомогою тестів перевірки функцій і вимог до них. Після цього перевіряється комплекс на виконавчих і іспитових тестах відповідно до вимог до ПС, апаратури і виконуваних функцій. Іспит системи проводиться в реальному середовищі, у якому система буде функціонувати надалі.

***Приклад*.** Оцінювання часу тестування та ризиків відмов модулів.

Нехай ми маємо приклад деякої системи інформаційно-аналітичної підтримки прийняття управлінських рішень із модулів, наведених у табл. 1, які функціонують у розподіленому середовищі Oracle RunTime та MS Office (Word, Excel).

Таблиця 1. Склад програмних комплексів ПС

| **Шифр ПК** | **Призначення** |
| --- | --- |
| ПК1 | Підготовка вхідних даних |
| ПК2 | Ведення діловодства і контролю виконавської діяльності |
| ПК3 | Контроль і введення регламентованої інформації до БД |
| ПК4 | Надання довільних та регламентованих довідок |
| ПК5 | Формування звітної доповіді |
| ПК6 | Діагностична експертиза |
| ПК7 | Моніторинг стану діяльності |

Найбільший внесок у ризик відмов робить ПК3 контролю і введення даних до БД Oracle, який функціонує на 10 робочих місцях. Його головне призначення – контроль даних у формах, підготовлених за допомогою ПК1 та їх експорт до БД. ПК3 виконує запис у більш як 90 таблиць БД, використовує понад 50 нормативно-довідкових таблиць та класифікаторів.

На процесі проектування для цього ПК був виконаний аналіз можливих сценаріїв функціонування (табл. 2) за декількома способами використання.

Таблиця 1. Функції модулів, що надають найбільший внесок у відмови ПС

| **Модуль** | **Функції** |
| --- | --- |
| М1. Реєстрація та імпорт | Реєстрація документів, їх форм і даних про стан комплексу. Перехід у середовище Excel для опрацювання форм |
| М2. Контроль стану | Контроль опрацювання форм. Визначення ступеню повноти завантаження даних з форм за кожним надісланим документом (повністю, не повністю, не виконане) |
| М3. Експорт | Завантаження даних з форми до БД (підключення до сервера БД, параметризація збережених процедур, завантаження) |
| М4. Контроль | Контроль правильності наданих форм у Excel. Синтаксичний контроль форми (типи та формати даних, коди класифікаторів тощо) та семантичний контроль (непротирічність даних, відповідність обмеженням) |
| М5. Запити до БД | Запити до БД з метою виключення можливого дублювання інформації, яка надходить у формах з різних джерел |

Функції кожного з модулів М1-М5 дійсно є критичними для ПС, оскільки від їх безвідмовної роботи залежить цілісність системи.

Для реєстрації часу виконання *t* та моментів відмов у модулі М1-М5 на час тестування потрібно вбудувати відповідні фрагменти коду, що мати моменти початку та завершення (нормального або аварійного) роботи модуля та їх реєстрація у журналі подій і відмов. Це надасть змогу отримати дані про відмови та оцінити параметри надійності, вартість тестування і усунення відмов. Останні розраховуються за такими чинниками:

– вартості часу роботи фахівців (тестувальників та розробників);

– визначеного реального часу виконання кожного модуля під час тестування та часу, витраченого на усунення дефектів.

**Інженерія керування тестуванням**

За функціональні і системні тести несуть відповідальність розробник і замовник, останній більше впливає на складання тестів для випробувань системи.

Цей процес реалізує група тестувальників, що не залежать від групи розробників ПС. Її очолює керівник групи, який повинен мати:

– досвід в області тестування;

– здатність бути лідером і керувати групою тестувальників;

– знання з задач предметної області ( і програмного продукту);

– знання з інфраструктури (апаратного і системного програмного забезпечення).

Рядовий тестувальник повинен знати:

– галузь виробництва продуктів/технологій створення ПС;

– елементи інфраструктури розроблення ПС;

– вимоги до системи і стандарти тестування;

– підходи до використання робочих продуктів процесу тестування;

– інструменти і стратегії тестування;

– вміти аналізувати результати і підбирати нові тестові дані або додавати дані для оцінювання процесу тестування.

Деякі члени цієї групи – досвідчені фахівці або навіть професіонали в цій галузі. До них також належать аналітики, програмісти, що працюють в галузі розроблення систем від її початку. Вони мають справу не тільки зі специфікаціями, а й з методами і засобами проектування, тестування, організують створення і виконання тестів. Із самого початку тестувальники складають плани тестування, тестові дані, сценарії, а також графіки виконання тестів.

Професійні тестувальники працюють разом із групою керування конфігурацією, щоб забезпечити їх документацією й іншими механізмами для зв'язку між собою тестів і вимог проекту, конфігурації і коду. Вони розробляють методи і процедури тестування. У цю команду включаються додаткові фахівці, що ознайомлені з вимогами системи або з підходами до її розробки. Аналітики входять до складу команди, тому що вони розуміють проблеми визначення специфікацій замовників.

Багато фахівців порівнюють тестування системи зі створенням нової системи, у якій аналітики визначають потреби і цілі замовника, працюючи разом із проектувальниками і намагаючись реалізувати ідеї і принципи роботи системи.

Проектувальники системи повідомляють групі тестувальників проектні цілі, щоб вони знали декомпозицію системи на підсистеми і її функції, а також принципи роботи. Після проектування тестів група тестувальників проводить аналіз можливостей системи.

Оскільки тести і тестові сценарії є прямим відображенням вимог до проекту в цілому, перспективи керування конфігурацією системи визначаються саме цією групою. Зміни, що виявляються в програмі, помилки в системі відбивають у документації, вимогах, проекті, а також в описах вхідних і вихідних даних або в інших розроблюваних артефактах. Внесені зміни в процесі розроблення призводять до модифікації тестових сценаріїв або більшою мірою до зміни планів тестування.

Фахівці з керування конфігурацією враховують ці зміни і координують складання тестів з її урахуванням.

До групи тестувальників входять також користувачі. Вони оцінюють отримувані результати, зручність використання, а також висловлюють свою думку про принципи роботи системи.

Уповноважені замовника планують роботи доти, поки використовується і супроводжується система. При цьому вони можуть привнести деякі зміни в проект через неповноту заданих вимог і сформулювати системні вимоги для проведення верифікації системи і прийняття рішень про її готовність і корисність.

**Планування тестування.**

Для проведення тестування розробляється план (Test Plan), у якому описуються стратегії, ресурси і графік тестування окремих компонентів і системи в цілому. У плані визначаються роботи для різних членів команди, що виконують свої ролі в цьому процесі. План містить у собі також визначення ролі тестів у кожнім процесі, ступінь покриття програми тестами і відсоток тестів, що виконуються зі спеціальними даними.

Тестові інженери створюють тестові сценарії (Test Cases), кожний з яких перевіряє результат взаємодії між актором і системою на основі перед- і посту мов використання таких сценаріїв. Сценарії в основному належать до тестування за типом «білої скриньки» і орієнтовані на перевірку структури й операцій інтеграції компонентів системи.

Для проведення тестування тестові інженери пропонують процедури тестування (Test Procedures), що вміщують валідацію об'єктів і верифікацію тестових сценаріїв відповідно до плану графіку. Оцінка тестів (Test Evaluation) полягає в оцінці результатів тестування, ступеня покриття програм сценаріями і статусу отриманих помилок. На рис. 2. наведено коло обов'язків інженера-тестувальника. Тестувальник інтегрованої системи перевіряє інтерфейси і дає оцінку виконання відповідних системних тестів, а потім аналізує результати тестування.

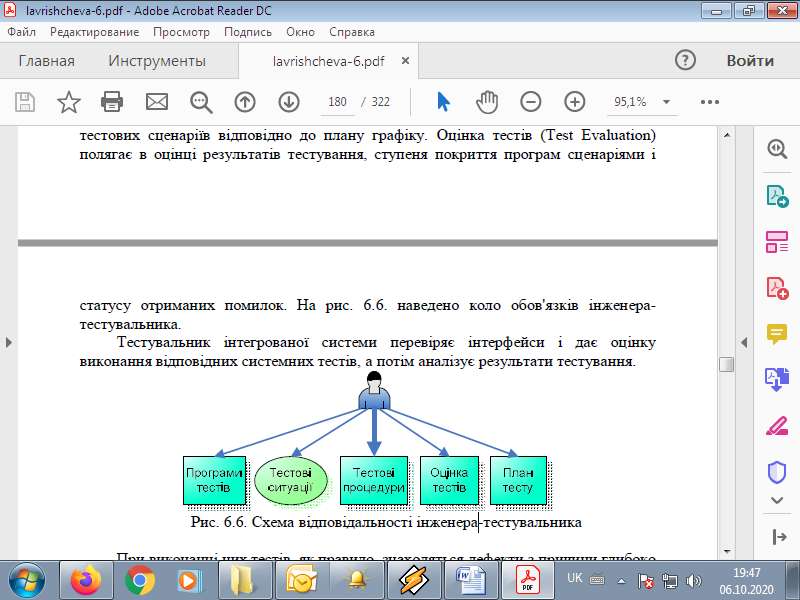


Рис. 2. Схема відповідальності інженера-тестувальника

При виконанні цих тестів, як правило, знаходяться дефекти з причини глибоко захованих недоліків у програмах, що виявляються при тривалому тестуванні системи на тестових даних.

**Керування тестуванням.**

Усі засоби тестування ПС об’єднуються базою даних, де містяться результати тестування системи, а також компоненти, тестові контрольні дані й інформацію про документування процесу тестування.

База даних проекту підтримується спеціальними інструментальними CASE-засобами, що забезпечують ведення аналізу предметної області, збирання даних про їхні об'єкти, потоки даних тощо. База даних проекту зберігає також початкові й еталонні дані, що використовуються для зіставлення даних, накопичених у базі, з даними, що отримані в процесі тестування системи.

При тестуванні виконуються різні види обчислень характеристик цього процесу за методами планування і керування.

1. *Розрахунок тривалості виконання функцій* шляхом збирання середніх показників швидкості виконання операторів без виконання програми на комп’ютері. Виявляються компоненти, що вимагають тривалого часу виконання в реальному середовищі.

2. *Керування виконанням тестування* шляхом підбору тестів перевірки, їхнього виконання, селекції результатів тестування і зіставлення їх з еталонними значеннями. Результати даного процесу відображаються на дисплеї, наприклад, гілки виконання у графічній формі, дані про відмови і помилки або конкретні значення вихідних параметрів програми. Ці дані аналізуються розробниками для формулювання висновків про напрями подальшої перевірки правильності програми або їхньому завершенні.

3. *Планування тестування* призначене для розподілу термінів робіт з тестування, розподілу тестувальників за окремими видами робіт і складання ними тестів перевірки системи. Визначається стратегія і шляхи тестування. У діалозі запитуються дані про реальні значення процесу виконання системи, структури розгалуження вершин графа і параметрах циклів. Перевірені цикли, як правило, вилучаються зі шляхів виконання програми. При плануванні шляхів виконання створюються відповідні тести, критерії і вхідні значення.

4. *Результати тестування документуються* відповідно до діючого стандарту ANSI/IEEE 829 і містять у собі:

– опис задач, призначення і зміст ПС, а також перелік функцій відповідно до вимог замовника;

– технологію розробки системи;

– плани тестування різних об'єктів, що відповідають технологічним прийомам проведення тестування;

– тести, контрольні приклади, критерії і обмеження, методику оцінки результатів виконання програмного продукту на процесі тестування;

– облік процесу тестування, складання звітів про аномальні події, відмови і дефекти в підсумковому документі системи.

***Контрольні запитання для самоперевірки***.

1. Надайте визначення *інфраструктури* перевірки правильності ПС?
2. Яким чином типи помилок розподіляються за процесами ЖЦ?
3. Наведіть приклад помилок на етапі виявлення вимог.
4. Які помилки виникають на процесі проектування?
5. Які помилки виникають на процесі кодування?
6. В чому причина появи помилок на етапі тестування?
7. В чому причина появи помилок на етапі супроводу?
8. Які кваліфікаційні вимоги висуваються до команди тестувальників?
9. Які дії виконуються на етапі планування тестуванням?
10. В чому полягає керування тестуванням?
11. Назвіть сутність інфраструктури організації робіт з тестування.

**Рекомендована література**

1. Бабенко Л.П., Лавріщева К.М*.* Основи програмної інженерії.– Навч. посібник.–К.: Знання, 2001.– 269с.
2. И. Соммервиль. Инженерия программного обеспечения,  
    6 изд. – И.д. "Вильямс", 2002.
3. Лавріщева К.М. Програмна інженерія. Електронний підручник: http://csc.knu.ua/uk/library/books/lavrishcheva-6.pdf
4. Лавріщева К.М. Програмна інженерія. – Підручник.–К.:Академперіодика, 2008.–415с.
5. Тамре Л. Введение в тестирование программного обеспечения / Л. Тамре. –М.: Издательский дом «Вильямс», 2003. –368 с.
6. Сайкс Д. Тестирование объектно-ориентированного программного обеспечения. Практическое пособие / Д.Сайкс, Д.Макгрегор. –К.: Диасофт, 2002. –432 с.
7. Фолк Д. Тестирование программного обеспечения / Д.Фолк, Е.К. Нгуен, С.Канер. – К.: Диасофт, 2003. –400 с.

***Для самостійного вивчення***: Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.