Лекція 1.
Тестування як спосіб забезпечення якості програмного продукту

Зміст

Цільова настанова: Розглянуто проблематику, цілі та вимоги до курсу. Обговорено основні теми курсу і практикуму.

Ключові слова: тестування, якість програмного продукту, вимоги до курсу, теми курсу і практикуму, споживачі курсу.

1.1. Вступ

Справжній курс присвячений вивченню способів забезпечення і контролю якості програмного забезпечення з позицій тестування. У цій області, поряд з вирішенням наукових і технічних проблем, важлива роль при-слід проблемі підготовки кадрів, здатних вирішувати завдання тестування і автоматизації тестування в умовах виробництва програмного продукту.

Завданням дисципліни, яка реалізуються через лекційний матеріал і практикум, є підготовка тестувальників програмного проекту. Це тим більш важливо, що в існуючих вузівських програмах підготовки професійних програмістів не передбачений достатній для рішення даного завдання обсяг лекційного матеріалу і практикумів.

1.2. Вимоги до курсу

Курс відповідає вимогам спеціальності 122 "Комп'ютерні науки та інформаційні технології" першого (бакалаврського) рівня, кричи-ентірованной на підготовку професійних програмістів, і, в першу чергу, тестувальників і покриває розділи курсу «Технологія програмування», присвячені тестуванню.

1.3. Основні теми лекційного курсу

- Основні поняття тестування: термінологія тестування, відмінності тестування і налагодження, фази і технологія тестування, проблеми тестування.
- Критерії вибору тестів: структурні, функціональні, стохастичні, мутаційні, оцінки покриття проекту.
- Різновиди тестування: модульне, інтеграційне, системне, регресійне, автоматизація тестування, витрати тестування.
- Особливості процесу і технології індустріального тестування: планування тестування, підходи до розробки тестів, особливості ручної розробки і генерації тестів, автоматизація тестового циклу, документування тестування, огляди і метрики.
- Регресійне тестування: особливості та види регресійного тестування, методи відбору тестів, оцінка ефективності.
- Термінологічний словник: містить глосарій термінології тести-вання відповідно до IEEE Standard Glossary of Software Engineering.

В курсі використані приклади, розроблені на мові C #, крім того, ці ж приклади продубльовані на C.C ++ в Додатку.

1.4. Основні теми практикуму

Для демонстрації і закріплення теоретичних знань розроблений практикум, що містить:

- опис лабораторних робіт;
- методичні вказівки по проведенню самостійної роботи студентів;
- рекомендації по підготовці комп'ютерної лабораторії до проведення лабораторних робіт.

В рамках практикуму студенти освоюють різні підходи до розробки тестів і тестування та умови їх застосування.

Практикум представлений у формі тренінгу, в якому розглянуті наступні теми:

- Розробка документації на тестируемую систему і її оточення: опис вимог (Requirement Specification) і специфікацій розробника (High Level Design).
- Планування тестування.
- Практикум модульного тестування.
- Практикум інтеграційного тестування.
- Практикум системного тестування.
- Ручне тестування і тестові процедури.
- Автоматизоване тестування на основі скриптів.
- Автоматизоване тестування на основі Msc-діаграм і генерація тестів.
- Засоби підтримки автоматизації тестування.

Використовуючи модель реальної системи управління, студенти можуть:

- розробляти різні види тестів і тестуючих програм;

• шукати дефекти системи в процесі тестування, брати участь в їх виправленні і модернізації тестованого програми;
• розробляти документацію - вимоги до системи, тести і тестові процедури - і відстежувати взаємозв'язок цих документів з розробленими тестами.

Прогнозовані результати

В результаті вивчення курсу:

1. Виробляється розуміння умов застосування верифікації, валідації та тестування.
2. Виробляються навички і прийоми тестування, що застосовуються на різних фазах розробки якісного програмного продукту.
3. Оцінюються умови ефективного застосування інструментальних засобів для розробки якісного програмного забезпечення.
4. Виробляються навички розробки тестових програм і тестових наборів в проекті Програми.
5. Виробляються навички розробки проектної документації для етапу тестування.
6. Виробляються навички планування та відстеження завдань тестування.
7. Забезпечуються основи навчання проектної команди, що складається з розробників і тестувальників.
8. Виробляються навички тестування програмного забезпечення проектів, розроблених на C #.

1.5. Мотивація до вивчення технологій тестування

1. Тестування - це перспективно. Спробуємо описати ситуацію з фахівцями з тестування в країнах СНД. Фахівців явно недостатньо. Тестерів не готує жоден ВНЗ. У той же час бізнес програмування - це один з найбільш динамічно розвиваються бізнесів. У Росії щорічний оборот становить майже 1 млрд. Доларів, в Україні - близько 300 млн. (Звичайно, ці цифри - неофіційні). Це більше, ніж в будь-якій іншій європейській країні. Потреба у фахівцях - величезна і, в першу чергу, в фахівцях з тестування.
2. Тестування - це вигідно. У ситуації, коли фахівців не вистачає, а що розвивається бізнес вимагає їх все більше і більше, компанія готова вкласти гроші в їх навчання в області тестування і це дуже хороша можливість для них у всіх відносинах (навіть якщо вони не збираються будувати свою подальшу кар'єру в цьому напрямку).
3. Тестування - це цікаво. Це не тільки функціональне тестування, яким в більшості випадків доводиться зараз займатися. Це тестування вимог, тестування дизайну (для того, щоб тестувати дизайн потрібно добре в цьому розбиратися), розробка документації, написання скриптів для автоматичного тестування, робота з різними програмами для оцінки якості коду, продуктивності програм і т.п. Компанії зобов'язані планувати і серйозно зайнятися поліпшенням підходу до тестування, і вони зможуть реально навчитися професійному підходу і навіть внести свій вклад в розвиток

цього підходу.

4. Тестування - це творчість. Помилково вважається, що програмування - це творча робота, а робота тестера - це рутина. Якщо уважно проаналізувати, порівнювати роботу програміста і роботу тестера, то все виглядає не настільки однозначно. Програміст робить те, що просить, замовляє клієнт. У нього є можливість проявити себе на початкових стадіях, наприклад, коли розробляється дизайн програми (це якщо його залучають до дизайну). Після цього він робить те, що йому говорять і не більше. З тестуванням інша справа. Наприклад, є два підходи до тестування програм - переконатися, що програма відповідає вимогам клієнта і довести, що вона не відповідає вимогам клієнта. Обидва підходи дають простір для творчості. В обох випадках у тестера є мета, а як її досягти - це залежить від креативності людини (навіть якщо є вся належна документація).
5. Тестування - це престижно. У всіх компаніях тестери, які роблять свою роботу якісно, користуються великим авторитетом. Саме від того, наскільки якісно виконано тестування залежить якість кінцевого продукту. Нормальний програміст відчуває подяку до хорошого тестеру за те, що він допомагає робити його (програміста) роботу краще (а іноді допомагає уникнути ганьби). А про ставлення керівництва компанії годі й говорити - природно, що хороші тестери - на вагу золота. Заробити хорошу репутацію, працюючи тестером і роблячи свою роботу на совість, можна досить швидко.
6. Тестування - це кар'єрне зростання. Компанія планує розвиватися і, крім усього іншого, планує розвивати якість. Для хороших фахівців з тестування з часом можливий кар'єрне зростання - керівники з тестування проектів, керівники проектів, аналітики і т.п.
7. Тестування і програмування - дві сторони одного процесу. Як, на перший погляд, не дивно, програміст повинен бути зацікавлений в тому, щоб тестер знайшов в програмі якомога більше проблем, помилок, нестиковок і т.п. Знати про проблему - це означає наполовину розв'язати цю проблему. Якщо важлива проблема не буде знайдена на початкових етапах, то виправити її в подальшому буде набагато складніше. Крім того, це дозволяє програмісту розвиватися - тобто він бачить проблеми, знайдені в його коді, і буде намагатися не повторювати їх у майбутньому. Зрештою, є така річ, як репутація та для програмістів - це дуже важлива річ. Якщо продукт вийде якісним - це на руку і програмісту, і тестеру. До речі, питання репутації в свою чергу безпосередньо пов'язаний з кар'єрою, преміюванням і т.п.

1.6. Історія розвитку тестування забезпечення

Перші програмні системи розроблялися в рамках програм наукових досліджень або програм для потреб міністерств оборони. Тестування таких продуктів проводилося строго формалізовано із записом всіх тестових процедур, тестових даних, отриманих результатів. Тестування виділялося в окремий процес, який починався після завершення кодування, але при цьому, як правило,

виконувалося тим же персоналом.

У 1960-х багато уваги приділялося «вичерпного» тестування, яке повинно проводитися з використанням всіх шляхів в коді або всіх можливих вхідних даних. Було відзначено, що в цих умовах повне тестування ПО неможливо, тому що, по-перше, кількість можливих вхідних даних дуже велике, по-друге, існує безліч шляхів, по-третє, складно знайти проблеми в архітектурі і специфікаціях. З цих причин «вичерпне» тестування було відхилено і визнано теоретично неможливим.

На початку 1970-х тестування ПО позначалося як «процес, спрямований на демонстрацію коректності продукту» або як «діяльність по підтвердженню правильності роботи ПЗ». У зароджувалася програмної інженерії верифікація ПО значилася як «доказ правильності». Хоча концепція була теоретично перспективною, на практиці вона вимагала багато часу і була недостатньо всеохоплюючою. Було вирішено, що доказ правильності - неефективний метод тестування ПО. Однак в деяких випадках демонстрація правильної роботи використовується і в наші дні, наприклад, приймально-здавальні випробування. У другій половині 1970-х тестування уявлялося, як виконання програми з наміри третьому знайти помилки, а не довести, що вона працює. Успішний тест - це тест, який виявляє раніше невідомі проблеми. Даний підхід прямо протилежний попередньому. Зазначені два визначення є «парадокс тестування», в основі якого лежать два протилежних твердження: з одного боку, тестування дозволяє переконатися, що продукт працює правильно, а з іншого - виявляє помилки в ПЗ, показуючи, що продукт не працює належним чином. Друга мета тестування є більш продуктивною з точки зору поліпшення якості, так як не дозволяє ігнорувати недоліки ПО.

У 1980-х тестування розширилося таким поняттям, як попередження дефектів. Проектування тестів - найбільш ефективний з відомих методів попередження помилок. В цей же час стали висловлюватися думки, що необхідна методологія тестування, зокрема, що тестування має включати перевірки на всьому протязі циклу розробки, і це повинен бути керований процес. В ході тестування треба перевірити не тільки зібрану програму, а й вимоги, код, архітектуру, самі тести. «Традиційне» тестування, яке існувало до початку 1980-х, відносилося тільки до компільованою, готової системі (зараз це зазвичай називається системне тестування), але в подальшому тестувальники стали залучатися в усі аспекти життєвого циклу розробки. Це дозволяло раніше знаходити проблеми у вимогах та архітектури і тим самим скорочувати терміни і бюджет розробки. В середині 1980-х з'явилися перші інструменти для автоматизованого тестування. Передбачалося, що комп'ютер зможе виконати більше тестів, ніж людина, і зробить це більш надійно. Спочатку ці інструменти були вкрай простими і не мали можливості написання сценаріїв на скриптових мовах.

На початку 1990-х в поняття «тестування» стали включати планування, проектування, створення, підтримку і виконання тестів і тестових оточень, і це означало перехід від тестування до забезпечення якості, що охоплює весь цикл розробки ПЗ. У цей час починають з'являтися різні програмні інструменти для

підтримки процесу тестування, а саме: більш просунуті середовища для автоматизації з можливістю створення скриптів і генерації звітів, системи управління тестами, ПО для проведення навантажувального тестування. В середині 1990-х з розвитком Інтернету і розробкою великої кількості веб-додатків особливу популярність стало отримувати «гнучке тестування» (за аналогією з гнучкими методологія-ми програмування).

У 2000-х з'явилося ще ширше визначення тестування, коли в нього було додано поняття «оптимізація бізнес-технологій» (business technology optimization, BTO). BTO направляє розвиток інформаційних технологій відповідно до цілей бізнесу. Основний підхід полягає в оцінці і максимізації значущості всіх етапів життєвого циклу розробки ПЗ для досягнення необхідного рівня якості, продуктивності, доступності.

1.7. Якість програмного продукту

Кожен день у своїй роботі ми стикаємося з досить абстрактним поняттям «якість ПО» і якщо поставити запитання тестувальників або програмісту - «що таке якість?», То у кожного знайдеться своє тлумачення.

Якість програмного продукту характеризується набором властивостей, що визначають, наскільки продукт «хороший» з точки зору зацікавлених сторін, таких як замовник продукту, спонсор, кінцевий користувач, розробники і тестувальники продукту, інженери підтримки, співробітники відділів маркетингу, навчання і продажів. Кожен з учасників може мати різне уявлення про продукт і про те, наскільки він хороший чи поганий, тобто про те, наскільки висока якість продукту.

Таким чином, постановка задачі забезпечення якості програмного продукту виливається в задачу визначення зацікавлених осіб, їх критеріїв якості і потім - знаходження оптимального рішення, що задовольняє цим критеріям.

Отже, що ж таке якість програмного забезпечення?

Спробуємо дати визначення термінам якість і якість програмного забезпечення. Основною проблемою є той факт, що визначення якості занадто неясне і неоднозначне. Це викликано тим, що зазвичай термін якість розуміється неправильно. Така плутанина може пояснюватися кількома причинами.

- Перша, якість - це не окремо взята ідея або поняття, швидше за багатовимірна і неоднозначна концепція.
- Друга, для будь-якого поняття і визначення існує кілька рівнів абстракції, наприклад, коли люди говорять про якість, одна частина розуміє під цим занадто широкий і розмитий сенс, в той час як інша може посилатися на конкретне визначення і значення.
- Третя, термін якість є невід'ємною частиною нашого повсякденного спілкування, проте загальноприйняте і професійне використання може бути досить сильно відрізнятися.

Популярний погляд на якість. Загальноприйнята думка про якість таке, що це щось нематеріальне і "невловиме" - про нього можуть вестися суперечки і дискусії, його можна критикувати і вихваляти, але зважити і виміряти якість

неможливо. Такі вирази як «хорошу якість» і «погана якість» служать наочним прикладом, як люди кажуть про щось невизначеному для них, то, що вони не можуть чітко характеризувати і визначити. Таку думку відображає той факт, що люди сприймають і інтерпретують якість по-різному. Мається на увазі, що якість не може бути контрольованим і керованим, і тим більше воно не може бути кількісно вимірюваних. Інша популярна думка, що якість нерозривно пов'язане з розкішшю, першим сортом і тонким смаком. Дорогий, досконально продуманий і більш технічно складний продукт розглядається як гарантія високої якості, ніж дешевші аналоги. Відповідно до такого підходу, якість обмежено певним класом дорогих продуктів з хитромудрої функціональністю і класовим продуктам. Простіше кажучи, чи недорогий продукт буде класифікований як якісний продукт. На жаль, таке невизначене і розпливчасте уявлення не може бути використано для поліпшення процесів розробки програмного забезпечення і явно контрастує з професійним підходом до управління якістю.

Професійний підхід до управління якістю стверджує, що якість - це чітко визначена величина, яку можна виміряти і проконтролювати, вона піддається управлінню і поліпшенню. Спробуємо дати чітке і зручне для роботи визначення. У літературі в 1970 році якість було визначено як «відповідність вимогам», а в 1979 його визначили, як «придатність до використання». Ці два визначення тісно пов'язані і прекрасно узгоджуються, як ми побачимо пізніше.

«Відповідність вимогам» передбачає, що вимоги повинні бути настільки чітко визначені, що вони не можуть бути зрозумілі і інтерпретовані некоректно. Пізніше, на етапі розробки, проводиться регулярне оцінки розробленого продукту, для визначення відповідності вимогам. Будь-які невідповідності повинні розглядаються як дефекти - відсутність якості. Наприклад, специфікація на певну модель радіостанції може вимагати можливості приймати певну частоту радіохвиль на відстані більш ніж 30 кілометрів від джерела мовлення. У разі якщо радіостанція не здатна виконати дану вимогу, вона не задовольняє вимоги до якості і повинна бути визнана непридатною і неякісною. Виходячи з тих же принципів, якщо Кадиллак відповідає всім вимогам до машин Кадиллак, значить це якісна машина. Якщо Шевроле відповідає всім вимогам до машин Шевроле, отже, це теж якісна машина. Ці дві машини можуть бути абсолютно різними за стилем, швидкості і економічності, але, якщо обидві вимірювати за стандартними для них наборам, тоді вони обидві будуть якісними машинами.

Визначення «придатність до використання» приймає до уваги вимоги та очікування кінцевих користувачів продукту, які очікують, що продукт або сервіс, що надається буде зручним для їх потреб. Однак різні користувачі можуть використовувати продукт по-різному, це означає, що продукт повинен володіти максимально різноманітними варіантами використання. Згідно визначення, кожен варіант використання - це характеристика якості і всі вони можуть бути класифіковані за категоріями в якості параметрів придатності до використання.

Ці два визначення якості («відповідність вимогам» і «придатність до використання») по суті рівнозначні. Різниця в тому, що варіант «придатність до

використання» вказує на важливу роль вимог і очікувань замовника. Роль замовника, пов'язана з якістю, ніколи не може бути переоцінена. З точки зору замовника, якість продукту, який він придбав, складається з безлічі різних факторів, таких як: ціна, продуктивність, надійність і т.п.

Тільки ваш замовник може розповісти вам про якість, тому що це єдине що він дійсно купує. Замовник не купує продукт. Він купує ваші гарантії того, що всі його очікування до продукту будуть реалізовані. Якість - це придатність до використання. Чи робить даний продукт то, в чому маю потребу полегшує він мою роботу, чи можу я його використовувати так, як мені зручно.

Якість– з точки зору замовника або кінцевого користувача – це придатність до використання. Дане визначення бере до уваги вимоги та очікування кінцевих користувачів продуктів в тому, що продукт або сервіс, що надається буде зручний для їх потреб.

А тепер подивимося на точку зору розробника.

Якість – з точки зору розробника – це відповідність специфікованою і зібраним вимогам.

Проблема в тому, що специфіковані і зібрані вимоги - це зазвичай лише частина всіх реальних вимог і очікувань замовника. Де ж правильне визначення якості? Якість - це відповідність реальним вимогам, явним і неявним. Дуже часто неявні вимоги настільки очевидні для замовника або користувача, що він навіть не припускає, що вони невідомі розробникам. Для прикладу повернемося до наших автомобілів - замовник може детально описати вимоги до дизайну, параметрам двигуна, оформлення салону, кольором кузова, але ніде не вказати, що шини повинні бути круглими, а лобове скло - прозорим.

Замовник буде задоволений тільки тоді, коли куплений товар буде повністю задовольняти його реальним і життєвим вимогам, як специфіковані, так і немає.

За численними відгуками західних представників основним недоліком компаній, що виробляють ПО, є слабке розуміння бізнес - потреб клієнтів.

Приклад: клієнт замовив додаток для розробки і друку триколірних візитних карток. Програміст Іванов подумав, що ми живемо в століття прогресу, кольорових лазерних принтерів і триколірні візитки - відстій. Тому візитки повинні бути кольоровими. Він реалізує свої ідеї і відправляє програму клієнтові в термін. А ось тут відбувається дивна річ - клієнт у люті. Вся справа в тому, що він планував випустити на ринок програму для друку триколірних візиток, продаючи її, скажімо за, 15 доларів за диск. А потім, продавши достатню кількість дисків, запропонувати нову версію програми, яка буде робити повнокольорові візитки, і продавати її по 30 доларів, а з тих клієнтів, які купили у нього попередню версію взяти по 10 доларів за оновлення. У підсумку, клієнт вимагає переробити програму за гроші компанії і вимагає у компанії неустойку за те, що програма не була готова до терміну.

Наступний приклад: клієнт домовляється з компанією про демо-версії програми, в якій буде реалізовано ядро нової системи і частина інтерфейсу. Про терміни домовилися. Клієнт замовляє конференц-зал, запрошує журналістів, потенційних споживачів і планує презентацію демо-версії програми, все оплачує вперед. Програміст Петров працює над ядром. У якийсь момент

він розуміє, що ядро можна і потрібно зробити краще. Якщо так зробити, то програма буде працювати швидше в 10 разів. На це буде потрібно всього лише 3 додаткових дні. В результаті ядро системи сяє, але до встановленого терміну не готове вікно зі звітами. У підсумку презентація провалюється, тому що вікно зі звітами є основною відмінністю програми від аналогів. Клієнт несе колосальні збитки, подає в суд на компанію і подальше виробництво продукту зупиняється. Сяюче ядро системи нікому не потрібно.

Сенс в тому, що потрібно намагатися зрозуміти, що для клієнта є найбільш важливим. Якась дрібна помилка в віконці Ебаут (About) з неправильно написаним прізвищем губернатора або мера, якого автори проекту дякують за сприяння, може привести до плачевних наслідків. І в такій ситуації якість всієї іншої системи може виявитися неоціненим, тобто вся робота - «нанівець». Дуже важливо, щоб і програмісти, і тестери це розуміли і думали про це. Будь-які відхилення від початкових домовленостей щодо термінів, по функціональності повинні обговорюватися з клієнтом.

1.8. Якість ПЗ в контексті міжнародних стандартів

На закінчення розглянемо визначення «якості програмного забезпечення» в контексті міжнародних стандартів.

Управління якістю на підприємствах, в тому числі і виробляють програмне забезпечення, регламентується стандартами серії ISO 9000.

ISO 9000 - це серія міжнародних стандартів, розроблених техні-ного комітетом міжнародної організації зі стандартизації (ISO).

Стандарти ISO прийняті більш ніж 90-та країнами світу в якості національних стандартів і застосовні до будь-яких підприємств, незалежно від їх місця розташування, чисельності, обсягу випуску продукції і сфери діяльності.

Сертифікація проводиться по єдиному стандарту з цієї серії ISO 9001. При цьому діє дворівнева система підтвердження відповідності. Сертифікацією окремих підприємств займаються спеціально сформовані аудиторські організації. Вони, в свою чергу, акредитуються національними акредитаційними товариствами. Втім, суті-ють і незалежні системи акредитації.

Важливо розуміти, що відповідність стандарту ISO не гарантує високу якість продукції. Відповідність вимогам і рекомендаціям цих стандартів говорить тільки про здатність підприємства підтримувати стабільність якості і покращувати результативність своєї роботи. Також відповідність вимогам ISO 9001 свідчить про деяке рівні надійності постачальника.

Мета серії стандартів ISO - стабільне функціонування документованої системи менеджменту якості підприємства-постачальника. Вихідна спрямованість стандартів серії ISO була саме на відносини між компаніями у формі потребітельпоставщік. З прийняттям у 2000 році черговий версії стандартів ISO більша увага стала приділятися якості всієї діяльності підприємства, а не тільки якістю продукції. Тепер стандарт ISO робить акцент на задоволенні всіх зацікавлених сторін, а не тільки споживачів. Стандарти ISO допомагають підприємствам формалізувати їх систему менеджменту, вводячи, зокрема, такі

системоутворюючі поняття, як внутрішній аудит, процесний підхід, коригувальні та запобіжні дії.

Отже, ISO існує як універсальний документ, який залишається практично незмінними незалежно від географії використання. Таку широку привабливість стандарт має завдяки своєму витонченому міні-малізму і в той же час широкої сфери застосування. Поняття споживача, процесу, ланцюжка постачання, постійного поліпшення якості - є універсальними. Ця універсальність і є ядром його успіху.

Безперечно, універсальність була б суперечливим фактором, якби не була джерелом чудової архітектури для систем управління якістю. Стандарт ISO успішно працює в самих різних галузях, від про-ництва до сервісу, від маленьких магазинчиків до мульти національних корпорацій, від суперприбутковими компаній до державних і громадських організацій - скрізь стандарт доводить, що він гідний, називаються тися ефективною моделлю управління.

На даний момент найбільш поширена і використовується багаторівнева модель якості програмного забезпечення, представлена в наборі стандартів ISO.

На верхньому рівні виділено 6 основних характеристик якості ПЗ, кожну з яких визначають набором атрибутів, що мають відповідними-ющие метрики для подальшої оцінки (рис. 1.1).

Іншими словами, з точки зору ISO, якість програмних засобів можна визначити, як сукупну характеристику досліджуваного ПО з урахуванням наступних складових:

- надійність;
- сопровождаемость;
- практичність;
- ефективність;
- мобільність;
- функціональність.

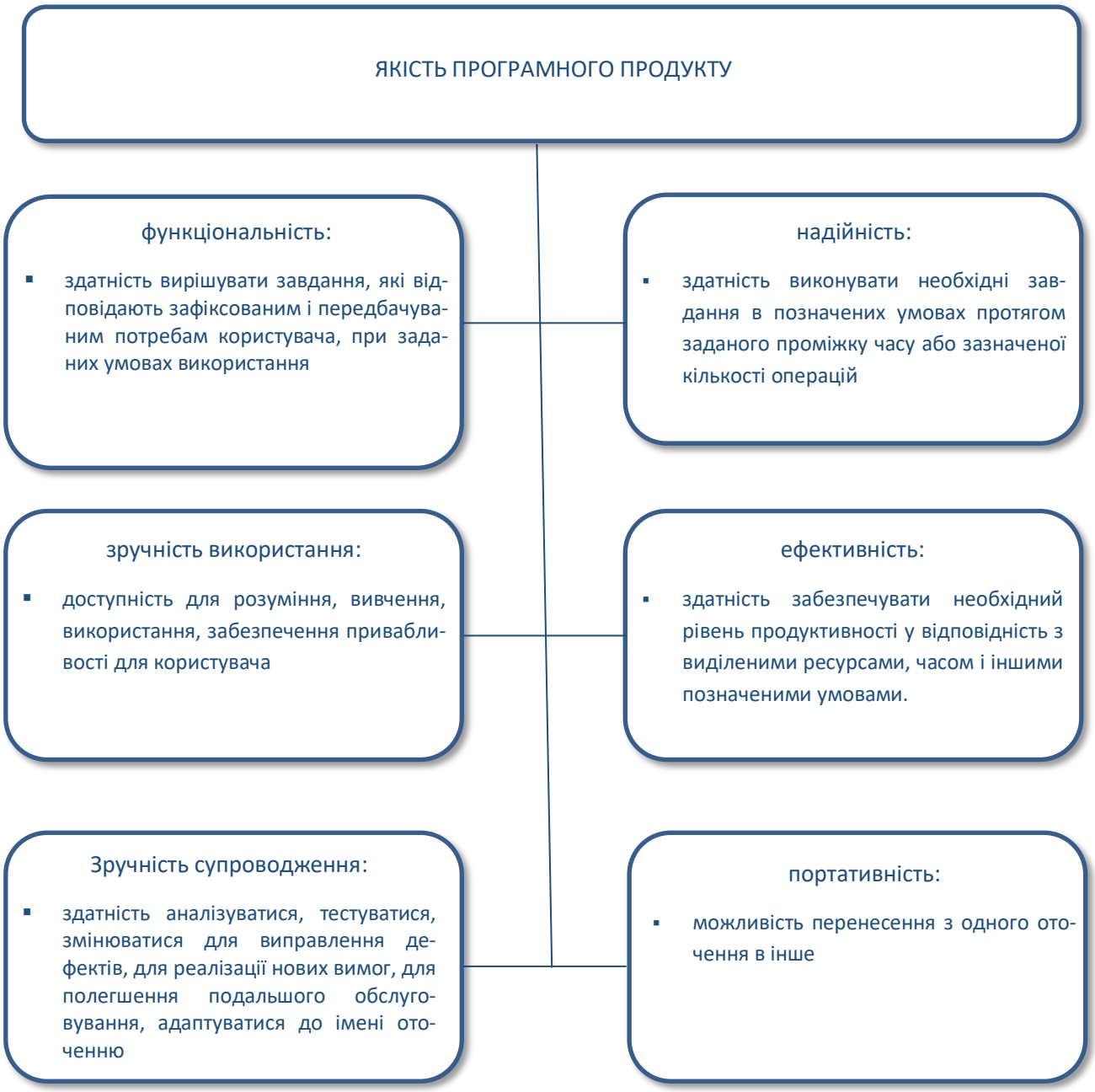

Рис.1.1. Модель якості програмного продукту

Розглянемо характеристики якості ПЗ:

- Функціональність - визначається здатністю ПО вирішувати завдання, які відповідають зафіксованим і очікуваним потребам користувача, при заданих умовах використання ПЗ. Тобто ця характеристика відповідає за те, що ПО працює справно і точно, функціонально сумісно, відповідає стандартам галузі і захищене від несанкціонованого доступу.
- Надійність - здатність ПЗ виконувати необхідні завдання в позначених умовах протягом заданого проміжку часу або зазначеної кількості операцій. Атрибути даної характеристики - це завершеність і цілісність всієї системи, здатність самостійно і коректно відновлюватися після збоїв в роботі, відмовостійкість.

- Зручність використання - доступність ПО для розуміння, вивчення, використання, забезпечення привабливості для користувача.
- Ефективність - здатність ПЗ забезпечувати необхідний рівень продуктивності у відповідність з виділеними ресурсами, часом і іншими позначеними умовами.
- Зручність супроводу - здатність ПЗ аналізуватися, тестуватися, змінюватися для виправлення дефектів, для реалізації нових вимог, для полегшення подальшого обслуговування та адаптуватися до імені оточенню.
- Портативність - характеризує ПО з точки зору легкості його перенесення з одного оточення в інше.

Розглянемо способи забезпечення якості програмного забезпечення.

Тестування є одним з найбільш усталених способів забезпечення якості програмного забезпечення. Тестування входить в набір ефективних засобів сучасної системи забезпечення якості програмних продуктів.

З технічної точки зору тестування полягає у виконанні програми на деякій множині вихідних даних і звірці одержуваних результатів із заздалегідь відомими (еталонними), з метою встановити відповідність різних властивостей і характеристик програми замовленим властивостями. Як одна з основних фаз процесу розробки програмного продукту (Дизайн додатка - Розробка коду - Тестування), тестування характеризується досить великим внеском в сумарну трудомісткість розробки продукту.

Широко відома оцінка розподілу трудомісткості між фазами створення програмного продукту: 40% -20% -40%, з чого випливає, що найбільший ефект у зниженні трудомісткості може бути отриманий насамперед на фазах Design і Testing.

Тому основні вкладення в автоматизацію або генерацію коду слід здійснювати, перш за все, на цих фазах. Хоча в сучасному індустріальному програмуванні автоматизація тестування є широко поширеною практикою, в той же час технологія верифікації вимог та специфікацій поки робить лише свої перші кроки.

Завданням найближчого майбутнього є рух в бік такого розподілу трудомісткості (60% -20% -20%), щоб сумарна вартість виявлення більшості дефектів прагнула до мінімуму за рахунок виявлення переважного числа на найбільш ранніх фазах розробки програмного продукту.

1.9. Метрики щодо забезпечення якості ПЗ

Згідно з міжнародним стандартом ISO 14598:

Метрика - це кількісний масштаб і метод, який може використовуватися для вимірювання.

У ведення і використання метрик необхідно для поліпшення контролю над процесом розробки, а зокрема над процесом тестування, який ми і будемо розглядати далі.

Мета контролю тестування полягає в отриманні зворотного зв'язку і візуалізації процесу тестування. Необхідну для контролю інформацію збира-

ють (як в ручну так і автоматично) і використовують для оцінки стану і прийняття рішень, таких як покриття (наприклад, покриття вимог або коду тестами) або критерії виходу (наприклад, критерії закінчення тестування). Метрики, так само можуть бути використані для оцінки прогресу виконання запланованих робіт і освоєння бюджету .

Створення, використання і аналіз метрик

Для більшої наочності має сенс згрупувати метрики за типами сутностей, що беруть участь в забезпеченні якості та тестуванні програмного забезпечення, а саме:

1. Метрики за тестовими випадків (Test Cases);
2. Метрики по Багам / дефектів;
3. Метрики за завданнями.
4. Розберемо кожну з них:

Метрики по тест кейсів

Назва	Опис
Passed/Failed Test Cases	Метрика показує результати проходження тест кейсів, а саме відношення кількості вдало пройдених до завершився з помилками. В ідеалі, до кінця проекту, кількість провальних тестів прагнути до нуля
Not Run Test Cases	Метрика показує кількість тест кейсів, які ще необхідно виконати в даній фазі тестування. Маючи цю інформацію, ми можемо проаналізувати і виявити причини, за якими тест не були проведені

Метрики по багам

Назва	Опис
Open/Closed Bugs	Метрика показує відношення кількості відкритих багів до закритих (виправленим і Перевірена)
Reopened/Closed Bugs	Метрика показує відношення кількості перевідкриття багів до закритих (виправленим і Перевірена
Rejected/Opened Bugs	Метрика показує відношення кількості відхилених багів до відкритих
Bugs by Severity	Кількість багів по серйозності
Bugs by Priority	Кількість багів за пріоритетом

Слід зазначити, що метрики " Open / Closed Bugs ", " Bugs by Severity "і" Bugs by Priority "добре візуалізують ступінь наближення продукту до досягнення критеріїв якості по Багам. Маючи вимоги до кількості відкритих багів, після кожної ітерації тестування ми порівнюємо їх з реальними даними, тим самим бачачи місця, де нам потрібно додати, для якнайшвидшого досягнення

мети.

Метрики " Reopened / Closed Bugs " і " Rejected / Opened Bugs " спрямовані на відстеження роботи окремих учасників груп розробки і тестування.

Приклад перший:

Припустимо, ми маємо ситуацію, коли кількість перевідкривається після усунення багів не зменшується або навіть зростає. Це є сигналом до того, що необхідно провести аналіз причин, тому що подібна ситуація може показати, що:

1. Вимоги до функції можна трактувати по-різному;
2. Тестувальник неточно описав проблему;
3. Неякісне поверхневе вирішення проблеми (фікс бага).

Другий приклад покаже для чого необхідна метрика " Rejected / Opened Bugs ":

Ми спостерігаємо, що відсоток відхилених (Rejected) багів дуже великий. Це може означати:

1. Вимоги до функції можна трактувати по-різному;
2. Тестувальник неточно описав проблему;
3. Розробник не бажає виправляти допущену їм помилку чи не вважає, що це насправді помилка. (Ця проблема є прямим наслідком 2-ий, що виникла через не точного опису) .

Всі ці проблеми помітно дестабілізують обстановку на проекті. Тому, при їх виникненні, рекомендується провести коротку бесіду з керівниками проектних груп, щоб надалі зменшити кількість перевідкриття і відхилених дефектів.

Метрики за завданнями

Назва	Опис
Deployment tasks	Метрика показує кількість і результати установок програми. Процедура установки програми була описана в статті Процедура проведення установки нової версії ПО (Deployment WorkFlow). У разі, якщо кількість відхилених командою тестування версій буде критично високим, рекомендується терміново проаналізувати і виявити причини, а також в найкоротші терміни вирішити наявну проблему.
Still Opened Tasks	Метрика показує кількість все ще відкритих завдань. До закінчення проекту всі завдання повинні бути закриті. Під завданнями розуміємо такі види робіт: написання документації (архітектура, вимоги, плани), імплементація нових модулі або зміна існуючих за запитами на зміни, роботи з налаштування стендів, різні дослідження і багато іншого.

Метрики за завданнями можуть різні, ми привели лише 2 з них. Так само цікава може бути метрика за часом виконання завдань і багато інших.

У висновку треба зазначити, що наявність необхідних метрик і графіків, які відображають зміни стану проекту в плині часу, дозволить вам поліпшити

не тільки процес тестування, а й розробки в цілому, а також полегшить процедуру проведення аналізу виконаного проекту , що дозволить в подальшому не допускати минулих помилок.

Контрольні запитання

1. Поясніть мотивацію до вивчення дисципліни. Ви згодні з такою моті вацією ?
2. Доведіть тезу, що тестування - це перспективно.
3. Доведіть тезу, що тестування - це престижно.
4. Проведіть історичний екскурс в історію тестування.
5. Чи існує зв'язок і аналогії між тестуванням промислової продукції і тестуванням програмних продуктів?
6. Як розуміють якість програмного забезпечення в контексті міжнародних стандартів?
7. Дайте визначення поняттю «якість».
8. Дайте визначення поняттю «якість програмного продукту» з точки зору замовника.
9. Дайте визначення поняттю «якість програмного продукту» з точки зору розробника.
10. Поясніть поняття «метрики щодо забезпечення якості програмного забезпечення.
11. Чим визначається якість ІС?
12. Які характеристики якості можна визначити?
13. Що визначає показник якості?
14. Охарактеризуйте дефектологічні властивості в залежності від цілей дослідження і етапів життєвого циклу ІС: дефектогенность, дефектабельность і дефектоскопічность.
15. Як формується показник якості?
16. Які існують види метричних шкал для вимірювання критеріїв?
17. Поясніть модель класифікації критеріїв якості інформаційних систем.
18. Що оцінюється за допомогою функціональних критеріїв?
19. Для чого призначені конструктивні критерії?