http://softuni.bg

Юнит тестване / Интеграционно тестване /

Системно тестване / Приемно тестване

Функционално и нефункционално тестване

Типове и нива на тестване

Софтуерен университет

Преподавателски екип

СофтУни

2

1. Нива на тестване

▪ Юнит тестване (компонентно тестване)

▪ Интеграционно тестване

▪ Системно тестване

▪ Тестване за приемане от краен потребител

(приемно тестване)

2. Типове тестване

▪

Функционално тестване

▪

Нефункционално тестване

Съдържание

sli.do

#QA-Basics

Въпроси

3

Нива на тестване

Юнит тестване / Интеграционно тестване /

Системно тестване / Приемно тестване

▪ Групи от тестови дейности

▪ Всяко ниво е част от тестовия процес

▪ Съответства на определена фаза от разработката на

софтуера

▪ Тест нива:

▪ Тестване за одобрение / Приемно тестване

(Acceptance testing)

▪ Системно тестване (System testing)

▪ Интеграционно тестване (Integration testing)

▪ Юнит тестване / Компонентно тестване (Unit testing)

Нива на тестване

5

6

Йерархия на тестовите нива

Приемно тестване

(Acceptance testing)

Интеграционно тестване

(Integration testing)

Юнит тестване

(Unit testing)

Системно тестване

(System testing)

Тестване на отделен компонент

Юнит тестване

8

▪ Проверяване на всяко

яйце от кутията, преди

закупуването ѝ

Юнит тестване в реалния живот

▪ Проверяване дали

фаянсовата плочка не е

счупена, преди

нанасянето на лепило

9

▪ Какво е юнит тестване?

▪ Нарича се още компонентно тестване

▪ Първото или най-базово ниво на тестване

▪ Тества отделни компоненти на софтуера

▪ Компонент може да бъде отделна функция, метод, модул или обект

▪ Обикновено се извършва от самите програмисти във фазата на писане на

код

▪ Изпълнява се изолирано

▪ Защо ни е необходимо?

▪ Проверява дали отделните компоненти работят коректно

▪ Позволява дефектите да бъдат отстранени рано, още във фазата на

разработка

Юнит тестване / Unit Testing

10

▪ Проверка на възрастта

Програмата задава въпрос: "На колко години си?"

Прочита вевъдените от потребителя "години"

ако годините са >= 18

връща вярно

в противен случай

връща невярно

Юнит тестване: Пример

Положителен тест(20) → очаква се вярно

Отрицателен тест (16) → очаква се невярно

Граничен тест (18) → очаква се вярно

Тестване на взаимодействието между компоненти

Интеграционно тестване

▪ Столове подходящи

за маса

Интеграционно тестване в реалния свят

▪ Врата и каса за врата

13

▪ Какво е интеграционно тестване?

▪ Второ ниво от процеса по тестване на софтуер

▪ Отделните компоненти или единици на софтуера се тестват в

група

Интеграционно тестване / Integration Testing (1)

▪ Извършва се от програмисти, QA

специалисти или специални

интеграционни екипи

▪ Предполага се, че компонентите

вече са тествани поотделно

14

▪ Защо ни е необходимо интеграционното тестване?

▪ След свързването на отделните компоненти един с друг,

може да възникне нова грешка

▪ Тестването трябва да потвърди, че всички свързани

компоненти си взаимодействат правилно

▪ Основната цел е да се открият грешките в:

▪ Интерфейси

▪ Взаимодействието между интегрирани компоненти

▪ Взаимодействието между системи

Интеграционно тестване / Integration Testing

(2)

15

▪ Вътрешно интеграционно тестване

▪ Разкрива дефекти в интерфейсите и взаимодействието между

интегрираните компоненти

▪ "Integration test in the small"

▪ Външно интеграционно тестване

▪ Тестване на съчетанието на системи и пакети

▪ Тестване на интерфейси към външни организации

▪ "Integration test in the large"

Поднива на интеграционното тестване

16

▪ GitHub има няколко модула (компоненти):

▪ Home Page → Login Page → User Dashboard

▪ Всеки от тях е тестван поотделно

▪ Искаме да проверим дали работят заедно

▪ Интеграционни тестове:

▪ Тестваме дали бутонът за вход води към формата за вход

▪ Тестваме дали след успешно влизане с потребителско име и

парола, се показва потребителският дашборд

▪ Тестваме дали след излизане от профила, потребителският

дашборд е недостъпен

Интеграционно тестване: Пример

Тестване на цялата система

Системно тестване

▪ В автомобилната индустрия, всеки произведен автомобил е щателно

тестван, в края на производствения процес:

▪ Двигател, джанти, волан, спирачки

▪ Врати, ключалки, седалки

▪ Електрическа система: светлини, чистачки,

климатик

▪ Мултимедийна система: радио, GPS, карти

▪ и т.н.

▪ QA-те вече са оценили всички тези функциoналности поотделно, а

също и взаимодействащи една с друга, но те трябва бъдат

тествани и като цялостна система

Системно тестване в реалния живот

19

▪ Какво е системно тестване ?

▪ Трето ниво от процеса по тестване на софтуер

▪ С фокус върху цялата система:

▪ Нейното поведение (какво прави системата)

▪ Колата правилно ли е сглобена, работи ли по предназначение?

▪ Нейните възможности (как се справя системата)

▪ Дали автомобилът е надежден, сигурен, в добро състояние, каква е

неговата производителност и ефективност

▪ Реализира се чрез тестване "от край до край" (E2E, end-to-end)

▪ Извършва се само от QA специалисти

Системно тестване / System Testing (1)

20

▪ Защо ни е необходимо системно тестване?

▪ Предишни тестове са били изпълнени спрямо технически

спецификации

▪ Системните тестове разглеждат системата от гледна точка на

крайния потребител

▪ Системните тестове проверяват дали всички компоненти на

дадена система функционират при реален сценарии

▪ Системното

тестване

може

да

бъде

функционално

и

нефункционално. Това гарантира работеща за крайния

потребител система

Системно тестване (2)

21

▪ Системното тестване

изисква специално

обособена "стейджинг"

среда

▪ Максимално точно копие

на сайта/приложението,

до което имат достъп

крайните потребители,

предназначено за

системно тестване

Системно тестване: Пример

•

Използва се от клиенти (на живо)

•

Пълни данни

•

Използва се от QA

специалисти

и/или клиенти за UAT

•

Ограничени данни

•

Използва се от QA специалисти

•

Без клиентски данни

•

Използва се от програмисти

•

Без клиентски данни

Тестване

(Testing)

Разработване

(Development)

Продукция

(Production)

Стейджинг

(Staging)

\*UAT - User Acceptance Testing /

Потребителско тестване за одобрение

Клиентът тества крайния продукт от бизнес

гледна точка

Тестване за приемане от крайния клиент

▪ Собственик на апартамент, проверява апартамента след

ремонт

▪ Всички стаи: подове, тавани, стени, врати, прозорци

▪ Уреди

▪ Мебели

▪ Вода и водопровод

▪ Електричество

▪ Газ

▪ и т.н.

Приемно тестване в реалния живот

24

▪ Какво е тестване за приемане от крайния клиент?

▪ Последното ниво, обикновено преди внедряване (deployment)

▪ Валидира цялостно функцонално бизнес решение

▪ Под внимание се вземат законовите и/или регулаторните изисквания

▪ Приемното тестване се изпълнява:

▪ От членове на бизнес екипа (алфа тестване)

▪ От крайни потребители (бета тестване)

▪ Следват се оперативни инструкции

▪ Гарантира се спазване на договорните и регулаторни насоки

Приемно тестване/ Acceptance Testing (1)

25

▪ Защо ни е необходимо тестване за приемане от крайния

потребител?

▪ Проверява работата на системата, обикновено преди

внедряване

▪ Основната цел е работещо бизнес решение

▪ Не се фокусира върху козметичните грешки

▪ Отговаря на въпроса, дали актуалното поведение на

системата съответства на очакванията на клиента

Приемно тестване (2)

26

▪ Най-новият Microsoft Windows се тества първо локално в Редмънд (алфа

тестване), след това от външни потребители (навсякъде по света)

Приемно тестване: Пример

Алфа тестери

▪ Група вътрешни потребители

▪ Запознати са с проекта

▪ Не участват пряко в развитието му

▪ Тестват дали приложението работи

правилно

▪ Дават обратна информация за това

как потребителското изживяване

може да се подобри

Бета тестери

▪ След алфа тестването, продуктът и

грешките се коригират

▪ Бета

тестването

се

извършва

от

избрана група крайни потребители

▪ Служи като "плавен старт"

▪ Обратна връзка от реални

потребители, които нямат

предварителни познания за

приложението и/или новите

функции

Типове тестване

Функционално и нефункционално тестване

28

▪ Група от тест дейности, които тестват специфични

характеристики на определена софтуерна система

▪ Типовете тестове се разделят на две основни групи:

▪ Функционално тестване

▪ Отговаря на въпроса "Какво?"

▪ Потвърждава правилното функциониране на софтуера

▪ Нефункционално тестване

▪ Отговаря на въпроса "Как?"

▪ Потвърждава ефективността и производителността на софтуера

Типове тестване

29

▪ Функционалното тестване на софтуер за онлайн банкиране

включва:

▪ Тестване дали средствата са точно преведени

▪ Дали лихвените изчисления са правилни

▪ Дали плащанията по сметки се извършват навреме

▪ Нефункционално тестване се фокусира върху сигурността на

системата:

▪ Да се гарантира, че достъпът е напълно безопасен

▪ Да се гарантира, че системата може да се справи с натоварването

▪ Особено в пикови периоди, като началото на месеца, когато масово се

изплащат заплати и други плащания към бюджета

Типове тестване: Пример

30

▪ Типовете тестване могат да се прилагат на много/всички

тестови нива

▪ Пример: тестване на сценарий "регистрирай потребител"

▪ Функционални тестове:

▪ Валидна потребителска информация, невалидна потребителска

информация, дублирана потребителска информация

▪ Нефункционални тестове:

▪ Производителност

(100

хиляди

потребители),

надеждност

(по 1 потребител в секунда за 24 часа), UX тест (навигацията

лесна ли е за потребителя)

Типове тестване и нива на тестване

Тестване на определени функции

Функционално тестване

32

▪ Сешоар: Функционално тестване

▪ Старт / Стоп

▪ Промяна на силата

▪ Горещ / студен въздух

▪ Йонни настройки

▪ Тестове на различни приставки

▪ Тестове на прибиращия се кабел

▪ и т.н.

Функционално тестване в реалния живот

33

▪ Тества функциите, които една система трябва да изпълнява

▪ Функциите са "какво" трябва да прави системата

▪ Потвърждава дали софтуерната система отговаря на

функционалните изисквания

▪ Функционалното тестване основно включва тестване тип черна

кутия (black box testing)

▪ Функционално покритие:

▪ Начин да се измери покритието на функционалните тестове

върху определена функционалност

Функционално тестване

▪ Тестване на основните функции на приложение

▪ Съобщения за грешка

▪ Проверка дали се извеждат

подходящи съобщения за грешка

▪ Базисно приложение

▪ Безпроблемна навигация през различните екрани

▪ Достъп

▪ Проверка на достъпа на потребителя до системата

34

Цели на функционалното тестване

35

▪ Функционални тестове за файлов архиватор (като 7-Zip или

WinRAR):

▪ Архивиране на папка в архив

▪ Разархивиране на папка

▪ Архивиране на един файл

▪ Архивиране на няколко файла

▪ Архивиране на празна папка

▪ и т.н.

Функционално тестване: Пример

Тества аспекти, които не са функционалност

Нефункционално тестване

37

▪ Сешоар: нефункционално тестване

▪ Тест за прегряване: ако сешоара работи 30 минути

▪ Колко време отнема да се промени температурата на

въздушния поток?

▪ Тест за шум

▪ Тест за падане

▪ Тест за тегло / размери

▪ Електрическият кабел достатъчно дълъг ли е?

▪ Лесно ли се използва с лява и дясна ръка?

Нефункционално тестване: Пример от реалния живот

38

▪ Нефункционалното тестване оценява:

▪ Надеждност

▪ Ефективност на работата

▪ Сигурност / безопасност

▪ Тества "Как" или какво е качеството, с което системата

изпълнява своите функции

Нефункционално тестване

39

▪ Нефункционалното тестване се фокусира главно

върху подобряване качеството на:

▪ Лесната употреба

▪ Ефективността

▪ Поддръжката

▪ Преносимостта на продукта

Цели на нефункционалното тестване

40

▪ Нефункционални тестове за файлов архиватор (като 7-Zip или

WinRAR):

▪ Тест за скорост: колко бързо се компресират файлове / папки

▪ Тест за скорост: колко бързо се разархивират файлове / папки

▪ Размер на архива: сравнение на различни нива на компресия

▪ Тест за сигурност: архивиране / разархивиране на файл, защитен

с парола

▪ Тест за препълване: компресиране / разархивиране на папка с

500 хил. файла

▪ Тест за претоварване: компресиране / разархивиране на 50

файла паралелно

Нефункционално тестване: Пример

▪ …

▪ …

▪ …

Какво научихме днес?

41

▪

Различни нива на тестване

▪

Юнит тестване: тестване на единичен компонент

▪

Интеграционно тестване: тестване на взаимодействието

между компонентите

▪

Системно тестване: QA-те тестват цялата система

▪

Приемно тестване: Клиентът тества крайния продукт

▪

Различни типове тестване

▪

Функционално тестване: тестване на софтуерната

функционалност

▪

Нефункционално тестване: производителност,

надеждност и др.

Въпроси?

Диамантени партньори на СофтУни

▪ Официален уеб сайт:

▪ Официален дискусионен форум:

▪ Официална фейсбук група:

Уеб сайт на курса, Форум и FB група

https://softuni.bg/trainings/4357/qa-basics-november-2023

https://softuni.bg/forum

https://www.facebook.com/groups/qabasicsnovember2023

44

45

Лиценз

▪ Този курс (презентации, примери, демонстрационен код,

упражнения, домашни, видео и други активи) представлява

защитено авторско съдържание

▪ Нерегламентирано копиране, разпространение или

използване е незаконно

▪ © СофтУни – https://softuni.org

▪ © Софтуерен университет – https://softuni.bg

46

Обучения в Софтуерен университет (СофтУни)

▪ Софтуерен университет – качествено образование,

професия и работа за софтуерни инженери

▪ softuni.bg

▪ Фондация "Софтуерен университет"

▪ softuni.foundation

▪ Софтуерен университет @ Facebook

▪ facebook.com/SoftwareUniversity

▪ Дискусионни форуми на СофтУни

▪ forum.softuni.bg

http://softuni.bg

Софтуерен университет

Преподавателски екип

СофтУни

QA, тестване, бъгове, принципи на тестване, процес на тестване

Софтуерно осигуряване на качеството:

Въведение

2

1. Какво означава софтуерно осигуряване на

качеството (SQA)?

2. Какво означава софтуерно тестване?

3. Софтуерни дефекти (бъгове)

4. Сравнение между Ръчно тестване и

Автоматизирано тестване

5. Седем принципа в тестването

6. Тест сценарии и тест случаи

(Test Scenarios and Test Cases)

Съдържание

3

sli.do

#QA-Basics

Въпроси

Какво означава софтуерно тестване?

Какво означава софтуерно осигуряване

на качеството?

5

▪ Какво е софтуерно осигуряване на качеството (SQA / QA)?

▪ SQA има за цел да гарантира, че софтуерът се държи според

очакванията

▪ SQA е методология за проверка на съответствието на софтуера

спрямо изискванията

▪ По-голяма част от работата на QA е софтуерното тестване: ръчно

и автоматизирано

▪ Софтуерните дефекти (бъгове) се докладват и проследяват чрез

системи за проследяване на дефекти (bug tracking systems)

▪ Процесът по софтуерно осигуряване на качеството се изпълнява

от QA специалисти

Софтуерно осигуряване на качеството (SQA)

6

▪ По-голяма част от QA работата е софтуерното тестване

▪ Ръчно тестване (кликни и провери резултатите)

▪ Автоматизирано тестване (QA автоматизация посредством

скриптове)

▪ Непрекъсната интеграция и непрекъснато внедряване (CI / CD

pipeline)

▪ Автоматизирано изграждане и

обновяване в тестова среда

(build and deploy)

▪ Автоматизирано изпълнение

на тестoве

▪ Известяване (notification) / доклад (report)

Софтуерно осигуряване на качеството (SQA)

7

▪ Софтуерното тестване е начин:

▪ За оценка на качеството на софтуера

▪ Да се провери дали софтуерът отговаря на определени

изисквания и да се открият бъгове

▪ Да се намали риска от повреда на софтуера при неговото

използване

▪ Процесът по анализиране на софтуерен продукт включва:

▪ Откриване на разликите между разработения софтуер и

разписаните спецификации

▪ Оценка на функционалностите на софтуерния продукт

Софтуерно тестване

8

▪ Основни цели на тестването

▪ Предотвратяване на дефекти

▪ Верификация на посочените изисквания

▪ Верификация на очакваното поведение на софтуера

▪ Да се намали рискът от възможен провал на софтуера

▪ Да предоставя информация на заинтересованите страни

▪ Да спомага за спазването на договорни, законови или

регулаторни изисквания

Цели на тестването

▪ Тестването проверява дали разработеният софтуер

отговаря на изискванията

▪ Тестването има за цел да открие и

докладва дефекти (bugs)

▪ Процесът по софтуерно тестване включва:

▪ Планиране на тестването: какво, кога, как?

▪ Дизайн на тестването: тест сценарии и тест случаи

▪ Настройка на тестовата среда: инсталиране, конфигуриране, подготовка

на тестови данни, …

▪ Реализация на тестовете: изпълнение на тестовете

▪ Отчет на тестването: регистриране на резултатите от теста и откритите

бъгове

Процесът на софтуерно тестване

9

Грешки, дефекти, бъгове и неизправности

Софтуерни дефекти (Bugs)

11

▪ Хората допускат грешки (пропуски)

▪ Грешките водят до дефекти

▪ Дефектите = бъгове в програмния код или

грешки в изискванията / дизайна / друго

▪ Ако бъгът бъде активиран, това може да

доведе до неизправност

▪ Софтуерът не успява да изпълни това, което се

очаква / изпълнява грешни неща

▪ QA / софтуерното тестване цели да намери бъговете

▪ Автоматизираното тестване и Непрекъсната

интеграция/внедряване (CI / CD) намаляват бъговете

Софтуерни дефекти

Човешки грешки

(пропуски)

Дефекти в софтуера

(бъгове)

Неизправности при

изпълнение/непра-

вилно поведение

12

▪ Програмна грешка

▪ Функция "събиране", която работи коректно, с

изключение на 5 + 3 = 7 (скрита/латентна грешка)

▪ Активирана грешка -> ефективен дефект/бъг

Извикване на функция "събиране", чрез 5 + 3

▪ Резултат 7 в някаква променлива (вместо 8)

▪ Неизправност – отклонение в поведението на

системата

▪ Насрочване на среща в 7:00 сутринта, вместо в

8:00 сутринта

Грешка / Дефект / Неизправност

Грешка

Дефект /

Бъг

Неизправност

13

▪ Причини за грешки на програмиста/тестващия могат да

бъдат:

▪ Липса на време

▪ Недостатъчно добро обучение

▪ Сложен код

▪ Сложна инфраструктура

▪ Променящите се технологии

Какво може да доведе до дефекти / бъгове? (1)

14

▪ Примери за липса на време

▪ За написването на дадена функция са

необходими 10 минути

▪ Разрешеното време е 2 минути

▪ По-голяма вероятност да се допусне грешка

▪ Примери за недостатъчно добро обучение

▪ В софтуер трябва да се имплементира физична формула

▪ Програмистът не разбира формулата

▪ Лошо/неправилно изпълнение

▪ Кодът прави нещо друго

Примери

15

▪ Организационни фактори

▪ Неефективна комуникация

▪ Неясно дефинирани изисквания

▪ Условия на околната среда

▪ Електронни полета, магнетизъм, радиация, замърсяване и др.

▪ Биха могли да повлияят на състоянието на хардуера

▪ Неправилна софтуерна среда (напр. грешен IP адрес)

▪ Пример за неясно дефинирани изисквания:

▪ "Софтуерът трябва да бъде лесен за използване."

Какво може да доведе до дефекти / бъгове? (2)

16

▪ Други причини за бъгове:

▪ Неправилна конфигурация или

неизправност в производствената или

тестовата среда

▪ Некоректни тест данни

▪ Правилен тест, който дава отрицателен резултат

▪ Грешен тест, който дава положителен резултат

▪ Некачествени тестове

▪ Невалидни очаквани резултати

Какво може да доведе до дефекти / бъгове? (3)

17

▪ Пример за некоректни тест данни

▪ Тестващия регистрира потребителско

име "john123"

▪ Не връща базата данни в първоначалния ѝ вид

▪ Втори тестващ се опитва да изпълни същия тест

▪ Регистрацията е неуспешна заради дублиране, т.е. коректен тест

дава отрицателен резултат

▪ Неправилна конфигурация на производствената среда

▪ Деактивирана функция "изпращане на имейл" на хостинг сървъра

▪ Потребител се регистрира, но не получава потвърждаващ имейл

Примери

Тежки загуби, причинени от софтуерни дефекти

Фатални софтуерни бъгове

19

Фатални софтуерни бъгове (1)

Катастрофата на Марс Клаймат Орбитър (1998)

▪ Предназначен за изучаване на марсианския

климат, атмосфера и повърхност

▪ Трябва да поддържа орбита на разстояние

140 – 150 км от Марс

▪ Достига 57 км и бива разрушен от налягането

▪ Загуба: $125 милиона

▪ Причини: използване на грешни мерни

единици (имперски спрямо посочените от

НАСА)

20

Фатални софтуерни бъгове (2)

Медицинският ускорител Therac-25

(1985-1987)

▪ Неизправност в машина за радиационно

облъчване на раково болни

▪ Смъртоносни дози радиация са

приложени на трима пациенти, други

трима са тежко ранени

▪ Причина: пропуск в алгоритъма за

проверка на грешки

21

Фатални софтуерни бъгове (3)

Космическата сонда Маринър 1 (1962)

▪ Маринър 1 е първият космически кораб

от американската програма Маринър

▪ Проектиран да изследва Венера

▪ Грешно функциониране на насочващите

команди

▪ Унищожен 5 минути след изстрелването

▪ Загуба: $18.2 милиона

▪ Причина: приликата на горната черта

с тире ('‾' вместо '-')

22

Фатални софтуерни бъгове (4)

Еърбъс A300-600R на

Китайските Авиолинии (1994)

▪ Разбива се и се запалва по време на

кацане на летището в Нагоя

▪ Загуба: $40 милиона + 264 човешки

живота

▪ Причина: пилотска грешка и липса на

препоръчаната актуализация на

софтуера (Китайските авиолинии

преценяват, че "не е спешно")

23

Фатални софтуерни бъгове (5)

Ариана 5, полет 501 (1996)

▪ Най-новата безпилотна ракета за

изстрелване на сателити в Европа

▪ Унищожена 36,7 секунди след

изстрелването

▪ Загуба: $8 милиона. Носи сателит на

стойност $500 милиона долара

▪ Причина: софтуерът се опитва да побере

64-битово число в 16-битово

пространство

▪ Видео: https://youtu.be/qnHn8W1Em6E

Ръчни кликвания или автоматични скриптове

Ръчно или автоматизирано тестване

25

▪ Ръчно тестване

▪ Тип софтуерно тестване, при което тестовете се изпълняват ръчно, без

използване на автоматизирани инструменти

▪ Човек изпълнява тестовете стъпка по стъпка, без тест скриптове

▪ Тестовете се изпълняват индивидуално, един по един

▪ Aвтоматизирано тестване

▪ Тип софтуерно тестване, при което тестовете се изпълняват

автоматично чрез "структура" за автоматизация на тестове (test

automation frameworks)

▪ Тестващите използват инструменти и скриптове, за да автоматизират

повтарящи се дейности

▪ Включва писане на код и поддръжка на тестове

Ръчно и автоматизирано тестване

26

Ръчно или автоматизирано тестване

Аспект на тестването

Ръчно

Автоматизирано

Изпълнение на теста

Изпълнява се ръчно от

QA специалисти

Изпълнява се автоматично с помощта

на инструменти и скриптове за

автоматизация

Ефективност на теста

Много време, по-ниска

ефективност

Повече тестове за

по-малко време и

по-висока ефективност

Видове дейности

Изцяло ръчни дейности

Повечето дейности могат да бъдат

автоматизирани,

включително реални потребителски

симулации

Покритие на

тестовете

Трудно е да се

гарантира задоволително покритие

на тестовете

Лесно се осигурява

по-голямо покритие на тестовете

27

▪ http://softuni-qa-amazonaws.com/manual-qa-demo

Пример за ръчно тестване

Ръчен тест

28

Пример за автоматизирано тестване

driver = webdriver.Chrome(options=chrome\_options)

driver.get("https://manual-qa-demo.softuniqa.repl.co")

sleep(3)

button = driver.find\_element(By.ID, "button")

button.click()

msg = driver.find\_element(By.ID, "msg")

assert msg.text == 'Button clicked'

QA-Automation-Demo.py

▪ https://replit.com/@SoftUniQA/AutomationDemo

Философия на софтуерното тестване

Седемте принципа в тестването

30

▪ Тестването може да покаже

наличието на дефекти

▪ Не може да докаже липсата

на дефекти

▪ Подходящото тестване

намалява вероятността за

наличие дефекти

"Софтуерното тестване може да

покаже наличието на дефекти, но

не и отсъствието им"

Седем принципа в тестването (1)

31

▪ Комбинациите от входни

данни и тестови условия са

безкрайни

▪ Да се тества всичко е

невъзможно

▪ След направена оценка на

риска, приоритет имат

тестовете с най-висок за

ситемата риск

Седем принципа в тестването (2)

"Изчерпателното

тестване е невъзможно"

32

▪ Дейностите по тестването

трябва да започнат

възможно най-рано

▪ Те трябва да са фокусирани

върху предварително

определени цели

▪ Колкото по-късно се открие

един бъг – толкова

по-висока е цената!

"Ранното тестване

спестява време и пари"

Седем принципа в тестването (3)

33

▪ Тестването трябва да бъде

правилно насочено

▪ 80% от проблемите са

породени от 20% от

модулите в системата

(Принцип на Парето)

▪ Фокусът пада върху 20%, от

които идват повечето

проблеми

"Струпване на дефекти"

Седем принципа в тестването (4)

34

▪ Повтарянето на едни и същи

тестове води до намаляване

на ефективността им

▪ Неоткритите по-рано бъгове

си остават неоткрити

▪ Необходима е разработка на

нови и/или модифицирани

тестове

"Парадокс на

пестицидите"

Седем принципа в тестването (5)

35

▪ Тестовете трябва да са

съобразени и подбрани в

зависимост от

приложението, което ще се

тества

▪ Софтуер изискващ високо

ниво за безопасност, се

тества по различен начин от

този за електронна търговия

"Тестването зависи от

контекста"

Седем принципа в тестването (6)

36

▪ Схващането, че софтуер с малък

брой бъгове е успешен продукт е

погрешно

▪ Самото намиране и отстраняване

на бъгове е безсмислено, ако:

▪ Изградената система е

неизползваема

▪ Не отговаря на нуждите и

очакванията на потребителите

Заблудата "Липса на

дефекти"

Седем принципа в тестването (7)

Истории за тестване

Тест сценарии (Test Scenarios)

38

▪ Какво е "тест сценарий"?

▪ Всяка функционалност / свойство / потребителска история, която може

да бъде тествана

▪ Нарича се още "story under test " или "feature under test"

▪ Пример: тествай формата за вход

▪ Защо ни е необходим?

▪ Сложните системи могат да бъдат разделени на няколко тест сценария

▪ Задава посоката, в която ще се тества

▪ За изучаване на функционалността на програмата от край до край (end-

to-end functioning)

Тест сценарий (Test Scenarios)

39

▪ Тест сценарий 1:

▪ Регистрирай се в платформа

Пример: Тест Сценарии

▪ Тест сценарий 2:

▪ Запиши се за курс

40

▪ Един тест сценарий включва няколко

тест случая (test cases)

▪ Пример:

▪ Потребителска история:

Потребителите трябва да могат да "влязат" с

потребителските си данни

▪ Tест сценарий : Вход с потребител + парола

▪ Тест кейс:

▪ Вход с валиден потребител + парола → успешно влизане

▪ Вход с невалиден потребител + парола → грешка

Тест Сценарии и Тест Случаи

Test Case #1

Test Case #2

Test Case #3

User Story

Test Scenario

41

▪ Запознайте се с документите с изискванията (requirements)

▪ Помислете за възможните потребителски действия за всяко едно

изискване

▪ Всяка функционалност трябва да има собствен тест сценарий

▪ Създайте тест кейсове, които покриват очакваното

потребителско поведение

▪ Създайте тест кейсове, които покриват неочакваното

потребителско поведение

▪ Уверете се, че сте покрили всички изисквания

▪ Предайте сценариите за преглед

Как се изготвя тест сценарий?

Тестове на единична, конкретна функция

Тест Случаи (Test Cases)

43

▪ Какво представляват тест кейсовете?

▪ Поредица от действия, изпълнявани с цел да проверят

конкретна пътека на изпълнение

▪ Могат да включват специфични входни и изходни условия

▪ Защо са ни необходими?

▪ За сравнение на очакваните с действителните резултати

▪ За проучване на начина на функциониране на даден

софтуерен компонент с определен вход и при определени

входни условия

Тест случаи (Test cases)

▪ Поредица от стъпки за проверка на правилното

поведение

▪ Поне два тест случая за тестване на определен

сценарий

▪ Положителен тест

▪ Отрицателен тест

▪ Тест случаите се състоят от:

▪ Заглавие (+ незадължително описание)

▪ Стъпки за изпълнение

▪ Очакван резултат

Тест кейс

44

45

Тест кейс – структура и пример

46

Тест сценарии и тест кейсове – пример

User Registration

•••••••••••

Password:

Register

Maria Steward

Full Name:

maria

Username:

▪ Примерен тест сценарий:

▪ Регистрирайте се в дадена платформа

▪ Тест кейс, част от този сценарий :

▪ Несъществуващо, валидно

потребителско име → успех

▪ Дублирано потребителско име → грешка

▪ Празно потребителско име или парола → грешка

▪ Твърде дълго потребителско име или парола → грешка

▪ Невалидни знаци в потребителското име или парола → грешка

47

▪ …

▪ …

▪ …

Какво научихме днес?

▪ Дефиниция за SQA

▪ Дефиниция за Софтуерно тестване

▪ Софтуерни дефекти / Bugs

▪ Ръчно и Автоматизирано тестване

▪ Седемте принципа в тестването

▪ Тест сценарий: тестване на определена

функционалност (форма за регистрация)

▪ Тест кейс: различните начини, по които

се тества функционалността

Въпроси?

Диамантени партньори на СофтУни

▪ Официален уеб сайт:

▪ Официален дискусионен форум:

▪ Официална фейсбук група:

Уеб сайт на курса, Форум и FB група

https://softuni.bg/trainings/4357/qa-basics-november-2023

https://softuni.bg/forum

https://www.facebook.com/groups/qabasicsnovember2023

50

51

Лиценз

▪ Този курс (презентации, примери, демонстрационен код,

упражнения, домашни, видео и други активи) представлява

защитено авторско съдържание

▪ Нерегламентирано копиране, разпространение или

използване е незаконно

▪ © СофтУни – https://softuni.org

▪ © Софтуерен университет – https://softuni.bg

52

Обучения в Софтуерен университет (СофтУни)

▪ Софтуерен университет – качествено образование,

професия и работа за софтуерни инженери

▪ softuni.bg

▪ Фондация "Софтуерен университет"

▪ softuni.foundation

▪ Софтуерен университет @ Facebook

▪ facebook.com/SoftwareUniversity

▪ Дискусионни форуми на СофтУни

▪ forum.softuni.bg

https://about.softuni.bg

Софтуерен университет

Преподавателски екип

СофтУни

Бъгове, доклади, жизнен цикъл и системи за

проследяване на дефекти

Дефекти и проследяване на дефекти

(Bugs and Bug Tracking)

2

1. Софтуерни бъгове (Bugs)

2. Доклад за дефекти (Bug Report)

3. Жизнен цикъл (Bug Lifecycle)

4. Съвети за докладване на бъгове

5. Системи за проследяване на бъгове

(Bug Tracking Systems)

6. Показатели за управление на инциденти

Съдържание

sli.do

#QA-Basics

Въпроси

3

Проблеми в кода → Причина за повреда/неправилно функциониране

Софтуерни дефекти / бъгове

5

▪ Хората допускат грешки (пропуски)

▪ Грешките водят до дефекти

▪ Дефектите = бъгове в програмния код или

грешки в изискванията / дизайна / друго

▪ Ако бъгът бъде активиран, това може да

доведе до неизправност / неправилно функциониране:

▪ Не успява да изпълни това, което се очаква / изпълнява грешни неща

Софтуерни дефекти - Обобщение

Софтуерен дефект/бъг е грешка в компютърна програма

или система, който я кара да се държи неочаквано или да не

работи по предназначение.

6

▪ Както гласят "Седемте принципа в тестването":

▪ "Ранното тестване спестява

време и пари"

▪ Колкото по-бързо бъде

отстранен един бъг, толкова

по-малки ще са причинените

от него щети

▪ "Струпване на дефекти"

▪ Непоправените бъгове

прикриват други бъгове

Колко е важно отстраняването на бъгове?

Описание на дефекта в подробности и стъпки

как да бъде пресъздаден

Доклад за дефекти (Bug Report)

8

▪ Какво представлява докладът за дефекти (bug report)?

▪ Писмен документ, описващ определен бъг, открит по време

на конкретна фаза от процеса на тестване

▪ Защо ни е необходим?

▪ Предоставя подробна информация за проблема

▪ Помага при поддържането на архив за бъдещи справки

▪ Категоризира бъговете, за да помогне при анализа на

първопричината

▪ Избягва се докладването на дублираращи се проблеми

Доклад за дефект (bug report)?

9

▪

Bug ID: SB-21

▪

Приоритет (priority):

▪

Сериозност (severity):

▪

Възложено на: Питър Уайт

▪

Докладвано от: Мария Нелсън

▪

Докладвано на: 22.09.2023

▪

Статус: Нов

▪

Среда: https://test.website.com/chatter

▪

Резюме: Създателите на групов чат не могат да

го преименуват

▪

Описание: Всеки участник в групов чат трябва

да

може

да

го

преименува.

Проблемът

съществува само за създателя на груповия чат.

Всички

останали

участници

могат

да

го

преименуват.

Пример: Bug Report

▪

Очаквано

поведение:

Всички

участници

трябва да могат да преименуват груповия чат.

▪

Реално поведение: бутонът [Преименуване на

чат] е деактивиран за създателя на груповия

чат.

▪ Стъпки за възпроизвеждане:

1) Отворете https://test.website.com

2) За вход: потр. име – test3 / парола –

testtest

3) Отворете

диалогов

прозорец

за

чат,

озаглавен "Test Chat"

4) Отворете [Settings] => [Add User]

5) Добавете

произволен

потребител

към

групов чат => кликнете [Done]

6) Щракнете отново върху настройките, за да

преименувате груповия чат

Следващата версия

Ниско ниво

10

▪ Резюме

▪ Кратко изложение/заглавие на проблема

▪ Описание

▪ Опишете проблема, включително:

▪ Аномалии

▪ Входни данни + очаквани + реални резултати

▪ Среда

▪ URLs

▪ Снимки

Какво включва един Bug Report? (1)

11

▪ Стъпки за възпроизвеждане

▪ Очаквано поведение

▪ Какво се очаква да се случи?

▪ Реално поведение

▪ Какво действително се случва?

Какво включва един Bug Report? (2)

1. ………

2. ………

3. ………

Последователност

12

▪ Препратки към външни източници

▪ Документи със спецификации

▪ Други работни документи

▪ Прикачени файлове

▪ Видеоклипове и снимки

▪ Всякаква допълнителна информация за

конфигурацията

Какво включва един Bug Report? (3)

13

▪ Сериозност (Severity) и приоритет (Priority) на дефекта (bug)

▪ Определят се от QA специалисти или на

специални срещи за определяне на приоритетите

(bug review, or bug triage)

▪ Определят се също рисковете, разходите, възможностите и ползите

от поправянето или непоправянето на дефекта

Какво включва един Bug Report? (4)

Приоритет

Priority

незабавно (immediate), при следваща версия

(next release), ако има възможност (on occasion),

отворен (open)

Сериозност

Severity

блокираща (blocking), критична (critical),

висока (high), средна (medium), ниска (low)

▪ Какво е "severity"?

▪ Степента на въздействие, която даден бъг

оказва върху работата на продукта

▪ Отнася се до функционалност или стандарти

▪ Индикатор за значимостта на дефекта

▪ Определя се от функционалността

▪ Сериозността на дефекта е обективен показател и е

малко вероятно неговият статус да се промени (напр. oт

high да стане low)

▪ Базира се на техническата страна на продукта

Доколко е сериозен дефектът (Bug Severity)

14

▪ Пречи на потребителя да използва

фунциoналността по предназначение

▪ Проблемът няма как да бъде преодолян по

заобиколен начин

Severity или "Сериозност" на дефекта - нива (1)

Blocking

Блокираща

▪ Повреда на данните

▪ Лесно и многократно хвърля грешка /

"изключение"

▪ Проблемът няма как да бъде преодолян по

заобиколен начин

▪ Функционалността не работи според

очакванията

Critical

Критична

15

▪ Хвърля изключение, когато не следва "щастливия" път

▪ Объркващ потребителски интерфейс

▪ Има заобиколно решение

Severity или "Сериозност" на дефекта - нива (2)

High

Висока

▪ Функционалността работи извън "щастливия" път с

малки проблеми

▪ Малки проблеми с потребителския интерфейс

▪ Едно или повече заобиколни решения

Medium

Средна

16

▪ Козметични проблеми

▪ Много заобиколни решения

▪ Малка вероятност да бъде забелязано от потребителите

Low

Ниска

▪ Какво е "приоритет"?

▪ Показва колко бързо трябва да бъде коригиран даден бъг

▪ Определя реда, в който бъговете трябва да бъдат отстранени

▪ Обвързан е с планиране / график

▪ Определя се след обсъждане с мениджъра / клиента

▪ В голяма степен зависи от финансови показатели

▪ Въз основа на изискванията на клиента

▪ Какъв да бъде преоритетът на даден бъг е субективен

показател и може да бъде променян във времето, според

текущата ситуация по проекта

Приоритет на дефекта / Bug Priority

17

18

▪ Примерна класификация:

▪ 1 – Незабавно / Висок – трябва да се разреши

възможно най-скоро

▪ 2 – Следваща версия / Среден – трябва да се

разреши в нормалния ход на дейностите по

разработката. Може да изчака, до пускането на

следващата версия

▪ 3 – Когато има възможност/ Нисък – може да бъде

отложено, докато не бъде коригирана по-сериозна

грешка

▪ 4 – Отворен – засега не е планирано

Приоритет (Priority) класификация

19

▪ Висок приоритет и високо ниво на сериозност

▪ Ключова функционалност не работи и няма

заобиколно решение

▪ Бутонът за вход на началната страница не работи и

клиентите не могат да влязат в приложението

▪ Висок приоритет и ниско ниво на сериозност

▪ Минимална грешка, с голямо значение за бизнеса на

клиента

▪ Правописна грешка на заглавната страница или

сгрешено фирмено лого

Примери за priority и severity (1)

20

▪ Нисък приоритет и високо ниво на сериозност

▪ Функционална грешка в приложението, няма

заобиколно решение, но рядко се използва от крайния

потребител

▪ Грешка в изчисленията на годишния отчет, който

крайният потребител не използва ежедневно

▪ Нисък приоритет и ниско ниво на сериозност

▪ Козметични или правописни проблеми, в рамките на

конкретен параграф

▪ Несъответствие на шрифтове в раздел за коментари

Примери за priority и severity (2)

Жизнен цикъл на дефекта

Bug Lifecycle

22

▪ Докладите за дефекти

(bug reports) се

управляват чрез жизнен

цикъл (lifecycle)

▪ Целта е, процесът по

коригиране на бъгове да

се систематизира

Жизнен цикъл на дефекта (Bug lifecycle)

Нов

Отворен

Възложен

Поправен

Проверен

Затворен

Повторно -

Отворен

/ Дублиран

/ Отхвърлен

/ Отложен

/ Не е бъг

▪ За бъга се съобщава за първи път

▪ Бъгът все още не е потвърден

▪ Водещият QA (test lead) потвърждава, че

бъгът е реален

▪ Променя статуса на "Отворен"

▪ Бъгът се възлага на съответния

програмист или екип от програмисти

Фази в жизнения цикъл (1)

Нов

New

Отворен

Open

Възложен

Assigned

23

▪ Бъгът е отстранен и е предоставен на

екипа, отговорен за тестването

▪ Ако програмистът смята, че бъгът не е

истински, той го отхвърля

▪ Бъгът се повтаря два пъти или два бъга се

отнасят до една и съща концепция

Фази в жизнения цикъл (2)

Поправен

Fixed

Отхвърлен

Rejected

Дублиран

Duplicate

24

▪ Ако бъгът не присъства в софтуера,

тестващият потвърждава, че грешката е

отстранена

▪ Дори и след отстраняването на бъга от

програмиста, той продължава да съществува

▪ Бъгът преминава отново през жизнения

цикъл

▪ Бъгът е отстранен, тестван и приключен

Фази в жизнения цикъл (3)

Проверен

Verified

Повторно

Отворен

Reopened

Затворен

Closed

25

Добри практики

Съвети за докладване на бъгове

27

▪ Ясно и кратко заглавие / резюме

▪ Повторяемост

Опорни точки за добър бъг репорт (1)

НЕ : Браузърът се срина

ДА : Грешка 404: Страницата не е

намерена при кликване върху бутона

[Еxport].

НЕ : Непопълнено

ДА : "Всеки път", "От време на

време", "Не може да се възпроизведе"

28

▪ Описание

▪ Допълнителна информация

Опорни точки за добър бъг репорт (2)

НЕ : Срина се

ДА : След кликване върху бутона [Export], "500

вътрешна грешка на сървъра" за потребители,

които не са администратори

НЕ : Непопълнено

ДА : Грешката възниква при записи на събития,

които имат такса, но приложението работи

коректно при записи на събития, които са

благотворителни

29

▪ Реални резултати

▪ Очаквани резултати

Опорни точки за добър бъг репорт (3)

НЕ : Не работи

ДА : Грешка 403: Forbidden

НЕ : Очаквах да работи

ДА: След натискане на бутона [Execute],

трябва да се отвори нов изскачащ прозорец

(popup window)

30

▪ Платформа

▪ Приложени снимки или прикачени файлове,

ако грешката го позволява

Опорни точки за добър бъг репорт (4)

НЕ : Windows

ДА : Windows 10, Google Chrome 103.0.5060.134

Как лесно да проследим дефектите

Системи за проследяване на дефекти

(Bug Tracking Systems)

32

▪

Какво са системите за проследяване на дефекти?

▪

Примери за системи за проследяване

на дефекти (bug tracking systems):

▪

Jira

▪

GitHub Issues

▪

BugZilla

▪

ClickUp

Системи за проследяване на дефекти

Системата

за

проследяване

на

дефекти

е

софтуерно

приложение,

в

което

се

въвеждат

и

пазят

докладваните

бъгове

по

софтуерни

проекти.

Може

да

се

разглежда

като

вид

система

за

проследяване

на

проблеми.

Регистриране на бъг с Jira

Демо

34

▪ Отидете на официалната страница на Jira

▪ Натиснете върху бутона [Get it free] в горната лента

▪ Натиснете бутона [Next]

▪ Ще бъдете пренасочени до "Sign up" страницата

Как се регистрира (логва) бъг в Jira (1)

35

▪ Изберете име за вашия сайт

▪ След това изберете темплейта за проследяване на дефекти (bug

tracking template)

▪ Вече имате страница с проект, в който можете да логвате бъгове

Как се регистрира (логва) бъг в Jira (2)

36

Как се регистрира (логва) дефект в Jira (3)

Списък с бъговете на

проекта – включително

кой отговаря за всеки

един, кой го е

докладвал и т.н.

Табло за

управление

на проекта

Бутон за създзване

(докладване) на нов

проблем

37

▪ Оттук можете да

попълните своя

bug report

Как се регистрира (логва) бъг в Jira (4)

Попълнете задължителните полета

(резюме, описание, стъпки за

възпроизвеждане и т.н.)

Проект, по който докладвате

За какъв вид проблеми съобщавате

(бъг, функция, подобрение и т.н.)

38

▪ След това ще видите страница с

новосъздадения дефект (бъг)

Как се регистрира (логва) бъг в Jira (5)

39

▪ Намерете и логнете в Jira изчезващия "Submit" бутон

▪ http://softuni-qa-amazonaws.com/disappearing-button/

Демо: Bug report - форма за регистрация

Как да определим дали целите са постигнати

Показатели за управление на инциденти

Показатели за управление на инциденти

41

▪ Общ брой на бъговете

▪ Брой на отворените (активни) бъгове/задачи

▪ Брой открити грешки преди пускане

▪ Съотношение на поправени бъгове

(поправени грешки)

▪ Съотношение на отхвърлени бъгове

(отхвърлени грешки)

▪ Съотношение на пропуснати бъгове

(неоткрити грешки)

▪ Брой бъгове от различни категории

42

Примерно табло за управление

▪ …

▪ …

▪ …

Какво научихме днес?

▪

Грешки в софтуерната програма могат да я накарат неочаквано да

се затвори или да се държи по нежелан начин

▪

Какво включва докладът за дефект?

▪

Докладите за дефекти се управляват чрез жизнен цикъл, който се

използва, за да се систематизира процеса на коригиране на

грешки

▪

Как да изготвяме добри доклади за дефекти

▪

Системи за проследяване на дефекти и как да ги използваме

▪

Показатели за управление на инциденти

43

Въпроси?

Диамантени партньори на СофтУни

▪ Официален уеб сайт:

▪ Официален дискусионен форум:

▪ Официална фейсбук група:

Уеб сайт на курса, Форум и FB група

https://softuni.bg/trainings/4357/qa-basics-november-2023

https://softuni.bg/forum

https://www.facebook.com/groups/qabasicsnovember2023

46

47

Лиценз

▪ Този курс (презентации, примери, демонстрационен код,

упражнения, домашни, видео и други активи) представлява

защитено авторско съдържание

▪ Нерегламентирано копиране, разпространение или

използване е незаконно

▪ © СофтУни – https://softuni.org

▪ © Софтуерен университет – https://softuni.bg

48

Обучения в Софтуерен университет (СофтУни)

▪ Софтуерен университет – качествено образование,

професия и работа за софтуерни инженери

▪ softuni.bg

▪ Фондация "Софтуерен университет"

▪ softuni.foundation

▪ Софтуерен университет @ Facebook

▪ facebook.com/SoftwareUniversity

▪ Дискусионни форуми на СофтУни

▪ forum.softuni.bg

http://softuni.bg

Юнит тестване / Интеграционно тестване /

Системно тестване / Приемно тестване

Функционално и нефункционално тестване

Типове и нива на тестване

Софтуерен университет

Преподавателски екип

СофтУни

2

1. Нива на тестване

▪ Юнит тестване (компонентно тестване)

▪ Интеграционно тестване

▪ Системно тестване

▪ Тестване за приемане от краен потребител

(приемно тестване)

2. Типове тестване

▪

Функционално тестване

▪

Нефункционално тестване

Съдържание

sli.do

#QA-Basics

Въпроси

3

Нива на тестване

Юнит тестване / Интеграционно тестване /

Системно тестване / Приемно тестване

▪ Групи от тестови дейности

▪ Всяко ниво е част от тестовия процес

▪ Съответства на определена фаза от разработката на

софтуера

▪ Тест нива:

▪ Тестване за одобрение / Приемно тестване

(Acceptance testing)

▪ Системно тестване (System testing)

▪ Интеграционно тестване (Integration testing)

▪ Юнит тестване / Компонентно тестване (Unit testing)

Нива на тестване

5

6

Йерархия на тестовите нива

Приемно тестване

(Acceptance testing)

Интеграционно тестване

(Integration testing)

Юнит тестване

(Unit testing)

Системно тестване

(System testing)

Тестване на отделен компонент

Юнит тестване

8

▪ Проверяване на всяко

яйце от кутията, преди

закупуването ѝ

Юнит тестване в реалния живот

▪ Проверяване дали

фаянсовата плочка не е

счупена, преди

нанасянето на лепило

9

▪ Какво е юнит тестване?

▪ Нарича се още компонентно тестване

▪ Първото или най-базово ниво на тестване

▪ Тества отделни компоненти на софтуера

▪ Компонент може да бъде отделна функция, метод, модул или обект

▪ Обикновено се извършва от самите програмисти във фазата на писане на

код

▪ Изпълнява се изолирано

▪ Защо ни е необходимо?

▪ Проверява дали отделните компоненти работят коректно

▪ Позволява дефектите да бъдат отстранени рано, още във фазата на

разработка

Юнит тестване / Unit Testing

10

▪ Проверка на възрастта

Програмата задава въпрос: "На колко години си?"

Прочита вевъдените от потребителя "години"

ако годините са >= 18

връща вярно

в противен случай

връща невярно

Юнит тестване: Пример

Положителен тест(20) → очаква се вярно

Отрицателен тест (16) → очаква се невярно

Граничен тест (18) → очаква се вярно

Тестване на взаимодействието между компоненти

Интеграционно тестване

▪ Столове подходящи

за маса

Интеграционно тестване в реалния свят

▪ Врата и каса за врата

13

▪ Какво е интеграционно тестване?

▪ Второ ниво от процеса по тестване на софтуер

▪ Отделните компоненти или единици на софтуера се тестват в

група

Интеграционно тестване / Integration Testing (1)

▪ Извършва се от програмисти, QA

специалисти или специални

интеграционни екипи

▪ Предполага се, че компонентите

вече са тествани поотделно

14

▪ Защо ни е необходимо интеграционното тестване?

▪ След свързването на отделните компоненти един с друг,

може да възникне нова грешка

▪ Тестването трябва да потвърди, че всички свързани

компоненти си взаимодействат правилно

▪ Основната цел е да се открият грешките в:

▪ Интерфейси

▪ Взаимодействието между интегрирани компоненти

▪ Взаимодействието между системи

Интеграционно тестване / Integration Testing

(2)

15

▪ Вътрешно интеграционно тестване

▪ Разкрива дефекти в интерфейсите и взаимодействието между

интегрираните компоненти

▪ "Integration test in the small"

▪ Външно интеграционно тестване

▪ Тестване на съчетанието на системи и пакети

▪ Тестване на интерфейси към външни организации

▪ "Integration test in the large"

Поднива на интеграционното тестване

16

▪ GitHub има няколко модула (компоненти):

▪ Home Page → Login Page → User Dashboard

▪ Всеки от тях е тестван поотделно

▪ Искаме да проверим дали работят заедно

▪ Интеграционни тестове:

▪ Тестваме дали бутонът за вход води към формата за вход

▪ Тестваме дали след успешно влизане с потребителско име и

парола, се показва потребителският дашборд

▪ Тестваме дали след излизане от профила, потребителският

дашборд е недостъпен

Интеграционно тестване: Пример

Тестване на цялата система

Системно тестване

▪ В автомобилната индустрия, всеки произведен автомобил е щателно

тестван, в края на производствения процес:

▪ Двигател, джанти, волан, спирачки

▪ Врати, ключалки, седалки

▪ Електрическа система: светлини, чистачки,

климатик

▪ Мултимедийна система: радио, GPS, карти

▪ и т.н.

▪ QA-те вече са оценили всички тези функциoналности поотделно, а

също и взаимодействащи една с друга, но те трябва бъдат

тествани и като цялостна система

Системно тестване в реалния живот

19

▪ Какво е системно тестване ?

▪ Трето ниво от процеса по тестване на софтуер

▪ С фокус върху цялата система:

▪ Нейното поведение (какво прави системата)

▪ Колата правилно ли е сглобена, работи ли по предназначение?

▪ Нейните възможности (как се справя системата)

▪ Дали автомобилът е надежден, сигурен, в добро състояние, каква е

неговата производителност и ефективност

▪ Реализира се чрез тестване "от край до край" (E2E, end-to-end)

▪ Извършва се само от QA специалисти

Системно тестване / System Testing (1)

20

▪ Защо ни е необходимо системно тестване?

▪ Предишни тестове са били изпълнени спрямо технически

спецификации

▪ Системните тестове разглеждат системата от гледна точка на

крайния потребител

▪ Системните тестове проверяват дали всички компоненти на

дадена система функционират при реален сценарии

▪ Системното

тестване

може

да

бъде

функционално

и

нефункционално. Това гарантира работеща за крайния

потребител система

Системно тестване (2)

21

▪ Системното тестване

изисква специално

обособена "стейджинг"

среда

▪ Максимално точно копие

на сайта/приложението,

до което имат достъп

крайните потребители,

предназначено за

системно тестване

Системно тестване: Пример

•

Използва се от клиенти (на живо)

•

Пълни данни

•

Използва се от QA

специалисти

и/или клиенти за UAT

•

Ограничени данни

•

Използва се от QA специалисти

•

Без клиентски данни

•

Използва се от програмисти

•

Без клиентски данни

Тестване

(Testing)

Разработване

(Development)

Продукция

(Production)

Стейджинг

(Staging)

\*UAT - User Acceptance Testing /

Потребителско тестване за одобрение

Клиентът тества крайния продукт от бизнес

гледна точка

Тестване за приемане от крайния клиент

▪ Собственик на апартамент, проверява апартамента след

ремонт

▪ Всички стаи: подове, тавани, стени, врати, прозорци

▪ Уреди

▪ Мебели

▪ Вода и водопровод

▪ Електричество

▪ Газ

▪ и т.н.

Приемно тестване в реалния живот

24

▪ Какво е тестване за приемане от крайния клиент?

▪ Последното ниво, обикновено преди внедряване (deployment)

▪ Валидира цялостно функцонално бизнес решение

▪ Под внимание се вземат законовите и/или регулаторните изисквания

▪ Приемното тестване се изпълнява:

▪ От членове на бизнес екипа (алфа тестване)

▪ От крайни потребители (бета тестване)

▪ Следват се оперативни инструкции

▪ Гарантира се спазване на договорните и регулаторни насоки

Приемно тестване/ Acceptance Testing (1)

25

▪ Защо ни е необходимо тестване за приемане от крайния

потребител?

▪ Проверява работата на системата, обикновено преди

внедряване

▪ Основната цел е работещо бизнес решение

▪ Не се фокусира върху козметичните грешки

▪ Отговаря на въпроса, дали актуалното поведение на

системата съответства на очакванията на клиента

Приемно тестване (2)

26

▪ Най-новият Microsoft Windows се тества първо локално в Редмънд (алфа

тестване), след това от външни потребители (навсякъде по света)

Приемно тестване: Пример

Алфа тестери

▪ Група вътрешни потребители

▪ Запознати са с проекта

▪ Не участват пряко в развитието му

▪ Тестват дали приложението работи

правилно

▪ Дават обратна информация за това

как потребителското изживяване

може да се подобри

Бета тестери

▪ След алфа тестването, продуктът и

грешките се коригират

▪ Бета

тестването

се

извършва

от

избрана група крайни потребители

▪ Служи като "плавен старт"

▪ Обратна връзка от реални

потребители, които нямат

предварителни познания за

приложението и/или новите

функции

Типове тестване

Функционално и нефункционално тестване

28

▪ Група от тест дейности, които тестват специфични

характеристики на определена софтуерна система

▪ Типовете тестове се разделят на две основни групи:

▪ Функционално тестване

▪ Отговаря на въпроса "Какво?"

▪ Потвърждава правилното функциониране на софтуера

▪ Нефункционално тестване

▪ Отговаря на въпроса "Как?"

▪ Потвърждава ефективността и производителността на софтуера

Типове тестване

29

▪ Функционалното тестване на софтуер за онлайн банкиране

включва:

▪ Тестване дали средствата са точно преведени

▪ Дали лихвените изчисления са правилни

▪ Дали плащанията по сметки се извършват навреме

▪ Нефункционално тестване се фокусира върху сигурността на

системата:

▪ Да се гарантира, че достъпът е напълно безопасен

▪ Да се гарантира, че системата може да се справи с натоварването

▪ Особено в пикови периоди, като началото на месеца, когато масово се

изплащат заплати и други плащания към бюджета

Типове тестване: Пример

30

▪ Типовете тестване могат да се прилагат на много/всички

тестови нива

▪ Пример: тестване на сценарий "регистрирай потребител"

▪ Функционални тестове:

▪ Валидна потребителска информация, невалидна потребителска

информация, дублирана потребителска информация

▪ Нефункционални тестове:

▪ Производителност

(100

хиляди

потребители),

надеждност

(по 1 потребител в секунда за 24 часа), UX тест (навигацията

лесна ли е за потребителя)

Типове тестване и нива на тестване

Тестване на определени функции

Функционално тестване

32

▪ Сешоар: Функционално тестване

▪ Старт / Стоп

▪ Промяна на силата

▪ Горещ / студен въздух

▪ Йонни настройки

▪ Тестове на различни приставки

▪ Тестове на прибиращия се кабел

▪ и т.н.

Функционално тестване в реалния живот

33

▪ Тества функциите, които една система трябва да изпълнява

▪ Функциите са "какво" трябва да прави системата

▪ Потвърждава дали софтуерната система отговаря на

функционалните изисквания

▪ Функционалното тестване основно включва тестване тип черна

кутия (black box testing)

▪ Функционално покритие:

▪ Начин да се измери покритието на функционалните тестове

върху определена функционалност

Функционално тестване

▪ Тестване на основните функции на приложение

▪ Съобщения за грешка

▪ Проверка дали се извеждат

подходящи съобщения за грешка

▪ Базисно приложение

▪ Безпроблемна навигация през различните екрани

▪ Достъп

▪ Проверка на достъпа на потребителя до системата

34

Цели на функционалното тестване

35

▪ Функционални тестове за файлов архиватор (като 7-Zip или

WinRAR):

▪ Архивиране на папка в архив

▪ Разархивиране на папка

▪ Архивиране на един файл

▪ Архивиране на няколко файла

▪ Архивиране на празна папка

▪ и т.н.

Функционално тестване: Пример

Тества аспекти, които не са функционалност

Нефункционално тестване

37

▪ Сешоар: нефункционално тестване

▪ Тест за прегряване: ако сешоара работи 30 минути

▪ Колко време отнема да се промени температурата на

въздушния поток?

▪ Тест за шум

▪ Тест за падане

▪ Тест за тегло / размери

▪ Електрическият кабел достатъчно дълъг ли е?

▪ Лесно ли се използва с лява и дясна ръка?

Нефункционално тестване: Пример от реалния живот

38

▪ Нефункционалното тестване оценява:

▪ Надеждност

▪ Ефективност на работата

▪ Сигурност / безопасност

▪ Тества "Как" или какво е качеството, с което системата

изпълнява своите функции

Нефункционално тестване

39

▪ Нефункционалното тестване се фокусира главно

върху подобряване качеството на:

▪ Лесната употреба

▪ Ефективността

▪ Поддръжката

▪ Преносимостта на продукта

Цели на нефункционалното тестване

40

▪ Нефункционални тестове за файлов архиватор (като 7-Zip или

WinRAR):

▪ Тест за скорост: колко бързо се компресират файлове / папки

▪ Тест за скорост: колко бързо се разархивират файлове / папки

▪ Размер на архива: сравнение на различни нива на компресия

▪ Тест за сигурност: архивиране / разархивиране на файл, защитен

с парола

▪ Тест за препълване: компресиране / разархивиране на папка с

500 хил. файла

▪ Тест за претоварване: компресиране / разархивиране на 50

файла паралелно

Нефункционално тестване: Пример

▪ …

▪ …

▪ …

Какво научихме днес?

41

▪

Различни нива на тестване

▪

Юнит тестване: тестване на единичен компонент

▪

Интеграционно тестване: тестване на взаимодействието

между компонентите

▪

Системно тестване: QA-те тестват цялата система

▪

Приемно тестване: Клиентът тества крайния продукт

▪

Различни типове тестване

▪

Функционално тестване: тестване на софтуерната

функционалност

▪

Нефункционално тестване: производителност,

надеждност и др.

Въпроси?

Диамантени партньори на СофтУни

▪ Официален уеб сайт:

▪ Официален дискусионен форум:

▪ Официална фейсбук група:

Уеб сайт на курса, Форум и FB група

https://softuni.bg/trainings/4357/qa-basics-november-2023

https://softuni.bg/forum

https://www.facebook.com/groups/qabasicsnovember2023

44

45

Лиценз

▪ Този курс (презентации, примери, демонстрационен код,

упражнения, домашни, видео и други активи) представлява

защитено авторско съдържание

▪ Нерегламентирано копиране, разпространение или

използване е незаконно

▪ © СофтУни – https://softuni.org

▪ © Софтуерен университет – https://softuni.bg

46

Обучения в Софтуерен университет (СофтУни)

▪ Софтуерен университет – качествено образование,

професия и работа за софтуерни инженери

▪ softuni.bg

▪ Фондация "Софтуерен университет"

▪ softuni.foundation

▪ Софтуерен университет @ Facebook

▪ facebook.com/SoftwareUniversity

▪ Дискусионни форуми на СофтУни

▪ forum.softuni.bg