Для того, щоб бути конкурентоспроможним, потрібно вміти швидко впроваджувати продукти на ринок. Швидкість тестування зазвичай досягається двома способами:

Автоматизація

Відмова від формалізації

І ми розглянемо тему відмови від формалізації, або, в даному випадку, підхід Experience-Based Testing.

Спочатку, поняття тестування, заснованого на досвіді не було, спочатку існувало тільки Дослідницьке тестування (Exploratory testing), про який вперше згадує Сем Канер ще в далекому 1987 році в книзі "Testing computer software". Але поступово, з часом з'являються інші підходи до тестування, засновані на досвіді тестувальників, які в 2004 році міжнародна організація IEEE в документі "IEEE: Guide to the Software Engineering Body of Knowledge. IEEE Computer Society "об'єднує в загальний підхід Experience-Based Testing.

Що ж в себе включає Experience-Based Testing?

Стандартно (в тому числі і в ISTQB), це 3 техніки тестування:

Error Guessing (передбачення помилок)

Checklist-based testing (тестування на основі чек-листів)

Exploratory testing (тестування методом вільного пошуку або дослідницьке тестування)

Трохи розповім про них.

Передбачення помилок.

Досить складна техніка тестування, яка вимагає хороших знань тестованого ПО. Суть техніки полягає в можливості тестувальника виконувати тестування тільки шляхом передбачення дефектів, ризиків або проблем в системі. Тобто, ви вже 6-й рік тестируете одну і ту ж систему, пам'ятайте ще ті часи, коли система була старт-апом і ви не спали ночами, намагаючись її впровадити, до вас біжать всі аналітики, щоб зрозуміти, як буде працювати функціонал . В такому випадку ця техніка тестування для вас. Навіщо писати тест-кейси, тестувати весь функціонал, якщо ви інтуїтивно знаєте, де виникнуть проблеми з системою.

Трохи відступлюся ...

Року 4 назад у мене був такий знайомий, який тестував систему 5 років. Так ось на питання, чому він все ще сидить і тестує її, він відповідав, що йому подобається працювати там, де він все знає. Але його тест-кейси були просто чорт голову зломить "чудовими". Кожен його тест складався завжди з 3 кроків:

Що то відкрити

Що то зробити

Що то перевірити

І все! Як зробити, як відкрити, що вводити, де перевіряти, в загальному доводилося сідати і вивчати систему.

Але це лірика, повернемося до теми ...

Тестування на основі чек-листів.

Це досить часто поширена практика, особливо для невеликих організацій, в якій зазвичай не більше 2-3 тестувальників. Навіщо писати досить великі і об'ємні тестові сценарії, витрачати на них багато часу, коли можна просто описати основні перевірки.

Візуально чек-лист виглядає ось так:



Ну і останнє і найцікавіше - це дослідницьке тестування.

Є брати визначення з ISTQB, то дослідницьке тестування характеризується тим, що тестувальник одночасно вивчає продукт і його дефекти, планує роботи по тестуванню, проектує і виконує тести і звітує про результати. Тестувальник постійно коригує цілі під час виконання тестування і готує тільки високорівневу документацію. Саме таке визначення дав Джеймс Уіттакер в книзі "Exploratory Software Testing".

Отже, якщо ми більш глибоко зануримося в дослідне тестування, то ми зрозуміємо, що дослідне тестування неможливо спланувати. Тестувальник починає тестувати систему і в залежності від отриманих результатів планує виконання наступних тестів. При такому тестуванні, в принципі, відвалюється необхідність глибоко вивчати документацію, проектувати детальні тестові сценарії, готувати тестові дані та інші види активностей, пов'язаних з підготовкою до тестування за тестовими сценаріями.

Якщо розглядати тестувальника, як дійсно фахівця, що виконує складну інтелектуальну роботу, а не тупо шмяк йдучи бездумно по кроках тест-кейса, що останнім часом зустрічається все частіше і частіше, то це дійсно величезна праця і аналітична робота, що вимагає певного складу розуму і бажання дізнатися і пізнати все неосяжне. Зараз же фахівці з тестування перестають думати, чи то не відчувають відповідальності, то чи хочуть просто працювати, як мавпочки і ні в чому не розвивати себе. Загалом картина пригнічує ...

Особисто я вважаю, що працюючи по тест-кейсів і не докладаючи особливо зусиль на тестування тільки "отупляє" тестувальника. Але при цьому, завжди є можливість навіть при сценарному підході до тестування, відхилятися від тест-кейсів і проводити дослідницьку тестування.

Тому, дослідне тестування:

Наголошує на персональній свободу і відповідальність тестувальника

Показує індивідуальність тестувальника

Дозволяє постійно проектувати нові тест-кейси за результатами роботи

Дуже детально систематичний підхід до дослідницького тестування описує Уіттакер, використовуючи для цього підхід з турами по різних об'єктах, таким, як "Тур по поганому району", "Музейний тур" і т.д.

Звучить смішно. Але це дійсно так!

Методика дослідного тестування за допомогою турів полягає в наступному. Ваше додаток, яке ви тестируете - це незнайоме місто. Ви - турист. Ваша мета - виконати конкретне завдання, яку ви собі ставите, тим самим досліджуючи ваше місто (додатки).

Наприклад, якщо ви берете методику "Музейний тур", то музей - це місце, яке містить велику кількість різних старовинних речей, тобто це старий код, який зазвичай часто залишається недоторканим розробником. Проте, він може містити дефекти, які тестувальник повинен знайти. Тому, завдання тестувальника - визначити ті функції, які вже дуже довгий час не були схильні до змін і протестувати їх. Адже саме код розробника, який був написаний дуже давно, може бути підданий дефектам просто тому, що він вже давно забутий розробником і швидше за все на нього вже ніхто не пише навіть unit тести.

Дуже детально система турів Уіттакера розглядається в цій статті Система турів дослідного тестування.

Таким чином, для того, щоб проводити дослідне тестування досить мати дуже гарне абстрактне мислення.

Але виникає питання. А чи дійсно досить володіти великим досвідом, щоб проводити дослідне тестування?

Візьмемо досвідченого тестувальника, який вже багато часу проводить тестування різних систем. Чим керується в такому випадку тестувальник? Досвідом, інтуїцією або техніками тест-дизайну? Дуже часто я стикався з тестувальниками, які дуже добре знали систему. Вони розуміли, де і як може виникнути проблема, дефект, що потрібно тестувати, а на що можна просто не звертати уваги, тому що "там ніколи не виникає проблем". Так, це досвід, це вміння скоротити тестування за рахунок того, що тестувальник розуміє, до чого може привести ту чи іншу зміну коду, але тоді ми говоримо вже не про дослідницький тестування, а про Risk-Based Testing!

Що таке дослідження?

Дослідження - це систематичне розслідування з метою встановлення фактів, тобто це процес вивчення чого-небудь.

Тому, кажучи про досвід, ми дуже часто маємо на увазі саме знання чого-небудь. Знання - це вже вміння чогось, але дослідження - це вивчення. Говорячи про дослідницький тестуванні ми повинні відкинути всі наші знання тестування, досвід роботи з системою, технік тест-дизайну і дивитися на системи новим, абсолютно не «замилений" поглядом.

Дослідницьке тестування - це в першу чергу вміння мислити, бажання звертати увагу на різні аспекти системи, відкинувши наші знання і досвід роботи з даною системою. Адже саме досвід часто не дозволяє нам бачити здавалося б такі прості речі, на які звичайний тестувальник відразу вкаже при тестуванні.

Кожен же хоч раз в житті, працюючи з молодими тестувальниками, стикався з ситуацією, коли до вас приходить співробітник і питає, а чому тут, наприклад, немає можливості змінити будь-яке поле, хоча воно по інтуїції напрошується само собою (бо в життя можуть бути ситуації, коли потрібно це поле змінити). Насправді це дефект, але ми настільки звикли тестувати по документації і, на жаль, в нашій країні не цінують тестувальників, як висококласних фахівців, дійсно відповідають за якість роботи програми (саме з точки зору користувача), що у відповідь ми чуємо від Замовника тільки одну фразу:

- Раз так написали в ТЗ, значить так має бути.

Тому, кажучи про дослідницький тестування, не можна говорити, що це тестування без тест-кейсів.

Дослідницьке тестування - це цілий напрямок в тестуванні, яке вимагає від тестувальника не так знання і досвід роботи в галузі, з системою, як вміння неординарно мислити і знаходити дефекти не в соответствии с ТЗ, а відповідно до логіки роботи програми.