

Теоретичні відомості 2. Звіти про дефекти, історія тестування та перший конвеєр

Що таке дефект і з чого складається звіт

Дефект (а bug) — це будь-що в продукті, що загрожує його цінності на думку когось, чия думка має значення.

Визначення належить методології Rapid Software Testing (RST) Джеймса Баха та Майкла Болтона і впливає з визначення якості як цінності для когось, хто має значення. Тестувальник — представник цих людей: користувачів, клієнтів, керівників, стандартів, які до них застосовуються. Ваші власні відчуття можуть привести вас до дефекту, але звіт має дати причину, з якої хтось інший поділив би їх.

Дефекти бувають двох видів. **Звичайний дефект** — це невиконання продуктом свого призначення. **Запит на вдосконалення (enhancement request)** — це пропозиція змінити саме призначення, бо воно не служить людям, які мають значення. Про них повідомляють по-різному: із запитом на вдосконалення ви берете на себе роль проєктувальника, тому назва починається зі слова *Будь ласка (Please)*, щоб було зрозуміло, що ви пропонуєте зміну обсягу продукту, а не вказуєте на невдачу.

Під кожним звітом, яким би неформальним він не був, лежать чотири елементи.

1. **Проблема, яку ви бачите.** Що поганого відбувається або що хорошого не відбувається. Будьте конкретними і запитайте себе, чи це суть справи, чи варто повідомити про щось більше.
2. **Як ви на неї натрапили.** Безпосереднє спостереження продукту; конкретні кроки та дані.
3. **Чому це проблема.** Оракул або кілька оракулів. Слабкий оракул, наприклад відчуття, що чимось важко користуватися, дає підозру, варту розслідування; після розслідування ви можете мати сильніший.
4. **Чому це важливо.** Наскільки ймовірно, що це побачать, і якої шкоди це завдасть, коли побачать.

Звіт без третього елемента — це спостереження, а не дефект; звіт без четвертого не приведе до виправлення.

Розслідування дефекту

Ваш оракул порушено, і ви думаєте, що бачите дефект. Розслідування має чотири мети: **підтвердити** дефект (який найавторитетніший оракул він порушує; чи не хибна це тривога?); знайти його **найцікавіший прояв** (ізолювати й узагальнити; яка найгірша версія?); **відрізнити** його від уже повідомлених дефектів; **зібрати дані, потрібні розробникові**, — зазвичай кроки відтворення і хибний результат разом із вводом, який, здається, його спричинив.

Бах і Болтон дають правило щодо зусиль: **десятихвилинний ліміт**. Розслідуйте загадковий дефект приблизно десять хвилин, а потім зверніться до розробника, який часто вже знає, що відбувається. Якщо розробник теж спантеличений, продовжуйте; якщо ні, повідомте те, що маєте, і йдіть далі. Так знаходять більше дефектів, ніж “вилизуванням” звітів.

Прийоми розслідування — невеликий набір, здебільшого про керування змінними.

Прийом	Що робити
Спиральне дослідження	Збережіть стан, у якому ви знайшли проблему; поверніться на один крок і повторіть
Вгадати причину	Перелічіть можливі причини, спираючись на знання технології; кожна здогадка — фокус
Прибирати кроки	Вилучайте кроки по одному, щоб з'ясувати, які з них мають значення
Спростити, узагальнити, нормалізувати	Складний ввід → простіший; один конкретний випадок → загальніший; штучний ввід → типовий
Фокусування	Змінюйте один чинник за раз, щоб знайти, який саме викликає проблему
Розфокусування	Змінюйте багато чинників одночасно, щоб підтвердити, що ця група чинників <i>не</i> причетна
Відстежувані дані	Обирайте ввід, який легко впізнати у виводі
Чиста машина	Очистьте кеш і cookie; дефект на чистій платформі турбує більше, ніж той, якому потрібні особливі налаштування
Моніторинг, запис, читання журналів	Відеозапис екрана, журнали запитів, проксі; потім шукайте механізм у записаному

Переміжні проблеми

Переміжна (intermittent) проблема — це загадкова й тривожна поведінка системи, яку спостерігали принаймні один раз і яку ми ще не можемо викликати на вимогу.

Завдання тестувальника — перетворити переміжний дефект на звичайний, розв'язавши загадку навколо нього. Варто пам'ятати кілька принципів з RST: якщо це сталося один раз, воно, ймовірно, станеться знову; якщо дефект зник без виправлення, він, ймовірно, зник не назавжди; стану, до якого зазвичай іти довго, можна досягти миттєво непередбаченим шляхом; заплутана поведінка часто має просту причину; і легко закохатися в теорію, яка розумна, дотепна й хибна. Із цього випливають два наслідки. Не чекайте завершення дослідження, щоб почати звітувати; а якщо ви не

змогли зловити проблему, судитимуть вас за сумлінне, вдумливе, енергійне зусилля її знайти.

Повідомлення, яке ви отримуєте, — це свідчення очевидця. Свідчення очевидців пропускають дуже багато і змішують спостереження з тлумаченням. Слухайте, але зберігайте відстороненість від тверджень; з'ясуйте, що свідок міг і чого не міг бачити і що він припустив.

Поради щодо пошуку поділяються на чотири групи.

- **Спершу перевірте спостерігача.** Перевірте найбазовіші припущення: чи тестуєте ви те, що думаєте, на тій збірці, що думаєте, і чи спостерігаєте те, що думаєте?
- **Поліпшіть спостереження і керування.** Здобудьте журнали та скриптовані інтерфейси; точно керуйте вводом (послідовність, час, розмір, джерело, повторення, комбінація); керуйте станом, знайшовши способи повертатися до відомого стану; потім систематично покривайте простори вводу і станів.
- **Тягніть в обидва боки.** Спрощуйте: одна змінна за раз, поділ системи на частини. Ускладнюйте: багато змінних одразу, дайте стану «забруднитися», додайте навантаження і повторення, вносьте випадковість і ставте пастку — безперервний запис, щоб наступний випадок навчив вас більшого.
- **Зберігайте розсудливість.** Запитуйте, чи варта проблема витраченого часу; коли все інше не допомагає, залиште її й подивіться, чи повториться вона сама.

Найкорисніший інструмент — перелік **п'яти можливостей** зі словами-підказками для причин під кожною.

Можливість	Слова-підказки
1. Система не поводитися по-різному; переміжність — артефакт спостереження	погане спостереження (сліпота неувважності); погана пам'ять; хибне приписування (помилкова кореляція двох подій); ненадійний оракул (не всі вважають це проблемою); ненадійна комунікація («більше ніхто не скаржиться»)
2. Це була інша система	розробник змінив її і повернув назад; випадкова зміна; оновлення платформи; стороння система (розширення, другий екземпляр); розподілена система, частини якої не відповіли вчасно; людина всередині системи
3. Система була в іншому стані	заморожена умова; неправильна ініціалізація; поступове пошкодження даних; поступова дестабілізація (двоетапний збій); <i>рідкісна функція (occasional function)</i> , яка виконується тихо й автоматично, — кеш, обробник винятків, збирання сміття, періодичне

	обслуговування; режим або параметр, який не виглядає інакше
4. Система отримала інший ввід	випадковий ввід (синдром Розумного Ганса); <i>приховані межі</i> в даних, які перемикають логіку; час як ввід; <i>часова лотерея (timing lottery)</i> — вузьке вікно, у якому дві події накладаються
5. Ваша модель системи неправильна або неповна	складний алгоритм поводить ся несподівано, переміжно і правильно

Висновок розслідування називає можливість, докази на її користь і докази, які виключили решту. Проблема з можливості 4 з прихованою межею лише виглядає переміжною: вона детермінована, щойно межу знайдено. Проблема з можливості 3 стає відтворюваною, щойно стан можна задати на вимогу.

Формальний звіт

Звітування має три соціальні форми. **Згадка мимохідь (mention in passing, MIP)**: речення голосом або в повідомленні, часто як запитання — «це так і має працювати?» — для знахідок, щодо яких ви доволі впевнені, що це *не* дефект, але підозрюєте, що може бути; без шаблону, без трекера. **Формальні звіти**: письмові записи в системі відстеження; повільніші й дорожчі; погані формальні звіти підривають довіру до всього тестування. **Чорний прапор (black flagging)**: крок за межі звіту — скликати нараду або ескалювати, коли дефект вказує на проблему процесу, що загрожує бізнесу, наприклад дефект безпеки чи захищеності, який міг виникнути лише тоді, коли дисципліна розробки зруйнована.

Формальний звіт має кілька полів, і перше вирішує, чи прочитають решту.

Поле	Правило
Назва	Одне речення, близько дванадцяти слів, що передає суть дефекту. Характерна, а не загальна («аварійне завершення під час входу як адміністратор», ніколи «продукт аварійно завершується»). Проблема на початку, бо список дефектів читають як список. Без префікса з місцем («Сторінка замовлень: ...»): місце належить до тексту або до поля «область продукту», а префікс ламає перегляд списку за впливом. Запити на вдосконалення починаються з «Будь ласка»
Версія	Остання версія, у якій ви бачили дефект; попередні версії згадайте в описі

Середовище	Браузер, операційна система, спосіб запуску продукту; все незвичне
Вкладення	Знімки екрана або короткі відео для всіх звітів, крім найпростіших; файли даних, потрібні для відтворення
Опис	Усе інше, що команді треба знати про цей режим збою; правила тут — правила стриманості
Кроки	Лише якщо не очевидні; ніколи «1. Відкрийте браузер». Останній крок починається словами «Зверніть увагу, що...», щоб сказати, що саме ви вважаєте неправильним
Оракул	Чому ви вважаєте це дефектом, якщо не очевидно. Ніколи «продукт не повинен аварійно завершуватися». Будьте обачні з «Очікувано:»; тестувальники пишуть це навіть тоді, коли очікуваний результат зрозумілий усім
Обхідний шлях	Якщо ви його знаєте

Фокус. Звітування про дефекти — почасти робота продавця: сформулюйте звіт найпереконливіше, але правдиво, щоб його зрозуміли і за ним діяли. Зосередьтеся на тому аспекті, який має найбільший вплив на користувача або найбільший публічний вплив; назвіть найсильніший оракул, який можете. Одна проблема на звіт: дві проблеми в одному звіті отримують одне виправлення.

Заперечення, з якими зустрінеться звіт, передбачувані, і звіт пишуть так, щоб він їх витримав: «Я не розумію, про що ви». «У мене це не відтворюється». «Я не бачу, чому це проблема». «Це складно виправити, і є простий обхідний шлях». «Вам це може не подобатися, але справжнім користувачам сподобається». За ними стоять запитання, які поставить розробник: що саме ви бачили; що ви робили безпосередньо перед цим; чи бачили ви це більше ніж один раз; які дані, які файли, який ввід; яка у вас вагома причина вважати це неправильним; яка версія і платформа; чи пробували ви прості варіації; наскільки легко це викликати; який найгірший наслідок. Якщо ви повідомляєте, що «великий файл» спричиняє помилку, скажіть, наскільки великий, і зробіть файл доступним.

Значущість

Тестувальник — перший суддя того, наскільки великий дефект, і це судження залежить від чотирьох речей.

Чинник	Питання
Частота	Як часто він з'являється або як легко його викликати? Скільки видів подій його викликають? Скільки користувачів

	його побачать? Наскільки він помітний, коли з'являється?
Шкода	Кому і наскільки він шкодить? Блокувальні дефекти, які зупиняють тестування, важливіші за все; далі — дефекти, які шкодять користувачам або блокують їх. Втрата даних зазвичай важливіша за плутанину, але не тоді, коли плутанина веде до небезпечних дій. Уявіть проблему, потім запитайте, кого вона стосується і наскільки ці люди будуть засмучені
Наслідки для інших ризиків	На що він вказує — на процес розробки, який, імовірно, породив подібні дефекти, ще не знайдені? Тут починається чорний прапор
Розголос	Як це звучатиме, коли люди публічно скаржитимуться? Дефекти з невеликою об'єктивною серйозністю можуть завдати великої шкоди довірі

Серйозність і пріоритет — різні судження. Серйозність, яку ви пропонуєте, — це ваша оцінка шкоди; пріоритет — рішення команди про порядок виправлення, і воно враховує вартість і графік. Ваша робота — обґрунтувати; ви не можете наполягати, щоб будь-який дефект виправили, але можете з'єднати факти так, щоб бізнес-ризик був зрозумілим.

Звітування про тестування

Відповідальний тестувальник відповідає за якість роботи і зобов'язується пояснювати й обґрунтовувати зроблене. Для цього потрібні два види розповіді: **звітування про тестування** — історія стану тестування, і **фреймінг тестування** — історія про те, чому саме ця стратегія і як вона служить бізнесу. Історія, за якою розсудливі люди діятимуть, відповідає на три запитання: **Що нового?** — чим сьогодні відрізняється від звичайного стану речей. **Хто це каже?** — чому цьому варто вірити; докази та особиста довіра. **І що з того?** — які рішення це допомагає ухвалити.

Трирівнева історія

Рівень	Зміст
1. Історія продукту	Що з продуктом: дефекти й ризики, які мають значення, приблизно за важливістю
2. Історія процесу	Як тестували: як продукт налаштовували, з ним працювали, за ним спостерігали і як його оцінювали; де ви дивилися і наскільки ретельно; чого не тестували і чому. Вона надає рівню 1 довіри

3. Історія цінності	Ризик продукту, нерозглянутий ризик, упевненість або обережність щодо процесу; чому ви тестували саме так; перешкоди та проблеми тестопридатності, які завадили роботі
---------------------	--

Не кожен звіт потребує всіх трьох рівнів. Давайте аудиторії не набагато більше, ніж вона вважає за потрібне, і не менше, ніж їй справді потрібно; мистецтво — у тому, що ви пропускаєте.

Нотатки

Тестування швидкоплинне; нотатки дають змогу відновити історію. Нотатки — не перелік натиснутих кнопок. Вони відповідають на питання проєктування тестів: що ви покрили, що виявили, чому це, а не те, чому саме ця техніка і ці дані. **Чернеткові нотатки** швидкі й грубі і служать годинами; **архівні нотатки** мають дати підготовленому сторонньому читачеві — або вам через кілька місяців — уявити виконане тестування. Ведіть чернеткові нотатки під час роботи і витрачайте п'ятнадцять–двадцять хвилин після неї на їх переписування. Нотатки з мітками часу під час розслідування — це чернеткові нотатки з часом, у яких спостереження та тлумачення записано в окремих рядках.

PROOF

Структура звіту про стан від Джона Баха. Кожна літера відповідає рівню історії.

Літера	Питання	Рівень
Past (минуле)	Що відбувалося під час тестування?	2
Results (результати)	Що з'ясовано про продукт?	1
Obstacles (перешкоди)	Що заважало або сповільнювало? Чим продукт був важким для тестування?	3
Outlook (перспективи)	Що далі? Що залишилося зробити?	2
Feelings (відчуття)	Як ви до всього цього ставитеся? Чи володієте ситуацією? Що вас турбує?	3

PROOF працює і як усний імпровізований звіт — п'ять питань, на які відповідають за хвилину-дві, — і як перелік для перевірки чужого звіту. Результати без перешкод не дають підстав довіряти результатам або сумніватися в них; минуле і результати без перспектив залишають клієнта гадати про завтра; звіт без відчуттів приховує те, що клієнтові найбільше

потрібно для калібрування, — чи впевнений тестувальник, чи стурбований, і чому.

Безпечна мова

Точність важить для тестувальника більше, ніж для більшості людей, і частина точності — чесно представляти свою невпевненість. Це робить **безпечна мова**: «сервер, здається, завис»; «схоже, це працює — *принаймні я ще не бачив збою*»; «я *припускаю*, що продукт налаштовано правильно»; «очевидно, це єдині випадки». Лінгвісти називають це епістемічною модальністю. Це також особиста дисципліна: змушуючи себе уточнювати твердження, ви нагадуєте власному мозку, що можете помилятися. Деякі клієнти чують у цьому слова-викрутаси; усвідомлюйте цей компроміс, бо відмова від уточнень означає обіцяти те, чого ви знати не можете.

Телескопічний звіт

Клієнтам потрібна від вас певна інформація, і рідко — вся. Упорядкуйте письмовий звіт так, щоб його можна було перервати в будь-якому місці і все одно мати повний звіт на якомусь рівні: підсумок в одне-два речення, який передає головне; абзац, який його пояснює; потім розділи. Пріоритет змісту: **чи все гаразд із продуктом** — так, ні або невідомо, і якщо ні, які проблеми роблять його негаразд, за важливістю; **чи все гаразд із тестуванням** — і якщо ні, що його блокує або йому загрожує і що вам потрібно; **все інше**, що клієнтам треба знати. Дієві судження першими; подробиці — наприкінці.

Низькотехнологічна панель тестування

Звіт на дошці, який Джеймс Бах придумав, коли віцепрезидент попросив щось, на що можна глянути щодня. Кожен рядок — область продукту.

Стовпець	Значення
Зусилля (Effort)	Поточна кількість людей на області: висока, низька, немає, заблоковано або дата початку
Покриття (C)	Якісна шкала: 0 — ще нічого; 1 — базова перевірка; 2 — типові випадки; 3 — відмінне покриття зі складними даними та потоками
Якість (Q)	Думка команди про продукт у цій області; порожньо, якщо тестування ще замало, щоб її скласти
Коментарі	Номери дефектів, блокери, запити: «автоматизація зламана», «потрібна допомога з тестуванням»

Кольори несуть судження. **Червоний**: команда стурбована; це може загрожувати релізу; керівництву варто втрутитися. **Зелений**: команда почувається особливо добре; цілі досягнуто або майже досягнуто. **Жовтий (amber)**: команда не може дійти згоди, червоний це чи зелений. **Чорний** (у стовпці якості не використовується): ні добре, ні тривожно; робота триває. Коли області завершено, їхні клітинки стають зеленими у значенні «зроблено» на тому рівні, який планувався.

На дошці немає жорстких метрик. Її ключові поля — суб'єктивні судження, але не довільні: тестувальник має бути готовий обґрунтувати кожен клітинку аргументами й доказами. Читач може прийняти дошку як є, вважати її підсумком або вказати на клітинку і заглибитися.

Покриття на схемі, а не в числах

Пол Голланд оцінює покриття якісно, за шестирівневою шкалою з лабораторної роботи 1: нічого; базова перевірка (один-два основні сценарії); більше ніж базова (кілька простих сценаріїв із помилками, все ще неглибоко); типові випадки (різноманіття, складні сценарії та сценарії з помилками, найуживаніші та найкритичніші шляхи глибоко); деякі межові випадки; глибокі межові випадки. Рівні позначають кольором на схемі покриття продукту, і конкретні тести за кожним кольором можна обговорити. Більшість автоматизованих регресійних перевірок досягає рівня базової перевірки; деякі — більше ніж базової; далі самими перевітками просунутися важко. **Схема покриття релізу** приховує нижчі рівні схеми й вилучає елементи, які не були ні новими, ні зміненими, щоб стейкхолдери одним поглядом бачили, які області протестовано в усьому продукті.

Підрахунок тест-кейсів і дефектів вводять в оману. Тест-кейси — непорівнянні одиниці: один займає п'ять хвилин, інший — день; один дефект потребує переписування, а двадцять виправляють за пів дня. Чотири ознаки поганої метрики за Голландом: вона порівнює речі, неоднорідні за розміром; створює змагання між людьми; її легко підіграти; вона дає хибне відчуття повноти. Числа можуть загострити історію; без історії вони — чорнильна пляма. Звітуйте, що покрито, наскільки глибоко, що ні і чому.

Перша автоматизована перевірка на кожен коміт

Безперервна інтеграція, тестування, доставка

Безперервна інтеграція (continuous integration, CI) за описом Мартіна Фаулера — це практика частого, принаймні щоденного, об'єднання роботи всіх учасників, щоб проблеми інтеграції виявлялися малими. Гаятрі Мохан описує її чотири складники: система керування версіями, яка містить усю кодову базу; автоматизовані тести; сервер CI, який виконує їх на найновішому коді при кожній зміні; інфраструктура, на якій це працює. Робочий цикл: розробник надсилає невеликий завершений фрагмент;

надсилання запускає конвеєр, перший етап якого збирає застосунок і виконує тести; якщо все пройдено, зміну інтегровано; якщо ні, її власник виправляє або відкочує її.

Безперервне тестування (continuous testing, CT) перевіряє якість застосунку після кожної зміни автоматизованими й ручними методами і сповіщає команду, коли зміна відхиляється від запланованого результату. **Безперервна доставка (continuous delivery, CD)** тримає застосунок готовим до продакшену в будь-який момент, з розгортанням одним натисканням; **безперервне розгортання** автоматично надсилає в продакшен кожен зелений коміт.

Етап збирання й тестування має завершуватися за хвилини — приблизно за час, потрібний, щоб зробити чашку чаю, — щоб команда чекала на нього, перш ніж братися за наступне завдання. Коли повний ланцюжок стає задовгим, тести розділяють на **димові тести (smoke tests)** — кілька тестів, які покривають основний потік кожної функції і виконуються на кожен коміт, — і **нічну регресію**, яка виконує весь набір раз на день. Тести мікрорівня залишаються на етапі збирання й тестування в будь-якому разі.

Етикет

- Комітьте часто, щойно завершено кожен малий фрагмент.
- Комітьте **самотестований код**: автоматизовані тести йдуть у тому самому коміті, що й зміна.
- Проходьте **сертифікаційний тест безперервної інтеграції**: переконайтеся, що ваш коміт пройшов, перш ніж рухатися далі, і поладуйте зламаній етап збирання й тестування за десять хвилин або відкотіть коміт.
- Не коментуйте тести, що не проходять, заради зеленого стану.
- Не надсилайте код у зламану збірку; робота, нагромаджена на зламаній основі, знову не пройде.
- Беріть на себе кожен збій, який спричинила ваша зміна, зокрема у коді, якого ви не писали.

Результат, до якого веде ця практика, вимірюють **чотири ключові метрики** дослідницької групи DORA: час постачання (lead time) від коміту до готовності до продакшену, частота розгортань, середній час відновлення і частка невдалих змін. Перші дві вимірюють темп, останні дві — стабільність. У цій роботі їх названо, але не виміряно.

Мінімальний робочий процес для .NET у GitHub Actions

Робочий процес (workflow) — це файл YAML у `.github/workflows/`. Він називає **події**, які його запускають, одне або кілька **завдань (jobs)**, кожне з яких виконується на свіжій віртуальній машині (**виконавці, runner**), і **кроки (steps)** кожного завдання —

опубліковані дії (actions) або команди оболонки. Заготовка, яка постачається з OrderFlow, містить тригер і перші два кроки; три команди, які залишилися додати, — ті самі три, що ви виконуєте локально.

```
name: ci
on:
  push:
  pull_request:
jobs:
  build-and-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-dotnet@v4
        with:
          dotnet-version: '10.0.x'
      - run: dotnet restore OrderFlow.slnx
      - run: dotnet build OrderFlow.slnx --no-restore
      - run: dotnet test OrderFlow.slnx --no-build
```

Запуск (run) читають на вкладці Actions: запуск, потім завдання, потім крок, який не пройшов, потім його журнал. `dotnet test` виводить по рядку на кожен тест, що не пройшов, із твердженням (assertion), яке не виконалося, очікуваним і фактичним значенням і трасуванням стека, а також підсумковий рядок: `Failed! - Failed: 1, Passed: 4.`

Щоб скасувати закомічену зміну, не переписуючи історію: `git revert <коміт>` створює новий коміт, який її скасовує; надішліть його — і конвеєр виконається знову.

Що говорить зелений запуск

Перевірка (check), у термінах RST, — це механістична верифікація твердження про продукт. Зелений запуск говорить, що кожне твердження, закодоване в перевірках, справдилося на цьому коміті, на цьому виконавці, з цими даними. Він нічого не говорить про твердження, яких ніхто не закодував. П'ять димових тестів, які запускають застосунок, виводять каталог, входять у систему, створюють чернетку, отримують рахунок-фактуру і запитують неіснуюче замовлення, верифікують п'ять тверджень на рівні базової перевірки за шкалою Голланда, і зміна ставки знижки не зачіпає жодного з них. Червоний запуск дає історію продукту про одне твердження та історію процесу про виконанні команди; історію цінності — що збій означає для користувачів і що ще могло зламатися — має написати людина. Конвеєр — це звіт, і, як будь-який звіт, він вартий рівно стільки, скільки історія за ним.

Трекер issue

Трекер дефектів для цієї роботи — GitHub Issues. Issue має назву, текст у Markdown, мітки (labels) і ланцюжок коментарів; зображення, вставлені в текст, додаються автоматично. Назва issue — це назва звіту, тож

правило дванадцяти слів стосується її. Використовуйте мітки для запропонованої серйозності та для області продукту, щоб список можна було переглядати за впливом і фільтрувати за місцем, — саме тому місцю не місце в назві. Коментарі — запис заперечень і відповідей; не редагуйте звіт, щоб приховати помилку, а додайте коментар, який її виправляє.