**Інженерія вимог**

Плутанина щодо термінології вимог поширюється на всю цю дисципліну. Деякі автори називають всю сферу інженерією вимог. Інші називають усе це управлінням вимогами. Отже, варто розуміти різницю: до інженерії вимог відносяться розробка вимог та управління вимогами; до розробки вимог відносяться виявлення, аналіз, специфікація та валідація вимог. Незалежно від життєвого циклу розробки проекту — водоспадного, ітеративного, інкрементального, agile чи гібридного — є певні процеси щодо вимог, що треба виконати. Залежно від життєвого циклу ці дії будуть виконуватися в різний час у проекті та з різним ступенем глибини чи деталізації. Процес розробки вимог включає всі дії, пов’язані з дослідженням, оцінкою, документуванням і підтвердженням вимог до продукту.

Процес **виявлення вимог** охоплює всі дії, пов’язані з виявленням вимог, такі як інтерв’ю, семінари, аналіз документів, створення прототипів тощо. Основні активності цього процесу:

* Визначення ймовірних класів користувачів продукту та інших зацікавлених сторін.
* Розуміння завдань і цілей користувача та бізнес-задач, з якими ці завдання узгоджуються.
* Вивчення середовища, в якому буде використовуватися новий продукт.
* Робота з представниками кожного класу користувачів для розуміння їхніх функціональних потреб та очікувань щодо якості.

Для виявлення вимог зазвичай використовується підхід, орієнтований або на користувача, або на продукт, хоча можливі й інші стратегії. Підхід, орієнтований на користувача спирається на розуміння та дослідження цілей користувача для визначення необхідної функціональності системи. Підхід, орієнтований на продукт, спирається на визначення функцій, які очікувано приведуть до успіху на ринку. Ризик із стратегіями, орієнтованими на продукт, полягає в тому, що ви можете реалізувати функції, які рідко використовуються, навіть якщо під час розробки вони здавалися гарною ідеєю. Рекомендується спочатку визначити бізнес-задачі та цілі користувачів, а потім використовувати цю інформацію для затвердження відповідних функцій і характеристик продукту.

**Аналіз вимог** передбачає досягнення більш повного та точного розуміння кожної вимоги та представлення наборів вимог різними способами. Нижче наведені основні активності процесу:

* Аналіз інформації, отриманої від користувачів, щоб відрізнити їхні вимоги від функціональних вимог, вимог якості, бізнес-правил, запропонованих рішень та іншої інформації.
* Розкладання вимог високого рівня на відповідний рівень деталізації.
* Виведення функціональних вимог з іншої інформації про вимоги.
* Розуміння відносної важливості атрибутів якості.
* Встановлення відповідності вимог до певних програмних компонентів, визначених в архітектурі системи.
* Встановлення пріоритетів впровадження.
* Виявлення прогалин у вимогах або непотрібних вимог, чи вони відносяться до визначеного обсягу.

**Специфікація вимог** включає представлення та зберігання зібраних знань про вимоги в усталеному та добре організованому вигляді. Основною активністю є перетворення зібраних потреб користувачів у письмові вимоги та діаграми, придатні для розуміння, перегляду та використання цільовою аудиторією.

Процес **валідації вимог** потрібен для підтвердження, що набір інформації про вимоги є коректним, і дозволить розробникам створити рішення, яке задовольняє бізнес-задачі. Основними активностями є:

* Перегляд задокументованих вимог для усунення будь-яких проблем до того, як група розробників отримає ці вимоги.
* Розробка приймальних тестів і критеріїв для підтвердження того, що продукт, що відповідає визначеним вимогам, відповідатиме й потребам клієнтів і, відповідно, досягне бізнес-цілей.

Ітерація є ключем до успішного процесу розробки вимог. Варто спланувати декілька циклів дослідження вимог для поступового вдосконалення вимог високого рівня для досягнення більшої точності та деталізації, а також підтвердження вимог користувачами. Це потребує багато часу, але, тим не менш, це невід'ємний аспект роботи з неточністю та невизначеністю на початку створення нової програмної системи.

**Управління вимогами** включає наступні активності:

* Визначення базової лінії вимог, певної дати (дедлайну) для представлення узгодженого, перевіреного і затвердженого набору функціональних і нефункціональних вимог. Часто це конкретна дата для певного релізу продукту або ітерації розробки.
* Оцінка впливу запропонованих змін вимог і контрольоване включення затверджених змін у проект.
* Підтримка планів проекту в актуальному стані з урахуванням вимог в процесі їх розробки/зміни.
* Обговорення змін обов’язків учасників проекту на основі оціненого впливу змін вимог.
* Визначення зв'язків і залежностей, які існують між вимогами.
* Відстеження індивідуальних вимог до відповідних проектів, вихідного коду та тестів.
* Відстеження статусу вимог і змін протягом усього проекту.

Метою процесу управління вимогами є не запобігання змінам, а передбачення та врахування реальних змін, яких можна очікувати, і мінімізація їх руйнівного впливу на проект.

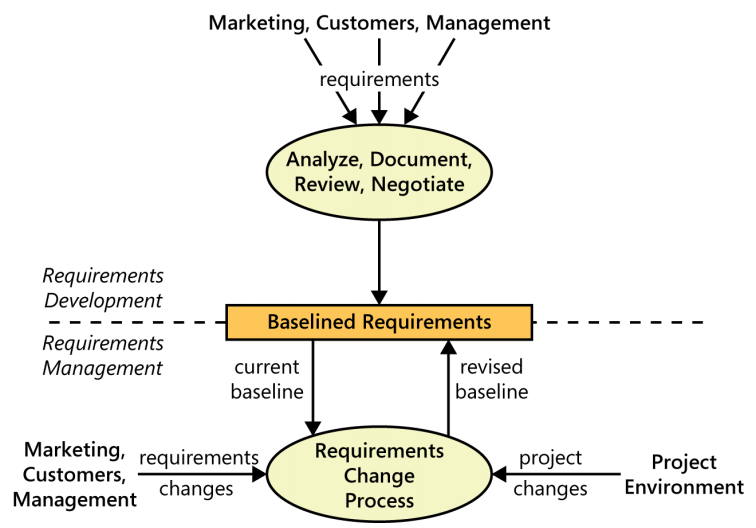


Рисунок 1 - Процеси інженерії вимог

На основі визначених бізнес-потреб, потреб ринку або концепції нового продукту менеджери або відділ маркетингу визначають бізнес-вимоги до програмного забезпечення, яке допоможе їхній компанії працювати ефективніше (для інформаційних систем) або успішно конкурувати на ринку (для комерційних продуктів). У корпоративному середовищі бізнес-аналітик зазвичай працює з представниками користувачів, щоб визначити користувацькі вимоги. Компанії, що розробляють комерційні продукти, часто призначають менеджера продукту, щоб визначити, які функції включити в новий продукт. Кожна користувацька вимога та характеристика повинні відповідати вимогам бізнесу. Виходячи з вимог користувача, бізнес-аналітик або менеджер продукту визначають функціональність, яка дозволить користувачам досягти своїх цілей. Розробники використовують функціональні та нефункціональні вимоги для розробки рішень, які реалізують необхідну функціональність, враховуючи усталені обмеження. Тестувальники визначають, як перевірити, чи вимоги були коректно реалізовані.

**Класифікація користувачів**

Залучення клієнтів — найкращий спосіб уникнути невиправданих очікувань, невідповідності між продуктом, який клієнти очікують отримати, і тим, що створюють розробники. Недостатньо просто запитати кількох клієнтів або їхніх представників/менеджерів, чого вони хочуть один чи два рази, а потім почати кодувати. Якщо розробники будують саме те, що клієнти спочатку вимагали, ймовірно, доведеться перероблювати все, оскільки клієнти часто не знають, що їм насправді потрібно. Крім того, бізнес-аналітики можуть розмовляти з не тими людьми або задавати неправильні запитання.

Характеристики, які користувачі представляють як свої «бажання», не обов’язково відповідають функціональним можливостям, необхідним для виконання завдань за допомогою нового продукту. Щоб отримати більш точне уявлення про потреби користувачів, бізнес-аналітик повинен зібрати широкий діапазон вхідних даних користувачів, проаналізувати та уточнити їх, а також визначити, що саме потрібно створити, щоб користувачі могли виконувати свою роботу. Бізнес-аналітик несе головну відповідальність за фіксування необхідних можливостей і властивостей нової системи, а також за передачу цієї інформації іншим зацікавленим сторонам. Це повторюваний процес, який вимагає часу. Якщо витрачається замало часу на досягнення спільного розуміння (спільного бачення запланованого продукту) це може призвести до переробки, порушення термінів, перевитрати коштів і незадоволення клієнтів.

Термін «користувач програмного забезпечення» часто використовують так, ніби всі користувачі належать до монолітної групи зі схожими характеристиками та потребами. Насправді більшість продуктів направлені на різноманітні групи користувачів із різними очікуваннями та цілями. Замість того, щоб думати про «користувача» як про певний юніт, варто приділити деякий час визначенню кількох класів користувачів, їхніх ролей і привілеїв стосовно продукту.

Клас кінцевих користувачів — це підмножина користувачів продукту, яка є підмножиною замовників продукту, яка є підмножиною стейкхолдерів. Певна особа може належати до кількох класів користувачів. Наприклад, адміністратор програми іноді також може взаємодіяти з нею як звичайний користувач. Користувачі продукту можуть відрізнятися за наступними ознаками (можна згрупувати користувачів у кілька різних класів на основі цих відмінностей):

* Їхні права доступу або рівні безпеки (наприклад, звичайний користувач, гість, адміністратор).
* Завдання, які вони виконують в рамках своїх обов’язків.
* Характеристики продукту, які потрібні користувачам.
* Частота, з якою вони використовують продукт.
* Їх досвід у сфері застосування продукту (або аналогів) та знання комп’ютерних систем.
* Платформи, які вони використовуватимуть (настільні ПК, ноутбуки, планшети, смартфони, спеціалізовані пристрої).
* Їх рідна мова.
* Чи взаємодіятимуть вони з системою прямо чи опосередковано.

Звісно, легше об’єднати користувачів у класи залежно від їхнього географічного розташування чи типу компанії, у якій вони працюють. Наприклад, компанія, яка розробляє програмне забезпечення для банківської сфери, спочатку розглядає можливість класифікувати користувачів залежно від того, чи працювали вони у великому комерційному банку чи малому, або у кредитній спілці тощо. Але ці відмінності представляють різні сегменти ринку, але не різні класи користувачів.

Кращий спосіб визначити класи користувачів — це створити список всіх завдань, які різні користувачі виконуватимуть із системою. Всі типи фінансових установ з попереднього прикладу мають однакові групи працівників в рамках однієї організації — це касири, працівники, які обробляють заяви на кредити, або оформлюють страхування тощо. Група осіб, які виконують певну діяльність (тобто мають певну посаду чи роль), матимуть однакові функціональні потреби для системи в усіх фінансових установах. Касири в різних банківських установах мають робити більш-менш однакові речі. Таким чином, більш логічні назви класів користувачів для банківської системи можуть включати касира, кредитного спеціаліста, спеціаліста зі страхування, менеджера відділення тощо. Додаткові класи користувачів можна виявити шляхом визначення всіх можливих варіантів використання, фіксування історій користувачів, ходу процесів і тих, хто може приймати участь в цих процесах.

Певні класи користувачів можуть бути важливішими за інші для конкретного проекту. До привілейованих класів відносяться користувачі, чия задоволеність найбільше пов’язана з досягненням бізнес-цілей проекту. При вирішенні конфліктів між вимогами з різних класів користувачів або прийнятті пріоритетних рішень, привілейовані класи користувачів отримують перевагу. Це не означає, що замовникам, які платять за систему (які можуть взагалі не бути кінцевими користувачами), або тим, хто має найбільший політичний вплив, обов’язково слід віддавати перевагу. Мається на увазі лише узгодження з бізнес-задачами.

Непривілейовані класи користувачів – це користувачі, які не повинні використовувати продукт з юридичних міркувань, міркувань безпеки або через відсутність певних прав. Можна вбудувати такі функції/характеристики, щоб навмисно ускладнити непривілейованим користувачам можливість виконання дій, які вони не повинні робити. Приклади включають в себе механізми безпеки, рівні привілеїв користувачів, функції захисту від зловмисного програмного забезпечення і логування використання. Наприклад, блокування облікового запису користувача після чотирьох невдалих спроб входу захищає від доступу непривілейованого класу користувачів «імітатори користувачів», хоча це і є ризиком створити незручності забудькуватим привілейованим користувачам. Якщо мій банк не розпізнає комп’ютер, яким я користуюся, він надсилає мені повідомлення на електронну пошту з одноразовим кодом доступу, який я маю ввести перед входом. Ця функція реалізована через непривілейований клас користувачів «люди, які могли вкрасти мою банківську інформацію».

Кожен клас користувачів матиме власний набір вимог до завдань, які мають виконувати члени класу. Потреби різних класів користувачів можуть збігатися. Наприклад, касирам, бізнес-банкірам і кредитним спеціалістам, можливо, доведеться перевірити баланс банківського рахунку. Різні класи користувачів також можуть мати різні очікування щодо якості, наприклад юзабіліті, що впливатиме на вибір дизайну інтерфейсу користувача. Нових або випадкових користувачів більше хвилює те, наскільки легко освоїти систему. Таким користувачам подобаються меню, графічний інтерфейс користувача, неперевантажені елементами екрани/вікна/сторінки, різноманітні майстри (wizards) та довідки. Коли користувачі набувають досвіду роботи з системою, вони стають більш зацікавленими в ефективності. Тоді їм вже більше потрібні комбінації клавіш, параметри налаштування, панелі інструментів і сценарії.

Класи користувачів не обов’язково повинні мати на увазі людей. Це можуть бути програмні агенти (боти), які виконують сервіси від імені користувача-людини. Програмні агенти можуть сканувати мережі в пошуках інформації про товари та послуги, створювати власні канали новин, обробляти вхідну електронну пошту, відстежувати фізичні системи та мережі на наявність проблем чи сторонніх втручань або виконувати data mining. Інтернет-агенти, які перевіряють веб-сайти на вразливості або генерують спам, є різновидом непривілейованого класу програмних агентів. Якщо визначите подібні класи користувачів, ви можете вказати певні вимоги не для задоволення їхніх потреб, а для того, щоб перешкодити їм. Наприклад, інструменти веб-сайту, такі як CAPTCHA, які перевіряють, чи є користувач людиною, намагаються заблокувати доступ «шкідливим користувачам».

Потрібно розглянути набагато ширший діапазон потенційних джерел вимог, ніж просто класи безпосередніх та опосередкованих користувачів. Наприклад, хоча члени команди розробників не є кінцевими користувачами системи, яку вони створюють, потрібна їхня думка/погляд щодо внутрішніх атрибутів якості, таких як ефективність, модифікованість, портативність, розширюваність і можливість повторного використання. Коли ви намагаєтеся визначити зацікавлених сторін, чиї вимоги є необхідними, варто дивіться не тільки на очевидних кінцевих користувачів.

Переконайтеся, що фокус-група представляє типи користувачів, чиї потреби мають стимулювати розвиток вашого продукту. Включайте як експертів, так і менш досвідчених клієнтів. Якщо ваша фокус-група складається лише з користувачів, які легко навчаються, або таких, які вважають, що чим більше функцій має система, тим краще, ви можете отримати багато складних і технічно складних вимог, які небагато клієнтів вважають корисними.

Різні моделі аналізу можуть використовуватися для визначення класів користувачів. Зовнішні сутності, показані за межами системи на контекстній діаграмі, є кандидатами на класи користувачів. Структурна діаграма корпоративної організації також може допомогти при виявленні потенційних користувачів та інших зацікавлених сторін. Майже всіх потенційних користувачів системи можна знайти десь на цій діаграмі.

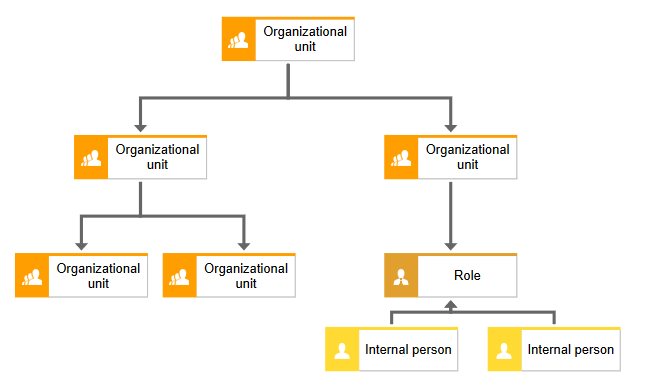


Рисунок 2 - Шаблон структурної (організаційної) діаграми

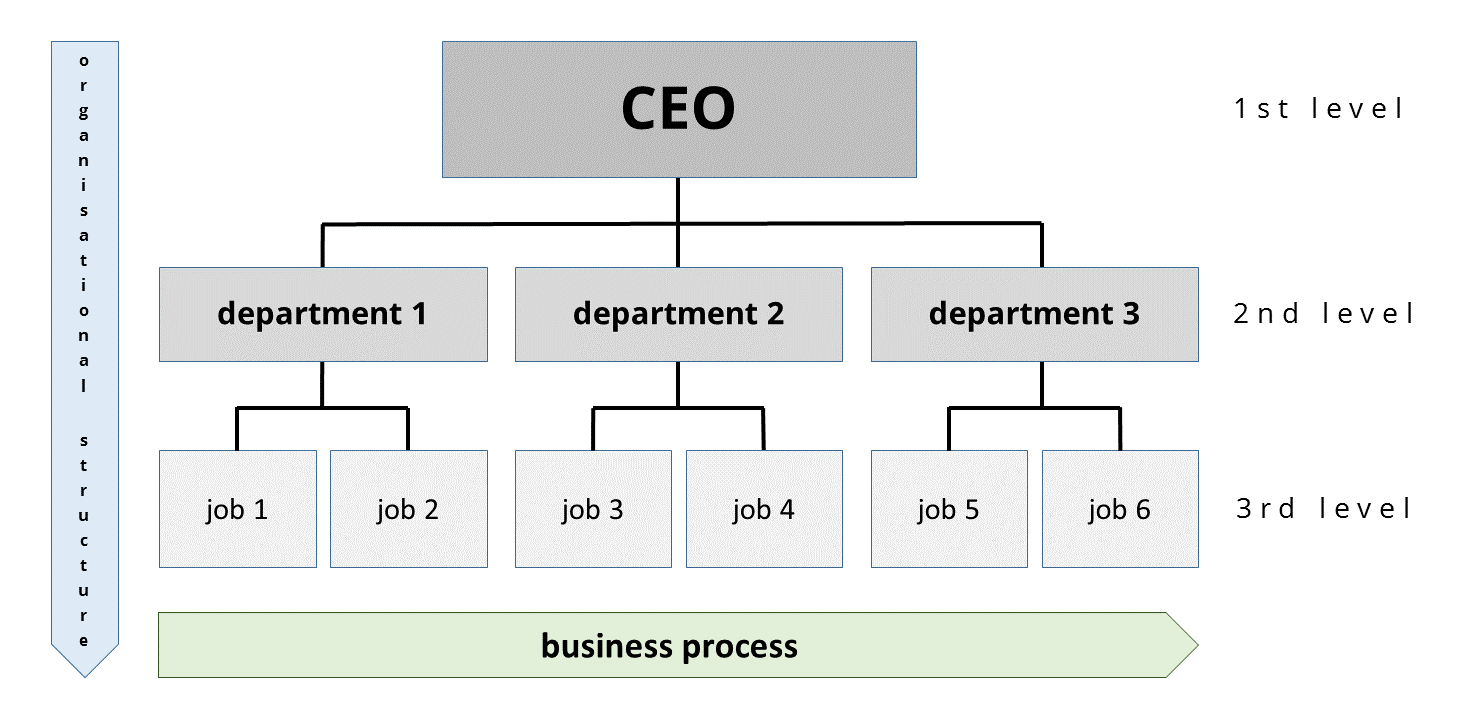


Рисунок 3 - Шаблон структурної (організаційної) діаграми

Виконуючи аналіз стейкхолдерів і користувачів, на структурній діаграмі організації треба знайти:

* Відділи, які беруть участь у бізнес-процесі.
* Відділи, на які впливає бізнес-процес.
* Відділи або ролі, у яких можна знайти прямих або непрямих користувачів.
* Класи користувачів, які охоплюють кілька відділів.
* Відділи, які можуть мати зв’язок із зовнішніми стейкхолдерами за межами компанії.

**Виявлення вимог**

Основою розробки вимог є процес виявлення вимог, тобто визначення потреб і обмежень різних зацікавлених сторін стосовно програмної системи. Виявлення – це не те саме, що «збір вимог». Це також не просто фіксування всього, що говорять користувачі. Виявлення – це колаборативний та аналітичний процес, який включає такі активності як знаходження, збір, відокремлення та визначення вимог. Виявлення використовується для формування бізнес-вимог, користувацьких, функціональних і нефункціональних вимог, а також інших типів інформації. Виявлення вимог — це, мабуть, найскладніший та критичний аспект розробки програмного забезпечення, схильний до появи помилок та інтенсивний з точку зору комунікацій між учасниками.

На цьому етапі бізнес-аналітикам треба зрозуміти процеси, що стоять за вимогами користувачів. Тобто знати алгоритми дій, які виконують користувачі, як приймають рішення стосовно своєї роботи, і витягнути звідти основну логіку. Цю інформацію треба донести і до інших учасників проекту, щоб всі розуміли, чому система повинна виконувати певні функції. Крім цього, із запропонованих вимог вилучаються такі, що відображають застарілі або неефективні бізнес-процеси або правила, які не слід включати в нову систему.

**Техніки виявлення вимог**

Техніки виявлення включають як групові активності, під час яких ви взаємодієте із зацікавленими сторонами, щоб виявити вимоги, так і незалежні активності, коли ви самостійно шукаєте інформацію. Групові активності в основному зосереджені на виявленні вимог бізнесу та користувачів. Робота безпосередньо з користувачами необхідна, оскільки користувацькі вимоги охоплюють завдання, які користувачі мають виконати за допомогою системи. Щоб отримати бізнес-вимоги, вам потрібно буде працювати з такими людьми, як спонсори проекту. Незалежні методи виявлення доповнюють вимоги користувачів і розкривають необхідну функціональність, про яку кінцеві користувачі можуть не знати. В більшості проектів використовується комбінація як групових, так і незалежних активностей для виявлення вимог. Кожна техніка пропонує різне дослідження вимог або може навіть виявити зовсім іншу групу вимог.

**Співбесіда**

Найбільш очевидний спосіб дізнатися, що потрібно користувачам програмної системи, це запитати їх. Співбесіда (або інтерв'ю) є традиційним джерелом збору вимог як для комерційних продуктів, так і для інформаційних систем для всіх підходів до розробки програмного забезпечення. Більшість бізнес-аналітиків впроваджують в певній формі індивідуальні співбесіди або співбесіди у малих групах, щоб зібрати вимоги для проекту. Agile проекти широко використовують інтерв’ю як механізм для прямого залучення користувачів. Співбесіди легше запланувати та провести, ніж активності у великих групах, такі як воркшопи.

Для підготовки до співбесіди треба попередньо створити будь-які матеріали (наприклад, список питань), щоб вести розмову в потрібному напрямку. Чернетки матеріалів дадуть користувачам відправну точку для міркувань. Люди часто легше критикують контент, ніж створюють його. Замість того, щоб просто фіксувати те, що говорять клієнти, креативний бізнес-аналітик пропонує ідеї та альтернативи в процесі виявлення. Іноді користувачі не усвідомлюють, які можливості можуть надати розробники, і відповідно будуть у захваті, коли буде запропонована функціональність, яка зробить систему ще більш цінною.

**Воркшопи**

Воркшопи (або семінари) заохочують співпрацю зацікавлених сторін у визначенні вимог. Семінар часто включає кілька типів стейкхолдерів, від користувачів до розробників і тестувальників. Робота в групі ефективніша для вирішення неузгодженостей у вимогах, ніж розмова з людьми окремо. Крім того, семінари корисні, коли потрібно швидко виявити вимоги через обмеження в графіку робіт.

Координатор (фасилітатор) відіграє вирішальну роль у плануванні семінару, відборі учасників і керуванні тематикою обговорення. Потрібен ще один член команди для фіксування всіх ідей/вимог, які виникають під час обговорення. Бізнес-аналітики повинні часто проводити подібні воркшопи. Коли команда тільки починає працювати за новим підходом до виявлення вимог, варто залучити координатора для допомоги бізнес-аналітику. Таким чином бізнес-аналітик може присвятити свою повну увагу обговоренню. Надзвичайно складно одночасно на високому рівні виконувати функції бізнес-аналітика і координатора, ще й брати активну участь в обговоренні.

**Фокус-групи**

Фокус-група — це репрезентативна група користувачів, яку збирають для групової активності з виявлення, щоб отримати вхідні дані та ідеї щодо функціональних вимог і вимог до якості продукту. Сеанси фокус-груп мають бути інтерактивними, щоб усі користувачі мали можливість висловити свої думки. Фокус-групи корисні для вивчення відношення, вражень, уподобань і потреб користувачів. Вони особливо цінні, якщо ви розробляєте комерційні продукти й не маєте прямого доступу до кінцевих користувачів у своїй компанії.

Необхідно координувати проведення зустрічей з фокус-групами. Потрібно дотримуватися теми обговорення, але не впливати на учасників, не нав’язувати ідеї. Варто також робити аудіо- або відеозапис сесії, щоб повторно уважно прослухати коментарі учасників. Від зустрічей з фокус-групами не варто очікувати кількісного аналізу, а скоріше багато суб’єктивних відгуків, які можна додатково оцінити та визначити пріоритети в процесі розробки вимог. На сесіях з фокус-групами можна скористатися порадами, описаними для семінарів. Учасники фокус-груп зазвичай не мають повноважень приймати рішення щодо вимог.

**Спостереження**

Коли ви просите користувачів описати, як вони виконують свою роботу, їм, швидше за все, буде важко описати всі деталі — щось буде пропущене або погано сформульоване. Часто це трапляється через те, що завдання складні, і важко запам’ятати і пересказати кожну деталь процесу. В інших випадках це відбувається тому, що користувачі виконують завдання “на автоматі” і не можуть сформулювати все, що вони роблять. Тобто, завдання настільки звичне, що вони навіть не замислюються в процесі. Отже, можна багато чому навчитися, спостерігаючи, як саме користувачі виконують свої завдання.

Спостереження за робочим процесом користувача в відповідному середовищі дозволяє бізнес-аналітику перевіряти інформацію, зібрану з інших джерел, визначати нові теми для співбесід, побачити проблеми з поточною системою та способи, за допомогою яких нова система може краще підтримувати робочий процес. Бізнес-аналітик має абстрагуватися й узагальнити дії користувача, щоб гарантувати, що визначені таким чином вимоги застосовуються до класу користувачів у цілому, а не лише до окремої особи. Бізнес-аналітик також може запропонувати ідеї щодо покращення поточних бізнес-процесів користувача.

**Анкетування**

Анкетування – це спосіб опитування великих груп користувачів, щоб зрозуміти їхні потреби. Вони недорогі, що робить їх логічним вибором для отримання такої кількості інформації, і крім того зникає обмеження щодо місцезнаходження опитуваних. Проаналізовані результати анкет можна використовувати як вхідні дані для інших методів виявлення вимог. Наприклад, ви можете використати анкету, щоб визначити найбільші проблеми користувачів із наявною системою, а потім використовувати результати для обговорення пріоритетів на семінарі із особами, які приймають рішення. Ви також можете використовувати анкети для опитування користувачів комерційних продуктів для отримання зворотного зв’язку.

**Аналіз інтерфейсу системи**

Аналіз інтерфейсу — це незалежний метод виявлення вимог, який передбачає перевірку систем, до яких підключається поточна система. Аналіз інтерфейсу системи виявляє функціональні вимоги щодо обміну даними та сервісами між системами. Контекстні діаграми та карти екосистем є очевидним вибором для початку пошуку інтерфейсів для подальшого вивчення.

Для кожної системи, яка взаємодіє з поточною, треба визначити функціонал, з якого можуть випливати вимоги до поточної системи. Ці вимоги можуть описувати, які дані передавати іншій системі, які дані отримуються з неї, а також правила щодо цих даних, наприклад критерії перевірки. Тут також можна виявити існуючу функціональність, яку не потрібно впроваджувати у системі.

**Аналіз інтерфейсу користувача**

Аналіз інтерфейсу користувача — це незалежний метод виявлення, за допомогою якого вивчаються існуючі системи для виявлення користувацьких та функціональних вимог. Найкраще взаємодіяти з існуючими системами безпосередньо, але при необхідності можна використовувати скріншоти роботи системи. Посібники (інструкції) користувача часто містять скріншоти, які добре підходять в якості відправної точки. Можна також розглянути інтерфейси користувача аналогічних продуктів.

Під час роботи з існуючою системою аналіз інтерфейсу користувача можна використати для визначення повного списку екранів, і таким чином виявити потенційні характеристики та функції. Взаємодіючи з інтерфейсом користувача існуючої системи, можна дізнатися про типові кроки, які виконують користувачі в системі, і створити черновики варіантів використання для перегляду з користувачами. Отже, замість того, щоб запитувати користувачів, як вони взаємодіють із системою та які кроки роблять, можливо, можна досягти початкового розуміння системи самостійно. Далеко не завжди певні функції потрібні новій системі лише тому, що ви знайшли їх у існуючій. Крім того, не припускайте, що через те, що користувальницький інтерфейс виглядає або поводиться певним чином у поточній системі, він має бути реалізований таким же чином у майбутній системі.

**Аналіз документів**

Аналіз документів передбачає перевірку будь-якої наявної документації на наявність потенційних вимог до програмного забезпечення. Найкорисніша документація включає специфікації вимог, бізнес-процеси та посібники користувача для існуючих або подібних програм. Документи можуть описувати корпоративні чи галузеві стандарти, яких необхідно дотримуватися, або правила, яким має відповідати продукт. Під час заміни/оновлення існуючої системи документація може виявити функціональність, яку, можливо, потрібно зберегти, а також застарілу функціональність. Порівняльний аналіз вкаже на недоліки в інших продуктах, від яких можна позбавитися в системі, що розробляється, щоб отримати конкурентну перевагу. Звіти про проблеми та запити на вдосконалення, зібрані від користувачів довідковою службою та персоналом підтримки, можуть запропонувати ідеї щодо вдосконалення системи в майбутніх релізах.

Аналіз документів може виявити інформацію, яку стейкхолдери не повідомляють, або тому, що вони про це не думають, або тому, що вони цього не знають. Наприклад, якщо при створенні нової програми, була знайдена складна бізнес-логіка, описана в посібнику користувача для існуючої програми. Можливо, користувачі навіть не знають про цю логіку. Тоді можна використовувати результати цього аналізу як вхідні дані для опитування користувачів.

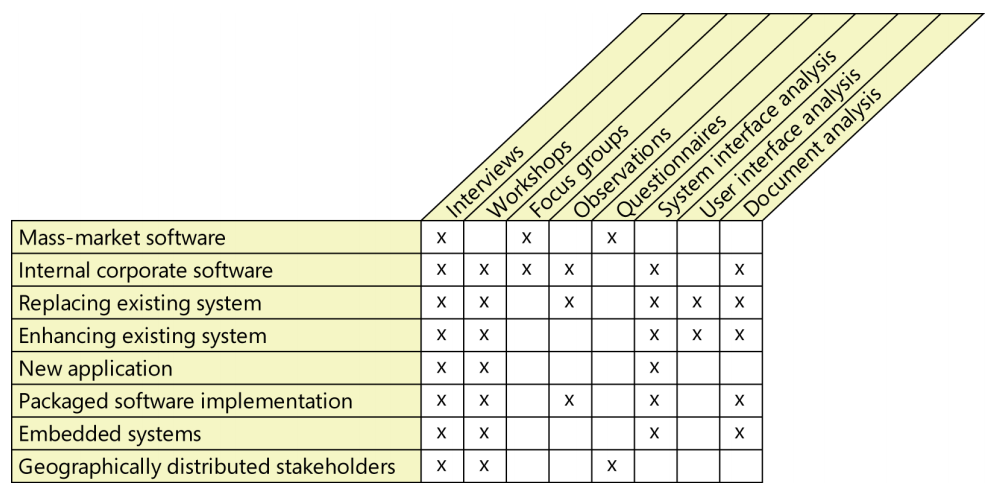


Рисунок 4 - Методи виявлення вимог в залежності від типу проекту

**Класифікація даних, отриманих від стейкхолдерів**

Учасники процесу виявлення вимог не скажуть так просто: «Ось список моїх бізнес-вимог». Аналітик повинен визначити, до якого типу інформації належить кожне твердження, отримане від клієнтів. Є кілька типових фраз, які допоможуть у процесі класифікації.

**Бізнес-вимоги.** Бізнес-вимогами є все, що описує фінансову, ринкову або іншу бізнес-вигоду, яку або клієнти, або організація, що розвивається, бажають отримати від продукту. Треба звертати увагу на твердження про цінність, яку отримають покупці або користувачі програмного забезпечення, наприклад:

* «Збільшити частку ринку в регіоні *a* на *b* відсотків протягом *c* місяців».
* «Заощадити *x* доларів на рік на електроенергії, яку тепер витрачають неефективні установки».

**Користувацькі вимоги.** Загальні формулювання цілей користувача або бізнес-завдань, які користувачі мають виконати, відносяться до користувацьких вимог, які найчастіше представлені у вигляді варіантів використання, сценаріїв або історій користувачів. Користувач, який каже: «Мені потрібно <щось зробити>», ймовірно, описує вимоги користувача, як у наступних прикладах:

* «Мені потрібно надрукувати поштову етикетку для посилки».
* «Мені, як головному оператору, щоранку потрібно в першу чергу калібрувати контролер насоса».

**Бізнес-правила.** Коли клієнт каже, що лише певні користувачі можуть виконувати дію за певних умов, це може бути бізнес-правило. Це не вимоги до програмного забезпечення в такому формулюванні, але з цього можна вивести деякі функціональні вимоги для дотримання правила. Такі фрази, як «Повинен відповідати . . . », «Якщо <деяка умова виконується>, то <щось трапиться>,» або «Потрібно обчислити відповідно до . . . » припускають, що користувач описує бізнес-правило. Приклади:

* «Новий клієнт повинен сплатити 30 відсотків передбачуваного гонорару за консультацію та витрати на відрядження заздалегідь».
* «Схвалення відпустки має відповідати політиці компанії щодо відпусток у відділі кадрів».

**Функціональні вимоги.** Функціональні вимоги описують поведінку системи за певних умов і дії, які система дозволяє виконувати користувачам. Ось кілька прикладів функціональних вимог, які можна почути від користувачів:

* «Якщо тиск перевищує 2.75 бар, повинна загорітися сигнальна лампа високого тиску».
* «Користувач повинен мати можливість сортувати список проектів у прямому та зворотному алфавітному порядку».

Ці твердження ілюструють, як користувачі зазвичай представляють функціональні вимоги, але це не є хорошим формулюванням функціональних вимог. Бізнес-аналітику потрібно буде перетворити їх у більш точні специфікації.

**Атрибути якості.** Твердження, які описують, наскільки добре система щось виконує, є атрибутами якості. Тут варто звертати увагу на слова, які описують бажані характеристики системи: швидка, проста, зручна, надійна, безпечна. Потрібно буде поспілкуватися з користувачами, щоб зрозуміти, що вони мають на увазі під цими неоднозначними та суб’єктивними термінами, щоб можна було написати чітко сформульовані цілі якості, такі які можна перевірити. Наведені нижче приклади показують, як можуть звучати атрибути якості, описані користувачами:

* «Мобільне програмне забезпечення має швидко реагувати на команди натискання».
* «Механізм кошика для покупок має бути простим у використанні, щоб нові клієнти не відмовлялися від покупки».

**Вимоги до зовнішнього інтерфейсу.** Вимоги в цій категорії описують зв’язки між поточною системою та зовнішнім світом. Шаблон SRS містить розділи для інтерфейсів користувачів, обладнання та інших програмних систем. Такі фрази, як «Потрібно зчитувати сигнали з . . . », «Потрібно надсилати повідомлення на . . . », «Повинен мати можливість читати файли <у форматі>» та «Елементи інтерфейсу користувача мають відповідати <стандарту>» вказують на те, що замовник описує вимоги до зовнішнього інтерфейсу. Приклади:

* «Виробнича система повинна контролювати сортувальник пластин».
* «Мобільний додаток має надіслати зображення чека в банк після того, як я сфотографую свій підписаний чек».

**Обмеження.** Обмеження проектування та реалізації, відповідно, обмежують можливості, доступні розробнику. Пристрої з вбудованим програмним забезпеченням часто мають відповідати фізичним обмеженням, таким як розмір, вага та гнізда/роз’єми/інтерфейси. Фрази, які вказують на те, що замовник описує обмеження щодо дизайну чи реалізації, включають: «Повинен бути написаний <конкретною мовою програмування>», «Не може перевищувати <певний ліміт>» і «Потрібно використовувати <особливий елемент керування інтерфейсом користувача>». Приклади обмежень:

* «Розмір файлів, надісланих в електронному вигляді, не може перевищувати 10 Мб».
* «Браузер повинен використовувати 256-бітове шифрування для всіх безпечних транзакцій».

Як і у випадку з функціональними вимогами, недостатньо просто фіксувати твердження користувача про обмеження. Треба дізнатися, чому існує таке обмеження, перевірити його доцільність/вірогідність і записати обґрунтування його включення до вимог.

**Вимоги до даних.** Клієнти пред’являють вимоги до даних кожного разу, коли описують формат, тип даних, дозволені значення або значення за замовчуванням для елемента даних, комплексну структуру бізнес-даних або звіт, який потрібно створити. Приклади вимог до даних:

* «Поштовий індекс складається з п’яти цифр, необов’язкового дефісу і чотирьох цифр, які за замовчуванням мають бути 0000».
* «Замовлення складається з інформації про клієнта, інформації про доставку та одного або кількох продуктів, кожен із яких містить номер продукту, кількість одиниць, ціну за одиницю та загальну вартість».

**Ідеї для реалізації.** Багато «вимог» від користувачів насправді є ідеями реалізації певних рішень. Хтось, хто описує певний спосіб взаємодії із системою для виконання певної дії, пропонує рішення. Бізнес-аналітику потрібно дослідити глибину ідеї рішення, щоб дійти до реальних вимог. Знову ж таки треба виявити справжню потребу користувача, що стоїть за ідеєю. Наприклад, паролі є лише одним із кількох можливих способів реалізації вимог безпеки. Приклади ідей реалізації деяких рішень:

* «Потім зі спадного списку я вибираю область, куди я хочу надіслати посилку».
* «Телефон повинен дозволяти користувачеві проводити пальцем в бік по дисплею, щоб переключатися між екранами».

У першому прикладі фраза про спадний список вказує на те, що це ідея реалізації рішення, оскільки вона описує певний елемент керування інтерфейсу користувача. Тут варто спитати: «Чому зі спадного списку?» Якщо користувач відповідає: «Це просто здавалося хорошим способом вибрати регіон», тоді справжня вимога виглядає приблизно так: «Система повинна дозволити користувачеві вказати регіон/область, куди він хоче надіслати посилку». Але, можливо, користувач скаже: «Ми робимо те саме в кількох інших місцях, і я хочу, щоб все було узгоджено і виглядало однаково. Крім того, спадний список запобігає введенню користувачем недійсних даних». Це вже виправдані підстави реалізувати ідею. Але така ідея реалізації накладає дизайнерське обмеження на цю вимогу: обмежує виконання вимоги лише одним способом. Це не обов’язково неправильно чи погано; просто варто переконайтися, що подібне обмеження існує з поважної причини.

**Варіанти використання та історії користувачів**

Варіант використання описує послідовність взаємодій між системою та зовнішнім актором, що призводить до того, що актор може досягти потрібного йому результату. Назви варіантів використання завжди пишуться у формі дієслова, після якого йде об’єкт. Вибирайте такі назви, щоб було однозначно зрозуміло, чим саме варіант використання буде цінним для певного користувача.

| Система | Прецеденти |
| --- | --- |
| Система відстеження хімікатів | Замовити хімікат  Змінити замовлення  Перевірити статус замовлення  Створити квартальний звіт про використання хімікатів |
| Стійка реєстрації в аеропорту | Зареєструватися на рейс  Роздрукувати посадкові талони  Змінити місце  Перевірити багаж |
| Система бухгалтерського обліку | Створити рахунок-фактуру  Звірити виписку з рахунку  Додати транзакцію з кредитною карткою  Роздрукувати податкові форми для постачальників  Шукати конкретну транзакцію |
| Інтернет-книгарня | Оновити профіль клієнта  Шукати товар  Купити товар  Відстежити надісланий пакунок  Скасувати невідправлене замовлення |

Історії користувача, які використовуються в agile-проектах, — це «короткий простий опис характеристики з точки зору людини (як правило, користувача або клієнта системи), яка бажає отримати нову можливість від системи». Історії користувачів часто пишуться за таким шаблоном, хоча використовуються й інші стилі:

Як <тип користувача>, я хочу <якась мета>, щоб <якась причина>.

Використання цього шаблону надає перевагу перед навіть коротшою назвою прецеденту, оскільки, хоча вони обидва вказують на мету користувача, історія користувача також визначає клас користувача та обґрунтування запиту цієї можливості для системи. Це цінні доповнення. Клас користувача, який не обов’язково повинен бути людиною, в історії користувача відповідає головному актору прецеденту. Обґрунтування можна надати в короткому описі варіанту використання. Таблиця нижче показує, як можна представити деякі варіанти використання з попередньої таблиці у формі історій користувачів.

| Система | Прецедент | Історія користувача |
| --- | --- | --- |
| Система відстеження хімікатів | Замовити хімікат | Як хімік, я хочу замовити хімікат, щоб я міг проводити експерименти. |
| Стійка реєстрації в аеропорту | Зареєструватися на рейс | Як пасажир, я хочу зареєструватися на рейс, щоб я міг дістатися до місця призначення. |
| Система бухгалтерського обліку | Створити рахунок-фактуру | Як власник малого бізнесу, я хочу створити рахунок-фактуру, щоб я міг виставити рахунок клієнту. |
| Інтернет-книгарня | Оновити профіль клієнта | Як клієнт, я хочу оновити свій профіль, щоб майбутні покупки були сплачені з нового номеру кредитної картки |

На цьому рівні варіанти використання дуже схожі на історії користувачів. Обидва зосереджені на розумінні того, що різні типи користувачів повинні досягти через взаємодію з програмною системою. Обидва підходи також можуть бути використані при створенні моделей візуального аналізу. Однак ці два підходи рухаються в різних напрямках: зі специфікації прецедентів виводяться функціональні вимоги та тести, а з користувацьких історій — приймальні тести.

Історії користувачів містять стислий опис потреб користувача. Варіанти використання занурюються глибше в опис того, як користувач уявляє взаємодію із системою для досягнення своєї мети. Варіант використання не повинен заглиблюватися в специфіку дизайну, а лише в представлення користувача про взаємодію. Історії користувачів пропонують перевагу простоти та лаконічності, але це є також недоліком. Варіанти використання надають учасникам проекту структуру та контекст, яких бракує колекції історій користувачів. Прецеденти забезпечують бізнес-аналітику організований спосіб вести дискусії щодо виявлення інформації, окрім простого збору списку речей, які користувачі повинні досягти за допомогою системи, як відправної точки для планування та обговорення.

**Підхід з використанням прецедентів**

Варіант використання описує послідовність взаємодій між системою та зовнішнім актором, що призводить до певного результату, що забезпечує цінність для актора. Актор — це особа (або іноді інша програмна система чи апаратний пристрій), яка взаємодіє з системою для виконання сценарію використання. Нижче наведено кілька запитань, які допоможуть представникам користувачів визначити акторів:

* Хто (або що) отримує сповіщення, коли щось відбувається в системі?
* Хто (або що) надає інформацію чи сервіси системі?
* Хто (або що) допомагає системі реагувати на завдання та виконувати його?

Діаграму прецедентів можна порівняти з контекстною діаграмою, яка створюється в процесі встановлення бізнес-вимог. Обидві визначають межу між об'єктами, які знаходяться поза системою, і об'єктами всередині системи. На діаграмі варіантів використання рамка відокремлює деякі внутрішні аспекти системи — варіанти використання — від зовнішніх учасників. Контекстна діаграма також зображує об’єкти, які знаходяться поза системою, але вона не забезпечує видимості внутрішніх елементів системи. Стрілки на контекстній діаграмі вказують на потік даних, керуючих сигналів або фізичних матеріалів (якщо ви визначили «систему», що включає ручні процеси) через границі системи. Навпаки, стрілки на діаграмі варіантів використання просто вказують на зв’язки між акторами та варіантами використання, в яких вони беруть участь, вони не представляють жодного потоку. Як і з усіма іншими формами представлення вимог, усі учасники, для яких ви створюєте моделі, повинні розуміти використовувані терміни.

| ID та назва | ВВ-4 Замовити хімікат | | |
| --- | --- | --- | --- |
| Створив (-ла): | User1 | Дата: | 12.08.2022 |
| Головний актор: | Замовник | Вторинні актори: | Покупець, Хімічний склад, База даних навчання |
| Опис | Замовник зазначає потрібний йому хімікат через назву або id хімікату або через імпорт з графічного редактора формул. Система пропонує замовнику контейнер з хімічного складу або дає можливість замовити хімікат у постачальника. | | |
| Тригер | Замовник робить запит на створення нового замовлення | | |
| Передумови | ПУ-1. Користувач був аутентифікований в системі.  ПУ-2. У користувача є права на замовлення хімікатів.  ПУ-3. Є інтернет-доступ до бд інвентаризації хімікатів. | | |
| Постумови | ПоУ-1. Замовлення зберігається в системі.  ПоУ-2. Замовлення було відправлене до Хімічного складу або до Постачальника. | | |
| Основний потік | **4.0 Замовити хімікат від хімічного складу**  1. Вказати потрібний хімікат.  2. Система надає список наявних на хімічному складі контейнерів із заданим хімікатом.  3. Система надає Замовнику можливість подивитися збережену історію будь-якого контейнера.  4. Замовник вибирає контейнер або перенаправляє замовлення на Постачальника (див. 4.1).  5. Замовник вносить дані для створення замовлення.  6. Система зберігає замовлення та повідомляє Хімічний склад. | | |
| Альтернативні потоки | **4.1 Замовити хімікат у Постачальника**  1. Замовник шукає хімікат в каталогах постачальників (див. 4.1.В1).  2. Система надає список постачальників хімікату, а також ціни, розміри та типи контейнерів.  3. Замовник обирає постачальника, кількість, розмір та тип контейнерів.  4. Замовник вносить дані для створення замовлення.  5. Система зберігає замовлення та повідомляє Покупця. | | |
| Винятки | 4.1.В1 Хімікат недоступний для продажу  1. Система дає повідомлення: Немає постачальників вказаного хімікату.  2. Система запитує Замовника, чи він хоче вибрати інший хімікат (3а) або вийти з системи (4а).  3а. Замовник вибирає інший хімікат.  3б. Система починає основний поток з початку.  4а. Замовник запитує вихід з системи.  4б. Система завершує прецедент. | | |
| Пріоритет | Високий | | |
| Частота використання | Приблизно 5 разів на тиждень для кожного хіміка, 200 разів на тиждень для працівників хімічного складу | | |
| Бізнес-правила | БП-28, БП-31 | | |
| Інше | Система повинна мати можливість імпорту хімікату у вигляді стандартної формули з будь-якого графічного редактору формул, що підтримується системою. | | |
| Припущення | Імпортовані хімічні формули вважаються дійсними. | | |

Передумови повинні бути виконані, перш ніж система зможе почати виконання варіанту використання. Система повинна мати можливість перевірити всі передумови, щоб побачити, чи можливо почати потік виконання. Передумови можуть описувати стан системи (для прецеденту зняття готівки з банкомата банкомат має містити гроші), але вони не описують наміри користувача («Мені потрібна готівка»).

Постумови описують стан системи після успішного виконання варіанту використання. Постумови можуть описувати:

* Щось, що може спостерігати користувач (система відобразила баланс рахунку).
* Фізичні результати (банкомат видав гроші та надрукував квитанцію).
* Змінюється внутрішній стан системи (з рахунку списано суму знятої готівки та комісії за транзакцію).

У багатьох застосунках користувач може об’єднати послідовність варіантів використання в «макро» сценарій використання, який описує більшу задачу. Деякі варіанти використання для веб-сайту електронної комерції можуть бути «Пошук у каталозі», «Додати товар до кошика» та «Оплата товарів у кошику для покупок». Якщо ви можете виконати кожну з цих дій незалежно, вони є індивідуальними варіантами використання. Тобто ви можете мати один сеанс із веб-сайтом, на якому ви щойно шукали в каталозі, другий сеанс, під час якого ви щойно додали товар у кошик без пошуку (можливо, ввівши номер продукту), і третій сеанс, коли ви заплатили за товари в кошику для покупок (це означає, що ваш кошик має зберігатися протягом сеансів входу в систему). Однак ви також можете виконати всі три дії послідовно як один великий варіант використання під назвою «Купити продукт». В описі сценарію використання «Купити продукт» можна просто сказати, що потрібно виконати по черзі кожен із інших трьох випадків використання: «Пошук у каталозі», «Додати товар у кошик для покупок», а потім «Оплатити товари в кошику для покупок».

Щоб цей процес працював, кожен варіант використання повинен залишати систему в стані, який дозволяє користувачеві негайно розпочати наступний варіант використання. Тобто постумови одного варіанту використання повинні задовольняти передумови наступного в послідовності. Подібним чином у застосунку обробки транзакцій, такому як банкомат, кожен варіант використання повинен залишати систему в стані, який дозволяє розпочати наступну транзакцію.

**Типові помилки при створенні прецедентів**

Як і у випадку з будь-якою технікою розробки програмного забезпечення, існує багато способів помилитися, застосовуючи підхід варіантів використання. Існують наступні типові помилки:

* **Забагато варіантів використання.** В такому разі треба змінити рівень абстракції. Не створюйте окремий варіант використання для кожного можливого сценарію. Зазвичай у системі набагато більше прецедентів, ніж бізнес-вимог і характеристик, і при цьому набагато більше функціональних вимог, ніж прецедентів.
* **Дуже складні варіанти використання.** Ви не можете контролювати складність бізнес-завдань, але ви можете контролювати, як ви їх представляєте у варіантах використання. Виберіть один успішний потік для варіанту використання та назвіть його основним. Використовуйте альтернативні потоки для інших логічних гілок, які призводять до успішного завершення прецеденту, і використовуйте винятки для обробки гілок, які призводять до збою. У вас може бути багато альтернатив, але кожна з них буде короткою та легкою для розуміння. Якщо довжина потоку перевищує 10–15 кроків, перевірте, чи справді він описує лише один сценарій. Проте не варто розділяти довгий потік навмання лише тому, що він має багато кроків.
* **Включення дизайну у прецеденти.** Варіанти використання повинні зосереджуватися на тому, що користувачі повинні досягти за допомогою системи, а не на тому, як це буде виглядати в ситемі. Підкресліть концептуальні взаємодії між акторами та системою. Наприклад, скажіть «Система пропонує варіанти» замість «Система відображає випадаючий список». Дизайнерські рішення не повинні впливати на процес пошуку вимог. Використовуйте ескізи екранів та карти діалогів, щоб допомогти візуалізувати взаємодію між актором і системою, якщо це дійсно необхідно, але не специфікацію дизайну.
* **Включення визначень даних у варіанти використання.** Пошук варіантів використання стимулює обговорення щодо того, які елементи даних служать вхідними та вихідними даними під час взаємодії. Деякі автори варіантів використання включають визначення відповідних елементів даних безпосередньо в специфікації варіантів використання. Це ускладнює пошук інформації, оскільки неочевидно, який варіант використання містить кожне визначення даних. Це також може призвести до дублювання визначень, які вже не будуть синхронізовані, коли один прецедент змінено, а інші ні. Зберігайте визначення даних у глосарії або моделі даних.
* **Варіанти використання, які користувачі не розуміють.** Якщо користувачі не можуть пов’язати варіант використання зі своїми бізнес-процесами чи цілями, це неправильно. Напишіть випадки використання з точки зору користувача, а не з точки зору системи, і попросіть користувачів переглянути їх. Дотримуйтеся максимально простих варіантів використання для досягнення чіткої та ефективної комунікації.

**Поради для проведення аналізу вимог**

Аналіз вимог включає уточнення вимог, щоб гарантувати, що всі зацікавлені сторони їх розуміють, і ретельну перевірку на наявність помилок, упущень та інших недоліків. Аналіз включає декомпозицію вимог високого рівня на відповідні рівні деталізації, створення прототипів, оцінку можливостей та узгодження пріоритетів. Мета полягає в тому, щоб розробити вимоги достатньої якості та точності, щоб менеджери могли скласти реалістичні кошториси проекту, а технічний персонал міг продовжити проектування, розробку та тестування.

Варто представляти деякі вимоги кількома способами — наприклад, як у текстовій, так і у візуальній формах або у формі вимог і тестів. Ці різні перспективи виявлять ідеї та проблеми, які не можна отримати лише з однієї перспективи. Численні перегляди також допомагають усім зацікавленим сторонам дійти спільного розуміння — спільного бачення — того, що вони матимуть після релізу продукту.

**Змоделюйте середовище для застосунку.** Контекстна діаграма — це проста модель аналізу, яка показує, як нова система вписується в своє середовище. Вона визначає границі та інтерфейси між системою, що розробляється, і зовнішніми об’єктами, такими як користувачі, апаратні пристрої та інші системи. Карта екосистеми показує різні системи в середовищі, які взаємодіють одна з одною, і характер їхніх взаємозв’язків.

**Створіть макети та технічні прототипи.** Якщо розробники або користувачі не впевнені щодо вимог, створіть прототип — часткову, можливу або попередню реалізацію — щоб зробити концепції та можливості більш зрозумілими. Прототипи дозволяють розробникам і користувачам досягти взаємного розуміння проблеми, що вирішується, а також допомагають перевірити вимоги.

**Проаналізуйте можливість реалізації вимоги.** Бізнес-аналітик повинен співпрацювати з розробниками, щоб оцінити можливість впровадження кожної вимоги за прийнятною ціною та продуктивністю в призначеному робочому середовищі. Це дозволить стейкхолдерам зрозуміти ризики, пов’язані з впровадженням кожної вимоги, включаючи конфлікти та залежності з іншими вимогами, залежність від зовнішніх факторів і технічні перешкоди. Вимоги, які є технічно нездійсненними або надто дорогими для реалізації, можливо, можна спростити для досягнення бізнес-цілей проекту.

**Встановіть пріоритети для вимог.** Важливо визначити пріоритети вимог, щоб переконатися, що команда спочатку впроваджує функції в правильному порядку. Застосуйте аналітичний підхід, щоб визначити відносний пріоритет реалізації характеристик продукту, варіантів використання, історій користувачів або функціональних вимог. На основі пріоритету визначте, який реліз або ітерація проекту міститиме яку характеристику чи набір вимог. Коригуйте пріоритети в рамках проекту в міру того, як пропонуються нові вимоги та змінюються потреби клієнтів, ринкові умови та бізнес-цілі.

**Створіть глосарій даних.** Визначення елементів даних і структур, пов’язаних із системою, містяться в глосарії даних. Це дає змогу всім, хто працює над проектом, використовувати узгоджені визначення даних. У міру розробки вимог глосарій повинен визначати елементи даних із проблемної області, щоб полегшити спілкування між клієнтами та командою розробників.

**Змоделюйте вимоги.** Модель аналізу – це діаграма, яка візуально відображає вимоги, на відміну від текстового представлення списку функціональних вимог. Моделі можуть виявити неправильні, непослідовні, відсутні та зайві вимоги. Такі моделі включають діаграми потоків даних, діаграми сутності-зв’язку, діаграми переходів станів, таблиці станів, діалогові карти, дерева рішень тощо.

**Проаналізуйте інтерфейси між системою та зовнішнім середовищем.** Усі програмні системи підключаються до інших елементів середовища через зовнішні інтерфейси. Інформаційні системи мають інтерфейс користувача і часто обмінюються даними з іншими програмними системами. Вбудовані системи передбачають взаємозв’язки між програмними та апаратними компонентами. Підключені до мережі програми мають комунікаційні інтерфейси. Аналіз цих даних допомагає переконатися, що застосунок, який розробляється, буде без проблем вписуватися в своє середовище.

**Розподіліть вимоги до підсистем.** Вимоги до комплексної системи, який містить кілька підсистем, повинні бути розподілені між різними програмними, апаратними та людськими підсистемами та компонентами.