# Лекція 4

# Управління вимогами

*Анотація: Види вимог: функціональні вимоги, нефункціональні вимоги. Властивості вимог: ясність і недвозначність, повнота і несуперечність, необхідний рівень деталізації, можливість відслідковування, тестування і перевірки, модифікується. Формалізація вимог. Цикл роботи з вимогами.*

## Проблема

Для прикладу розглянемо роботу будівельної компанії. Будівельники будують будинки, нехай різні: багатоповерхові, окремі котеджі, офісні будівлі і ін. – проте, весь цей спектр цілком може охопити одна компанія. Будівельній компанії не доводиться будувати тарілку, що літає, гіперболоїд інженера Гаріна, місяцехід, систему миттєвої телепортації і ін. А розробники ПЗ, багато в чому, знаходяться саме в такій ситуації.

Велике розмаїття систем, які створює одна компанія, одна команда. Хоча зараз і намічаються тенденції до спеціалізації ринку розробки ПЗ, проте, чудасії світової економіки і багато інших причин приводять до того, що строго спеціалізованих компаній не так багато, як хотілося б. Багато галузей відчувають великий дефіцит окремих програмістів і цілих колективів і компаній, що добре розбираються в їх специфіці. Прикладом такої галузі може служити телебачення, де про дану проблему відкрито говорять на засіданнях різних міжнародних співтовариств.

Крім того, ПЗ продовжує проникати у все нові і нові галузі людської діяльності. Сформулювати адекватні вимоги в цьому випадку взагалі виявляється супер важким завданням.

Але навіть якщо мова йде про одну, певну галузь, то відсоток нових, унікальних рис систем, що належать до цієї галузі, високий: за поєднанням призначених для користувача характеристик, за особливостями середовища виконання і вимогами до інтеграції, через розподіленість інформації про вимоги серед працівників компанії-замовника. Все це несе на собі дуже великий відбиток індивідуальності замовника – персонально або його компанії, – сильно пов'язано із специфікою його бізнесу або устаткування, яке використовується в цій галузі.

Крім того, існують труднощі в розумінні між замовником і програмістами, а ще – в мінливості ПЗ (вимоги мають тенденцію змінюватися підчас розробки).

У результаті, далеко не очевидно, що та система, яку хоче замовник, взагалі може бути створена. Важко знайти чорну кішку в темній кімнаті, особливо якщо її там немає. Або те, як зрозуміли і втілили завдання розробники, виявиться зручним та затребуваним на ринку.

Помилки і різночитання, які виникають підчас з’ясування вимог до системи, виявляються одними з найдорожчих. Вимоги – це те початкове розуміння завдання розробниками, яке є основою всієї розробки.

Декілька слів про важкість порозуміння замовника і розробників (Рис. 5.1). Тут позначається великий розрив між програмістами і іншими людьми. По-перше, щоб добре розібратися, якою повинна бути система автоматизації лікарні або система підтримки хімічних експериментів, слід попрацювати у відповідній галузі достатньо часу. Інший підхід – це навчитися якимось іншим способом бачити проблеми даної наочної галузі зсередини. По-друге, позначається специфічність програмування як сфери діяльності. Для більшості користувачів і замовників украй не просто точно сформулювати знання, які необхідні програмістам. На питання, скільки типів аналізів існує у вашій лабораторії, доктор, подумавши, відповідає – 43. І вже потім, випадково, програміст уточнив, а чи немає інших типів аналізів? Звичайно, є, відповів доктор, проте вони трапляються не часто і можуть бути в деякому розумінні, якими завгодно. З першого разу він назвав лише типові. Але, звичайно ж, інформаційна система повинна зберігати інформацію про всі аналізи, які проводяться в лабораторії.

Тепер трохи докладніше про мінливість програмного забезпечення і про її причини.

- Змінюється ситуація на ринку, для якого призначалася система, або змінюються вимоги до системи через перспективи продажу ще неготової системи.
- Потягом розробки виникають проблеми і труднощі, через які підсумкова функціональність змінюються (видозмінюється, урізується).
- Замовник може міняти своє власне бачення системи: чи то він краще розуміє, що ж йому насправді треба, чи то з'ясовується, що він щось упустив із самого початку, чи то з'ясовується, що розробники його не так зрозуміли. Загалом, буває різне, важливо лише, що тепер замовник безумовно хоче іншого.

Годі і говорити, що мінливість вимог під час розробки дуже хворобливо позначається на продукті. Були випадки, коли ще не створену систему відділ продажу починає активно продавати, через що надходить величезний потік додаткових вимог. Всі їх реалізувати у повному обсязі не вдається, як результат, система виявляється набором демо-функціональності.

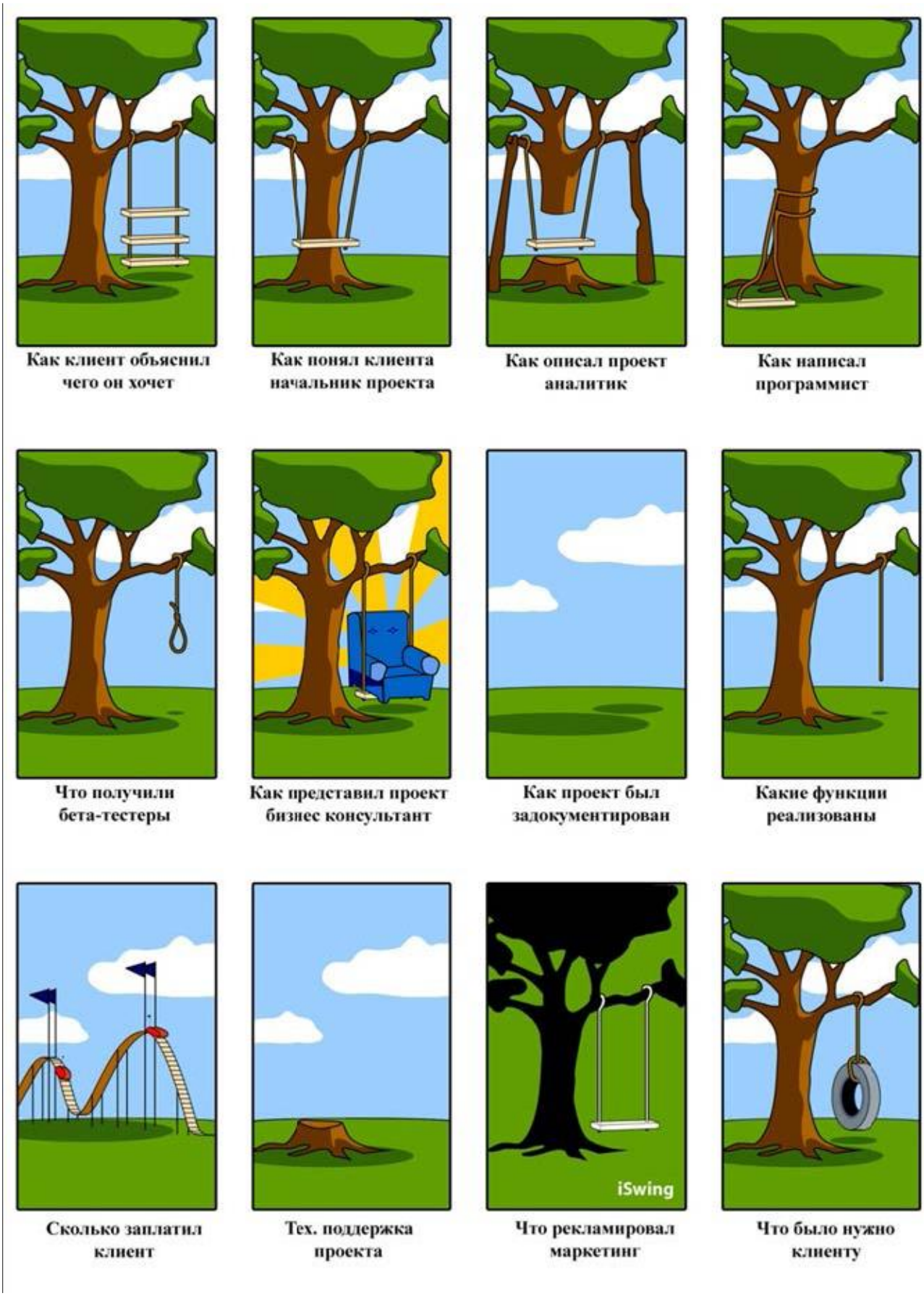


**Рис. 5.1** Результати проекту через непорозуміння замовника і розробників

## Види і властивості вимог

Розділимо вимоги на дві великі групи – **функціональні** і **нефункціональні**.

**Функціональні вимоги** є детальним описом поведінки і сервісів системи, її функціоналу. Вони визначають те, що система повинна уміти робити.

**Нефункціональні вимоги** не є описом функцій системи. Цей вид вимог описує такі характеристики системи, як надійність, особливості постачання (наявність інсталятора, документація), певний рівень якості (наприклад, для нової Java-машини це буде значити, що ПЗ має задовольняти вимогам щодо набору тестів, які підтримуються компанією Sun). Сюди ж можуть відноситися вимоги до засобів і процесів розробки системи, вимоги до перенесення, відповідності стандартам і так далі. Вимоги цього виду часто відносяться до всієї системи в цілому. На практиці, особливо початківці, часто забувають про деякі важливі нефункціональні вимоги.

Нефункціональні вимоги — це вимоги до програмного забезпечення, які задають критерії для оцінки якості його роботи.

Нефункціональні вимоги можна поділити на дві категорії:

- покращення (безпека, надійність, швидкодія, зручність у використанні ...)
- вдосконалення (масштабування, відновлюваність ...) властивостей системи

Види нефункціональних вимог:

- Вимоги до інтерфейсу (Interface Requirements)
- Апаратні інтерфейси (Hardware Interfaces) — апаратні інтерфейси, які необхідні для підтримки системи, включаючи логічну структуру, фізичні адреси і очікувану поведінку.
- Інтерфейси ПЗ (Software Interfaces) — інтерфейси програмного забезпечення з якими аплікація повинна взаємодіяти.
- Комунікаційні інтерфейси (Communications Interfaces) — інтерфейси для комунікацій (взаємодії) з іншими системами та/або пристроями.
- Апаратні та програмні вимоги (Hardware/Software Requirements) — опис апаратної та програмної платформ, необхідних для роботи (і підтримки) системи.
- Операційні вимоги (Operational Requirements)
- Безпека та конфіденційність (Security and Privacy)
- Надійність (Reliability)
- Відновлювальність (Recoverability)
- Продуктивність (Performance)

- Потенціал (Capacity)
- Збереження даних (Data Retention)
- Керування помилками (Error Handling)
- Правила перевірки (Validation Rules)
- Узгоджені стандарти (Convention Standards)

На відміну від нефункціональних вимог, які визначають якою система повинна бути, функціональні вимоги визначають, що система повинна робити. Функціональні вимоги до програмного забезпечення визначаються на першій стадії процесу розробки ПЗ — на етапі аналізу вимог.

**Сформулюємо ряд важливих властивостей вимог.**

- **Ясність, недвозначність** – однозначність розуміння вимог замовником і розробниками. Часто цього важко досягти, оскільки кінцева формалізація вимог, яка виконана з погляду потреб подальшої розробки, є важкою для сприйняття замовником або фахівцем наочної галузі, котрі повинні проінспектувати адекватність формалізації.
- **Повнота і несуперечність**.
- **Необхідний рівень деталізації**. Вимоги повинні мати чітко визначений рівень деталізації, стиль опису, спосіб формалізації:
  - або це опис властивостей наочної галузі, для якої призначене ПЗ;
  - або це технічне завдання, яке додається до контракту;
  - або це проектна специфікація, яка повинна бути потім уточнена підчас детального проектування.

  Важливо також ясно бачити і розуміти тих, для кого даний опис вимог призначений, інакше не уникнути непорозуміння і подальших труднощів. Адже в розробці ПЗ задіяно багато різних фахівців – інженерів, програмістів, тестувальників, представників замовника, можливо, майбутніх користувачів – і всі вони мають різну освіту, професійні навички і спеціалізацію, часто спілкуються різними мовами. Тут також важливо, щоб вимоги були максимально абстрактні і незалежні від реалізації.
- **Можливість відслідковування** – важливо бачити ту або іншу вимогу в різних моделях, документах, нарешті, в коді системи. Іноді виникають питання типу – "Хто знає, чому ми вирішили, що такий-то модуль повинен працювати саме таким чином?". Можливість відслідковування функціональних вимог досягається шляхом їх розбиття на окремі, елементарні вимоги, надання їм ідентифікаторів і створення моделі трасування, яка в ідеалі повинна сягати програмного коду. Важливо знати, де потрібно змінити код, якщо певна вимога змінилася. На практиці формальна можливість цілковитого відслідковування є важко

досяжною, оскільки логіка і структура реалізації системи можуть сильно відрізнятись від останніх для моделі вимог. Як результат, одна вимога виявляється сильно "розмазаною" за програмним кодом, а та або інша ділянка коду може впливати на багато вимог. Але слід прагнути можливості відслідковування стану вимог, розумно поєднуючи формальні і неформальні підходи.

- **Можливість тестування та перевірки** – необхідно, щоб існували способи тестування і перевірки даної вимоги. Причому, важливі обидва аспекти, оскільки часто перевірити замовник може, а ось тестувати дану вимогу дуже важко, або неможливо з причини обмеженості доступу до оточення системи для команди розробника (наприклад, з міркувань безпеки). Отже, необхідні процедури перевірки-виконання тестів, проведення інспекцій, проведення формальної верифікації частини вимог і ін. Потрібно також визначати "щабель" якості (чим вище якість, тим воно дорожче коштує!), а також критерії повноти перевірок, щоб керівники проекту, які їх виконують, чітко усвідомлювали, що саме перевірене, а що ще ні.
- **Можливість модифікації**. Визначає процедури внесення змін у вимоги.

## Варіанти формалізації вимог

Загалом, вимоги як такі – це деяка абстракція. У реальній практиці вони завжди існують у вигляді якогось уявлення – документа, моделі, формальної специфікації, списку і так далі. Вимоги важливі як такі, тому що існують у вигляді розуміння розробниками потреб замовника і майбутніх користувачів створюваної системи. Але оскільки в програмному проекті багато різних аспектів, видів діяльності і фаз розробки, то це розуміння може набувати дуже різних форм. Кожне унаочнення вимог виконує певне завдання:

- служить "містком", фіксацією угоди між різними групами фахівців,
- використовується для оперативного управління проектом (відслідковується, в якій фазі реалізації знаходиться та або інша вимога, хто за неї відповідає і ін.),
- використовується для верифікації і модельно-орієнтованого тестування.

І у першому, і в другому, і в третьому прикладі ми маємо справу з вимогами, але формалізовані вони будуть по-різному.

Отже, формалізація вимог в проекті може бути дуже різною – це залежить від його величини, прийнятого процесу розробки, які інструментальні засоби використовуються, а також тих завдань, які вирішують формалізовані вимоги. Більш того, може існувати паралельно декілька формалізацій для вирішення різних завдань. Розглянемо варіанти.

1. **Неформальна постановка вимог в листуванні електронною поштою.** Добре працює в невеликих проектах, у випадку залучення замовника в розробку (наприклад, команда виконує субпідряд). Може також застосуватись, коли є взаєморозуміння між замовником і командою, тобто зайві формальності не потрібні. Проте, електронні листи в такій ситуації часто опиняються важливими документами – важливо уміти вести ділове листування, підводити підсумки, зберігати важливі листи і користуватися ними у разі розбіжностей. Важливо також вчасно зрозуміти, коли такий спосіб перестає працювати і необхідні більш формальні підходи.

2. **Вимоги у вигляді документа** – опис наочної галузі і її властивостей, технічне завдання як додаток до контракту, функціональна специфікація для розробників і так далі.

3. **Вимоги у вигляді графа** із залежностями в одному із засобів підтримки вимог (IBM Rational RequisitePro, DOORS, Borland CALIBERRM і деякі інші). Таке уявлення зручне у разі частої зміни вимог, для відслідковування виконання вимог, у випадку організації «прив'язки» до вимог завдань, людей, тестів, коду. Важливо також, щоб була можливість легко створювати такі графи з текстових документів, і навпаки, створювати презентаційні документи за такими графами.

4. **Формальна модель вимог** для верифікації, модельно-орієнтованого тестування і так далі.

Отже, кожен спосіб подання вимог повинен відповідати на питання: хто споживач, користувач цього подання, як саме, з якою метою це подання використовується.

**Деякі помилки підчас документування вимог**.

Перерахуємо деякі помилки, які зустрічаються підчас створення технічних завдань і інших документів з вимогами.

- Опис можливих рішень замість вимог.
- Нечіткі вимоги, які не допускають однозначну перевірку, залишають недомовленості, мають відтінок порад, обговорень, рекомендацій: «Можливо, має сенс реалізувати також...», і так далі.
- Ігнорування аудиторії, для якої призначене подання вимог. Наприклад, якщо специфікацію складає інженер замовника, то часто зустрічається надлишок інформації про устаткування, з яким повинна працювати програмна система, відсутній глосарій термінів і визначень основних понять, використовуються численні синоніми і так далі. Або присутній дуже сильний ухил у бік програмування, що робить дану специфікацію незрозумілої всім непрограмістам.
- Відсутність важливих аспектів, які пов'язані з нефункціональними вимогами, зокрема, інформації про оточення системи, про

терміни готовності інших систем, з якими повинна взаємодіяти ця система. Останнє трапляється, наприклад, коли ця програмна система є частиною більшого проекту. Типовою проблемою підчас створення програмно-апаратних систем є ситуація, коли апаратура вчасно не постачається і ПЗ неможливо тестувати, а в термінах і вимогах це не передбачено.

## Цикл роботи з вимогами

У зведенні знань щодо програмної інженерії SWEBOK визначаються такі види діяльності для роботи з вимогами.

- **Виділення вимог** (***requirements elicitation***), націлене на виявлення всіх можливих джерел вимог і обмежень на роботу системи та отримання вимог з цих джерел.
- **Аналіз вимог** (***requirements analysis***), метою якого є виявлення і усунення суперечностей і неоднозначностей у вимогах, їх уточнення і систематизація.
- **Опис вимог** (***requirements specification***). В результаті цієї діяльності вимоги повинні бути оформлені у вигляді структурованого набору документів і моделей, які можуть систематично аналізуватися, оцінюватися з різних позицій. Як результат, такий набір документів повинен бути затверджений як офіційне формулювання вимог до системи.
- **Валідація вимог** (***requirements validation***), яка вирішує задачу оцінки зрозумілості сформульованих вимог і їх характеристик, що є необхідними для розробки на їх основі ПЗ, в першу чергу, несуперечності і повноти, а також відповідності корпоративним стандартам на технічну документацію.