АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ЛЕКЦІЇЇ 1-2. ВИЗНАЧЕННЯ ВИМОГ ДО ПРОГРАМНИХ СИСТЕМ

Кожна програмна система – це перетворювач, функцією якого є визначене оброблення даних і вивід отриманих результатів. З метою побудови програмної системи до неї, насамперед, формулюються вимоги до умов виконання функції і обробки даних. Ці вимоги є предметом практичного контракту між замовником і розробником системи [1].

У загальному випадку під *вимогами* до ПС розуміють властивості, які повинна мати система для виконання запропонованих замовником функцій. Прикладами таких функцій можуть бути бізнес-функції, документообіг, керування даними і структурою інформації, що необхідна для прийняття системних рішень, та ін. У ядрі знань SWEBOK викладено основні концепції й особливості інженерії вимог, які подано на рис. 1.1.

Software Engineering Body of Knowledge (SWEBOK) — це документ, що готує комітет Software Engineering Coordinating Committee зі спільнотою IEEE Computer Society. Призначення SWEBOK — в об'єднанні знань з інженерії програмного забезпечення.[1]

Це документ є одним з трьох документів, що були створені у співпраці IEEE-CS та ACM. Цими документами бажають забезпечити наступне:

- визначити чіткий необхідний набір знань та рекомендовані практики;
- визначити етичні та професійні стандарти;
- визначити навчальну програму для студентів, аспірантів та тих, хто продовжує навчання.

В редакції 2004 року визначаються десять областей знань в контексті програмної інженерії:

- Вимоги до ПЗ, англ. Software requirements.
- Проектування, англ. Software design.
- Конструювання, англ. Software construction.
- Тестування, англ. Software testing.
- Супроводження, англ. Software maintenance.
- Керування конфігурацією, англ. Software configuration management.
- Керування проектами, англ. Software engineering management.
- Процеси програмної інженерії, англ. Software engineering process.
- Засоби та інструменти, англ. Software engineering tools and methods.
- Якість ПЗ, англ. Software quality.

На даний момент співтовариством розробляється нова, доповнена версія, що включає 15 областей:

- Software Requirements - вимоги до ПЗ.
- Software Design - проектування ПЗ.
- Software Construction - конструювання ПЗ.
- Software Testing - тестування ПЗ.
- Software Maintenance - супровід ПЗ.
- Software Configuration Management - управління конфігурацією.
- Software Engineering Management - управління IT проектом.
- Software Engineering Process - процес програмної інженерії.
- Software Engineering Models and Methods - моделі та методи розробки.
- Software Engineering Professional Practice - опис критеріїв професіоналізму та компетентності.
- Software Quality - якість ПЗ.
- Software Engineering Economics - економічні аспекти розробки ПЗ.

- Computing Foundations - основи обчислювальних технологій, застосовних у розробці ПЗ.
- Mathematical Foundations - базові математичні концепції і поняття, застосовні в розробці ПЗ.
- Engineering Foundations - основи інженерної діяльності.

Також SWEBOK визначає дисципліни, що відіграють велику роль в програмній інженерії:

- Комп'ютерна інженерія
- Комп'ютерні науки
- Менеджмент
- Математика
- Контроль якості
- Ергономіка ПЗ
- Системне адміністрування

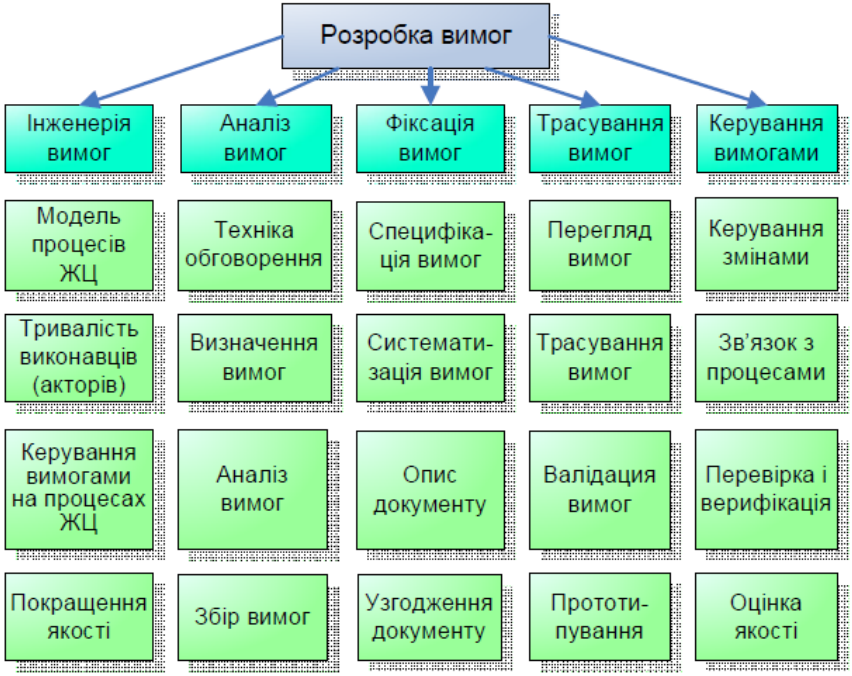

Рис. 1.1. Основні розділи розробки вимог

Найпоширеніший інструмент інженерії вимог – це UML, у якому вимоги визначаються й уточнюються через подання можливих варіантів використання ПС (use case), що трансформуються у проектні рішення і архітектуру системи. Розглянемо загальні й спеціалізовані (зокрема, об’єктно-орієнтовані) підходи до розробки вимог до системи у контексті розділів, поданих на рис. 1.1.

1.1. Загальні підходи до визначення вимог

Визначення вимог є нетривіальною задачею і проводиться, як правило, шляхом обговорення поглядів замовника на систему з майбутніми її розробниками.

Замовник висловлює свої потреби і представляє погляди щодо автоматизації

функцій і задач системи, які далі набувають формулювання у вигляді різнопланових вимог до ПС, класифікація яких подається нижче.

1.1.1. Класифікація вимог

Дотепер ще відсутні загальноприйняті терміни, якими користуються спеціалісти для опису вимог замовника до ПС, а саме, функціональних, системних,
технологічних. У різних тематичних джерелах наведені різні визначення поняття вимог, виходячи з особистих поглядів замовників на програмний продукт, який потрібно побудувати.

У ряді публікацій формування вимог до ПС розглядається як певна ділова гра, під час якої виявляються інтереси зацікавлених у розробці ПС осіб, правила реалізації цих інтересів у конкретному продукті. При цьому висловлюються різного
роду претензії й обмеження на властивості і способи забезпечення вимог для отримання кінцевого програмного продукту. Отже, зрозуміло що нема формалізованих методів збирання й опису вимог, а також відсутнє загальноприйняте визначення самого поняття *вимога*. Наведемо тлумачення цього слова з джерел [1–3].

Вимоги – це твердження про функції й обмеження системи.
Вимоги – це властивості, які повинен мати продукт, щоб являти бути собою цінним для свого користувачів.
Вимоги – це специфікація того, що і як повинно бути реалізовано.__

Відповідно до міжнародного глосарію з термінології комп'ютерної науки

вимога містить у собі опис:

1) умов або можливостей, необхідних користувачеві для вирішення поставлених проблем або для досягнення цілей;

2) функцій і обмежень, які повинна мати система або системні компоненти, щоб виконати контракт замовника на розробку;

3) положень і регламенту, встановлених використаними стандартами і відображених у специфікаціях або інших формальних документах на систему;

4) умов, можливостей і обмежень середовища, необхідних для проектування і виконання системи.

Розглянемо основні типи вимог.

Вимоги до продукту охоплюють умови користувачів щодо зовнішнього поводження системи і погляди розробників на деякі параметри системи. Термін користувач стосується осіб, зацікавлених у створенні системи.

Вимоги до ПЗ є такі: системні, функціональні і нефункціональні вимоги.

Вимоги користувачів (user requirements) задаються умовами досягнення цілей і задач, віддзеркалюють вимоги споживачів до спектра розв'язуваних майбутньою системою задач. Вони подаються як текстовий опис або сценарії, прецеденти, таблиці «подія-відгук» тощо.

Системні вимоги (system requirements) визначають зовнішні умови виконання системних функцій і обмежень на створення продукту, а також вимоги до опису програмно-апаратних підсистем. Системні вимоги накладають обмеження на архітектуру системи, засоби її візуального подання і функціонування. Для опису системних вимог використовують спеціальні шаблони і форми, що допомагають уявити вхідні і вихідні дані й автоматизувати ці вимоги.

Вимоги до атрибутів якості (quality attributes) – це деякі обмеження на властивості функцій або системи, важливі для користувачів або розробників. Наприклад, переносність, цілісність, стійкість системи до збоїв.

Функціональні вимоги – це перелік функцій або сервісів, які повинна надавати система, а також обмежень на дані і поводження системи при їхньому виконанні.

Специфікація функціональних вимог (software requirements specification) – опис функцій та їхніх властивостей, які не містять у собі протиріч і виключень.

Нефункціональні вимоги визначають умови виконання функцій (наприклад, захист інформації у БД, аутентифікація доступу до ПС тощо) у середовищі, що безпосередньо не пов'язані з функціями, а відбивають потреби користувачів щодо їх виконання. Ці вимоги характеризують принципи взаємодії із середовищами або іншими системами, а також визначають показники часу роботи, захисту даних і досягнення якості з урахуванням рекомендацій використовуваного стандарту.

Вони можуть встановлюватися як числові значення (наприклад, час чекання відповіді, кількість клієнтів, що обслуговуються і ін.) у різних одиницях виміру, включаючи, наприклад, ймовірність (значення ймовірності безвідмовної роботи системи – показника її надійності). Для більшості сучасних ПС вимоги складаються з умов й обмежень типу:

– конфіденційність, безпека і захист даних;

– відмовостійкість;

– одночасність доступу користувачів до системи;

– час чекання відповіді при звертанні до системи (продуктивність);

– склад виконуваних функцій системи (запуск, швидкість реакції й ін.);

– положення стандартів з виконання сформульованих вимог.

Дані вимоги визначаються і формалізуються аналітиками на процесі аналізу і проектування структури системи. Так, у випадку вимог з безпеки функціонування системи, в системі виділяються категорії користувачів, що мають право доступу до тих або інших функцій (програмних компонентів) або даних, та передбачаються додаткові функції системи з перевірки доступу (санкціонований доступ до них чи ні). Якщо потрібно обмежити доступ до конкретних даних (наприклад, до окремих записів, полів у таблиці), то в системі може передбачатися, наприклад, мандатний захист. Для захисту всієї системи від несанкціонованого доступу користувачі реєструються і проходять автентифікацію для роботи із системою.

Для відновлення і збереження резервних копій БД, архівів баз даних аналізуються можливості СУБД і способи забезпечення необхідного рівня безперебійної роботи системи, правил доступу авторизованих користувачів і заходів боротьби з різними загрозами, що надходять ззовні від користувачів, які не мають прав доступу до деяких або до всіх даних системи.

До вихідного продукту пред'являються нефункціональні вимоги до:

– застосування (якість інтерфейсу, продукту й ін.);

– продуктивності (пропускна здатність, час реакції й ін.);

– надійності виконання (без помилок і відмов);

– зовнішніх інтерфейсів, за якими виконується взаємодія з іншими компонентами або підсистемами.

Опис усіх видів вимог проводиться з урахуванням стандартів, наприклад, стандарту з якості **ISO/IEC ДСТУ 9126** і стандартизованого понятійного і термінологічного довідника, що містить у собі загальноприйняті терміни щодо структури ПрО і призначення функцій системи. Специфікація вимог відображає принципи взаємодії проектованої системи з іншими середовищами, платформами і загальносистемним забезпеченням (БД, СКБД, мережі та ін.).

Формування документа зі специфікаціями вимог завершується на процесі проектування архітектури, після чого він узгоджується з замовником системи і використовується як керування дій при виконанні задач розробки програмного продукту на процесах ЖЦ і отриманні готового продукту. При виявленні на них неузгоджених вимог, проводиться їхнє уточнення і, відповідно, вносяться зміни у деякі задачі процесу розроблення системи або характеристики продукту.

1.1.2. Аналіз і збирання вимог

У сучасних технологіях процес ЖЦ, у якому фіксуються вимоги до розробки системи, є визначальним для задання функцій, термінів і вартості робіт, а також показників якості, які необхідно досягти в процесі розроблення. Виявлення вимог проводиться під час обговорення проекту, аналізу особливостей предметної області і визначення підходів до її проектування на процесах ЖЦ.

Вимоги відбивають потреби людей (замовників, користувачів, розробників), зацікавлених у створенні ПС. Замовник і розробник спільно обговорюють проблеми проекту, збирають вимоги, проводять їхній аналіз, перегляд і визначають необхідні обмеження.

Обговорення проекту системи відбувається з метою вивчення думки і вироблення перших висновків щодо доцільності виконання проекту і прогнозування реальності його виконання в заданий термін і за кошти, що дає замовник. Природно, особа, яка замовила проект системи, бажає отримати від розробника набір необхідних послуг, за якими будуть звертатися різні категорії користувачів: оператори, менеджери, фахівці у ПрО.

Розробники системи можуть оцінити можливість реалізації послуг у проекті системи, що замовляється, у заданий термін і бюджет. Серед розробників призначаються головний аналітик, відповідальний за вимоги до системи, і головний програміст, відповідальний за їхню реалізацію. Вони узгоджують вимоги і визначають сферу дії проекту на спільних переговорах із замовником з метою уточнення наступних питань:

– спектра проблем ПрО, при вирішенні яких будуть визначатися послуги системи;

– функціонального змісту послуг;

– регламенту операційного обслуговування системи тощо.

В обговоренні вимог до системи беруть участь:

– представники замовника з декількох професійних груп;

– оператори, що обслуговують систему;

– аналітики і розробники майбутньої системи.

Погоджена сфера дій у проекті дає можливість оцінити необхідні інвестиції в проект, заздалегідь визначити можливі ризики і здатності розробників щодо виконання проекту. Підсумком обговорення проекту може бути рішення про розгортання реалізаційних робіт на проекті або відмови від нього.

Аналіз вимог починається після обговорення проблематики проекту. Серед обговорюваних вимог можуть виявитися:

– неочевидні або не однаково важливі, які були взяті з застарілих джерел і документів замовника;

– різні типи умов, що відповідають різним рівням деталізації проекту і потребують застосування методів керування ними;

– постійно змінювані або уточнювані, залежно від унікальних властивостей або значень;

– складні за формою і змістом, тяжкі для узгодження їх із замовником.

Розробники вимог повинні мати відповідні знання в даній предметній області і вміти здійснювати:

– аналіз проблем, задач предметної області, а також потреб замовника і користувачів системи,

– виявлення функцій системи, що мають бути реалізовані в проекті,

– внесення змін в окремі елементи вимог у процесі їх виконання.

У вимогах до ПС, крім проблем системи, формулюються реальні потреби замовника щодо функціональних, операційних і сервісних можливостей майбутньої системи. Результати дії дослідження й аналізу предметної області фіксуються в документі з опису вимог і в договорі між замовником і виконавцем проекту.

Помилки через нечіткі або неоднозначні формулювання вимог можуть призвести до того, що виготовлена система не буде задовольняти замовника. Тому на процесах розробки вимоги повинні постійно уточнюватися і знову затверджуватися замовником. В окремих випадках внесені зміни у вимоги можуть обумовити необхідність перепроектування окремих частини або всієї системи в цілому. Відповідно до статистики, частка помилок у постановці вимог і у визначенні задач системи перевищує частку помилок, що допускається під час кодування системи. Це обумовлюється суб'єктивним характером процесу формулювання вимог і відсутністю способів їхньої формалізації. У США, наприклад, щорічно витрачається до 82 млрд.дол. на проекти, визнані після реалізації такими, що не відповідають вимогам замовників.

Існуючі стандарти (**ДСТУ 34.601–90 і ДСТУ 34.201–89**) з розробки вимог до автоматизованої системи (АС) і відповідної документації фіксують результати створення програмного, технічного, організаційного та інших видів забезпечення автоматизованих систем на процесах ЖЦ.

Збирання вимог.

Джерелами відомостей для збирання вимог є:

– мета і задачі проекту, що формулює замовник майбутньої системи, і які повинні осмислюватися розробниками;

– колектив, який виконує реалізацію функцій системи.

Вивчення і фіксація реалізованих функціональних можливостей у діючій системі є підґрунтям для накопичення досвіду для формулювання нових вимог до неї. При цьому необхідно відокремлювати нові вимоги до системи від старих вимог, щоб не повторити невдалі розв’язки щодо старої системи в новому її виконанні.

Вимоги до системи формулюються замовником у термінах понять його предметної області з урахуванням відомих словників, стандартів, існуючих умов середовища функціонування майбутньої системи, а також трудових і фінансових ресурсів, виділених на розробку системи.

Методи збирання вимог такі:

– інтерв'ю з виразниками інтересів замовника системи;

– вивчення прикладів можливих варіантів виконання функцій, ролей відповідальних осіб, які пропонують ці варіанти, або тих, що взаємодіють із системою при її функціонуванні;

– спостереження за роботою діючої системи для відокремлення властивостей, що обумовлені кадровими аспектами.

Зовнішні і внутрішні аспекти вимог пов’язують з характеристиками якості і відносяться до властивостей створюваного продукту, а саме, функцій системи, її призначення і виконання в заданому середовищі. На прикінці користувач очікує досягнення максимального ефекту від застосування вихідного продукту та орієнтується на його кінцеву експлуатаційну якість.

Отримання зовнішніх і внутрішніх характеристик якості досягається спеціально розробленими методами з виконання процесів ЖЦ. Внутрішні характеристики, які досягаються в ході ЖЦ, позначаються на зовнішніх показниках і використовуються при оцінюванні робочих та кінцевих продуктів ПС.

Остаточно сформульовані вимоги – основа для підпису контракту між замовником і розробником системи.

1.1.3 Інженерія вимог

Інженерна дисципліна аналізу і документування вимог передбачає планування і перетворення запропонованих замовником вимог до системи на опис вимог до ПЗ, їх специфікацію і валідацію. Вона базується на моделях процесів розроблення вимог, діях акторів і керуванні поступовим перетворенням вимог до проектних рішень і опису компонентів у мові програмування.

Модель процесу визначення вимог – це схема процесів ЖЦ, що виконуються від початку проекту і доти, поки не будуть визначені і погоджені вимоги.

Керування вимогами до ПЗ полягає в плануванні і контролі формування вимог, заданих на основі проектних рішень, у перетворенні їх на специфікації компонентів системи.

Якість і процес поліпшення вимог – це методи досягнення і перевірки характеристик і атрибутів якості (надійність, реактивність та ін.), які повинна мати система і ПЗ, у процесах ЖЦ і під час закінчення розробки продукту.

Керування вимогами до системи – це планування і керування формуванням вимог на всіх процесах ЖЦ, а саме, керування змінами вимог, відновлення їхнього джерела й уточнення вимог. Невід'ємна складова процесу керування – трасування вимог, що полягає у відстеженні правильності завдання і реалізації вимог до системи і ПЗ на процесах ЖЦ і зворотного процесу звіряння ПЗ із заданими вимогами.

Основні задачі керування вимогами це:

– розроблення атрибутів вимог,

– керування варіантами вимог,

– керування ризиками, що виникають при неточному визначенні вимог,

– контроль статусу вимог, вимірювання зусиль при формуванні вимог;

– реалізація вимог на процесах ЖЦ.

Розроблення і керування вимогами зв'язана з іншими областями знань (рис.1.2).

Рис. 1.2. Керування вимогами і зв'язок із задачами SWEBOK

Планування робіт на проекті стосується питань організації інтеграції компонентів, керування ризиками, версіями системи, на які впливають задані вимоги та їхні зміни.

1.1.4. Фіксація вимог

Початковий процес розроблення ПС – збирання вимог, який завершується формуванням списку вимог до системи, що називається у вітчизняній практиці технічним завданням. Фіксація вимог (Requirement Capturing) у технічному завданні обумовлена потребою замовника зафіксувати і одержати задані ним властивості у реалізованій системі. При цьому передбачається специфікація, верифікація і валідація вимог на правильність, відповідність і повноту.

Специфікація вимог до ПЗ – це формалізований опис функціональних, нефункціональних і технічних вимог, вимог до характеристик якості, до структури ПЗ і принципів взаємодії з його компонентами.

Приклад. Скласти вимоги до облікової і статистичної функцій ПЗ системи обробки даних.

Згідно з стандартом **ДСТУ 34.601–92 («Розробка АС»)** функціональні вимоги до ПЗ даної системи можливо представити так:

– система повинна мати 12 функцій, з них 8 облікових та 4 статистичних;

– кожна функція повинна бути ретельно реалізована, бути коректною і повинна давати точні результати;

– дані для функцій подаються в табличному вигляді і зберігаються в БД СКБД Oracle, їхній обсяг – 10000 записів на рік;

– дані повинні контролюватися і бути захищені від несанкціонованого доступу до БД тощо.

Дані вимоги відносять до характеристик функціональності системи. Після її реалізації повинні бути перевірені функції на відповідність установленим вимогам, діючим нормам і стандартам. Оцінка функцій виконується після їхньої валідації, верифікації та реалізації. Як оцінки використовують метрики (коректність, точність, повнота тощо) для перевірки різних аспектів реалізації функцій. Ці метрики наведені у першій, а їхні формули – у другій колонках табл.3.1. У ці формули підставлені значення, яким надано кількісний опис у специфікації вимог.

На основі отриманих метрик розв’язується задача правильної реалізації вимог до ПЗ або до всієї програмної системи.

Таблиця 1.1. Перевірка реалізації функцій системи

Назва метрики	Опис результату перевірки
Повнота реалізації функції	$\Psi = 1 - N/M$ N – число нереалізованих (пропущених) функцій M – число функцій в описі вимог
Коректність реалізації функцій	$\varphi = 1 - N/M$ N – число некоректно розроблених функцій M – число функцій в описі вимог
Точність реалізації функцій	$\delta = N/T$ N – число відхилених результатів виконання функцій T – час використання компонентів функції
Ретельність реалізації функцій	$\nu = N/M$ N – число функцій, для яких специфікації вимог були точно реалізовані, M – число функцій, для яких вимоги до точності були встановлені в специфікації вимог

Здатність до обміну даних	$\mu = N/R$ N – число даних, що беруть участь в обміні даних із БД, R – загальне число форматів даних, що беруть участь в обміні із БД
Контроль доступу до даних у БД	$k = N/M$ N – число несанкціонованих операцій, M – число нелегальних операцій, наведених в описі вимог
Точність обчислення даних	$k = N/M$ N – число елементів даних, для яких забезпечений рівень точності обчислень, M – число елементів даних, для яких у специфікації встановлений рівень точності обчислень
Ступінь контролю доступу	$k = N/M$ N – число вимог до контролю доступу стосовно специфікації вимог, M – число вимог до контролю доступу, встановлених у специфікаціях вимог
Функціональна відповідність	$\varphi = N/M$ N – число коректно розроблених компонентів, до яких пред'являються функціональна відповідність, M – загальне число компонентів, до яких установлені норми і правила відповідності

Валідація вимог – це перевірка вимог для переконання, що вони визначають саме дану систему. Замовник проводить експертизу зафіксованого варіанта вимог для того, щоб розробник міг далі виконувати його проектування. Один з методів валідації – прототипування, тобто швидке відпрацювання окремих вимог на конкретному інструменті, аналіз масштабу виконання і зміни вимог, вимірювання функціональності і вартості системи, а також визначення зрілості процесів визначення вимог.

Верифікація вимог – це процес перевірки правильності специфікації вимог на їхню відповідність стандартам і функціям системи. Внаслідок перевірки вимог створюється остаточний і погоджений документ, що встановлює повноту і коректність вимог до ПЗ, а також можливість продовжити його проектування.

1.1.5. Трасування вимог

Одна з головних проблем збирання вимог – їхня зміна. Вимоги створюються ітераційно шляхом постійного спілкування представників замовників з аналітиками і розробниками майбутньої системи з метою виявлення необхідних потреб. Вимоги змінюються в міру уточнення функцій і задач, умов їхнього визначення на процесі укладання договору на створення системи і, зрештою, відповідають поглядам замовника на систему [4].

Одним з інструментів установлення залежності між сформульованими вимогами та їхніми змінами є трасування, тобто розвиток і обробка вимог із простежуванням ідентифікованих зв'язків, що повинні бути зафіксовані за двома напрямками – від потреб до робочих продуктів, і навпаки (рис. 1.3.). На процесі розроблення вплив змін у вимогах поширюється в першому напрямку, від потреб до робочих продуктів, а на процесі експлуатації – у зворотному напрямку. Під час цього виявляють причини виникнення різних неточностей, а потім виносять рішення про трасування в одному з наведених напрямків.

Рис. 1.3. Схема трасування вимог

Якщо після розроблення деякого робочого продукту виникає потреба в зміні окремих вимог або необхідність простежити за проходженням внесених вимог в одному з напрямків даної схеми трасування, то уточнюють зв'язки між окремими вимогами й елементами робочих продуктів. У випадку трасування вимог від продукту здійснюється рух у зворотному напрямкі, тобто рух до вимог шляхом з'ясування правильності написання рядків коду продукту і відповідності їх окремим атрибутам вимог. Трасування в обох напрямках допомагає знайти незаплановані, але реалізовані, деякі функції або фрагменти програм, які не відповідають вимогам, і, навпаки, виявити нереалізовані вимоги до функціональності. Взаємозв'язки і залежності між окремими вимогами можуть зберігатися в таблиці трасування і видалятися або модифікуватися за різних змін.

Трасування базується на специфікаціях усіх зв'язків між елементами вимог або обмежується описами функцій, ситуацій, контексту і можливих рішень.

Трасуванню піддаються:

– вимоги, що змінюються при їхньому формуванні;

– деякі деталі виконання функцій у робочому продукті системи, що не передбачалися, але з'явилися в зв'язку з практичною ситуацією, що виникла;

– зв'язки між різними моделями процесу проектування системи на ЖЦ і прийняті рішення про необхідність зміни вимог через виявлені недоліки в проміжному або робочому продукті;

– інформація про узгодження атрибутів вимог на різних рівнях даної схеми трасування та її матриць;

– системні вимоги, наприклад, до повторного використання готових компонентів;

– результати тестування, за якими можна визначити найбільш ймовірні частини коду, що вимагають зміни для виправлення виявлених дефектів.

У матриці вимог у рядках указані вимоги користувача, а у стовпцях – функціональні вимоги, елемент проектування, варіант версії й ін. У цих стовпцях заповнюються дані про ступінь виконання системних вимог на кожному об'єкті створюваного продукту. Механізм посилань у таблиці дозволяє перевіряти зв'язки між об'єктами системи. Процедура трасування передбачає:

– вибору елемента вимог з матриці, за яким буде відбуватися простежування на процесах ЖЦ;

– складання списку питань, за якими на кожному процесі ЖЦ перевіряються зв'язки при реалізації вимог, і, якщо змінюється будь яка ланка в ланцюжку вимог (рис.1.3), то може модифікуватися процедура розроблення цього елемента на наступному процесі ЖЦ;

– проведення моніторингу кожної вимоги на відповідність прийнятому плану;

– уточнення ресурсів проекту при зміні вимоги або елемента проекту.

Умова прийняття рішення про можливі модифікації вимог і результатів проміжного проектування – оновлена інформація про зв'язки між різними частинами системи і первісно заданими вимогами до них. Трасування забезпечує:

– введення складних зв'язків замість простих;

– використання різних шляхів трасування (між моделями або ієрархічними зв'язками);

– трасування об'єктів і зв’язків між ними.

Трасування може бути вибірковим для окремих об'єктів або зв'язаним з іншими об'єктами, а також з можливими переходами від однієї моделі проектування до іншої.

1.2. Об’єктно-орієнтована інженерія вимог

В об’єктно-орієнтованих підходах і методах розробки програмних систем головним є об'єкт. Для нього задаються вимоги за допомогою варіантів використання (use case), сценаріїв або прецедентів.

Наведені сценаріями або прецедентами вимоги до системи в UML послідовно трансформуються до інших сценаріїв, що наближають до логічної та виконуваної структури системи. Головні їх елементи – сценарії і актори, що задають дії щодо виконання сценаріїв системи.

1.2.1. Візуальний підхід

Один з методів побудови моделі системи, логічної і фізичної моделей – це use case, що використовується для візуального зображення вимог у моделі системи, яка уточнюється і доповнюється новими сценаріями для одержання остаточних логічної і фізичної моделей системи. Термін сценарій позначає деякий варіант подання моделі виконання системи [1, 5, 6].

При застосуванні сценарного підходу загальна метасистеми декомпозується на окремі підцілі, для яких визначаються функціональні або не функціональні вимоги і проектні рішення. Мета як джерело вимог до системи дає змогу виявити протиріччя й обмеження на функції й встановити залежності між ними, усунути конфлікти між цільовими функціями, а також об'єднати деякі з них між собою [11].

Після виявлення цілей визначаються носії інтересів, яким відповідає кожна мета, і можливі варіанти задоволення складених цілей у вигляді сценаріїв роботи системи, що допомагають користувачу одержати уявлення про призначення і виконання функції системи. Це відповідає першій ітерації визначення вимог до системи.

Далі виробляється послідовна декомпозиція складної проблеми до вигляду сукупності цілей, кожна з яких трансформується в сукупність можливих сценаріїв використання системи, а потім у сукупність взаємодіючих об'єктів. Тобто, маємо ланцюжок трансформацій:

$$проблема \rightarrow ціль \rightarrow сценарій \rightarrow об'єкт,$$

що характеризує ступінь концептуалізації аналізованої проблеми та її декомпозицію на сценарії з варіантів використання. Трансформація даного ланцюга виражається в термінах базових понять предметної області й активно використовується для подання і розвитку моделей системи.

Кожен сценарій ініціює актор, що виступає в ролі користувача визначеної роботи в системі, що зображена цим сценарієм. Фіксацію ролей акторів можна розглядати як визначений крок при виявленні цілей системи і постановки задач, а також рішення, що буде виконувати система.

Актор – це зовнішній чинник і його дії мають недетермінований характер. У його ролі може виступати і програмна система, якщо вона ініціює виконання деяких робіт, що задовольняють поставлені цілі системи. Ним може бути абстракція зовнішнього об'єкта, людина або зовнішня система. У моделі системи актор може бути поданий класом, а користувач – екземпляром класу, хоча це і не обов’язково. Якщо актор – це система, то він репрезентує її інтереси. При цьому одна особа може бути екземпляром декількох акторів.

Якщо актор знаходиться поза системою, то він взаємодіє з нею через зовнішній сценарій, що ініціює послідовність операцій для виконання системи.

Коли користувач як екземпляр актора ініціює певну подію для старту відповідного сценарію, то це приводить до виконання ряду дій у системі, що завершуються тоді, коли екземпляр сценарію перебуває в стані очікування чергової події або завершення сценарію.

Екземпляр сценарію існує, поки він виконується і його можна вважати екземпляром класу, він має свій стан і у нього своє поводження. Взаємодія між актором і системою породжує новий сценарій або об'єкт, що змінює внутрішній стан системи. Якщо кілька сценаріїв системи мають однакове поводження, вони створюють клас сценаріїв.

При внесенні змін відбувається повторне моделювання дій акторів і сценаріїв, які запускаються ними в дію. Сценарій ініціюється актором і кожний з них обслуговує відповідну сукупність сценаріїв.

Для завдання моделі сценаріїв використовується графічна нотація UML з такими правилами:

– актор позначається зображенням – іконка людини і можливо з назвою;

– сценарій подається овалом, у середині якого назва зображення іконки;

– актор зв'язується лінейкою з кожним овалом сценарію, що запускається ним в дію.

Приклад діаграми сценаріїв для читача бібліотеки в ролі актора подано на рис. 1.4.

Рис. 1.4. Приклад діаграми сценаріїв для читача

Актор починає заданий сценарій при звертанні до автоматизованої системи обслуговування бібліотеки. Усі сценарії, що містяться у системі, обведені рамкою, яка визначає межі системи, а актор знаходиться поза рамкою як зовнішній чинник системи.

Відношення між сценаріями. Між сценаріями відношення задаються стрілками з указівкою назви типу відносин.

Для сценаріїв можна задавати два типи відношення:

1) відношення «розширює» означають, що функція одного сценарію є доповненням до функції іншого і використовується при наявності декількох варіантів одного й того самого сценарію (рис. 1.5).

Рис. 1.5. Приклад відношення «розширює»

Інваріантна частина сценарію зображується у вигляді головного сценарію, а окремі варіанти – як розширення. При цьому головний сценарій є стійким, не змінюється при розширенні варіантів функцій і не залежить від них;

2) відношення «використовує» означають, що деякий сценарій використовується як розширення інших сценаріїв (рис. 1.6).

На рис. 1.6 показано сценарій «ведення репозитарію», що зв'язаний відносинами «використовує» з декількома сценаріями – розроблення інтерфейсу, опис компонента, створення схеми розгортання.

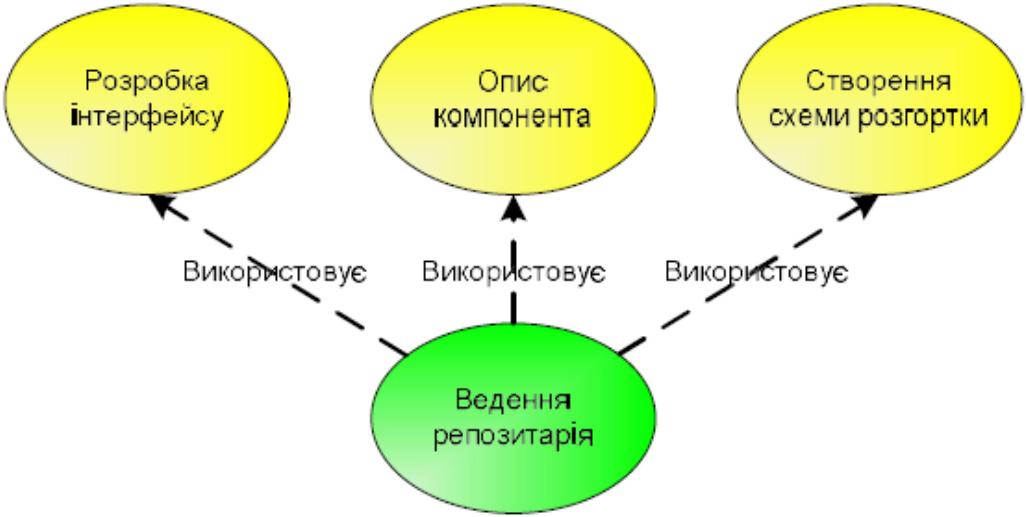

Рис. 1.6. Приклад відносин «використовує»

Інженерія вимог завершується побудовою моделі вимог, що містить у собі:

1) опис вимог і основних понять ПрО;

2) модель сценаріїв;

3) інтерфейси сценаріїв.

Модель сценаріїв – це неформальний опис кожної з діаграм сценарію, що входять у нього і описується послідовністю таких елементів:

– назва сценарію на діаграмі моделі вимог у вигляді посилання до іншого сценарію;

– короткий зміст сценарію в неформальному зображенні;

– список акторів, що будуть запускати в дію сценарії;

– параметри взаємодії системи з акторами, їх заборонені дії і можливі наслідки;

– передумови, що визначають початковий стан сценарію на момент його запуску і умови успішного виконання;

– функції, що реалізуються при виконанні сценарію;

– нестандартні ситуації, що можуть з'явитися при виконанні сценарію (наприклад, помилка в діях актора або системи).

На наступних процесах ЖЦ сценарій актора в моделі вимог трансформується в сценарій поводження системи, до елементів моделі можуть додаватися нефункціональні вимоги, що забезпечують запуск сценарію, введення даних і відпрацювання нестандартних ситуацій.

У процесі проектування виконується трансформація сценарію в опис функціональних компонентів системи і перевірка їх за допомогою верифікації.

Вимоги користувачів до системи відбивають в описі інтерфейсів компонентів, що розміщаються в репозитарії. За допомогою сценаріїв можна побудувати прототип системи для моделювання дій акторів у процесі їхнього виконання і відпрацювання різних їхніх деталей.

1.2.2. Текстовий підхід

Альтернативним терміном для сценарію є прецедент. Як і у випадку сценаріїв, задача опису вимог прецедентами зводиться до аналізу дерева цілей системи і до опису реакції системи у випадку недосяжності тієї або іншої поставленої мети щодо проектованої системи. Головною умовою завдання вимог прецедентами є повнота системних вимог до інтерфейсу користувача, до протоколів і форматів ведення [2,5].

Прецедент – це деякий випадок у системі, що міститься у декількох екземплярах. Екземпляр – це послідовність дій виконання системою, що може бути ініційована конкретним екземпляром актора. Опис прецеденту містить у собі назву і те, що відбудеться в системі, коли прецедент буде виконаний. Набір прецедентів установлює всі можливі шляхи використання системи.

При визначенні вимог створюється модель прецедентів, що моделює те, що повинно робити система з погляду потреб користувачів. На рівні реалізації проекту в цю модель додаються технічні вимоги, що зображуються в термінах класів.

Змістовна сторона системних вимог – опис функцій, даних і умов функціонування. Методологія формування вимог за допомогою прецедентів реалізована в середовищі Rational Rose (www.rational.com.uml) і передбачає побудову ряду моделей на їхній основі. Прецеденти відіграють визначену їм роль у кожному з основних процесів проектування: розроблення вимог, аналіз і проектування, виконання й випробування системи. Екземпляр прецеденту у реалізації відображає послідовність дій, виконуваних системою, і спостережень за одержанням результату.

У керованому прецедентами проекті розробляються два зображення системи – зовнішнє і внутрішнє. Зовнішнє зображення ПрО визначає, що повинно відбуватися в системі, щоб забезпечити замовнику необхідні результати. Після подання цілей системи прецедентами розробляються принципи взаємодії системи і її суб'єктів.

Внутрішнє зображення – це принципи організації роботи системи для досягнення запланованих результатів. Воно містить у собі сутності, що беруть участь у виконанні прецеденту, і зв'язки між ними. При цьому кожний із прецедентів виконує визначену дію для досягнення цілі і необхідних результатів у системі.

У процесі аналізу проблеми і формування вимог створюється модель прецедентів з відображенням мети системи. Вона складається з:

– використовуваних термінів (глосарія) предметної області;

– головних діючих осіб і їхніх цілей;

– використовуваних технологій і принципів взаємодії з іншими системами;

– вимог до форматів і протоколів взаємодії;

– вимог до тестування і до процедури розгортання системи у замовника;

– організації керування процесом розробки системи.

На процесі аналізу і проектування модель прецедентів реалізується в моделі проекту в термінах взаємодіючих об'єктів, тобто дається опис того, як прецедент буде виконуватися в системі.

Із синтаксичної точки зору ця модель має такий вигляд.

<Модель прецеденту ::= <ім'я прецеденту/діючої особи>, <ім'я ролі / короткий опис ролі діючої особи>, <опис меж системи>, <список усіх зацікавлених осіб при аналізі ключових цілей системи>, <вихідні умови>, <результат успішного закінчення визначення цілей системи>, <кроки сценарію для формування шаблону досягнення цілей проекту>, <опис інформації, необхідної розробнику для реалізації системи>.

Даний підхід до зображення системи за допомогою прецедентів можна задавати у формі шаблонів [5], які застосовуються в офісній сфері, де діловий прецедент відбиває зображення цієї сфери з зовнішньої сторони, щоб забезпечити суб'єкта необхідними результатами. При виконанні ділового прецеденту визначається взаємодія ділової сфери і суб'єкта. Сукупність ділових прецедентів установлює межі системи.

Внутрішнє зображення ділового прецеденту – це реалізація, що охоплює функції ділових працівників і, відповідно, бере участь у їхньому виконанні, а також зв'язки між ними. Таке завдання системи розробляється для того, щоб вирішити, як повинна бути організована робота системи за допомогою ділового прецеденту.

Висновки.

Проаналізовано підходи і методи формування вимог до системи і ПЗ, що створюється. Розглянуто функціональні і нефунціональні вимоги. Значна увага приділена об’єктно-орієнтованим методам інженерії вимог, що використані для побудови моделей предметних областей і на їхній основі проектування програмної системи. Наведені приклади проектування вимог за сценаріями і прецедентами.

ЛІТЕРАТУРА

1. *Вигерс К.И.* Розробка вимог до ПЗ. – М.: Російська редакція Microsoft, 2004. – 575 с.
2. *Леонов И.В.* Введення в методологію розробки програмного забезпечення за допомогою Rational Rose // Ескейп, 2004. – 301 с.
3. *Zave P., Jackson M.* Four Dark Corners of Requirements Engineering // ACM Transactions on Software Engineering, January 1997.– № 1.
4. *Pinheiro Francisco A. C., Goguen Joseph A.*. An Object-Oriented tool for Tracing Requirements // Software.– Mach 1996.– № 3.
5. *Guckkenheimer S., Peter J.* Software Engineering With Microsoft Visual Studio. Team System. – Adison Wesley, 2006. – 273 p.