МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра комп’ютерних наук

ОПОРНИЙ КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

«Аналіз вимог до програмного забезпечення»

Тернопіль - 2016

ЛЕКЦІЯ 1
ВВЕДЕННЯ

В даний час історія розвитку систем, призначених для зберігання та обробки інформації з використанням ПК, налічує вже понад півстоліття. В останнє десятиліття минулого століття ситуація зазнала якісних змін. Якщо спробувати сформулювати "портрет" сучасної інформаційної системи масштабу підприємства у вигляді десятка тез, то ми побачимо, що вона має:

• в основі - методологію управління, спрямовану на досягнення стратегічних цілей вищого менеджменту підприємства, виражену в інформаційній системі у вигляді системи управляючих впливів, що регламентує діяльність користувачів,

• можливість доступу до даних для безлічі користувачів, об'єднаних в локальну мережу підприємства, а часто - і для користувачів, віддалених від центрального офісу на сотні і тисячі кілометрів,

• засоби комунікації та елементи корпоративного рішення завдань колективом користувачів;

• розвинений, дружній графічний інтерфейс кінцевого користувача,

• режими обробки оперативної інформації, близькі до режиму реального часу,

• засоби аутентифікації і розмежування доступу, що дозволяють дозувати інформацію у відповідності з посадовими обов'язками користувача; високий рівень захищеності від несанкціонованого доступу,

• один або більше серверів баз даних, сумарний обсяг яких вимірюється в гіга - або терабайт; можливість обробки тисяч і мільйонів записів при складанні звітності,

• інваріантність (у певних межах) до апаратних і операційних середовищ функціонування серверних і клієнтських додатків,

• використання стандартизованих мов і протоколів для представлення та маніпулювання даними.

Основу інформаційної системи складають "три кити". Це - база даних, як правило, реляційного типу, підтримуюча доступ на основі стандарту SQL, програмні засоби, що забезпечують логіку обробки даних, інтерфейс користувача.

Визначення ІС

Далі по тексту ***інформаційною системою (ІС)*** *будемо називати програмно-апаратну систему, призначену для автоматизації цілеспрямованої діяльності кінцевих користувачів, що забезпечує, відповідно до закладеної в неї логікою обробки, можливість отримання, модифікації і зберігання інформації.*

Ключовим моментом у цьому визначенні є поняття "цілеспрямованої діяльності". Йдеться про діяльність, спрямовану на вирішення конкретного завдання, що стоїть перед користувачем (колективом користувачів).

Деякі дослідники визначають ІС трохи іншим чином. ІС в широкому сенсі - взаємозв'язана сукупність засобів, методів і персоналу, використовуваних для зберігання , обробки та видачі інформації в інтересах досягнення поставленої мети.

Основні відмінності такого підходу: 1) введення користувачів системи "всередину" ІС; 2) необов'язковість використання засобів обчислювальної техніки. Такий підхід також має право на життя. Так, наприклад, в ньому зручно простежувати загальну історію виникнення і розвитку систематичних засобів обробки інформації в бізнесі, яка почалася, очевидно, в докомпьютерную епоху. Однак, так як метою нашого курсу є вивчення аналізу вимог до комп'ютерних систем обробки інформації, будемо користуватися наведеним вище першим визначенням.

Розглянемо **приклади** деяких програмних засобів, що є, або не є ІС.

• 1С-Бухгалтерія 8.0. Використовується в цілях формування бухгалтерської звітності підприємства перед податковими органами. Є інформаційною системою.

• MS Excel. Програмний засіб універсального характеру, призначений для маніпуляцій з даними, представленими в табличній формі, автоматизації розрахунків, формування різноманітних діаграм для аналізу даних. Не є інформаційною системою.

• Книга MS Excel, що містить відомості про штатний розклад, працівників підприємства і оснащена макросами, що дозволяють розраховувати заробітну плату і формувати платіжні відомості. Є інформаційною системою .

• Система Axapta Retail комплексної автоматизації діяльності мережі роздрібних магазинів. Є інформаційною системою.

• Реляційна база даних DB-2 фірми IBM. Не є інформаційною системою.

Класифікація ІС

Інформаційні системи можуть бути класифіковані за різними ознаками. Розглянемо найбільш важливі з них.

Класифікація за масштабом

За масштабом ІС поділяються на однокористувацькі, групові та корпоративні .

Однокористувацькі ІС призначені для використання на одному робочому місці. В даний час на світовому та вітчизняному ринку представлено безліч рішень , призначених для автоматизації діяльності окремо взятого користувача. Як правило, це - рішення, орієнтовані на спеціаліста в тій чи іншій області, будь то складання специфікацій для складання виробів з комплектуючих, планування ремонтів обладнання, облік витрат і доходів приватного підприємця оптової торгівлі, або складання розкладів занять у деканаті.

В даний час альтернативу таким вузькоспеціалізованим системам склали табличні процесори, що не мають проблемної спеціалізації, в першу чергу - MS Excel. Системи цього класу важко віднести до класу ІС, але часто вони дозволяють непрограмуючому спеціалісту створити і, що дуже важливо, самостійно розвивати власні рішення, що замінюють, а місцями і перекривають функціонал однокористувальницьких систем зразка 90-х років.

В основі більшості однокористувальницьких систем лежить стандарт X-Base (Clipper, FoxPro, dBase). Широко використовуються також рішення на базі систем Paradox, Clarion, MS Access. Кожна з перерахованих конкуруючих систем володіє власним високорівневим інструментальним середовищем, що

дозволяє спроектувати базу даних, логіку обробки, користувальницький інтерфейс, звіти. На рубежі тисячоліть з'явилися також і однокористувацькі рішення на базі промислових реляційних СУБД. В цьому випадку ПЗ сервера інсталюється безпосередньо на робочу станцію користувача. Прикладом може служити Personal Oracle. Дані рішення пред'являють значні вимоги до ресурсів робочої станції, проте несуть в собі багато переваг промислових СУБД.

Групові системи призначені для автоматизації діяльності в робочій групі (відділі, кластері, групі проекту і т.д.). На відміну від однокористувальницьких ІС, групові системи, як правило, представляють спеціалізовані клієнтські рішення (їх часто називають автоматизованими робочими місцями, АРМ) для різних учасників групи. Наприклад, для оптової фірми, ІС може представляти набір таких АРМ, як "Менеджер з продажів", "Комірник", "Постачальник", "Директор". Для навчального планування - "Викладач", "Працівник відділу планування", "Працівник навчального відділу", "Спеціаліст з планування на кафедрі", "Працівник деканату".

Групове використання рішень на базі табличних процесорів можливе, але має суттєві обмеження, пов'язані з розмежуванням доступу, регламентацією і синхронізацією внесених змін. По суті єдиний режим їх використання, що забезпечує коректність даних - "файловий *сервер*, один автор, N читачів".

При створенні групових ІС в цілому використовуються ті ж засоби та інструментальні середовища, що і при створенні однокористувальницьких ІС. Однак, слід відзначити, що для використання в групі при виборі між системами з файловим і реляційним *сервером* слід віддавати перевагу реляційному *серверу* (причому доцільно використання виділеного сервера). Це може бути, наприклад, сервер Oracle, DB2, MS SQL, Sybase, Informix.

Корпоративні ІС (КІС) призначені для автоматизації діяльності підприємства. В англомовній літературі поняття "КІС" нерозривно пов'язане з поняттям "ERP" (Enterprise Resource Planning). В основі ERP-систем лежить міжнародний стандарт управління підприємством MRP-II (Manufacture Resource Planning), що забезпечує можливість обліку, аналізу та планування основних ресурсів - фінансів, людських, матеріальних. Відповідно, корпоративні ERP-системи - набір інтегрованих програм, які комплексно, в єдиному інформаційному просторі підтримують всі основні аспекти управлінської діяльності підприємств: планування ресурсів (фінансових, людських, матеріальних) для виробництва товарів (послуг), оперативне управління виконанням планів (включаючи постачання, збут, ведення договорів), всі види обліку та аналіз результатів господарської діяльності.

Серед вимог, що висуваються до сучасних КІС:

- централізація даних у єдиній базі (в основі - завжди промислова СУБД),
- близький до реального часу режим роботи,
- збереження загальної моделі управління для підприємств різних галузей,
- підтримка територіально-розподілених структур,
- робота на широкому колі апаратно-програмних платформ і СУБД.

Приклади ERP-систем - SAP R3, "Галактика", MS Navision Axapta.

Класифікація з архітектури

1. Архітектура "Файл-сервер". Історично перша архітектура інформаційних систем. Як виконувані модулі, так і дані розміщуються в окремих файлах операційної системи. Доступ до даних здійснюється шляхом вказівки шляху (path) і використання файлових операцій (відкрити, вважати, записати). Для зберігання даних використовується виділений сервер (окремий комп'ютер), який і є файловим сервером. Виконувані модулі зберігаються або на робочих станціях, або на файловому сервері. В останньому випадку спрощується процедура їх адміністрування, але при цьому зростають вимоги до надійності мережі.

2. Архітектура "Клієнт-сервер". Клієнт-сервер - це не тільки архітектура, це - нова парадигма, що прийшла на зміну застарілим концепціям. Суть її полягає в тому, що клієнт (виконуваний модуль) запитує ті чи інші сервіси у відповідності з певним протоколом обміну даними. При цьому, на відміну від ситуації з файловим сервером, немає необхідності у використанні прямих шляхів операційної системи: клієнт їх "не знає", йому "відомі" лише ім'я джерела даних та інші спеціальні відомості, що використовуються для авторизації клієнта на сервері. Сервер, який фізично може перебувати на тому ж комп'ютері, а може - на іншому кінці земної кулі, обробляє запит клієнта і, зробивши відповідні маніпуляції з даними , передає клієнту запитувану порцію даних.

У рамках напрямку "клієнт- сервер" існують два основних "діалекти": "тонкий" і "товстий" клієнт.

У системах на основі тонкого клієнта використовується потужний сервер баз даних, це – високопродуктивний комп'ютер та бібліотека так званих збережених процедур, що дозволяють робити обчислення та реалізують основну логіку обробки даних безпосередньо на сервері. Клієнтський додаток, відповідно, пред'являє невисокі вимоги до апаратного забезпечення робочої станції. Основна перевага таких систем - відносна дешевизна клієнтських станцій.

Системи з товстим клієнтом, навпаки, реалізують основну логіку обробки на клієнті, а сервер являє собою в чистому вигляді сервер баз даних, що забезпечує виконання тільки стандартизованих запитів на маніпуляцію з даними (як правило - читання, запис, модифікацію даних в таблицях реляційної бази даних). У системах такого класу вимоги до робочої станції вище, а до сервера - нижче. Перевага архітектури - переносимість серверної компоненти на сервери різних виробників: всі промислові сервери баз даних реляційного типу підтримують роботу із стандартизованою мовою маніпулювання даними SQL, але внутрішня вбудована мова обробки даних, необхідна для реалізації логіки обробки, у кожного з серверів своя.

3. Тришарова архітектура. Базується на подальшій спеціалізації компонент архітектури: клієнт займається тільки організацією інтерфейсу з користувачем, сервер баз даних - тільки стандартизованою обробкою даних. Для реалізації логіки обробки даних архітектура передбачає окремий шар - шар бізнес-логіки. Цей шар може являти собою або виділений сервер (сервер додатків), або розміщуватися на клієнті в якості динамічної бібліотеки. Дана архітектура дозволила з'єднати переваги тонкого і товстого клієнта: хороша переносимість з'єднується в ній з невисокими вимогами до клієнта.

З розвитком інтранет- інтернет технологій з'явився різновид тришарової архітектури на онові використання web-технологій. У цьому різновиді роль сервера додатків грає web-сервер, а в якості клієнта використовується стандартний web-браузер. Переваги - в знижених вимогах до клієнта і в легкій вбудованості даної архітектури у світові інформаційні мережі. Основний недолік - відомі обмеження, що накладаються на інтерфейс користувача web-браузерами.

Класифікація за характером використання інформації

З деяким ступенем наближення всі ІС можна розділити на 2 класи: інформаційно-пошукові та керуючі.

Кінцеві користувачі *інформаційно-пошукових систем* (ІПС), як правило, мають доступ до збережених даних тільки з "читання" і використовують дані системи для пошуку відповідей на ті чи інші питання. Доступ до модифікації даних має адміністратор системи, у функції якого входить забезпечення актуальності інформації , усунення помилок.

Класичні приклади ІПС - системи пошуку в бібліотеках, на транспорті (довідки про наявність квитків). На сучасному етапі розвитку інформаційних технологій класичні ІПС поступово витісняються пошуковими серверами Інтернет - загального призначення і спеціалізованими.

Альтернатива ІПС - керуючі системи, які автоматизують (повністю або частково) діяльність, пов'язану з прийняттям рішень. Дії кінцевих користувачів таких систем призводять до модифікації інформації, що, звичайно, не виключає можливості просто отримувати інформацію, як в ІПС.

Приклади керуючих систем - системи бухгалтерського обліку, системи планування виробничих ресурсів і т.п.

Класифікація за системою подання даних

Серед найбільш поширених засобів і моделей подання даних слід виділити:

• "саморобні" формати зберігання даних, що зберігаються у файлах (текстових, бінарних);

• спеціалізовані формати зберігання даних, що використовувалися в "дореляційний" період (наприклад, x-Base, paradox);

• мови структурованої розмітки на основі формату xml;

• реляційна модель; SQL сервер;

• об'єктна, об'єктно-реляційна модель;

• документоорієнтоване сховище (IBM Lotus / Domino).

Класифікація за підтримуваним стандартом управління і технологіям комунікації

Епоха стихійної розробки АІС закінчена. Сучасні автоматизовані інформаційні системи розробляються, виходячи з реалій автоматизованого управління бізнесом. Існує значна кількість концепцій, технологій, підходів, що знайшли своє ефективне застосування в різних галузях промисловості по всьому світу. Деякі з них набули статусу міжнародних стандартів. У специфікації АІС, розроблюваної для масового продажу, як правило, вказується

- які стандарти і технології управління вона підтримує. Менш суворі вимоги до АІС, створюваним під замовлення для конкретного підприємства. Однак і в цьому випадку не враховувати світовий позитивний досвід просто нерозумно. Нижче перераховані деякі, найбільш важливі, технології і стандарти.

MRP (Material Requirements Planning) - планування поставок матеріалів, виходячи з даних про комплектації виробленої продукції і плану продаж.

CRP (Capacity Requirements Planning) - планування виробничих потужностей, виходячи з даних про технології виробленої продукції і прогнозу попиту.

MRP II (Manufacture Resource Planning) - планування матеріальних, потужних і фінансових ресурсів, необхідних для виробництва. Стандартизовано ISO.

ERP (Enterprise Resource Planning) - фінансово -орієнтоване планування ресурсів підприємства, необхідних для отримання, виготовлення, відвантаження та обліку замовлень споживачів на основі інтеграції всіх відділів і підрозділів компанії.

SCM (Supply Chain Management) - управління ланцюгами поставок. Реалізація бізнес-процесів на базі зовнішніх підприємств і торговельних майданчиків Засновано на основі референтної моделі SCOR, стандартизованої Supply Chain Council.

CRM (Customer Relationship Management) - управління взаємовідносинами із замовниками. Комплекс методів і засобів, націлений на завоювання, задоволення вимог та збереження платоспроможних клієнтів.

ERP II (Enterprise Resource & Relationship Processing) - управління ресурсами і взаєминами підприємства. Об'єднує в собі 3 вищеперелічені технології.

Workflow - технологія, керуюча потоком робіт за допомогою програмного забезпечення, здатного інтерпретувати опис процесу, взаємодіяти з його учасниками і, при необхідності, викликати відповідні програмні додатки.

OLAP (Online Analytical Processing) - оперативний аналіз даних. Технологія підтримки прийняття управлінських рішень на основі концепції багатовимірних кубів інформації.

Project Management - управління проектами. Підтримується рядом міжнародних стандартів.

CALS (Continuous Acquisition and Lifecycle Support) - безперервна інформаційна підтримка поставок і життєвого циклу. Описує сукупність принципів і технологій інформаційної підтримки життєвого циклу продукції на всіх його стадіях. Об'єднує в собі практично всі перераховані вище підходи і технології.

Настільки лаконічні визначення, звичайно ж, дозволяють лише ознайомитися з сучасною термінологією. Завдання їх детального вивчення не входить в план занять.

Класифікація за ступенем автоматизації

Ручні ІС характеризуються відсутністю сучасних технічних засобів обробки інформації та виконанням всіх операцій людиною. Наприклад, про

діяльність менеджера у фірмі, де відсутні комп'ютери, можна говорити, що він працює з ручною ІС.

Автоматичні ІС виконують всі операції з переробки інформації без участі людини.

Автоматизовані ІС припускають участь в процесі обробки інформації і людини, і технічних засобів, причому головна роль відводиться ПК. У сучасному тлумаченні в термін "інформаційна система", як правило, вкладається поняття автоматизованої системи.

Роль вимог в задачі впровадження АІС

Які можна зробити висновки з розглянутої класифікації ІС? Навіть не кажучи про різноманіття виробників ІС, що не розглянуті в цій лекції, на підставі тільки сформульованих ознак стає очевидним, що існує значна кількість ІС і вони істотно різняться між собою. Отже, вибір ІС для підприємства - досить нетривіальне завдання. Для того, щоб успішно її вирішити, необхідно добре знати об'єкт впровадження (підприємство), особливості його діяльності, стратегію розвитку і багато інших аспектів, які спричиняють характеристики ІС. Зазначені знання в кінцевому підсумку формалізуються в документі вимог до ІС, на основі якого і здійснюється вибір і подальша настройка ІС. Ще більшою мірою вимоги до АІС важливі при розробці АІС на замовлення. Детальніше про це - в наступних лекціях.

ЛЕКЦІЯ 2
ПОНЯТТЯ ВИМОГИ. КЛАСИФІКАЦІЇ ВИМОГ

Існує значна кількість різних методів класифікації вимог, найбільш суттєві з яких будуть розглянуті в лекції.

Визначення поняття вимоги

В російській редакції нотації RUP подає таке визначення: "Вимога - це умова або можливість, якій повинна відповідати система".

У IEEE Standard Glossary of Software Engineering Terminology (1990) дане поняття трактується ширше. *Вимога - це:*

1. умови або можливості, необхідні користувачу для вирішення проблем або досягнення цілей;

2. умови або можливості, якими повинна володіти система або системні компоненти, щоб виконати контракт або задовольняти стандартам, специфікаціям або іншим формальним документам;

3. документоване уявлення умов або можливостей для пунктів 1 і 2.

Введемо ще одне визначення. Вимоги - це вихідні дані, на підставі яких проектуються і створюються автоматизовані інформаційні системи.

Первинні дані, які надходять з різних джерел, характеризуються суперечливістю, неповнотою, нечіткістю, мінливістю. Вимоги потрібні зокрема для того, щоб Розробник міг визначити і погодити із Замовником часові та фінансові перспективи проекту автоматизації. Тому значна частина вимог повинна бути зібрана і оброблена на ранніх етапах створення АІС. Однак зібрати на ранніх стадіях всі дані, необхідні для реалізації АІС, вдається тільки у виняткових випадках. На практиці процес збору, аналізу й обробки розтягнутий у часі протягом усього життєвого циклу АІС, часто нетривіальний і містить безліч підводних каменів.

Класифікація вимог

Вимоги до продукту і процесу

Вимоги до продукту. У своїй основі вимоги - це те, що формулює Замовник. Мета, яку він переслідує - отримати хороший кінцевий продукт: функціональний і зручний у використанні. Тому вимоги до продукту є основоположним класом вимог.

Вимоги до проекту. Питання формулювання вимог до проекту, тобто до того, як Розробник виконуватиме роботи по створенню цільової системи, здавалося б, не лежать в компетенції Замовника. Без регламентації процесу Замовником легко можна було б обійтися, якби всі проекти завжди виконувалися точно і в строк. Однак, на жаль, світова статистика результатів програмних проектів говорить про зворотне. Замовник, вступаючи в договірні відносини з Розробником, несе різні ризики, головними з яких є ризик отримати продукт із запізненням, або неналежної якості. Основні заходи з контролю і зниження ризику - регламентація процесу створення програмного забезпечення та його аудит.

Наскільки детально Замовнику слід регламентувати вимоги до проекту - питання риторичне. Відповідь на нього залежить від множини факторів, таких, як:

- цінність кінцевого продукту для Замовника,
- ступінь довіри Замовника до Розробника,
- сума підписаного контракту,
- прив'язка терміну здачі продукту в експлуатацію до бізнес-планів Замовника і т.д.

Однак, з усією визначеністю можна сказати наступне:

1. регламентація процесу Замовником дозволяє знизити його ризики;

2. заходи Замовника з регламентації процесу призводять до додаткових накладних витрат. Потрібно знайти розумний компроміс між ступенем контролю ризиків і величиною витрат.

В якості вимог до проекту може бути внесений регламент звітів Розробника, спільних семінарів з оцінки проміжних результатів, визначено характеристики компетенцій учасників робочої групи, виконуючих проект, їх кількість, вказана методологія управління проектом. Нижче сформульований приклад формулювання вимоги до офшорного проекту (Замовник і Розробник фізично знаходяться в різних державах) - у цій ситуації Замовнику необхідний жорсткий контроль над Розробником.

1. Розробник представляє Замовнику узгоджений план робіт із деталізацією (WorkBreakdownStructure - WBS) з точністю до конкретних виконавців.

2. Розробник здійснює щоденні зібрання, регресійне тестування компонент продукту, що розробляється, і тестування продукту в цілому.

3. Усі управлінські та проектні артефакти, вихідні коди і тестові приклади розміщуються в режимі online в інтегрованому середовищі розробки Rational ClearCase з можливістю для Замовника здійснення online-моніторингу на базі web-технологій.

Рівні вимог

Впровадження ІС на підприємстві завжди переслідує конкретні бізнес-цілі - такі, як:

підвищення прозорості бізнесу,

скорочення термінів обробки інформації,

економія накладних витрат і т.д.

Сучасні інформаційні системи - це великі програмні системи, що містять в собі безліч модулів, функціональних та інтерфейсних елементів, звітів і т.д. Як охопити єдиним поглядом такі різнорідні речі, як цілі, переслідувані топ-менеджером підприємства Замовника, опис інтерфейсу користувача та характеристики модуля, що здійснює розрахунок собівартості?

На щастя, людство вже давно винайшло прийоми боротьби зі складністю, що широко застосовуються в моделюванні складних об'єктів - абстракцію і декомпозицію.

Стосовно до дисципліни аналізу вимог до програмних систем ці принципи працюють таким чином. Вимоги поділяються за рівнями. Рівні вимог

пов'язані, з одного боку, з рівнем абстракції системи, з іншого - з рівнем управління на підприємстві.

Зазвичай виділяють три рівня вимог.

• На верхньому рівні представлені так звані *бізнес-вимоги* (business requirements). Приклад бізнес- вимоги: система повинна скоротити термін оборотності оброблюваних на підприємстві замовлень в три рази. Бізнес-вимоги зазвичай формулюються топ-менеджерами, або акціонерами підприємства.

• Наступний рівень - рівень *вимог користувачів* (user requirements). Приклад вимоги користувача: система повинна представляти діалогові засоби для введення вичерпної інформації про замовлення, подальшої фіксації інформації в базі даних і маршрутизації інформації про замовлення до співробітника, відповідальному за його планування і виконання. Вимоги користувачів часто бувають погано структурованими, дублюючими, суперечливими. Тому для створення системи важливий третій рівень, в якому здійснюється формалізація вимог.

• Третій рівень – *функціональні вимоги* (functional requirements). Приклад функціональних вимог (або просто функцій) по роботі з електронним замовленням: замовлення може бути створене, відредаговане, видалене і переміщене з ділянки на ділянку.

Існують об'єктивні протиріччя між вимогами різних рівнів. Так, очевидною бізнес-вимогою є вимога про повноту інформації, яка збирається на робочих місцях користувачів в єдину базу даних. Чим повніше інформація - тим глибше база для аналізу діяльності та прийняття рішень. З іншого боку, конкретному користувачеві системи цілком може бути достатньо використання лише тієї частини інформації, яка впливає на виконання його основних функцій.

Важливі правила впровадження та використання АІС на підприємстві - "Одна точка збору", "Дані збираються там, де вони з'являються". Використання цих правил дозволяє уникнути витрат на необгрунтоване дублювання інформації і, що важливіше - втрат від помилок обліку, неминуче виникають при дублюванні точок введення.

Впровадження АІС на підприємстві призводить до необхідності оснащення всіх точок введення інформації автоматизованими робочими місцями (АРМ), навчання персоналу і, найчастіше, оптимізації та підвищення рівня формалізації робочих процесів, виконуваних персоналом. Тому впровадження АІС - непростий процес, часто вимагає "перекроювання людського матеріалу" і зустрічає опір з боку користувачів, які не готові, або не хочуть працювати по- новому.

Системні вимоги та вимоги до програмного забезпечення

Існують різні трактування поняття "Системні вимоги" (system requirements). INCOSE (International Council on Systems Engineering) дає більш детальне визначення *системи: "комбінація взаємодіючих елементів, створена для досягнення певних цілей; може включати апаратні засоби, програмне забезпечення, вбудоване ПЗ, інші засоби, людей, інформацію, техніки (підходи), служби та інші підтримуючі елементи"*. Таким чином , відбувається поділ між

системними вимогами, як узагальнюючим поняттям, та вимогами до програмного забезпечення, як виділеній підмножині системних вимог, спрямованих виключно на програмні компоненти системи. Цей же підхід простежується в стандарті ДСТУ ISO / IEC 12207 / 99: роботи, пов'язані з системою в цілому і з програмним забезпеченням виділяються в окремі групи з метою зручності оперування.

У практиці комп'ютерної інженерії побутує інший, більш вузький контекст використання даного поняття: під системними вимогами у вузькому сенсі розуміються вимоги, висунуті прикладною програмною системою (зокрема - інформаційною) до середовища свого функціонування (системного, апаратного). Приклад таких вимог - тактова частота процесора, об'єм пам'яті, вимоги до вибору операційної системи.

Функціональні, нефункціональні вимоги і характеристики продукту

Функціональні вимоги регламентують функціонування або поведінку системи (behavioral requirements).

Функціональні вимоги відповідають на питання "що повинна робити система" в тих чи інших ситуаціях.

Функціональні вимоги визначають основний "фронт робіт" Розробника, і встановлюють цілі, завдання та сервіси, що надаються системою Замовнику.

Функціональні вимоги записуються, як правило, при посередництві розпорядчих правил, наприклад: "система повинна дозволяти комірнику формувати прибуткові та видаткові накладні".

Іншим способом є так звані варіанти використання (uses cases) - популярний і вельми продуктивний спосіб представлення вимог.

Це - основний , визначальний вид вимог , що розглядатиметься на протязі всього лекційного курсу.

Нефункціональні вимоги, відповідно, регламентують внутрішні і зовнішні умови або атрибути функціонування системи.

Виділяють такі основні групи функціональних вимог:
- зовнішні інтерфейси (External Interfaces),
- атрибути якості (Quality Attributes),
- обмеження (Constraints).

Серед зовнішніх інтерфейсів в більшості сучасних ІС найбільш важливим є інтерфейс користувача (User Interface, UI). Крім того, виділяються інтерфейси із зовнішніми пристроями (апаратні інтерфейси), програмні інтерфейси та інтерфейси передачі інформації (комунікаційні інтерфейси).

Основні атрибути якості:
- застосовуваність,
- надійність,
- продуктивність,
- експлуатаційна придатність.

Обмеження - формулювання умов, що модифікують вимоги або набори вимог, звужуючи вибір можливих рішень щодо їх реалізації. Вибір платформи реалізації та/або розгортання (протоколи, сервери додатків, баз даних, ...), які, у свою чергу, можуть ставитися, наприклад, до зовнішніх інтерфейсів.

Характеристики продукту

Характеристика ("фіча" feature) формулюється, як набір логічно пов'язаних функціональних вимог, які забезпечують можливості користувача і задовольняють бізнес-цілі.

Існує і більш загальний погляд на дане поняття: "features можуть бути приналежні як до функціональних, так і до нефункціональних вимог, і можуть змінюватися від версії до версії продукту".

Роль характеристик виявляється в першу чергу в області маркетингу: не кожний потенційний споживач продукту стане читати його функціональні описи, а набір ключових показників, що характеризують конкурентні переваги, можна зробити лаконічним і вмістити на одній сторінці рекламної листівки, або надрукувати на компакт-диску.

Класифікація RUP

У специфікаціях Rational Unified Process при класифікації вимог використовується модель FURPS+ з посиланням на стандарт IEEE Std 610.12.1990.

Акронім FURPS визначає наступні категорії вимог:

- functionality (функціональність);
- usability (застосовність);
- reliability (надійність);
- performance (продуктивність);
- supportability (експлуатаційна придатність).

Символ "+" розширює FURPS-модель, додаючи до неї:

- обмеження проекту;
- вимоги виконання;
- вимоги до інтерфейсу;
- фізичні вимоги.

Крім того, у специфікаціях RUP виділяються такі категорії вимог, як

- вимоги, що вказують на необхідність узгодженості з деякими юридичними та нормативними актами;
- вимоги до ліцензування;
- вимоги до документування.

Методології та стандарти, що регламентують роботу з вимогами

Серед основоположних нормативних документів у галузі роботи з вимогами можна виділити наступні.

1. Розробки IEEE:

- IEEE 1362 "Concept of Operations Document".

• IEEE 1233 "Guide for Developing System Requirements Specifications".
• IEEE Standard 830-1998, "IEEE Recommended Practice for Software Requirements Specifications".
• IEEE Standard Glossary of Software Engineering Terminology / IEEE Std 610.12-1990.
• IEEE Guide to the Software Engineering Body of Knowledge (1) - SWEBOK®, 2004 .

2. Вітчизняні ДСТУ:

• ГОСТ 34.601-90. Інформаційна технологія. Автоматизовані системи. Стадії створення.
• ГОСТ 34.602-89. Інформаційна технологія. Технічне завдання на створення автоматизованої системи.
• ГОСТ 19.201-78. Єдина система програмної документації. Технічне завдання. Вимоги до змісту та оформлення.

ЛЕКЦІЯ 3
ВЛАСТИВОСТІ ВИМОГ

У практиці розробки програмних систем накопичилися певні уявлення про те, якими властивостями повинні володіти вимоги до програмної системи. У цій лекції ми розглянемо докладніше дані властивості

Найсуворіше і єдине правило побудови систем програмного забезпечення (ПО) - вирішити точно , що ж будувати. Ніяка інша частина концептуальної роботи не є такою важкою, як з'ясування деталей технічних вимог, у тому числі і взаємодію з людьми , з механізмами та з іншими системами ПЗ. Ніяка інша частина роботи так не псує результат , якщо вона виконана погано. Помилки ніякого іншого етапу роботи не справляються так важко .

Наука витягання і формалізації якісних (іноді говорять "хороших" , "правильних") вимог носить багато в чому емпіричний характер. Однак , у практиці розробки програмних систем накопичилися певні уявлення про те , якими властивостями повинні володіти вимоги до програмної системи. Це :

- повнота ,
- ясність ,
- коректність ,
- узгодженість ,
- верифіковані ,
- необхідність ,
- корисність при експлуатації ,
- здійсненність ,
- модифицируемость ,
- трасуванню ,
- впорядкованість за важливістю і стабільності ,
- наявність кількісної метрики .

Розглянемо зазначені вище властивості докладніше.

Повнота.

Як відомо з теорії штучного інтелекту , неповнота - одне з фундаментальних властивостей людського знання. При створенні програмних систем нам доводиться мати справу з характеристиками ще неіснуючої системи. Ідея про те , що необхідно сформулювати всі вимоги повністю, тобто вичерпним чином , до початку проектування , а тим більше - реалізації системи, зжила себе разом з так званим каскадним подходом, який підтримував послідовну модель реалізації системи . Спіральний підхід , на якому базується більшість сучасних методологій , передбачає поетапне виділення і деталізацію вимог на всьому протязі циклу розробки системи .

Проте , вимога повноти пред'являється до вимог , що формулюються до системи. Треба розуміти , що дана вимога - це скоріше тенденція , мета, до якої потрібно постаратися максимально наблизитися на якомога більш ранніх стадіях проекту.

Вимога повноти можна розглядати в двох аспектах: повнота окремого вимоги і повнота системи вимог .

Повнота окремого вимоги - властивість, що означає, що текст вимоги не вимагає додаткової деталізації, тобто в ньому передбачені всі необхідні нюанси, особливості і деталі даної вимоги.

Повнота системи вимог - властивість, що означає, що сукупність артефактів, що описують вимоги, вичерпним чином описує все те, що потрібно від розроблюваної системи.

Ясність (недвозначність, визначеність, однозначність специфікацій).

Кожен із співвласників розроблюваної системи має своїм особистим досвідом сприйняття подій зовнішнього світу. Слово , вимовлене вголос , викликає індивідуальні асоціації в семантичному просторі кожного окремого сприймає суб'єкта . Те , що є ясним , припустимо, для кардіохірурга , зовсім необов'язково буде таким для фахівця в галузі програмної інженерії.

Відповідно, вимога має властивість ясності , якщо воно схожим чином сприймається всіма співвласниками системи . На практиці ясність вимог досягається в тому числі і в процесі консультацій , в ході яких відбувається " вирівнювання тезаурусів " співвласників системи . Гарною підмогою в цьому служить узгоджений сторонами глосарій ключових понять предметної області.

Ще однією стороною поняття " ясність вимоги " є його простежуваність (див. також поняття трасуванню нижче по тексту) . Вимога, яка сформульовано ясно , може бути простежено , починаючи від того документа , де воно сформульовано вперше , аж до робочих специфікацій.

Коректність і узгодженість (несуперечність) .

Коректність - одне з найважливіших властивостей вимог. К. Вігерс вводить поняття коректності вимоги через точність опису функціональності. У цьому сенсі коректність певною мірою конкурує з повнотою. Але є й відмінність: якщо властивість повноти носить скоріше якісний характер: абсолютна повнота представляє недосяжний ідеал, до якого можна наближатися, то властивість коректності носить оціночний характер і задає дихотомію: кожну з вимог або коректно, або ні. Крім того, можна розмірковувати про взаємну коректності вимог або узгодженості (несуперечності): якщо дві вимоги вступають у конфлікт , значить - як мінімум одне з них некоректно. В ієрархії вимог можна виділити вертикальну і горизонтальну узгодженість . Іншими словами , вимоги не повинні суперечити , відповідно, вимогам свого рівня ієрархії і вимогам " батьківського "рівня. Так, вимоги користувачів не повинні суперечити бізнес-вимогам , а функціональні вимоги - вимогам користувача.

Верифіковані (придатність до перевірки) .

Ознаки (властивості) вимог , що розглядаються в цій лекції , не можна вважати незалежними. У математичній статистиці такі ознаки називаються корельованість. Так, властивість веріфіцируємості істотно пов'язано з властивостями ясності і повноти : якщо вимога викладена мовою, зрозумілою і однаково сприйманих учасниками процесу створення інформаційної системи , причому воно є повним , тобто жодна з важливих для реалізації деталей не втрачу - значить , це вимога можна перевірити. При цьому в ході перевірки у сторін (приймаючої і здає роботу) не повинно виникнути нерозв'язних протиріч в оцінках.

Так як добре сформульовані вимоги складають основу успішного створення системи - роль веріфіцируємості важко переоцінити. Вимоги до системи представляють основу контракту між Замовником та Виконавцем і якщо дані вимоги не можна перевірити - значить і контракт не має ніякого сенсу, отже, успіх або невдача проекту будуть залежати тільки від емоційних оцінок сторін і їх здатності домовитися, а це - надто хитка основа для здійснення робіт.

Необхідність і корисність при експлуатації.

Одні з найбільш суб'єктивних і важко перевіряються властивостей вимог.

Повертаючись до ієрархії вимог в лекції 2 , найбільш безспірними вимогами слід вважати бізнес- вимоги. Дані вимоги формулюють перші особи, які представляють Замовника, і навряд чи хто-небудь краще них зможе сказати, яким умовам повинна відповідати створювана інформаційна система, щоб відповідати бізнес-цілям підприємства. Тим не менш, якщо у представника Виконавця виникають сумніви в необхідності того чи іншого бізнес - вимог , викликані інтуїтивними міркуваннями, або досвідом впровадження інформаційних систем на аналогічних підприємствах , він повинен проявити ініціативу і зібрати спільну нараду сторін. Аргументи на користь відсутності необхідності вимоги безсумнівно будуть сприйняті , особливо якщо вони будуть мотивовані в бізнес- термінології Замовника і підтверджені викладками , прогнозує співвідношення витрат на виконання вимоги і очікуваної від нього ефективності .

Необхідність вимог користувача може випливати з відповідних бізнес - вимог . Крім того , вимоги користувача можуть мотивуватися ергономічністю продукту і особливостями функціонування його відділу (підрозділу), недостатньо повно розкритими на попередньому рівні ієрархії вимог.

Більшість функціональних вимог випливають з вимог перших двох рівнів. Інші функціональні вимоги можуть лежати поза сферою компетенції Замовника (який , взагалі кажучи , не зобов'язаний бути експертом в області IT) та їх повинен сформулювати Виконавець . Так , наприклад , інформаційна система в процесі її використання може почати знижувати свою продуктивність через великі обсяги накопичуваних даних. Тому доцільно закласти функції архівування інформації , перемикання облікових періодів і т.п. , необхідність яких слід не з особливостей бізнесу підприємства впровадження , а із загальних принципів побудови інформаційних систем .

Більш слабкої , ніж "необхідність" формулюванням володіє властивість "корисність при експлуатації". Розмежування між даними властивостями можна провести наступним чином . Необхідними слід вважати властивості , без виконання яких неможливо, або утруднене виконання автоматизованих бізнес-функцій користувачів; корисними при експлуатації слід вважати будь-які властивості , що підвищують ергономічні якості продукту .

Здійсненність (здійснимість) .

Є в деякій мірі конкуруючим по введених вище двох властивостями .

В принципі ніхто не заважає сформулювати вимогу , здійснимість якого обмежується сьогоднішнім рівнем розвитку техніки і технології , хоча багато з того , що було нездійсненно десять років тому , цілком здійснимо сьогодні. Можна сформулювати вимогу, здійснимість якої обмежена науковими

уявленнями про будову Всесвіту, наприклад - вимога миттєвої передачі інформації з земної станції на Марс, хоча й фундаментальні уявлення інколи змінюються , хай і не так швидко, як розвиток IT -технологій.

Повертаючись з небес на землю , відлинемо ті вимоги , які можна визнати абсурдними і зупинимося на тип , які здійснимі принципово. З точки зору науки управління вимог, далеко не всі з них є здійсненними .

Здійснимість вимоги на практиці визначається розумним балансом між цінністю (ступенем необхідності та корисності) і потреби ресурсів . Так , якщо вартість контракту на розробку інформаційної системи складає $ 10000 , а витрати на виконання нової вимоги , що виникло в момент , коли проект виконаний наполовину , оцінюється в $ 4000, є воно нездійсненним ? Швидше за все , так, якщо Виконавець доведе Замовнику новизну вимоги (вимога не входило в узгоджені специфікації) і складність його виконання. Але , якщо вимога є критично важливим , необхідним , але випало з поля зору при підписанні контракту Замовник готовий виділити додатково фінансування , а Виконавець - трудові ресурси - значить , вимога здійснимо. Таким чином , вимога здійсненності в ряді випадків також слід вважати суб'єктивним , а критерії його оцінки лежать в області домовленостей між Замовником та Виконавцем .

Чудовою ілюстрацією балансування між цінністю і здійсненними вимог є так званий трикутник компромісів.

Добре відома взаємозалежність між ресурсами проекту (людськими і фінансовими), його календарним графіком (часом) і реалізованими можливостями (рамками). Ці три змінні утворюють трикутник, показаний на рис. 3.1. Після досягнення рівноваги в цьому трикутнику зміна на будь-який з його сторін для підтримки балансу вимагає модифікацій на інший (двох інших) боках і / або на спочатку зміненої стороні.

Необхідно забезпечити можливість переробки вимог , якщо знадобиться , і підтримувати історію змін для кожного положення . Для цього всі вони повинні бути унікально помічені і позначені , щоб ви могли посилатися на них однозначно. Кожна вимога має бути записано в специфікації тільки одного разу. Інакше ви легко отримаєте неузгодженість, змінивши тільки одне положення з двох однакових . Краще використовуйте посилання на первинні твердження , а не дублюйте положення . Модифікація специфікації стане набагато легше , якщо ви складете зміст документа і покажчик . Збереження специфікації в базі даних комерційного інструменту управління вимогами зробить їх придатними для повторного використання.

Трасуванню

Трасуванню вимоги визначається можливістю відстежити зв'язок між ним та іншими артефактами інформаційної системи (документами , моделями , текстами програм та ін) . Окрема траса являє собою спрямований бінарне відношення , задане на множині артефактів ІС , де перший елемент відносини представляє відповідну вимогу , а другий - артефакт , залежний від даної вимоги . На практиці трасування аналізуються при посередництві графових , або табличних моделей.

Процес трасування дозволяє, з одного боку , виявити вже на стадії проектування системи проектні артефакти , до яких не веде зв'язок ні від одного з артефактів , що описують вимоги , з іншого - артефакти , що описують вимоги , не пов'язані з проектними артефактами . У першому з випадків доцільно переконатися в тому , що проектний артефакт дійсно має право на існування , а не є надлишковим . У другому випадку необхідно проаналізувати корисність виявлених вимог : або ці вимоги несуть недостатню корисне навантаження і можуть бути ігноровані , або мають місце помилки проектування: пропущені відповідні артефакти.

Інша мета трасування - підвищити керованість проектом : при зміні окремо взятого вимоги стає зрозуміло - які з проектних , робітників та інших артефактів підлягають зміні

Впорядкованість за важливістю і стабільності

Пріоритет вимоги являє собою кількісну оцінку ступеня значимості (важливості) вимоги. Пріоритети вимог зазвичай призначає представник Замовника. Дизайнер, відштовхуючись від пріоритетності вимог, управляє процесом реалізації інформаційної системи.

Стабільність вимоги характеризує прогнозну оцінку незмінності вимог в часі.

Наявність кількісної метрики

Кількісні метрики відіграють важливу роль у верифікації та атестації інформаційних систем . У першу чергу це відноситься до нефункціональним вимогам , які , як правило , повинні мати під собою кількісну основу (запит повинен відпрацьовуватися не більше , ніж ___ секунд ; середнє напрацювання на відмову повинна становити не менше , ніж ___ годин). Функціональні вимоги також можуть розширюватися кількісними заходами за допомогою так званих аспектів застосовності.

Яких вимог не повинно бути

Специфікація вимог не повинна містити деталей проектування або реалізації (крім відомих обмежень). Іншими словами , вимоги повинні відповідати на запитання: " що повинна робити система" , абстрагуючись від питання " як вона це повинна робити". Прагнення приймати детальні проектні рішення на етапі аналізу вимог - одна з типових " пасток " , типових для недосвідчених команд розробників . Варіантів реалізації завжди більше , ніж один , а для прийняття виваженого рішення потрібна максимально більш повна інформація . Тому етапи роботи з вимогами , проектування та реалізації плануються по черзі , хоча і можуть бути частково Запаралеленими в рамках ітераційного підходу до створення програмних систем.

ЛЕКЦІЯ 4
ПРОЦЕС АНАЛІЗУ ВИМОГ

Аналіз вимог - один з основних потоків програмної інженерії, поряд з такими, як проектування інтерфейсу користувача, або програмування, цьому питанню буде присвячена ця лекція.

Робочий потік аналізу вимог

Аналіз вимог - один з основних робочих потоків (workflow) програмної інженерії , поряд , припустимо , з такими , як проектування інтерфейсу користувача , або програмування .

Для його позначення в англомовній літературі , як правило , використовується поняття "Requirement Process" . У вітчизняній практиці , поряд з узагальнюючим терміном "аналіз вимог" , прийнятим, зокрема , в ДСТУ ISO / IEC 12207-99 , зустрічаються також такі терміни, як " потік робіт" вимоги ","робота з вимогами", "визначення вимог"і т.д. , що вносить неабияку плутанину. Для того , щоб внести деяку ясність, розглянемо декомпозицію робочого потоку Requirement Process на складові , прийняту в SWEBOK, і введемо термінологію , якою будемо дотримуватися протягом лекційного курсу.

SWEBOK пропонує виділити в Requirement Process такі основні складові:

• Requirements Elicitation (Витяг вимог) ,
• Requirements Analysis (Аналіз вимог у вузькому сенсі) ,
• Requirements Specification (специфицирования вимог) ,
• Requirements Validation (Перевірка вимог) .

Як приклад альтернативної декомпозиції потоку робіт можна розглянути погляд , запропонований у RUP . RUP пропонує виділити в основному потоці аналізу вимог такі компоненти , як:

• Analyze the Problem (Аналіз проблеми) ,
• Understand Stakeholder Needs (Розуміння потреб співвласників) ,
• Define the System (Визначення системи) ,
• Manage the Scope of the System (Управління контекстом системи) ,
• Refine the System Definition (Уточнення визначення системи).

Узагальнюючи зазначені вище декомпозиції, далі будемо дотримуватися наступної , більш зручною в методичному плані схемою декомпозиції потоку робіт " Робота з вимогами" :

• Формування бачення ;
• Виявлення вимог;
• Класифікація і специфікація вимог;
• Розширений аналіз вимог (моделювання і прототипування) ;
• Документування вимог;
• Перевірка вимог;
• Управління вимогами ;
• Удосконалення процесу роботи з вимогами .

Перші 6 робіт у лекційному курсі розглядаються , як компоненти процесу аналізу вимог .

Для того , щоб успішно створити автоматизовану інформаційну систему (або ширше , програмну систему) , необхідно , по-перше , визначити

компоненти потоку робіт , які будуть використовуватися командою розробників і , по-друге , правильно їх організувати. У питання організації входить упорядкування робіт у часі , інтерфейси між ними , паралелізм , робота з ризиками і багато іншого.

Знайти відповідь на перше питання може допомогти загальна класифікація завдань , робіт і операцій програмної інженерії , представлена в ДСТУ ISO / IEC 12207-99 . Інша , пізніша за часом класифікація , присутній у SWEBOK . Однак потрібно зазначити , що дані керівні документи розглядають загальний випадок , а в приватному проекті може бути задіяний далеко не весь арсенал робіт .

Складніше - з рішенням другого питання . На сьогодні існують і мають приклади успішного застосування десятки і сотні різних методологій (процесів) , серед найбільш відомих - MSF , RUP, Oracle PJM , XP , FDD , SCRUM , PSP , Crystal , DSDM . Методології поділяються на 3 " хвилі": каскадні (історично перші) , що прогнозують (наприклад , RUP) і " гнучкі " (agile) , що увійшли в широку практику на межі тисячоліть.

Описи методологій істотно різняться обсягом (від десятків до тисяч сторінок тексту) , наборами базових робіт і робочих кваліфікацій , акцентами (робота з моделями , управління ризиками , побудова команди та ін.) Але автори їх описів зазвичай сходяться в одному: найкраща з методологій , якої потрібно дотримуватися , щоб домогтися успіху - саме та , яку пропонує (описує , рекламує) автор. Рідкісним винятком є роботи А. Коберна , автора групи методологій Crystal, де він пропонує брати за основу не « найкращий" з процесів , а той , який , по-перше , найкращим чином відповідає проектної задачі , а по друге - команді , яка буде його реалізовувати. В вводиться кілька метрик , що дозволяють частково формалізувати процес підбору методології .

Чому потрібно аналізувати вимоги?

Зі сказаного вище випливає, що не всі роботи і операції , відомі в програмної інженерії , використовуються в тій чи іншій методології і, тим більше , конкретному проекті. Виникає питання: чи є робочий потік АТ необхідним у ланцюжку робочих потоків створення інформаційної системи , чи варто витрачати на нього час ? Який необхідний обсяг результатів , очікуваних від АТ ?

З усією очевидністю можна стверджувати : так , АТ , як етап розробки ІС , неможливо пропустити : цей етап закладає фундамент всього процесу проектування і реалізації системи . Ступінь опрацювання АТ може бути різною: від абсолютно неформальній записки , представленої на одній сторінці , до розгорнутої системи документів , моделей і прототипів , побудованої відповідно до принципів однією з прогнозуючих методологій , наприклад , RUP. Вона залежить від наступних основних факторів: розмірів проекту , величини наявних ресурсів і ступеня ризиків . Невисока глибина опрацювання прийнятна для невеликих проектів , що характеризуються невеликим ресурсом та невисокими ризиками . Добре опрацьовані вимоги дозволяють :

- виробити спільне розуміння між Замовником та Розробником ;
- визначити рамки проекту ;
- більш точно визначити фінансові та часові характеристики проекту ;

• убезпечити Замовника від ризику отримати продукт , в якому він не зможе працювати ,

• убезпечити Розробника від ризику потрапити в ситуацію " неконтрольованого розмиття кордонів" , яке може привести до непередбачених витрат ресурсів понад початкових очікувань .

Аналіз вимог - це процес (бізнес- процес) і, отже , до нього підходять методи і засоби процесного підходу до управління (див. , наприклад , [1]).

Одне з ключових питань , що дозволяють оцінити результативність процесу - що є виходом (результатом) процесу. У чому результат АТ ? Результатом робочого потоку " аналіз вимог " є набір артефактів. Це можуть бути текстові документи , моделі UML , або інших мов моделювання , прототипи програмного забезпечення.

Інше важливе питання - які цілі переслідує процес .

RUP пропонує наступні цілі для потоку робіт АТ :

• домогтися однакового розуміння з замовником і користувачами про те , що повинна робити система ;

• дати розробникам найкраще розуміння вимог до системи;

• визначити межі системи;

• визначити інтерфейс користувача і системи.

Хто створює і використовує вимоги

Як і ким використовуються вимоги?

• Спеціаліст з АТ - постановка задачі , визначення рамок проекту;

• Представник замовника - постановка задачі , визначення рамок проекту , контроль роботи виконавця , приймання результатів роботи ;

• Архітектор системи - розроблення архітектури , проектування підсистем;

• Програміст - розробка програмного коду;

• Тестувальник - складання тест- плану , тестових сценаріїв;

• Менеджер проекту - планування і контроль виконання робіт .

У рамках курсу лекцій для всіх згаданих вище осіб будемо використовувати узагальнюючий термін " Співвласники (зацікавлені сторони) " (stakeholders) . Співвласниками , слідом за розробниками RUP і MSF **Microsoft Solutions Framework** , будемо називати всіх учасників проекту створення програмної системи , прямо або побічно зацікавлених в його успіху . Автори більшості сучасних методологій розробки програмних систем сходяться в тому , що в групі співвласників ключову роль грають дві групи представників Замовника - ті , хто ставить стратегічні цілі та виділяє фінансування і ті , хто буде безпосередньо використовувати розроблений продукт. Причому , на відміну від каскадних методів , де Замовник підключався в початковій фазі - складанні технічного завдання та кінцевої - приймання готової роботи , в сучасних методологіях Замовник , дійсно зацікавлений в успіху проекту автоматизації , повинен брати участь у ньому **безперервно .**

Організація роботи з вимогами на прикладі MSF

У MSF для позначення ролі учасників команди софтверного проекту використовується поняття рольових кластерів .

MSF заснований на постулаті про шість якісних цілях , досягнення яких визначає успішність проекту . Ці цілі обумовлюють модель проектної групи. У

той час як за успіх проекту відповідальна вся команда , кожен з її рольових кластерів , що визначаються моделлю , асоційований з одного зі згаданих шести цілей і працює над її досягненням .

Шість рольових кластерів моделі проектної групи - це "Управління продуктом " (product management) , " Управління програмою " (program management) , "Розробка " (development) , " Тестування " (test) , " Задоволення споживача " (user experience) і " управління випуском " (release management) . Вони відповідальні за різні області компетенції (functional areas) та пов'язані з ними цілі і завдання .

MSF організований на базі комбінації каскадної і спіральної моделей. Окрема стадія роботи містить у собі 5 фаз:

- Envisioning (вироблення концепції) ,
- Planning (планування) ,
- Developing (розробка) ,
- Stabilizing (стабілізація) ,
- Deploying (впровадження) .

У фазі вироблення концепції робота з вимогами найбільш інтенсивна.

Рольовий кластер	**Фокус**
Управління продуктом	Загальні цілі проекту; виявлення потреб і вимог замовника; документ загального опису і рамок проекту.
Управління програмою	Цілі дизайну; концепція рішення; структура проекту.
Разробка	Прототипування; аналіз технологічних можливостей; аналіз здійсненності.
Задоволення споживача	Необхідні експлуатаційні характеристики рішення та їх вплив на його розробку.
Тестування	Стратегії тестування; критерії прийнятності, їх вплив на розробку рішення.
Управління випуском	Вимоги впровадження та їх вплив на розробку рішення; вимоги супроводу.

Як видно з таблиці, всі 6 кластерів працюють зі своїми групами вимог.
Триває спільна робота з вимогами і на наступній фазі - фазі планування, див.табл. 4.2.

Таблиця 4.2.

Рольовий кластер	**Фокус**
Управління продуктом	Аналіз бізнес-вимог
Управління програмою	Функціональна специфікація
Задоволення споживача	Сценарії / приклади використання, призначені для користувача вимоги, вимоги локалізації та загальнодоступності (accessibility).

Тестування	Вимоги тестування.
Управління випуском	Експлуатаційні вимоги.

У фазах розробки та впровадження робота з вимогами зосереджується в кластерах управління продуктом і програмою, див, відповідно, табл. 4.3,4.4.

Таблиця 4.3.

Рольовий кластер	**Фокус**
Управління продуктом	Очікування замовника.
Управління програмою	Управління функціональної специфікацією.

Таблиця 4.4.

Рольовий кластер	**Фокус**
Управління продуктом	Отримання відгуків і оцінок замовника; акт про прийом виконаної роботи.
Управління програмою	Зіставлення рамок проекту з поставленим рішенням; управління стабілізацією.

ЛЕКЦІЯ 5
КОНТЕКСТ ЗАВДАННЯ АНАЛІЗУ ВИМОГ

Результати аналізу вимог багато в чому визначають успіх проекту, але роль бізнес-аналізу та бізнес-моделювання не настільки очевидна. Тому варто розібратися в якому випадку слід застосовувати аналіз вимог, бізнес-аналіз або бізнес-моделювання

Аналіз вимог , бізнес-аналіз , аналіз проблемної області

В даний час існують вже сотні методик , методологій , процесів , стандартів, що регламентують ті чи інші деталі вибору ікомплексування потоків робіт при розробці автоматизованих інформаційних систем . Те , що в АТ стоїть на початку ланцюжка робіт і що її результати багато в чому визначають успіх проекту , мало у кого викликає сумніви. Інша справа - роботи , пов'язані з бізнес - аналізом і бізнес- моделюванням . Їх роль не настільки очевидна і приймається далеко не всіма методологіями . Отже , чи варто збирати інформацію про підприємство , для якого розробляється (вибирається) АІС у вигляді бізнес - моделей або варто пропустити цей етап і відразу формувати артефакти АТ ?

Автори [5.1] , " батьки-засновники " RUP і UML , в цьому питанні дають певну свободу : можна створювати бізнес - моделі за допомогою відповідних розширень UML і рекомендацій RUP, а можна обмежитися виробленням глосарію об'єктів предметної області. Як і в питанні вибору глибини розрахунку артефактів АТ , питання - проводити чи не проводити бізнес -аналіз (або , точніше кажучи , аналіз проблемної області) , вирішується залежно від конкретного завдання .

Роль глосарію при АТ .

Трохи перебільшуючи , можна сказати , що Замовник і Дизайнер завжди говорять на різних мовах. Загальне розуміння виробляється насилу , цей процес займає час , але важливість його важко переоцінити : адже успішна реалізація проекту в області та впровадження АІС в чому залежить від того, чи вдасться виробити і документувати їх загальне уявлення про предмет розробки. Якщо ж Дизайнер йде ще далі і вникає в особливості ведення справ на підприємстві Замовника - він , по-перше , зможе домогтися кращого розуміння вимог до АІС і , по-друге , брати участь поряд із Замовником у формулюванні цих вимог , аналізі пропущених вимог і пр. глосарій (докладніше див влекції 8) можна розглядати як документ , що засвідчує спільне розуміння основної термінології Замовником та Розробником .

Задачу аналізу бізнес- процесів (ділове моделювання) , настільки популярну в останні десятиліття зважаючи стійкої кон'юнктури , слід розглядати як частину більш загальної задачі систем для аналізу проблемної області. Роботи , присвячені аналізу проблемної області , з'явилися у вітчизняній літературі в середині минулого століття ; дана тематика нерозривно пов'язана з задачного підходу і інженерією експертних систем . Чи застосовні методи , прийняті при побудові інтелектуальних систем для такої , "більш приземленою " завдання, як завдання побудови АІС - безумовно , так. Так , стратегії вилучення знань, багато в чому перетинаються з рекомендаціями по роботі аналітика, методи розв'язання задачі шляхом редукції на підзадачі і пошуку в просторі станів знайшли своє відображення у безлічі методик бізнес-аналізу , аналізу та синтезу програмних систем , і цей список можна продовжувати. Інше питання - наскільки результативно застосування тих чи інших моделей і методів при описі організаційних систем .

Ключ до вирішення цього питання лежить у наступному : спочатку треба визначити цілі і завдання самого бізнес-аналізу , як етапу побудови КІС.
З позицій моделювання , аналіз вимог (АТ) і аналіз проблемної області (АПО) - принципово різні процеси .

АПО переслідує класичні цілі створення моделі: у наявності об'єкт (автоматизується підприємство або організаційна система , ОС) і завдання аналітика - відобразити цей об'єкт в створюваній моделі з необхідної ступенем точності.

Аналіз вимог , навпаки , спрямований на моделювання уявного , ще не існуючого об'єкта (АІС). Тобто спочатку створюється модель , а потім , на її підставі , синтезується об'єкт .

Для того , щоб прояснити зв'язок між цими процесами , необхідно зауважити , що створювана АІС також є моделлю стосовно ОС. Таким чином , створюючи документ АТ , ми тим самим породжуємо як би " модель другого порядку " , тому що документ АТ є нічим іншим , як моделлю моделі ОС. Не володіючи моделлю АПО , ми , звичайно , можемо створити модель АТ . Але при цьому ми ризикуємо тим , що при синтезі оригіналу моделі (тобто АІС) , не володіючи знаннями про ОС , ми можемо потрапити в ситуацію неузгодженості : результуюча АІС НЕ буде інгерентно (узгоджена з) ОС і , тим самим, не стане життєздатною.

Чи випливає з цього , що етап АПО є необхідною ланкою створення КІС ? Ні, не завжди. Тут доречно звернутися до класифікації завдань і методологій А.

Коберна . Крім того , це залежить від складу третьої компоненти " трикутника моделювання " - моделює суб'єкта , в нашому випадку - колективу Розробника . Якщо моделюючий суб'єкт має неявними знаннями про ОС в достатньому обсязі - значить , АПО можна виключити. На практиці це можливо в наступних випадках :

• Розробник є частиною (структурним підрозділом , дочірнім підприємством і т.д.) ОС , в колектив Розробника входять експерти , які добре знають предметну область;

• Замовник нарівні з Розробником бере участь у створенні документа АТ і розділяє з ним відповідальність за прийняття рішень. Це - шлях " agile методологій ".

Розглянемо тепер узагальнену " формулу " створення АІС . ОС - > М (ОС) -> М (АІС) -> М ' (АІС) -> М " (АІС) -> М '" (АІС) -> АІС .

Аналіз організаційної системи дозволяє створити її модель М (ОС). Це - модель бізнес- аналізу (проблемної області) .

Аналізуючи модель проблемної області , в ній можна виокремити ,

• з одного боку , завдання і функції, реалізовані всередині ОС і функції комунікації ОС і середовища ,

• з іншого - пристрій предметної області (на початку - на рівні концептуальної моделі) ,

• з третього - вимоги до інформації та її обробці.

Виділивши серед функцій ті , які підлягають автоматизації , ми отримуємо основу для виявлення функціональних вимог до системи . Решта зібрана на етапі АПО інформація служить для пошуку нефункціональних вимог. В результаті отримуємо модель АТ , як перше наближення моделі АІС , М (АІС) .

Потім , шляхом поглибленого аналізу і проектування , формуються , відповідно, аналітична модель М ' (АІС) , проектна модель М " (АІС) і модель реалізації М '" (АІС) .

Модель рівня реалізації дозволяє синтезувати власне АІС , як сукупність програмних , інформаційних , організаційних та ін артефактів.

АІС в свою чергу представляє собою модель організаційної системи М ' (ОС) , замикаючи цикл моделювання.

Методології бізнес-аналізу

Методології бізнес аналізу можна розділити на три категорії за типами моделей:

• моделі , які мають на меті аналізу та покращення організаційної системи (наприклад , SWOT , VCM , BPR , CPI/TQM/ISO9000 , BSC) ,

• моделі загального призначення , такі , як SADT , DFD , IDEF1 , IDEF3 , IDEF5 та інші ,

• моделі , спеціально розроблені для використання при автоматизації (наприклад , ISA , BSP , ARIS , RUP) .

Найбільш розвинена модель опису проблемної області пропонується в методології ARIS .

Архітектура ARIS виділяє в організації наступні підсистеми .

• Організаційна . Визначає структуру організації - ієрархію підрозділів , посад і конкретних осіб , різноманіття зв'язків між ними , а також територіальну прив'язку структурних підрозділів .

• Функціональна . Визначає функції , виконувані в організації .

• Підсистеми входів / виходів . Визначають потоки використовуються і виробляються продуктів і послуг.

• Інформаційна (підсистема даних). Описує отримання , поширення та доступ до інформації (даних).

• Підсистема процесів управління . Визначає логічну послідовність виконання функцій за допомогою подій і повідомлень. Можна сказати , що підсистема управління - це сукупність рознесених у часі повідомлень різного роду .

• Підсистема цілей організації. Описує ієрархію цілей, що досягаються в ході виконання того чи іншого процесу .

• Підсистема засобів виробництва . Описує життєвий цикл основних і допоміжних засобів виробництва.

• Підсистема людських ресурсів . Описує прийом на роботу , навчання і просування по службі персоналу організації .

• Підсистема розташування організаційних структур. Описує територіальне розташування організаційних одиниць.

Дане поділ є до певної міри умовним; виділені "підсистеми" не є підсистемами в сенсі системного аналізу, тому що взаємопроникають і перетинаються. Вони представляють швидше сукупність предметів дослідження, різних поглядів на досліджуваний об'єкт.

Слухачеві курсу пропонується самостійно проаналізувати, які групи і категорії вимог до системи дозволяє прояснити та чи інша компонента прийнятої в ARIS структуризації об'єкта дослідження.

Вимоги та архітектура АІС

Говорячи про архітектуру АІС , зазвичай підкреслюють поділ на апаратні , програмні , інформаційні , організаційні компоненти , їх зв'язність і деталізацію .

Метафора архітектури RUP описується у вигляді 4 +1 уявлень : логічне , уявлення процесів , уявлення реалізації та фізичне уявлення зв'язуються між собою поданням варіантів використання (use case) , яке відіграє центральну роль у виробленні архітектури системи.

Вимоги первинні по відношенню до архітектури . Але це не означає , що вимоги та архітектура повинні розроблятися послідовно .

Навпаки , ці процеси взаємопов'язані і мають бути суттєво Запаралеленими . Як тільки зібраний мінімальний набір ключових вимог , що дають розуміння про те , що потрібно робити - повинна бути знайдена архітектура , яка пояснює , як це може бути реалізовано . У великих , відповідальних проектах зазвичай розглядається кілька альтернативних архітектур , їх достоїнства і недоліки стосовно до вихідним вимогам .

У процесі роботи з вимогами вони деталізуються , деталізується та архітектура . У разі множинності альтернативних архітектур на певному рівні деталізації необхідно зупинитися на одній , щоб не розпорошувати ресурси . Але природа вимог така, що , крім деталізації вони ще й змінюються . Зі зміною

вимог змінюються і деталі архітектури. Стійкість архітектури проявляється в незначних її змінах при додаванні , деталізації та зміні вимог . Якщо настав момент , при якому поява нової інформації про вимоги перестало впливати на архітектуру - значить , архітектура стабілізувалася.

Це - нормальний , природний шлях розвитку вимог і архітектури. Але що робити , якщо вимоги змінилися настільки , що архітектура перестала їм відповідати ? Причини тому можуть бути різні , наприклад : недосвідчена архітектурна група не виявила достатньо далекоглядності ; група по збору вимог пропустила на ранніх стадіях критичні , архітектурно значущі вимоги ; в бізнес- сфері Замовника відбулися великі зміни , що викликали корінну зміну вимог до системи . Слідства також можуть бути різними: домовленість про збільшення термінів і сум за контрактом; виправлення ситуації за рахунок Розробника ; розрив контракту .

Альтернативний вихід пропонується в методології XP : архітектура - не догма , а всього лише метафора. Якщо вимоги увійшли врозріз з існуючою архітектурою - значить , архітектуру потрібно просто змінити. Слід розуміти , що шлях і рецепти XP при уявній простоті орієнтовані далеко не на будь-який колектив . Команда XP складається з професіоналів, які мають позитивний досвід роботи в цій методології .

Аналіз вимог та інші робочі потоки програмної інженерії

Розглянемо короткий огляд робочих потоків RUP і їх зв'язок з потоком робіт АТ.

Потік робіт " ділове моделювання" служить основою для аналізу та формування вимог до АІС , дозволяє уникнути помилок.

Потік робіт "управління середовищем " надає вихідну інформацію для робочої групи АТ , яка регламентує формати оформлення , CASE -засоби , регламенти роботи .

Потік робіт "управління проектом" грунтується на специфікації вимог. Стратегічне і тактичне планування , формування проміжних віх (очікуваних результатів) тісно ув'язано з вимогами до системи .

Потік робіт " аналіз і проектування " здійснюється на основі вихідних даних , наданих АТ . Певною мірою ці потоки робіт проводяться паралельно. При виявленні проблем , пов'язаних з вимогами , виникає зворотній зв'язок від цього потоку робіт до потоку робіт АТ .

Потік робіт " випробування" багато в чому базується на моделі вимог та додаткових специфікаціях , що регламентують процес тестування (тестові сценарії та ін.)

Для потоку робіт "реалізація " зв'язок з вимогами не вказана . Тим часом автор вважає , що вимоги повинні аналізуватися і враховуватися у ВСІХ робочих потоках проекту , навіть якщо це формально не передбачено обраним групою процесом . Людям властиво помилятися і помилки, здійснені на ранніх стадіях проекту , при русі від етапу до етапу наростають , як снігова куля. Тому будь-якому учаснику команди , зацікавленій в успіху проекту , не зайве зазирнути в специфікацію вимог і переконатися в тому , що та робота , яка йому доручена , відповідає тому або іншому вимогу. Це дозволяє організувати зворотний зв'язок, що дозволяє відстежити помилки в специфікаціях . Багато

проектів зайшли в глухий кут саме через відірваність групи , що відповідає за реалізацію від групи збору і аналізу вимог .

ЛЕКЦІЯ 6
ВИЯВЛЕННЯ ВИМОГ

У цій лекції будуть розглянуті основні джерела вимог. Детальніше зупинимося на стратегіях виявлення даних вимог: інтерв'ю, анкетування, спостереження і т.п.

Джерела вимог

Основним джерелом вимог до інформаційної системи , безумовно , є міркування , висловлені представниками Замовника. Згідно з ієрархічною моделлю вимог дана інформація структурується як мінімум на 2 рівня:

• бізнес- вимоги і

• вимоги користувачів .

Проблема полягає в тому , що вимоги формулюються до створюваної , ще не існуючій системі , тобто по суті вирішується початкова підзадача задачі проектування АІС , а представники Замовника далеко не завжди бувають компетентні в даному питанні. Тому , поряд з вимогами , висловленими Замовником , доцільно збирати і вимоги від інших співвласників системи : співробітників аналітичної групи виконавця , зовнішніх експертів і т.д.

Результуючий , часто досить сирий матеріал розглядається , як документ "Вимоги співвласників " 1) . На вимоги співвласників зазвичай не накладається ніяких спеціальних обмежень.

Продовжуючи міркування , розпочаті в попередній лекції , модель створюваної інформаційної системи в певній мірі повинна відображати модель ОС.

Тому іншим важливим джерелом інформації , крім виявлення вимог , є артефакти , що описують предметну область . Це можуть бути документи з описом бізнес -процесів підприємства , виконані консалтинговим агентством , або просто документи (посадові інструкції, розпорядження , склепіння бізнес - правил) , прийняті на підприємстві . Однією з небагатьох методологій , в якій спеціально виділяється робочий потік ділового моделювання , є Rational Unified Process .

Ще одна альтернатива , яка використовується при виявленні вимог - так звані "кращі практики " , широко використовуються в даний час в бізнес-консалтингу і при впровадженні корпоративних інформаційних систем . Кращі практики являють собою описи моделей діяльності успішних компаній галузі , використовувані тривалий час у сотнях і тисячах компаній по всьому світу.

Підсумовуючи сказане , зазначимо , що основними джерелами , що утворюють "вхід" процесу виявлення вимог , є вимоги , висловлені співвласниками , як такі або (і) артефакти , що описують об'єкт дослідження . Однак , це - досить спрощений погляд : щоб дані надійшли " на вхід" , аналітики вимог повинні виконати чималу роботу , пов'язану з підбором респондентів та інформаційних матеріалів , організацією інтерв'ю і т.д.

Стратегії виявлення вимог

Інтерв'ю

Ключовою стратегією виявлення вимог було і залишається інтерв'ю з експертами.

У тепер класичною, але нітрохи не загубила актуальність монографії Д.Марко в процесі проведення інтерв'ю пропонується виділити три підлеглих процесу: підготовку, проведення інтерв'ю (опитування) і завершення. Нижче наводиться короткий огляд рекомендацій Д.Марко з акцентом на виявлення вимог (у монографії дано рекомендації з інтерв'ювання з метою формування моделі об'єкта дослідження).

1. Підготовка

Підготовка дозволяє спланувати процес опитування і виробити стратегію управління цим процесом. Важливість підготовчого етапу виростає , якщо респондент є " дефіцитним " корисним ресурсом , наприклад - президентом великої компанії.

При підготовці Д.Марко рекомендує наступні кроки:

• виберіть потрібного співрозмовника ;

• домовтеся про зустріч ;

• встановіть попередню програму зустрічі ;

• вивчіть супутню інформацію ;

• узгодьте свої дії з групою проектування .

При виборі співрозмовника для цілей збору вимог визначальними є дві речі:

• Він дійсно є експертом з даного питання ;

• Його думка дійсно є цінним при формуванні цільового набору вимог .

Важливо заздалегідь обумовити мету зустрічі і обмежити бесіду в межах години -менш . Практика показує , що активне спілкування в процесі інтерв'ю , як правило , обмежується годиною. Якщо цього часу недостатньо , можна спланувати кілька зустрічей.

Корисними прийомами є формування програми бесіди і ознайомлення з нею респондента , докладний планування бесіди аж до запису підготовлених питань . Підготовлене таким чином інтерв'ю називають структурованим . На додаток до так побудованому інтерв'ю автор пропонує проводити неструктуроване інтерв'ю , " представляє собою неформальну зустріч , якої не властиві заготовлені про запас питання або заздалегідь поставлені цілі " . Мета такого інтерв'ю - пробудити респондента до креативу в області , в якій інтерв'юер недостатньо добре орієнтується .

2. Проведення опитування

У проведенні опитування найважливіше - правильно організувати і підтримувати потік інформації від експерта до вас. Рекомендується витратити час на обмірковування вірного початку опитування, при зборі інформації по можливості використовувати записи , закінчувати розмова плавно . Обговоримо докладніше кожен з цих пунктів.

Починаючи розмову , не забудьте представитися і сформулювати мету зустрічі. Це допоможе уникнути непорозумінь і дасть бесіді правильний напрямок . Крім того , обговоріть можливість ведення записів .

Потім сформулюйте перше питання . Пам'ятайте , що перше питання часто задає тон всьому розмови , тому добре продумайте його .

Збирайте інформацію , роблячи записи про все (про спеціальні термінах , взаємозв'язках між частинами системи і т.п.) і обмежуючи час бесіди. Запишіть SADT- функції і дані , спробуйте накидати діаграму. Підтримуйте потік інформації , ставлячи питання , які уточнюють і підтверджують відповіді .

Насамперед, не заперечуйте .

Ніколи не ставте навідних питань або питань з короткими відповідями "так" або "ні". Замість цього записуйте те , що вам говорять , і просите підвести підсумок або дати пояснення.

Ви отримаєте від опитування більше , якщо ви дасте експерту можливість говорити те , що він хоче сказати , а не те , що ви хочете почути.

3. Завершення

Слідкуйте за виникненням наступних ситуацій:

• ви вже отримали достатньо інформації ;

• ви отримуєте великий обсяг невідповідної інформації;

• велика кількість інформації вас пригнічує ;

• експерт починає втомлюватися ;

• у вас з експертом часто виникають конфлікти.

Будь-яка з цих причин - достатня підстава для завершення бесіди .

Коли ви вважаєте за потрібне закінчити опитування, завершуйте бесіду плавно. Коротко Підсумуйте основні пункти та зробіть огляд отриманих відомостей , які можуть бути опущені або невірно витлумачені . Домовтеся про час наступної зустрічі, якщо вона потрібна, та отримайте рекомендації для найближчих опитувань. Поставте експерта до відома , коли і як ви збираєтеся використовувати отриману інформацію і коли ви надішлете йому матеріал на рецензування .

Завжди оформляйте матеріали опитування відразу ж після зустрічі з експертом. У цьому випадку негайно виникає зворотній зв'язок , і ви мінімізуєте можливість втрати важливої інформації.

Що потрібно пам'ятати при опитуванні

Наступні рекомендації допомагають підтримувати безперервність потоку і достовірність інформації, що надходить від експерта:

• робіть паузи , поки експерт думає. Дайте експерту можливість вирішувати , що сказати далі . Ніколи не перебивайте , підказуючи відповідь або задаючи інше питання ;

• намагайтеся не задавати навідних питань , питань -підказок , питань, що містять відповідь , тому що це не дозволяє експерту ділитися своїми знаннями . Намагайтеся не ставити контрольних питань , оскільки це перериває потік інформації;

• робіть записи , щоб зосередитися на предметі розмови і щоб підготуватися до наступного питання , але не стаєте стенографом , інакше ви можете втратити контроль над опитуванням.

Анкетування

Анкетування - самий маловитратний для аналітика спосіб добування інформації , він же - і найменш ефективний . Звичайно застосовується як доповнення до інших стратегіям виявлення вимог.

Недоліки анкетування очевидні: респонденти часто бувають нездатні , або слабо мотивовані в тому , щоб добре і інформативно заповнити анкету.

Велика ймовірність отримати неповну або зовсім неправдиву інформацію. Перевага - в тому , що підготовка і аналіз анкет вимагають невеликий ресурс .

Рекомендується формулювати в анкетах питання із замкнутим циклом відповідей в одній з наступних трьох форм .

Багатоальтернативного питання. Ця форма анкети відома всім , хто коли небудь проходив тестування; може розширюватися коментарями респондента у вільній формі.

Рейтингові питання. Представляють визначений набір відповідей на сформульовані запитання . Використовуються такі значення, як "абсолютно згоден", "згоден", "ставлюся нейтрально", "не згоден", "абсолютно не згоден" , " не знаю" .

Запитання з ранжируванням. Передбачає ранжування (упорядкування) відповідей шляхом привласнення ним порядкових номерів, процентних значень і т.п.

Спостереження

Спостереження за роботою модельованої організаційної системи - корисна стратегія отримання інформації (хоча , строго кажучи , за результатами спостереження можна отримати модель ОС , а не модель АТ) .

Розрізняють пасивне і активне спостереження. При активному спостереженні аналітик працює , як учасник команди , що дозволяє поліпшити розуміння процесів .

Через спостереження, а можливо, і участь аналітики отримують інформацію про події день за днем операціях з перших рук. Під час спостереження за роботою системи часто виникають питання, які ніколи б не з'явилися , якби аналітик тільки читав документи або розмовляв з експертами.
Недоліком цієї стратегії є те, що спостерігач, як і всякий " вимірювальний прилад", вносить перешкоди в результати вимірювань: співробітники організації , перебуваючи "під ковпаком" можуть почати поводитися принципово по іншому , ніж зазвичай.

Самостійний опис вимог

Если опытный аналитик уже исследовал большое число систем такого же типа, что и на предприятии внедрения, он обладает фундаментальными знаниями в соответствующей предметной области, относительно определенного класса систем. Авторы методологии SADT рекомендуют проводить самоопрос с тем, чтобы получить максимальную пользу от своих знаний.

По результатам анализа документов и собственных знаний аналитик может составить описание требований и предложить его представителям Заказчика в качестве информации к размышлению, либо - основы для формирования технического задания.

Недостаток этой стратегии - опасность пропуска знаний, специфичных для объекта исследования (в случае самоопроса), либо - неформализованных знаний, эмпирических правил и процедур, широко используемых на практике, но не вошедших в документы.

Спільні семінари

Крім класичного інтерв'ю " тет а тет" , існує значна кількість методик, які передбачають широку участь представників Замовника та Виконавця .

Правила мозкового штурму припускають повну розкутість і свободу думок , навіть самих химерних і на перший погляд " маячних " . Перше правило мозкового штурму - " повна заборона на будь-яку критику " . Всяке висловлену думку представляє цінність , а повна відсутність заборон дозволяє повноцінним чином підключити творчу фантазію.

Потім , на другому етапі , всі висловлені думки ретельно обговорюються , завідомо неприйнятні варіанти відсіваються , формуються колективні пропозиції .

Правила JAD - методу , що вважається одним із сучасних способів вилучення вимог , були вперше сформульовані в кінці 1970 - х років компанією IBM. Учасники JAD -наради:

Ведучий - фахівець в області міжособистісних комунікацій . Повинен орієнтуватися в предметній області , але не обов'язково добре орієнтуватися в проблемах IT.

Секретар - стенографіст зустрічі. Фіксує її результати на комп'ютері. Можливе застосування CASE -засобів.

Замовники - користувачі або керівники , основні учасники , що формують , які обговорюють вимоги та приймають рішення .

Розробники - аналітики та інші учасники проектної команди. Працюють в більшій частині в пасивному режимі з метою найкращого розуміння проблемної області.

Спільні семінари , зберігаючи всі переваги режиму інтерв'ю , привносять додаткові бонуси: робота в групі більш продуктивна , групи швидше навчаються , більш схильні до кваліфікованих висновків , дозволяють виключити багато помилок .

Ця стратегія , очевидно , одна з найбільш витратних , проте вона окупається за рахунок меншої кількості помилок і відмову від формалізації на користь живого спілкування , вироблення спільної мови і пр. Деякі методології (наприклад , XP) грунтуються на постійному тісному контакті між Замовником та Виконавцем та , якщо такої можливості немає - XP- проект просто не зможе відбутися .

" Роз'яснює зустрічі" або " запланований мозковий штурм " - термін, що прийшов із загальної практики менеджменту і базований на ідеях співробітництва зацікавлених осіб для спільного аналізу шляхів вирішення проблем , визначення та попередження ризиків і т.п.

Прототипування

Прототипування - ключова стратегія виявлення вимог в більшості сучасних методологій. Програмний прототип - "дзеркало" , в якому видно відображення того, як зрозумів Виконавець вимоги Замовника. Процес виявлення вимог шляхом прототипування тим більш інтенсивний, ніж це дзеркало кривее. Документальний спосіб виявлення вимог завжди поступається живому спілкуванню. Аналіз того, що зроблено у вигляді інтерфейсів користувача дає ще більший ефект. Підключається правопівкульний канал сприйняття, який , як відомо, працює у більшості людей на порядок ефективніше , ніж вербальний .

Метод RAD - один з найбільш відомих способів швидко створювати прототіпи1) .

RAD базується на наступних базових принципах:

• Еволюційний прототипування ;

• CASE -засоби , як основний інструмент , включаючи можливості прямого і зворотного проектування і автоматичної генерації коду;

• Висококваліфіковані фахівці , що добре володіють розвиненими інструментальними засобами ;

• Інтерактивний JAD - метод , в якому спілкування поєднується з розробкою в режимі online ;

• Жорсткі часові рамки , як протиотруту від "розповзання кордонів" проекту: якщо команда не вкладається в строк - функціонал звужується.

ЛЕКЦІЯ 7
ФОРМУВАННЯ БАЧЕННЯ

Роботи з формування бачення продукту і меж проекту зазвичай починаються на самій ранній фазі проекту, до початку широкомасштабних консультацій з виявлення підробних вимог, тому це питання не можна залишати без уваги.

Бачення продукту і межі проекту

Роботи з формування бачення продукту і меж проекту зазвичай починаються на самій ранній фазі проекту, до початку широкомасштабних консультацій з виявлення підробних вимог, хоча в цілому наявність і послідовність даних кроків залежить від обраної методології. На практиці дані роботи часто поєднуються. Правила вилучення вимог, розглянуті в лекції 6, можуть бути використані і при формуванні бачення

Аналізуючи літературу з даної тематики, можна виділити наступні широко вживані ключові слова : з одного боку - концепція , бачення , образ , з іншого - рамки , межі , контекст .

У першому випадку мова йде про бачення того , якою має бути система . Обговорюються високорівневі вимоги (можливості , властивості) продукту і найбільш істотні обмеження. Ряд авторів , навпаки , наполягає на тому , що бачення повинно бути " нічим не обмеженим " .

Поняття бачення широко вживано в бізнес- аналізі . Якщо у топ - менежменту компанії є уявлення про те, які ключові цілі , сегменти ринку , товарні позиції , прибуток повинні бути досягнуті , припустимо , через 5 років - значить , компанія має довгострокове бачення себе на ринку . Спосіб зняття обмежень при виробленні бачення дозволяє виробити новий погляд на речі , " піднятися над ситуацією" , планувати майбутнє , відштовхуючись немає від поточних ресурсів і обмежень , а від стратегічних цілей , застосовуючи інновації , ноу -хау тощо

Даний досвід формування бачення багато в чому переносимо і на процес розробки інформаційних систем : потрібно " побачити" в горизонті середньо-і (або) довгострокового планування , як АІС впишеться в організаційні процеси підприємства , які ключові вигоди вона дасть , які проблеми дозволить вирішити. При пошуку нових методів і засобів управління підприємством на основі інформаційних технологій часто доводиться " перекроювати " існуючі бізнес - процеси ; по суті впровадження АІС , яка зачіпає істотний відсоток

процесів підприємства , неминуче призводить до перебудови цих процесів з метою оптимізації діяльності підприємства , досягнення ключових факторів ефективності та пр.

У другому випадку (рамки , межі , контекст) обговорюються такі питання , як межа системи і середовища , необхідні ресурси на створення системи , терміни. Побудувавши " нічим не обмежене бачення " , рано чи пізно доводиться повернутися до таких прозовим речам , як бюджет , календарне планування , підбір персоналу , віхи проекту.

Чи завжди потрібно створювати документ " Концепція " ? Чи слід розділяти бачення і межі ?

Найчастіше Замовник усвідомлює необхідність автоматизації , як спосіб вирішення накопичених проблем. Сформулювавши для себе проблему , Замовник часто бачить і варіант її рішення , з яким приходить до Виконавця ("мені потрібен сайт " , " потрібно CRM -система" і т.п.). Кваліфікований Виконавець не повинен , стрімголов , поспішати вирішувати завдання в формулюванні Замовника. За образним висловом Г.Калянова1) автоматизувати процеси " як є" - все одно , що асфальтувати доріжки , по яких ходять корови.
У нотації RUP присутній важлива метафора : "Побачити проблему за проблемою". Концепція якраз і служить для того , щоб допомогти Замовнику виявити саме ті вимоги до системи, які допоможуть йому оптимізувати роботу свого підприємства в довгостроковій перспективі.

Тому етап формування концепції важливий, але він пред'являє і до Замовника та до Виконавця досить високі вимоги : Замовник повинен виділити ресурси і бути готовим до трудовитрат на спільний пошук рішень ; Виконавець повинен володіти достатньою кваліфікацією як у сфері IT- , так і в сфері управління підприємствами , щоб розроблювальний засіб автоматизації дійсно принесло користь. Все вищесказане нітрохи не виключає можливість роботи без концепції: або мова йде про невеликий проекті , закладати в бюджет якого етап вироблення концепції просто нерентабельно , або Замовник сам володіє достатньою кваліфікацією , щоб сформулювати вимоги до АІС , маючи " концепцію в голові" і час для консультування розробника.

Деякі аргументи за розділення бачення і кордонів були наведені вище . Провести чітку межу між цими поняттями пропонує, зокрема , процес MSF . Зрештою , питання " розділяти або не розділяти " визначається обраною методологією.

Розглянемо основні вимоги до вироблення концепції , закладені у вітчизняних ГОСТ , методологіях RUP і MSF .

Концепція в ГОСТ РФ

Відповідно до ГОСТ 34.601-90 "Автоматизовані системи. Стадії створення ", після етапу формування (виявлення) вимог до системи виконується етап розробки концепції системи .

Основні роботи етапи:

- Вивчення об'єкта;
- Проведення науково -дослідних робіт (НДР) ;
- Розробка варіантів концепції АС ;
- Оформлення звіту про виконану роботу .

Так як даний етап хронологічно стоїть на другому місці , до його початку у Розробника на руках вже є документ , в якому зібрані основні вимоги користувачів .

Роботи над концепцією починаються з обстеження об'єкта автоматизації . Виконуються НДР , спрямовані на дослідження принципової реалізованості вимог і можливих варіантів реалізації .

ГОСТ , на відміну від більшості сучасних методологій , в загальному випадку закладає Багатоальтернативність варіантів концепції системи і планів їх реалізації. Кожен з опрацьованих варіантів оцінюється з позицій необхідних ресурсів і функціональності. Для варіантів повинні бути подані оцінки переваг і недоліків. Корисність опрацювання кількох варіантів концепції полягає в тому , що Замовнику важко сформулювати самостійно бачення системи , в той час , як вибір з набору варіантів , представлених Розробником - цілком посильне завдання .

Крім того , концепція повинна відображати оцінки якості , умови приймання системи , оцінку ефекту , очікуваного від реалізації . При оформленні звіту необхідно привести обгрунтування пропонованого варіанту .

Бачення в RUP

Кроки, які необхідно пройти для формування документа "Бачення":

- Формулювання проблем.
- Ідентифікація співвласників
- Визначення меж системи
- Ідентифікація обмежень
- Формулювання постановки завдань
- Визначення можливостей системи
- Оцінка результатів

Для опису проблем пропонується шаблон, показаний в табл. 7.1

Таблиця 7.1.	
Проблема	**(опис проблеми)**
Зачіпає	(співвласники, які поставлені проблемою).
Її наслідком є	(який вплив проблеми).
Успішне вирішення	(список деяких ключових переваг від успішного рішення).

Ідентифікація співвласників передбачає пошук і фіксацію інтересантів проекту - представників Замовника та Виконавця , інвесторів , зовнішніх експертів і пр.

Визначення меж системи являє собою нетривіальний процес . Для цього використовують контекстні діаграми. RUP в пошуку кордонів пропонує відштовхуватися від акторів і варіантів використання.

Серед джерел обмежень зазвичай виділяють:

- Політичні ,
- Економічні ,
- Середовища ,
- Технічні ,
- Виконання ,
- Системні .

Опис можливостей системи являє собою формулювання високорівневих вимог.

Шаблон документа "Vision" RUP містить такі основні розділи :

1 . Введення

2 . Позиціонування

3 . Описи співвласників і користувачів

4 . Короткий огляд вироби

5 . Можливості продукту

6 . Обмеження

7 . Показники якості

8 . Старшинство і пріоритети

9 . Інші вимоги до виробу

10 . Вимоги до документації

11 . Додаток .

У вступі описуються мета документа , його контекст (зв'язок і взаємовплив з різними проектами) , визначення , акроніми та скорочення, посилання на інші документи , короткий зміст .

У розділі " позиціонування " поміщається визначення розв'язуваної проблеми (проблем) , вказується цільової замовник і досліджуються ділові переваги вироби перед аналогічними на ринку.

В описі співвласників і користувачів , крім власне опису цих двох груп , досліджується демографія ринку: цільові ринкові сегменти , розмір і темпи зростання ринку , існуючі конкурентні пропозиції на ринку , репутація Розробника на ринку.

Короткий огляд виробів містить резюме вироби , опис його перспектив і ключових можливостей , припущення і залежності , вказується вартість та її калькуляція , розглядаються питання ліцензування та інсталяції.

У розділі , присвяченому можливостям продукту , вони описуються більш докладно , кожна - в окремому параграфі.

В розділ " Обмеження " слід виносити існуючі технічні , технологічні та ін обставини , які необхідно враховувати на даній стадії .

Розділ " Показники якості " містить опис найбільш істотних функціональних вимог до системи (ефективності , надійності , відмовостійкості та ін.)

Розділ " Старшинство та пріоритети" ранжує сформульовані раніше вимоги і можливості системи за ступенем важливості , черговості реалізації тощо

Розділ " Інші вимоги до виробу " описує застосовувані стандарти , системні вимоги , експлуатаційні вимоги , вимоги до навколишнього середовища.

У вимогах до документації наводяться ключові характеристики керівництва користувача , інтерактивної довідки , керівництва по установці і конфігуруванню , файла**Read Me .**

У додаток виносяться атрибути можливостей. RUP рекомендує наступний набір атрибутів: статус, вигода, обсяг робіт, ризик, стабільність, цільової випуск, призначення, причина.

Бачення / рамки в MSF

На фазі вироблення концепції (envisioning phase) закладається одна з фундаментальних основ успіху проекту - **створення і згуртування проектної групи** на основі вироблення єдиного бачення . Проектна група повинна **чітко уявити собі** , що вона хоче зробити для замовника і сформулювати свою мету таким чином , щоб максимально мотивувати як замовника , так і саму проектну команду. Вироблення високорівневого погляду на цілі та умови проекту може розглядатися як рання форма планування; вона готує грунт для процесів створення детальних планів , які будуть здійснені безпосередньо під час фази планування .

Основними завданнями фази вироблення концепції є створення ядра проектної групи (див. нижче) і підготовка **документа загального опису і рамок проекту** (vision / scope document). Формування бачення проекту та специфікування його рамок - не одне і теж, хоча для успіху проекту необхідно і те, і інше. Бачення (vision) - це нічим не обмежується уявлення про те, яким має бути рішення. Рамки (scope) ж дають чіткі межі того, що із запропонованого цим баченням буде реалізовано в умовах існуючих проектних обмежень.

Управління ризиками являє собою ітеративний процес , здійснюваний протягом усього життєвого циклу проекту. Під час фази вироблення концепції проектна група готує документ оцінки ризиків та представляє головні ризики проекту разом із загальним описом і рамками проекту. Для отримання подальшої інформації про управління ризиками , см. "Білу книгу" дисципліни управління ризиками MSF .

Також під час фази вироблення концепції проводиться виявлення та аналіз **бізнес -вимог** . Більш детально ці вимоги розглядаються під час фази планування .

Провідним рольовим кластером на фазі вироблення концепції є "Управління продуктом".

Шаблон MSF містить такі розділи:

- Бізнес-переваги,
- Опис переваг
- Формулювання бачення
- Аналіз вигод
- Концепція рішення
- Цілі, завдання, припущення та обмеження
- Аналіз застосовності
- Вимоги
- Рамки
- Список характеристик / функцій
- Поза рамками
- Стратегія підготовки релізів
- Критерії застосовності
- Експлуатаційні критерії
- Стратегії проектування рішення
- Стратегія проектування архітектури
- Стратегія технічного проектування

ЛЕКЦІЯ 8

КЛАСИФІКАЦІЯ І СПЕЦИФІКУВАННЯ ТРЕНОВАНІЙ

Підвищити рівень інформативності вимог можливе за допомогою оформлення їх у вигляді варіантів використання . Перш , ніж приступити до специфицирования вимог у формі варіантів використання , RUP рекомендує виявити реєстр акторів і варіантів іспользованія.Как раз про це ми і поговоримо в цій лекції

Актори і варіанти використання

Для того , щоб підвищити рівень інформативності вимог , усунути взаємні суперечності і домогтися виконання їх інших основних характеристик , здійснюється перехід від повністю неформалізованих текстів до частково регламентованим (наприклад , шаблонами MS Word) текстам , класифікація , привласнення наборів атрибутів , побудова моделей , прототипування .

Найпопулярнішим і досить ефективним способом підвищення інформативності вимог є оформлення їх у вигляді варіантів використання (use case).

Перш , ніж приступити власне до специфицирования вимог у формі варіантів використання , RUP рекомендує виявити реєстр акторов1) (actors) і варіантів використання.

Актор - це хтось або щось, що володіє активністю по відношенню до програмної системи . Якщо ви розробляєте простий текстовий редактор , то , швидше за все , вибір актора не складе особливих труднощів : це буде користувач , що набирає текст. Однак не завжди все так просто. Крім користувача в якості актора може розглядатися інша програмна система , апаратний пристрій , в ряді випадків - активна компонента самої системи. Пошук акторів корпоративної інформаційної системи зазвичай зводиться до аналізу ролей різних користувачів . Менеджер з продажу , старший менеджер і начальник відділу продажів - один актор , два чи три? Це залежить від їх функціональних обов'язків , розмежування доступу , способів використання інформаційної системи . Пошук акторів може здійснюватися , наприклад методом мозкового штурму . Надалі при необхідності знайдені актори можуть узагальнюватися , переглядатися і об'єднуватися .

Варіант використання в першому наближенні можна розглядати , просто , як функцію , реалізовану системою . Однак , сучасний погляд на організацію бізнесу говорить про те , що всяка функція повинна мати цінність для кінцевого споживача продукту або послуги. Філософія варіанту використання передбачає виділення серед всього функціоналу системи підмножини , корисного конкретному кінцевому користувачеві (точніше кажучи , типом кінцевого користувача) . Інша сторона - варіант використання повинен не тільки бути корисний , а ще й дозволяти отримувати КП конкретні закінчені результати. Так , однією з функцій текстового редактора , очевидно , є створення порожнього файлу. Але навряд чи КП буде використовувати редактор з метою виготовлення порожніх файлів. Отже, створення порожнього файлу - функція , але не варіант використання системи. Варіантом використання може бути , наприклад , підготовка в текстовому редакторі службової записки . Варіант використання реалізується через функції системи .

Глосарій

Крім формування вимог співвласників іншим результатом початкової фази виявлення вимог є концептуальний аналіз проблемної області. Найпершим результатом його є формування глосарію (словника) основних використовуваних термінів . Значення глосарію важко переоцінити: він є основою , ключем для однакового розуміння описів вимог Замовником та Розробником .

Крім того , глосарій є відправною точкою для побудови більш розгорнутих моделей проблемної області , які , на стадії реалізації інформаційної системи , лягають в основу об'єктної моделі (для об'єктно - орієнтованих додатків) і моделі даних (для генерації схеми бази даних).

Глосарій оформляється , як текст, що складається з абзаців , кожний з яких визначає значення одного з термінів проблемної області. Термін зазвичай виділяють напівжирним кеглем . Іноді проблемну область доцільно сегментувати на ряд " підобластей " (subject areas) . Тоді кожній з них в глосарії виділяється окремий параграф.

Специфікація варіанту використання

Існують різні шаблони опису варіантів використання. Основні стилі описи:

- Вільний формат,
- Повний формат (запропонований А. Коберном),
- Таблиця у дві колонки,
- Таблиця в три колонки,
- Стиль RUP.

Крім того, іноді доцільно використовувати:

- Псевдокод,
- Діаграму активності UML,
- Інші графічні моделі.

Вільний формат

Вільний формат передбачає опис дій користувача і системи в розповідному стилі , наприклад : "Менеджер запитує у Системи список замовлень за період.Системавідображаєнаекранізнайденізамовленняданого Менеджераіззазначеннямїхосновнихатрибутів ". Вільнийстильдопускаєвикористання конструкцій "Якщо те ". "Якщо Менеджер має повноваження Начальника Відділу, то Система надає можливість перегляду замовлень всіх менеджерів цього відділу ".

Шаблон повного опису варіанту використання за А. Коберн

Назва< коротка фраза у вигляді дієслова в невизначеною формі доконаного виду, що відображає мету >

Контекст використання< уточнення мети , при необхідності - умови її нормального завершення > .

Область дії< посилання на рамки проекту > . Наприклад - підсистема бухгалтерського обліку .

Рівень<один з трьох: узагальнений, цілі користувача, подфункции>. Автор ставить зумовлену трирівневу класифікацію вимог, в цілому відповідну класифікації вимог на бізнес-вимоги, вимоги користувачів і функціональні вимоги, див. лекцію 2.

Основна діюча особа<ім'я ролі основного актора або його опис>.

Учасники та інтереси<список інших акторів-учасників прецеденту із зазначенням їх інтересів>.

Передумова<те, що очікується, вже має місце>.

Мінімальні гарантії<що гарантується акторам-учасникам>. Наприклад - у разі невдалої транзакції всі дані, що були в системі до її початку, зберігаються незмінними.

Гарантії успіху<що отримають актори-учасники в разі успішного досягнення мети>.

Тригер<те, що "запускає" варіант використання, зазвичай - подія в часі>.

Основний сценарій<тут перераховуються кроки основного сценарію, починаючи від тригера і аж до досягнення гарантії успіху>.

Формат опису: <Номер кроку><Опис дії>

Розширення <тут послідовно описуються всі альтернативні сценарії>. Кожна з альтернатив прив'язана до кроку основного сценарію.

Формат опису: <Номер шага.Номер розширення><Умова>: <Дія або посилання на підлеглий варіант використання>.

Будь-який з кроків основного сценарію може мати 1 або більше розгалужень. Кожне розгалуження оформляється у вигляді розширення. У блоці "Розширення" всі розширення описуються послідовно.

У разі, якщо альтернативний сценарій не вдається описати одним рядком - застосовується наступний формат.

Починаючи з рядка, наступною після опису розширення , йде опис його дій у форматі основного сценарію:

< Номер шага.Номер расширенія.Номер кроку розширення >< Дія >

Опис розширення закінчується описом виходу з розширення. Основні варіанти виходу з розширення : повернення до чергового по номеру кроку основного сценарію , закінчення прецеденту , перехід до іншого кроку основного сценарію .

Список змін в технології і даних < що гарантується акторам - учасникам > . Наприклад - у разі невдалої транзакції всі дані , що були в системі до її початку , зберігаються незмінними .

Допоміжна інформація<додаткова інформація, корисна при описі варіанту використання>.

Табличні представлення варіанту використання

Іноді представляється зручним поміщати сценарії варіантів використання в таблицю, як це показано нижче. Інформація при цьому приймає більш структурований вигляд.

Таблиця 8.1. Таблиця в 2 колонки:

Актор	**Дії**
Користувач	Формує запит на пошук замовлень
Система	Відображає список замовлень
Користувач	Вибирає необхідний замовлення
Система	Показує детальну інформацію щодо замовлення

Таблиця 8.2. Таблиця в 3 колонки:

№ кроку	**Користувач**	**Система**
1	Формує запит на пошук	Відображає список замовлень

	замовлень	
2	Вибирає необхідний замовлення	Показує детальну інформацію щодо замовлення

Шаблон варіанту використання RUP

З шаблоном опису варіанту використання RUP і прикладами можна ознайомитися в інтерактивній версії RUP1).

Нижче наведено короткий огляд його розділів.

1. Найменування та короткий опис. У цьому розділі вказується: найменування варіанту використання, актори варіанту використання, короткий (в один абзац) опис варіанту використання.

2. Потік подій

2.1. **Основний потік подій**

Так само, як в "Основний сценарій".

2.2**. Альтернативні потоки подій**

Кожен з альтернативних сценаріїв описується в окремому параграфі, в тому ж стилі, що й основний потік подій. Альтернативні сценарії описують поведінку системи при будь-яких відхиленнях від основного сценарію, а також поведінка у виняткових ситуаціях.

3. **Спеціальні вимоги**

Тут перераховуються нефункціональні вимоги, що мають безпосереднє відношення саме до цього варіанту використання.

4 . **Предусловия**

Події, що описуються предусловием або постусловіем , повинні бути станами , які користувач може спостерігати [8.3] . Передумова описує стан , в якому система повинна перебувати до початку виконання прецеденту .

5 .**Післяумови**

Постусловіем RUP по суті описує те ж, що і мінімальна гарантія у Коберна . Л.Новіков [8.3] акцентує увагу на тому , що коректно сформульоване постусловіем повинно бути істинним при будь-якому можливому сценарії прецеденту , а не описаному в основному потоці.

6 . **Точки розширення**

Даний параграф визначає положення точок , що розширюють потік подій.

Вибір форми опису варіанту використання

При виборі форми і ступеня подробиці варіанту використання слід враховувати такі фактори , як:

• Розміри проекту ,

• Важливість проекту та варіанту використання ,

• Традиції , що склалися в колективі " Замовник - Дизайнер " .

Для невеликого проекту навряд чи буде доцільним застосовувати опису варіантів використання в розгорнутому форматі , досить використовувати коротку форму вільного стилю. Для проекту , в якому задіяно більше десяти учасників , в якому виникають проблеми розбиття на мікро - колективи , координації учасників , слід вибрати більш формалізований і більш детальний варіант.

Ступінь подробиці залежить також від критичності проекту в цілому і критичності варіанту використання в даному проекті. А.Коберн ділить всі

програмні проекти за ступенем критичності на 4 категорії: виходячи з ціни помилок : " проекти , помилки в яких можуть призвести до ..." :

• небезпеки для життя людей ,

• непоправних фінансових втрат ,

• фінансових втрат в обмеженому обсязі ,

• зниження комфортності кінцевого користувача.

Очевидно , що військові системи, або системи управління складними технічними об'єктами вимагають більш скрупульозного документування , у тому числі - і на рівні опису варіантів використання .

Крім того , в одному і тому ж проекті можуть зустрічатися більш важливі - з позицій частоти і масовості використання , складності для розуміння , технічних ризиків і т.д. і менш важливі прецеденти . У цьому випадку для різних прецедентів одного і того ж проекту цілком припустимо опис з різним ступенем подробиці.

Нарешті , специфікація варіантів у стилі Коберна , стилі RUP, в табличній формі , з використанням псевдокод або графічних конструкцій багато в чому визначається суб'єктивним вибором учасника прецедентів і сформованим досвідом роботи з замовником проекту.

Специфікація функціональних вимог

Опис функціональних вимог зазвичай здійснюється у формі, близькій до вільного формату опису варіанту використання. RUP рекомендує концентрувати нефункціональні вимоги в документі, що описує варіант використання у всіх випадках , коли це можливо. У разі , якщо нефункціональні вимоги носять загальний характер і не можуть бути прив'язані до конкретного прецеденту - вони виносяться в документ " Додатковаспецифікація " .

Атрибути вимог

Описи вимог повинні бути операбельними . Для цього всі вимоги повинні враховуватися в тій чи іншій обліковій системі , будь то електронна таблиця MS Excel , спеціалізована база даних, або інтегроване середовище управління змінами . При реєстрації вимоги воно проходить класифікацію у відповідності з певною системою ознак . Основні ознаки (атрибути) вимог були розглянуті в лекції 2 . Крім того , для оперативного керування вимогами буває корисно призначити їм такі властивості , як проект , відповідальна особа , статус , ризик , ступінь закінченості і т.п. У RUP для управління атрибутами вимог передбачений артефакт " Атрибути вимог".

Артефакт " Атрибути вимог", пропонований RUP, являє собою репозиторій текстів вимог , їх атрибутів і трасуванню .

Атрибути вимог представлені матрицею атрибутів вимог , де для кожного типу вимог перераховуються вимоги по одній осі і атрибути вимог цього типу за іншою . Для кожної вимоги вказуються значення його відповідних атрибутів. Приклади атрибутів : статус в часі , пріоритет , важливість , ризик , № ітерації (етапи) в плані.

Трасуванню описується у вигляді дерева , що показує в графічному вигляді вхідні і (або) вихідні зв'язку трасуванню.

ЛЕКЦІЯ 9
РОЗШИРЕНИЙ АНАЛІЗ ВИМОГ . МОДЕЛЮВАННЯ

У цій лекції ми поговоримо про моделювання аналізу вимог . Будуть розглянуті докладно діаграми UML , що пояснюють функціональність системи і внутрішній устрій системи , а також альтернативні мови моделювання

Які моделі використовувати

Вербальні описи варіантів використання системи , розглянуті в попередній лекції , на сьогодні є стандартом де- факто для формулювання угоди між Замовником та Виконавцем . Якщо обидві сторони готові виділити достатню кількість часу на уважний і всебічний аналіз вимог до системи і на початковій фазі її створення сформулювали 80 % всіх можливих сценаріїв використання системи на зрозумілій сторонами мовою - значить , ключові ризики створення системи - ризик різного розуміння проблеми і ризик розмиття кордонів багато в чому подолані.

Однак , далеко не всякий Замовник готовий скрупульозно обговорювати нудні томи опису варіантів використання , які навіть для систем середнього розміру можуть досягати сотні сторінок.

Щоб полегшити процес формулювання і розуміння вимог для Замовника , існує ряд прийомів. По-перше , вимоги можна формулювати на різних рівнях абстракції. Так , рівень опису вимог , підтримуваний в документі " Бачення" , є досить збалансованим. Те ж можна сказати і про короткі (в один абзац) опису ключовою функціональності системи . Діючи таким чином , ми , очевидно , вирішимо проблему залучення Замовника в аналіз завдань , однак вказані вище ризики будуть знижені недостатньо : концептуальні описи функціональності залишають багато свободи для тлумачення в ту чи іншу сторону.

Гарною підмогою у вирішенні завдання є застосування візуальних засобів опису вимог. Як відомо , у більшості людей правополушарное (образне) мислення дозволяє сприймати інформацію в рази і порядки більш прискореному темпі , ніж левополушарное (вербальне) .

На сьогодні в арсеналі аналітика існують десятки методик , мов , візуальних уявлень , що дозволяють моделювати вимоги до системи. При створенні інформаційних систем стандартом де -факто є універсальна мова моделювання , UML. Деякі інші нотації були згадані в лекції 5 .

Як вже зазначалося в лекції , присвяченій аналізу контексту АТ , процес аналізу вимог тісно пов'язаний , з одного боку , з аналізом проблемної області , з іншого - з архітектурним аналізом і проектуванням . Часто на практиці буває важко виокремити кордону компетенцій цих потоків робіт . Так, модель аналізу потоків даних , широко використовувана в аналізі проблемної області , згадується багатьма авторами , як модель , корисна в аналізі вимог. Ряд дослідників вважає доцільним ілюструвати опису вимог діаграмами класів , хоча , строго кажучи , виділення класів відноситься не до аналізу вимог , а до архітектурного аналізу .

Як визначити доцільність використання тих чи інших прийомів , методик, мов моделювання при аналізі вимог? Тут можна запропонувати три практичні рекомендації.

По-перше , аналіз вимог покликаний вивчати взаємодії автоматизованої інформаційної системи та її середовища , тобто користувачів , мережевих і системних компонент , що знаходяться поза системою . Отже, бізнес - моделі,

які описують взаємодії між компонентами організаційної системи , строго кажучи , можна розглядати лише як " сировину" для вилучення вимог , але не як моделі, що описують вимоги .

По-друге , аналіз вимог повинен знаходити відповідь на те , ЩО робить система , абстрагуючись від деталей реалізації, тобто того , ЯК вона це робить. Тому , припустимо , діаграма взаємодії об'єктів , що реалізують той чи інший варіант використання , можна розглядати швидше , як ілюстрацію внутрішнього устрою системи , корисну для програміста , ніж модель для замовника.

Однак , найбільш важливим є третя міркування , в чомусь " опозиційний " по відношенню до перших двох . Для моделювання аналізу вимог слід застосовувати моделі , найбільш адекватно проясняють функціональність системи та особливості її використання. Однак , аналітик вільний вибирати ті мови та методики , які дозволять домогтися цільової функції: зниження ризиків нерозуміння між Виконавцем та Замовником та розмиття кордонів. Тому , ілюструючи варіанти використання , починайте з " канонічних " способів , які будуть розглянуті трохи нижче , але , якщо порахуєте доцільним відхилитися від них - експериментуйте сміливо .

Моделі UML , що пояснюють функціональність системи

Діаграма варіантів використання

Діаграма варіантів використання UML, Use Case Diagram - одне з найпростіших уявлень системи. Її базові "будівельні елементи" - актори і варіанти використання. Діаграма задумана так, щоб дати найбільш загальне уявлення про функціональність системи (її компоненти), не вдаючись у деталі взаємозв'язків функцій. Тому основний вид відносини, використовуваний в діаграмі - асоціація між актором і варіантом використання.

Інші види відносин - відношення включення (include), розширення (extend) і узагальнення / генералізації.

Включення служить для позначення підлеглих варіантів використання (коли один або більше варіантів використання містять виклики однієї і тієї ж функціональності).

Розширення в точності відповідає точці розширення, використовуваної при описі варіанту використання, див лекцію 8.

Ставлення узагальнення може застосовуватися як до акторам, так і доваріантів використання, з метою вказівки спеціалізації одних щодо інших.

Діаграма дій

Якщо діаграма варіантів використання дає "вигляд зверху" на функціональність системи, діаграма дій UML, навпаки, дозволяє докладно ілюструвати окремий варіант використання та його сценарії.

Основні компоненти опису системи:

• Функції (дії),
• Символи "старт" і "стоп",
• Потоки управління,
• Розгалужувачі,
• Лінійки синхронізації.

Діаграма дій дозволяє проілюструвати варіант використання з необхідної ступенем подробиці. Лінійний варіант використання призводить до діаграми

дій з лінійним потоком управління між діями . Дії варіанту використання з альтернативними сценаріями реалізується через розгалужувачі . Лінійки синхронізації дозволяють описувати такі складні конструкції , як синхронізацію початку (закінчення) паралельних у часі процесів .

Крім стандартного формату опису , UML пропонує варіант з " плавальними доріжками " . Цей формат зручний для опису випадку , коли у варіанті використання беруть участь кілька акторів.

Діаграма станів

Діаграма станів в аналізі вимог використовується, коли потрібно дослідити поведінку системи, як кінцевого автомата. Це подання прийшло в UML з теорії систем.

У загальному випадку діаграма станів описує, як система себе веде в більш, ніж одному варіанті використання. Синтаксис діаграм станів багато в чому збігається з синтаксисом діаграм дій.

Основні компоненти опису системи:

• Прості стану,
• Складові стану,
• Символи "старт" і "стоп",
• Переходи,
• Лінійки синхронізації.

В UML під станом розуміється абстрактний метаклас, використовуваний для моделювання окремої ситуації, протягом якої має місце виконання деякої умови. Стан може бути задане у вигляді набору конкретних значень атрибутів класу або об'єкта, при цьому зміна їх окремих значень буде відображати зміну стану модельованого класу або об'єкта.

Перехід системи зі стану в стан здійснюється при настанні подій. При цьому йдеться, що перехід спрацьовує. Перехід може бути безальтернативним, або містити альтернативи. У другому випадку перехід обумовлений настанням сторожових умов. Нарешті, подія може супроводжуватися виразом дії, який відбувається у разі, якщо спрацьовує перехід. Повний синтаксис опису переходу (написи на стрілці) наступний:

Подія [сторожова умова] / вираз дії

Іноді буває корисним об'єднати частину станів в одне мета-стан. Графічно це виглядає, як символ стану (прямокутник з округленими кутами), що містить всередині себе кілька символів станів. При цьому можливі переходи між підлеглими станами, переходи між підлеглим і зовнішнім станами і переходи між складовим і зовнішнім станом.

Діаграми UML, що пояснюють внутрішній устрій системи

Деякі автори рекомендують використовувати при описі вимог діаграми UML, що описують створювану систему через її компоненти (класи, об'єкти), відносини і взаємодії між ними. Даний підхід має свої обмеження (див. Принцип 2).

Діаграма класів

Для створення діаграми класів необхідно:

1 . Здійснити пошук класів (ключових компонент проблемної області) .

2 . Для кожного знайденого класу визначити його ім'я , основні атрибути , операції та (або) відповідальності .

3 . Дослідити відносини знайдених класів .

Класи на діаграмі представляються у вигляді прямокутників , відносини - у вигляді ліній, що пов'язують прямокутники . Лінії різного типу графічно відрізняються різною штрихуванням і стрілками .

Прийнято виділяти 3 рівня абстракції класів:

• концептуальний рівень ,

• рівень специфікації ,

• рівень реалізації .

Аналіз вимог розумно супроводжувати діаграмами, що відбивають концептуальний рівень. На даному рівні при описі класів доцільно вказати їх найменування і відповідальності. Атрибути і операції можна не вказувати , або ввести тільки самі основні , відклавши їх дослідження на більш пізні стадії деталізації.

Діаграма класів показує статичну структуру проблемної області. Для аналізу взаємодії об'єктів - екземплярів класу в ході реалізації варіанту використання в UML передбачені дві діаграми взаємодії: діаграма кооперації та діаграма послідовності .

На думку автора , якщо діаграма класів у ряді випадків і може розглядатися , як артефакт , що пояснює структуру проблемної області , то діаграми взаємодії навряд чи варто розглядати в якості виразного мовного кошти , що ілюструє вимоги до системи в діалозі " Замовник - Виконавець " . Тим не менш, у відповідність з Принципом 3 свободи вибору мовних засобів , аналітик , який вирішив використовувати дані діаграми , може ознайомитися з ними в спеціальній літературі .

Альтернативні мови моделювання

Діаграма потоків даних

Діаграма потоків даних (data flow diagram , DFD) - один з основних інструментів структурного аналізу і проектування інформаційних систем , що існували в " доюмеельную " епоху. Незважаючи на що має місце в сучасних умовах зміщення акцентів від структурного до об'єктно-орієнтованого підходу до аналізу і проектування систем , " старовинні " структурні нотації і раніше широко і ефективно використовуються як в бізнес- аналізі , так і в аналізі інформаційних систем .

Історично склалося так , що для опису діаграм DFD використовуються дві нотації - Йодана (Yourdon) і Гейна - Сарсона (Gane - Sarson), що відрізняються синтаксисом . На наведеній нижче ілюстрації використана нотація Гейна - Сарсона .

Інформаційна система приймає ззовні потоки даних. Для позначення елементів середовища функціонування системи використовується поняття зовнішньої сутності. Усередині системи існують процеси перетворення інформації , які породжують нові потоки даних. Потоки даних можуть надходити на вхід до інших процесів , поміщатися (і вилучатись) у накопичувачі даних , передаватися до зовнішніх сутностей .

Модель DFD , як і більшість інших структурних моделей - іереархіческая модель. Кожен процес може бути підданий декомпозиції , тобто разбиению на структурні складові , відносини між якими в тій же нотації можуть бути показані на окремій діаграмі . Коли досягнута необхідна глибина декомпозиції -

процес нижнього рівня супроводжується міні - специфікацією (текстовим описом) .

Крім того , нотація DFD підтримує поняття підсистеми - структурної компоненти розроблюваної системи .

Нотація DFD - зручний засіб для формування контекстної діаграми , тобто діаграми , що показує розроблювану АІС в комунікації із зовнішнім середовищем. Це - діаграма верхнього рівня в ієрархії діаграм DFD . Її призначення - обмежити рамки системи , визначити , де закінчується розробляється система і починається середу . Інші нотації , часто використовуються при формуванні контекстної діаграми - діаграма SADT1) , діаграма Діаграма варіантів використання .

Інші види моделей

Серед різноманіття моделей, що використовуються в аналізі систем, хочеться особливо відзначити ще дві нотації, що дозволяють описати складні багатоальтернативного взаємодії компонент інформаційної системи - нотацію IDEF3 і eEPC-діаграму ARIS .

Для моделювання вимог до систем з розгалуженою логікою К.Вігерс рекомендує використовувати таблиці та дерева рішень. Часто на практиці бувають корисні діаграми сутність-зв'язок і SADT-діаграми.

ЛЕКЦІЯ 10
РОЗШИРЕНИЙ АНАЛІЗ ВИМОГ . ІЛЮСТРОВАНІ СЦЕНАРІЇ І ПРОТОТИПИ

Особливості сприйняття людиною вербальної і невербальної інформації стосовно моделей слід відносити до візуальних прототипам. У цій лекції ми поговоримо про прототіпіруванії , розглянемо основні цілі, що вимагають застосування прототипів, а також розглянемо ілюстровані сценарії прецендентов , які поряд з прототипами дозволяють досягти кращого розуміння між Замовником та розробником

Прототипи дозволяють побачити за сухими рядками документа опису вимог фрагменти реальної системи , " пограти " в експлуатацію системи до її створення .

Цілі прототипування

Все те , що говорилося в попередній лекції про особливості сприйняття людиною вербальної і невербальної інформації стосовно моделей , у ще більшою мірою слід віднести і до візуальних прототипам.

Розглянемо основні цілі, що вимагають застосування прототипів :

• прояснити неясні вимоги до системи;

• вибрати одне з різних концептуальних рішень ;

• проаналізувати здійсненність .

1 . **Неясні вимоги .** Часто Замовнику буває важко сформулювати вимоги до того , що він очікує від системи . У цьому випадку прототип інтерфейсу користувача (User Interface , UI) , оперативно створений за результатами інтерв'ю , дає йому можливість побачити схематичну реалізацію того , як Виконавець побачив відповідну частину системи . Що цікаво - в даному випадку корисний будь-який результат прототипування : якщо Виконавець

зрозумів вимоги добре - користь очевидна ; якщо не дуже - користь полягає в тому , що Замовник може вказати , в чому полягає нерозуміння , тим самим вирішивши основне завдання - зробити неясне ясним .

2 . **Різні варіанти рішення.** Будь-яку технічну задачу можна вирішити різними способами. Це стосується як завданняформулювання вимог , так і її реалізації в UI .

Розглянемо приклад. Постачальнику надходить вхідний потік вимог на комплектацію замовлень матеріалами. Різні позиції одного і того ж вимоги можуть бути закуплені у різних постачальників . Постачальник повинен зіставити постачальника кожної позиції кожного з вимог. Є як мінімум два сценарії розв'язання цієї задачі .

А) Сценарій послідовної обробки вимог.

о А1. Система відображає реєстр вимог , наявних у вхідній черзі .

о А2. Користувач вибирає чергову вимогу .

о А3. Система відображає перелік матеріалів вимоги та довідник постачальників .

о А4. Користувач зіставляє кожній з позицій вимоги постачальника з довідника постачальників .

о А5. Система надає вимогу статус " оброблено " , висилає по електронній пошті автору вимоги сповіщення .

о А6 . Продовжувати з кроку А1 , поки черга не спорожніє .

Б) Сценарій угруповання за матеріалами .

о Б1 . Система відображає позиції всіх вимог і довідник постачальників .

о Б2. Користувач групує позиції по типу (так , щоб однотипні позиції , що поставляються одним і тим же постачальником , знаходилися поруч).

о Б3 . Користувач вибирає групу позицій і зіставляє їй постачальника.

о Б4 . Система перевіряє - чи не з'явилися повністю оброблені вимоги . При позитивному результаті перевірки присвоює цим вимогам статус " оброблено " і висилає по електронній пошті автору вимоги сповіщення .

о Б5 . Продовжувати з кроку Б1 , поки черга не спорожніє .

Перший сценарій зручний тим , що дозволяє постачальнику працювати в розрізі авторів вимог , почати з найбільш критичних з часу вимог , контролювати процес їх обробки. Другий сценарій зручний тим , що дозволяє одночасно спостерігати на екрані рядка різних вимог , об'єднуючи їх в єдиний замовлення .

Після реалізації прототипів UI по першому і другому сценаріями Замовник , оцінивши їх переваги і недоліки , зміг у діалозі з Розробником сформулювати третій , комбінований сценарій , поєднує переваги перших двох .

3 . **Аналіз здійсненності** . Часто буває так , що комбінація функціональних , нефункціональних вимог і обмежень така, що виникає ризик неможливості їх реалізації. Як правило , такий ризик пов'язаний з вимогами до швидкодії системи при відомих обмеженнях середовища її реалізації . У цьому випадку створюються прототипи (не обов'язково , пов'язані з UI) , реалізують відповідну частину системи , що імітують потоки даних , що надходять на її вхід і їх обробку .

Класифікація прототипів

Розглянемо наступні класифікації прототипів :

• горизонтальні і вертикальні ;
• одноразові й еволюційні ;
• паперові й електронні, раскадровкі1) .

Горизонтальний прототип

Горизонтальний або поведінковий прототип (horizontal prototype , behavioral prototype) моделює інтерфейс користувача програми , не зачіпаючи логіку обробки і базу даних.

Якщо в розробленому інтерфейсі використовуються фрагменти бази даних - вони імітуються в програмному коді. При цьому тексти , що відображаються на екрані , повинні відображати реальну специфіку проблемної області , інакше користувачеві буде важко зосередитися . Якщо при натисканні на елемент управління повинні проводитися якісь розрахунки або запити в зовнішні системи - результати запитів також імітуються . Бажано реалізувати ту частину коду , яка відповідає за переміщення між екранами в процесі виконання варіантів використання , щоб користувач зміг зрозуміти , як буде діяти система у відповідь на його дії.

Горизонтальні прототипи слід використовувати для досягнення мети прояснення неясних , або багатоальтернативного вимог.

Вертикальний прототип

Вертикальний або структурний прототип (vertical prototype, structural prototype) не обмежується інтерфейсом користувача. Він реалізує вертикальний "зріз" системи, зачіпаючи всі рівні її реалізації. При створенні такого роду прототипів рекомендується використовувати ті мови та середовища реалізації, що і при виготовленні цільової системи (що, взагалі кажучи, зовсім не обов'язково для горизонтальних прототипів).

Основні цілі застосування такого роду прототипів - аналіз застосовності, перевірка архітектурних концепцій.

Одноразовий прототип

Одноразовий чи дослідницький прототип (throwaway prototype , exploratory prototype) створюється , коли потрібно швидко промакетіровать ті чи інші аспекти і компоненти системи .

Цілям створення дослідних прототипів служить технологія RAD (rapid application development) - швидка розробка додатків , див. лекцію 6 .

Одноразовий прототип повинен створюватися швидко. При його розробці не слід приділяти увагу питанням повторного використання коду , якості , швидкодії , технологічності і т.п.

У результаті виходить "сирий" код , який може містити значну кількість дефектів. Необхідно вжити заходів до того , щоб фрагменти коду , що реалізують такого роду прототипи , не стали частиною цільової системи .

Еволюційний прототип

Еволюційний прототип (evolutionary prototype) створюється , як перше наближення системи , покликане стати згодом самою системою.

Код еволюційного прототипу повинен послідовно , в перебігу однієї або більше ітерацій , перерости в код цільового програми . Тому даний вид прототипів вимагає всього того , від чого слід відмовитися при створенні одноразових прототипів : скрупульозної розробки , застосування технологічних методів і прийомів , тестування результатів тощо

У таблиці наведено співвідношення між розглянутими вище 4 видами прототипів

Таблиця 10.1.

	Одноразові	**Еволюційні**
Горизонтальні	• Прояснення і уточнення прикладів використання функціональних вимог • Виявлення пропущених вимог • Дослідження можливих варіантів інтерфейсу користувача	• Реалізація базових варіантів івикористання • Реалізація додаткових варіантів використання за пріоритетами • Реалізація та доопрацювання web-сайтів • Адаптація системи до швидко мінливих вимог бізнесу
Вертикальні	• Демонстрація технічної здійсненності	• Реалізація та нарощування ключовою клієнт-серверної функціональності і рівнів комунікації • Реалізація та оптимізація основних алгоритмів • Тестування і налаштування продуктивності

Паперовий прототип

Паперовий прототип (paper prototype) - відмінна альтернатива розглянутим вище різновидам електронних прототипів у разі , коли Дизайнер обмежений у ресурсах. Начерки інтерфейсів на папері , звичайно , не замінять інтерфейс , створений в середовищі розробки. Однак , при всіх недоліках , у таких прототипів є два істотні гідності.

1 . Замовник не стане акцентувати увагу на кольорі , формі кнопок і т.п. , відволікаючись від аналізу функціональності.

2 . Замовник ніколи не скаже , дивлячись на паперовий інтерфейс : "Та ви, я бачу , вже створили систему на 85%! Давайте закінчимо її в перебігу тижня".

Розкадровка

Рішенням проміжного між електронним і паперовим варіантами прототипів UI класу є презентації , виготовлені за допомогою засобів електронного офісу (наприклад , комбінації Microsoft Visio і Microsoft PowerPoint) . У цьому випадку користувач позбавлений свободи вибору , наданої йому поведінковим прототипом . Але ідею покрокової зміни екранів у процесі реалізації сценарію варіанту використання цілком можна реалізувати . Даний вид рішення визначається в [10.3] , як пасивна розкадровка . Активна розкадровка є подальшим розвитком поняття пасивної розкадровки , із застосуванням засобів анімації і т.п. Третій вид розкадровки , що вводиться в - інтерактивна являє собою електронний одноразовий горизонтальний прототип .

Ілюстровані сценарії прецедентів

Ілюстровані сценарії прецедентів , ІСП , поряд з прототипами дозволяють досягти кращого розуміння між Замовником та Розробником . Але якщо прототипи адресовані швидше Замовнику , ніж Розробнику , то з ІСП ситуація йде навпаки : вони містять додаткові відомості , що допомагають Розробнику

краще зрозуміти специфіку проблемної області і, тим самим , краще відобразити її в інтерфейсі користувача.

Основна ідея ІСП - " розбавити " текст опису сценарію варіанту використання аспектами застосовності .

Аспект застосовності - інформація, що дозволяє розширити опис прецеденту описами , конкретизують ті чи інші його особливості і, в кінцевому підсумку , підвищити ступінь комфортності користувача.

Розрізняють 3 різновиди аспектів застосовності : орієнтири , середні значення атрибутів і обсяги об'єктів , середня інтенсивність використання.

Орієнтири

Орієнтири - це опис опціональних функціональних можливостей системи . Відсутність таких можливостей не призводить до фатальної невдачі. Присутність - покращує застосовність , постачаючи корисною інформацією. Орієнтири слід розцінювати не як вимоги , а як побажання чи рекомендації .

Приклад. Опис потоку подій ІСП для прецеденту "Оформити замовлення " , розширеного орієнтирами (текст у квадратних дужках) .

У процесі виконання прецеденту менеджер з прийому замовлень вибирає замовника з клієнтської бази , визначає товарні позиції з довідника і вказує їх кількість. Система відображає на моніторі найменування позицій , ціну , суму і кількість на складі . Менеджер призначає знижку і визначає порядок оплати. Система розраховує підсумкову суму. [Менеджер повинен мати можливість бачити поточне сальдо розрахунків з клієнтом і дані за останніми десяти операціях із статистикою з дисципліни дотримання договірних зобов'язань] .

Середні значення атрибутів і обсяги об'єктів

Дана інформація дозволяє оптимальніше побудувати користувальницький інтерфейс і оцінити на ранніх стадіях проекту " вузькі місця " в обробці даних , які можуть вплинути на продуктивність системи .

Так , при виборі з 2 можливостей краще підійде елемент управління checkbox , при виборі , обмеженому 2-3 десятками позицій - список, що випадає , при різноманітті , вимірюваному тисячами варіантів , будуть потрібні додаткові засоби фільтрації і пошуку .

Приклад . Опис потоку подій ІСП для прецеденту "Оформити замовлення " , розширеного обсягами і середніми значеннями об'єктів (текст у фігурних дужках) .

У процесі виконання прецеденту менеджер з прийому замовлень вибирає замовника з клієнтської бази { до 10000 клієнтів } , визначає товарні позиції з довідника { товари розбиті на 10 категорій , кількість позицій в категорії не перевищує 500 } і вказує їх кількість { до 100 позицій , середня закупівля - 8 позицій } . Система відображає на моніторі найменування позицій , ціну , суму і кількість на складі . Менеджер призначає знижку і визначає порядок оплати { на даний момент існують 3 варіанти порядку оплати } . Система розраховує підсумкову суму.

Середня інтенсивність використання

Середня інтенсивність використання дозволяє виділити сценарії " масового "використання, в яких все має бути ідеально (швидкодія , зручність користування, мінімум дій на виконання операцій). Наприклад – інтерфейс касира в супермаркеті. Інша крайність - сценарії ,що виконуються від випадку

до випадку, не щодня і не потребують особливої оперативності (наприклад, розрахунок заробітної плати замісяць). Ці дані дозволяють, структурувати подачу інформації, прибрати з "головних "інтерфейсів рідко використовувані опції і т.п.

Приклад . Частковий опису потоку подій ІСП для прецеденту "Оформити замовлення для нового клієнта" , розширеного значеннями середньої інтенсивності використання (текст у круглих дужках) .

У процесі виконання прецеденту менеджер з прийому замовлень вибирає замовника з клієнтської бази (в 95 % випадків), або викликається інтерфейс реєстрації нового клієнта (у 5% випадків).

ЛЕКЦІЯ 11
ДОКУМЕНТУВАННЯ ВИМОГ

Щоб вимоги , виявлені і описані , взяли чинності угоди між Замовником та Розробником , їх необхідно оформити у вигляді документа . Ця лекція буде присвячена документування вимог

Щоб вимоги , виявлені і описані (прийняли чинності угоди між Замовником та Розробником , їх необхідно оформити у вигляді документа . У російській практиці для цього зазвичай використовується документ " Технічнезавдання" , ТЗ , у західній - " Software Requirements Specification " , SRS (специфікація програмних вимог) . По суті це - один і той же документ , тому далі по тексту будемо вживати термін "ТЗ " , розглядаючи різні шаблони його побудови - як на основі російських ГОСТ , так і західних методологій і стандартів.

Структура ТЗ у відповідність з ГОСТ 34.602-89

У завдання лекції не входить перерахування всіх правил і рекомендацій даного ГОСТ , бажаючі можуть ознайомитися з ним безпосередньо . Нижче будуть перераховані розділи , передбачені ГОСТ та розглянуто основні моменти , на які слід звернути увагу.

Загальні відомості - в цьому розділі , крім юридичних реквізитів сторін та іншої ділової інформації ГОСТ рекомендує вказати джерела і порядок фінансування робіт .

Призначення і цілі створення (розвитку) системи - тут необхідно вказати показники об'єкта автоматизації , які повинні бути досягнуті і критерії оцінки досягнення цих показників. Даним розділом на практиці часто нехтують і абсолютно марно - адже саме в цьому розділі закладаються високорівневі бізнес- вимоги і формулюються критерії їх досягнення .

Характеристика об'єктів автоматизації - досить важливий розділ. Його основні " розрізи " - організаційна структура , структура управління, структура розташування підприємства та його філій . Хороший опис об'єкта автоматизації дозволяє заощадити час на визначення класів користувачів, для великих територіально - розподілених систем - закласти структуру і топологію мережевих комунікацій .

Вимоги до системи - ключовий розділ цього документа , тому він буде розглянутий нижче , більш докладно.

Розділ **" Склад і зміст робіт зі створення системи"** , кажучи сучасною мовою , описує процес створення системи , включаючи вибір методології , що визначає зміст стадій , етапів і фаз і його конкретизацію для проекту (кількість етапів і ітерацій , їх основний зміст) .

Порядок контролю і приймання системи - також один з ключових компонент ТЗ. Він розподіляє ролі Замовника та Розробника у підготовці системи до випробувань та проведення випробувань . Тут доречно обумовити правила проведення випробувань , сформулювати основні тестові сценарії та критерії приймання.

Вимоги до складу і змісту робіт з підготовки об'єкта автоматизації до введення системи в дію , знову ж , апелюючи до сучасної термінології , обумовлюють порядок проведення реінжинірингу підприємства , який необхідно здійснити для того , щоб домогтися від впровадження АІС належного ефекту (підбір і навчання персоналу , зміни в організаційній структурі і т.п.).

Документ закінчується розділами **" вимоги до документування "** і **"джерела розробки"**, визначальними , відповідно, перелік і форми документації, підлягає розробці та перелік вже наявних документів, що містять передумови для розробки.

В якості додатків ГОСТ рекомендує використовувати розрахунок очікуваної ефективності системи та оцінку науково-технічного рівня системи.

Опис вимог до системи у відповідність з ГОСТ 34.602-89

ГОСТ поділяє всі вимоги до системи на три класи:

• вимоги до системи в цілому;

• вимоги до функцій (завдань), що виконуються системою;

• вимоги до видів забезпечення.

Серед вимог до системи в цілому (системні вимоги) зазначаються вимоги до:

• структурі системи (тут закладаються високорівневі архітектурні рішення , або структурні обмеження , вводиться поділ на підсистеми , комплекси і модулі , вирішуються питання комунікації компонент системи і системи з зовнішнім світом) ,

• режимам функціонування системи;

• персоналу (вказується чисельність , необхідна кваліфікація і режим роботи);

• надійності;

• безпеки;

• ергономіки та технічної естетики ;

• транспортабельності для рухомих АС ;

• експлуатації , технічного обслуговування , ремонту і збереження компонентів системи;

• захист інформації від несанкціонованого доступу;

• збереження інформації при аваріях ;

• захист від впливу зовнішніх впливів;

• патентної чистоти ;

• стандартизації та уніфікації ,

а також показники призначення (параметри, що характеризують ступінь відповідності системи її призначенню) та додаткові вимоги (поширюються на навчальні підсистеми , засоби контролю працездатності системи тощо) .

Вимоги ГОСТ до функцій (завдань) , в перекладі на сучасну мову , поділяються на:

• перелік функціональних вимог у прив'язці до підсистем і чергам автоматизації;

• тимчасової регламент реалізації функціональних вимог;

• вимоги до якості реалізації кожного з функціональних вимог (в тому числі - формі представлення вихідної інформації , характеристики необхідної точності і часу виконання , вимоги одночасності виконання групи функцій , достовірності видачі результатів);

• перелік і критерії відмов для кожного функціонального вимоги , за яким були задані вимоги по надійності.

Вимоги до видів забезпечення . Серед видів забезпечення ГОСТ вказує математичне , інформаційне , лінгвістичне , програмне , технічне , метрологічне, організаційне , методичне .

Документування вимог у RUP

Шаблон SRS , запропонований у RUP1) , по суті являє собою контейнер, в який необхідно " упакувати " артефакти , отримані в процесі специфицирования вимог (див. матеріали лекції 8). Крім того , SRS частково перегукується з документом " Бачення" (див. матеріали лекції 7). Шаблон зручний своєю компактністю і лаконізмом .

1 . Введення .

1.1 . Мета . Документ повинен вичерпним чином описувати зовнішню поведінку системи , а також нефункціональні вимоги та обмеження.

1.2 . Короткий звіт можливостей .

1.3 . Визначення , акроніми та скорочення.

1.4 . Посилання .

1.5 . Короткий зміст.

2 . огляд системи

2.1 . Огляд прецедентів . Містить список імен і коротких описів варіантів використання і акторів з ілюстраціями у вигляді діаграм прецедентів .

2.2 . Припущення і залежності. Дана секція описує ключові технічні можливості , компоненти , підсистеми , пов'язані проекти , які можуть впливати на життєздатність системи, що розробляється .

Припущенням (assumption) називається положення , яке вважається істинним за відсутності докази або визначальною інформації. [1].

При визначенні залежностей (dependencies) проекту від зовнішніх факторів , необхідно проаналізувати , які нові операційні системи , регламенти бізнес - процесів , стандарти якості , інформаційні системи можуть з'явитися на підприємстві впровадження і як це може вплинути на функціонування виготовленої АІС .

3 . опис вимог

3.1 . Опис варіантів використання . Параграф містить опис варіантів використання і пов'язаних з ними нефункціональні вимог , або посилання на відповідні артефакти.

3.2 . Спеціальні вимоги . Параграф містить опис функціональних вимог (не описані , як варіанти використання) , а також опис функціональних вимог загального характеру (НЕ зіставлених жодному прецеденту в попередньому розділі), або посилання на відповідні артефакти.
4 . Допоміжна інформація . Сюди включається інформація , що полегшує розуміння документа . Це може бути зміст і додатки , наприклад , описують прототипи інтерфейсу користувача.

Документування вимог на основі IEEE Standard 830-1998

Розглянемо шаблон документа опису вимог , складений К.Вігерсом на основі стандарту . Даний стандарт містить розгорнутий опис вимог , яке може бути оптимізовано для потреб конкретної організації.
1 . введення
1.1 Призначення документа .
1.2 . Підтримувані угоди .
1.3 . Передбачувана аудиторія та рекомендації щодо послідовності роботи з документом для кожного класу читачів.
1.4 . Межі проекту. Тут міститься посилання на документ "Концепція " , якщо такий є , або коротке резюме продукту .
1.5 . Посилання .
2 . Загальний опис .
2.1 . Загальний погляд на продукт. Тут необхідно визначити - чи є описуваний продукт новим членом зростаючого сімейства продуктів , новою версією існуючої системи , заміною існуючого додатка або зовсім новим продуктом. Якщо специфікація вимог визначає компонент більш великої системи , вкажіть , як це ПЗ співвідноситься з усією системою і визначте основні інтерфейси між ними.
2.2 . Особливості продукту . Перераховуються ключові особливості продукту або його головні властивості . Тут доречно помістити контекстну діаграму (у вигляді діаграми варіантів використання , потоків даних або ін специфікацій) .
2.3 . Класи і характеристики користувачів . Документується процес пошуку акторів , в якому виявляються всі користувачі системи і здійснюється узагальнення (виділення класів) користувачів. Знайдені класи описуються (наприклад - рівень кваліфікації , доступний функціонал і т.д.).
2.4 . Операційне середовище . Розглядається середовище функціонування АІС , включаючи апаратні засоби , операційні системи , для розподілених систем - географічне розташування користувачів і серверів , топологія мережі .
2.5 . Обмеження проектування та реалізації . Розглянемо класифікацію обмежень :
о певні технології , засоби , мови програмування та бази даних , які слід використовувати або уникати ;
о обмеження, що накладаються операційної середовищем продукту ;
о обов'язкові угоди або стандарти розробки;
о зворотна сумісність з продуктами , випущеними раніше ;
о обмеження, що накладаються бізнес- правилами;
о обмеження , пов'язані з обладнанням , наприклад вимоги до швидкодії , обмеження пам'яті або процесора ;

о угоди , пов'язані з призначеним для користувача інтерфейсом існуючого продукту , які необхідно дотримуватися при його поліпшенні
о формати і протоколи обміну даними.
2.6 Документація для користувачів.
2.7 Припущення і залежності
3 . функції системи
Для кожної і -й функції складається наступний опис .
3.і Найменування і -й функції системи .
3.і.1 Опис і пріоритети. Наводиться короткий опис функції і вказується її пріоритет (ступінь важливості / черговості реалізації) .
3.і.2 Послідовності " вплив - реакція " . Необхідно перерахувати послідовність впливів, які надають на систему (дії користувачів , сигнали зовнішніх пристроїв та ін) , і відгуки системи, що визначають реакцію конкретної функції.
3.і.3 Функціональні вимоги . Необхідно дати деталізацію і -й функції , перерахувати деталізовані функціональні вимоги , включаючи реакцію на очікувані помилки і невірні дії . Кожному детальному функціональному вимогу присвоюється унікальний ідентифікатор.

4 . Вимоги до зовнішнього інтерфейсу

Нижче розглянуті конкретні рекомендації з написання розділів цього параграф:

4.1 Інтерфейси користувача

Основні характеристики UI :

• посилання на стандарти графічного інтерфейсу користувачів або стильові рекомендації для сімейства продукту , які необхідно дотримуватися ;
• стандарти шрифтів , значків , назв кнопок , зображень , колірних схем , послідовностей полів вкладок , часто використовуваних елементів управління тощо ;
• конфігурація екрану або обмеження дозволу;
• стандартні кнопки , функції або посилання переміщення , однакові для всіх екранів , наприклад кнопка довідки ;
• швидкі клавіші;
• стандарти відображення повідомлень ;
• стандарти конфігурації для спрощення локалізації ПЗ ;
• спеціальні можливості для користувачів з проблемами із зором.

4.2 Інтерфейси обладнання

Опишіть характеристики кожного інтерфейсу між компонентами ПЗ і обладнання системи . В опис можуть входити типи підтримуваних пристроїв , взаємодії даних та елементів управлінь між ПЗ і обладнанням , а також протоколи взаємодії , які будуть використовуватися.

4.3 Інтерфейси ПО

Опишіть з'єднання продукту та інших компонентів ПЗ (ідентифіковані по імені і версії) , у тому числі бази даних , операційні системи, засоби , бібліотеки та інтегровані комерційні компоненти . Вкажіть призначення елементів повідомлень , даних і елементів управління , обмін якими відбувається між компонентами ПЗ. Опишіть служби , необхідні зовнішнім компонентів ПЗ , і природу взаємодії між компонентами. Визначте дані , до яких матимуть доступ компоненти ПЗ.

4.4 Інтерфейси передачі інформації

Вкажіть вимоги для будь-яких функцій взаємодії , які будуть використовуватися продуктом , включаючи електронну пошту , Web -браузер , протоколи мережевого з'єднання і електронні форми . Визначте відповідні формати повідомлень. Опишіть особливості безпеки взаємодії або шифрування, частоти передачі даних і механізмів синхронізації.

5 . Інші нефункціональні вимоги

У цьому розділі описуються інші нефункціональні вимоги , що не відносяться до вимог до інтерфейсу , які представлені в розділі 4, і до обмежень , описуваних у розділі 2.5 .

5.1 Вимоги до продуктивності

Вкажіть спеціальні вимоги до продуктивності для різних системних операцій . Обгрунтуйте їх необхідність для того , щоб допомогти розробникам прийняти правильні рішення, що стосуються дизайну. Наприклад , через жорсткі вимоги до часу відгуку бази даних розробники можуть зеркалізованнимі базу даних у кількох географічних місцях розташування або денормалізовать зв'язані таблиці баз даних для отримання більш швидкої відповіді на запит.

• **Додаток А. Словник термінів** (глосарій) .

• **Додаток Б. Моделі аналізу** . У цей розділ поміщаються всі моделі , побудовані в процесі аналізу вимог (див. матеріали лекції 9).

• **Додаток В. Перелік запитань .**

Це динамічний список ще не розв'язаних проблем , пов'язаних з вимогами . Це можуть бути елементи , позначені як " TBD " (to be determined - необхідно визначити) , відкладені рішення , необхідна інформація , невирішені конфлікти і т.п.

Документування вимог у MSF

На початку фази проектування проектна група працює з проектними вимогами. Вони поділяються на:

• бізнес- вимоги ,
• вимоги до експлуатації ,
• системні вимоги ,
• вимоги користувача .

Одним з основних результатів фази проектування є функціональна специфікація , яка служить :

• інструкцією команді розробників про те , що вони повинні будуть створити ;
• основою для оцінки обсягу робіт ;
• чітке угоду з Замовником про те , що повинно бути зроблено;
• основою для синхронізації роботи всієї проектної команди.

З шаблонами відповідних документів можна ознайомитися на сайті Microsoft.

ЛЕКЦІЯ 12
ПЕРЕВІРКА ВИМОГ

Ця лекція присвячена перевірці вимог. Будуть розглянуті такі процеси , як верифікація та валідація . Детально зупинимося на методах і засобах перевірки вимог , а також приділимо увагу деяким типовим проблемних ситуацій процесу формування та оцінки вимог

Верифікація та валідація

Термін " верифікація " (verification) в російськомовній літературі зазвичай перекладають , як " перевірка". Термін " валідація " - як " перевірка правильності" , " атестація " , "затвердження" .

Відповідно до стандарту IEEE 1012-1986 , верифікація являє собою процес оцінювання системи або компонента з метою визначити , чи задовольняють результати якоїсь фази умовам , накладеним на початку даної фази . Валідація в цьому ж стандарті визначається , як процес оцінювання системи або компонента під час або після закінчення процесу розробки з метою визначити , чи задовольняє вона зазначеним вимогам .

Важко очікувати від читача , вперше зіткнулася з цією термінологією і її визначеннями в цьому та інших стандартах , ясності розуміння . Принаймні , у автора справжнього курсу лекцій при першому знайомстві з визначеннями тих же понять у ISO IEC 12207 виникло чітке відчуття , що автори стандарту явно чогось не договорюють . Посудіть самі: згідно з цим стандартом , верифікація - це підтвердження експертизою та поданням об'єктивних доказів того , що конкретні вимоги повністю реалізовані. З іншого боку , валідація - це підтвердження експертизою та поданням об'єктивних доказів того , що конкретні вимоги до конкретних об'єктів повністю реалізовані. Правда , до честі авторів останнього стандарту , вони призводять примітка, яка кілька наближає читача до розуміння : " валідація пов'язана з експертизою продукту в цілях визначення його відповідності потребам користувача " . У цьому і полягає суть відмінності: якщо верифікація пов'язана із з'ясуванням того , чи задовольняє розроблюваний об'єкт, або процес його створення сформульованим вимогам , то валідація відповідає на питання - чи правильно розроблений цільовий об'єкт (продукт) , чи задовольняє він потребам замовника. Інший аспект валідації полягає в тому , що вона зазвичай ув'язується з формальної прийманням (атестацією) системи.

Деякі стандарти , наприклад SWEBOK , IEEE 1059-93 " IEEE Guide for Software Verification and Validation Plans " , вводять для цих двох процесів узагальнююче поняття V & V (Validation and Verification) . Згідно IEEE 1059-93 , верифікація та валідація програмного забезпечення - впорядкований підхід в оцінці програмних продуктів , застосовуваний протягом усього життєвого циклу . Зусилля, прикладені в рамках робіт з верифікації та валідації , спрямовані на забезпечення якості як невід'ємної характеристики програмного забезпечення і задоволення користувача вимог .

З вищесказаного ясно , як здійснити перевірки та затвердження АІС і (або) процесу її створення : у першому випадку необхідно переконатися , що АІС (компонента , процес) відповідає сформульованим вимогам , у другому - що АІС дійсно працює! Але якщо критерієм перевірки АІС служать вимоги , то що може послужити критерієм перевірки самих вимог? Відповідь полягає в тому , що вимоги повинні задовольняти властивостям , сформульованим у лекції 3 , Крім того , слід переконатися в тому , що :

• в специфікації вимог до ПЗ належним чином описані передбачувані можливості і характеристики системи , які задовольнять потреби різних зацікавлених у проекті осіб;

• вимоги до ПЗ точно відображають системні вимоги , бізнес-правила та ін;

• вимоги забезпечують якісну основу для проектування і збірки ПЗ .

Деякі типові проблемні ситуації процесу формування та оцінки вимог

Двозначність вимог

В ряду проблем і недоліків вимог двозначність , є, мабуть , найбільш критичним фактором ризику проекту , що закладається у фазі формування вимог. Двозначність (невідповідність властивості ясності , визначеності) закладає під проект " бомбу сповільненої дії" . На практиці вимога, сформульована двозначним чином , може привести до різних його інтерпретацій представниками Розробника і Замовника. Дизайнер , керуючись своєю інтерпретацією , визначить на її основі архітектурну основу , створить аналітичні та проектні моделі і в кінцевому підсумку створить програмний код . Як показують дослідження , ціна виправлення помилки виростає приблизно на порядок при переході між робочими потоками (від аналізу вимог до проектування , від проектування до реалізації і т.д.).

Тим самим , якщо не закласти кошти на перевірку вимог на предмет двозначності в момент їх формування - існує ризик неприйняття готової системи в момент приймально - здавальних випробувань , тому що кожна зі сторін буде дотримуватися своєї версією інтерпретації вимог , що веде до збитків , судовим процесам і т.п. і тому є маса прикладів .

" Позолота " продукту

Під " золоченням " розуміють такі ситуації , коли розробники додають функції , яких немає в специфікації , але їм здається , що це сподобається користувачам . Найчастіше ж клієнтам не потрібні такі надлишкові можливості, виходить , що час , відведений на реалізацію , витрачається даремно.

Ця ситуація виникає у разі , коли , по-перше , в колективі Розробника присутні творчі особистості (адже далеко не всяка команда стане проявляти ініціативу і робити понад те , про що її просили) , в других - існує розрив у проходженні інформації від Замовника до Розробнику . Ініціативний розробник " золотить " продукт з найкращих спонукань , але , можливо , він погано знайомий з бізнес- процесом Замовника та закладені ним " фічі " просто не будуть затребувані.

Інший бік " золочення " полягає в тому , що група представників Замовника неоднорідна за своєю структурою і може виникнути ситуація , коли представник Замовника , який формулює "дорогі" вимоги , не володіє відповідними повноваженнями. Це - специфіка російських підприємств , де часто все буває влаштовано істотно неформально .

Автору довелося одного разу бути присутнім в одному дуже показовому діалозі , де він брав участь в якості Розробника , що здає продукт (АІС для складу матеріалів) , ще були присутні користувач нової системи та інвестор проекту. Ірина (користувач) : мені незручно працювати в цій програмі . Вона не враховує розміри баночок в літрах. Павло (інвестор) : без цього спокійно можна обійтися , достатньо того , що є облік матеріалів по класифікатору та 2 одиниці виміру - в літрах і кілограмах . Ірина : а мені цього недостатньо , я веду облік в баночках ! Павло : а я тут господар , тут все моє і мені твій баночний облік не потрібен! Ірина : а я все одно буду працювати , як я звикла ! Діалог

тривав близько півгодини і останнє слово залишилося , звичайно , за Павлом , програму переписувати не довелося.

Мінімальна специфікація

Створювати повну документацію вимог відповідно до вищевикладених принципів , або обмежитися начерком вимог на 2-3 сторінки , як це часто робить автор лекцій у невеликих проектах - як кажуть, справа смаку.

Однак , для роботи " не за правилами " , по-перше , повинні бути об'єктивні передумови , по-друге - слід віддавати собі звіт в вигодах та ризики цього вибору .

Мінімальна специфікація доречна , якщо має місце наявність одночасно трьох обставин (об'єднання за " І ") :

• а) ціна контракту і розміри проекту такі, що розробка розгорнутого ТЗ економічно недоцільна ;

• б) колектив Розробника володіє достатнім ступенем професіоналізму та досвіду виконання проектів у суміжних областях , щоб вміти створювати по короткій специфікації продукт , який пройде приймання Замовником ;

• в) між Замовником та Розробником існують конструктивні відносини і обидві сторони розуміють і приймають ризики міні - специфікації .

Інший варіант роботи з міні- специфікації : Замовник і Дизайнер розуміють , що створення розгорнутої специфікації відтягує закінчення випуску готового продукту , що головна мета проекту - продукт , а не документація і готові до щільному співробітництву в процесі його створення. Це - шлях так званих agile - методологій розробки ПЗ , докладніше див вhttp://www.agile.org .

Основний ризик застосування міні - специфікації полягають в тому , що вони базуються на людському факторі . Хороші і конструктивні відносини між сторонами " на березі " повинні зберегтися на всьому протязі проекту , в іншому випадку у сторін виникнуть суттєві проблеми у формальному доведенні того - що повинна робити програма , тому що міні - специфікація для цього недостатньо повна.

Пропуск типів користувачів

Корпоративні АІС створюються для того , щоб бути використаними різними групами користувачів . Може скластися ситуація , в якій до групи представників Замовника , що беруть участь у формуванні вимог , потраплять найбільш ініціативні персони підприємства , які , по всій видимості , зможуть донести свій голос до представників Розробника . Ті ж категорії користувачів , у яких не знайдеться активних представників , можуть виявитися "за бортом" автоматизації . Саме ця помилка формування вимог називається " пропуск класів користувачів" . Щоб її уникнути , представник Розробника повинен об'єктивно оцінити організаційну структуру підприємства і його бізнес - процеси і вдумливо підійти до вибору ключових персон , проведення інтерв'ю з якими допоможе сформувати цілісну картину вимог до створюваної АІС .

Методи і засоби перевірки вимог

Напрацьовано значну кількість методів і засобів перевірки вимог . Вони різняться за низкою параметрів. Так , розрізняють :

• за широтою аналізу - перегляд (вибіркова перевірка) і наскрізний контроль (тотальна перевірка) ;

• за ступенем формалізації - неофіційні процедури , процедури, що проводяться за формальними правилами (інспекції , експертизи) ;

• за складом групи перевірки - з (без) участю автора , з (без) участю менеджера проекту , з (без) участю представників зовнішніх організацій ;

• по використовуваних засобів - тексти вимог , тестові сценарії , критерії прийнятності , прототипи .

Поняття і методи прототипування були розглянуті в лекції 10 . Деякі інші, найбільш важливі з перерахованого вище , методи і засоби , розглянуті далі по тексту.

Неофіційні перегляди вимог

Розрізняють кілька способів неофіційних переглядів вимог :

• перегляд " за столом " ,

• колективна перевірка ,

• критичний аналіз .

У перших двох випадках автор вимог звертається за допомогою до колег (відповідно , до одного , або до кількох) з метою видачі практичних рекомендацій щодо поліпшення продукту . У третьому випадку автор здійснює презентацію розроблені ним вимоги на нараді з подальшим обговоренням .

Неофіційні перегляди використовують для знайомства з розробкою , збору відгуків , формування зворотного зв'язку. За статистикою неофіційні перегляди дозволяють виявити до 60 % помилок у вимогах .

Інспекції

Поняття інспекції, стосовно до IT -індустрії , вперше було сформульовано Майклом Феганом (Michael Pagan) з IBM в середині 70- х гг.1).

Особи , що займають управлінські позиції (менеджери) відносно до будь-яких членам команди інспектування , не повинні брати участь в інспекціях .

Інспекція повинна вестися під керівництвом неупередженого (незалежного від проекту і його цілей) лідера , навченого технікам інспектування .

Інспектування завжди залучає авторів проміжного або кінцевого продукту.

У групу інспекції входять лідер , реєстратор , рецензент і кілька (від 2 до 5) інспекторів. Члени команди інспектування можуть спеціалізуватися в різних областями експертизи (володіти різними областями компетенції) , наприклад , предметної області , методах проектування , мовою тощо У заданий момент (проміжок) часу інспекції проводяться відносно окремого невеликого фрагмента продукту (в більшості випадків , фокусуючись на окремих функціональних чи інших характеристиках ; часто , відштовхуючись від окремих бізнес -правил , функціональних вимог або атрибутів якості , прим. Автора). Кожен член команди повинен дослідити оцінюваний продукт та інші вхідні дані до проведення інспекційної зустрічі , застосовуючи , можливо , ті чи інші аналітичні техніки до невеликих фрагментах продукту або до продукту , в цілому , розглядаючи в останньому випадку тільки один його аспект , наприклад , інтерфейси . Будь-яка знайдена аномалія повинна документуватися , а інформація передаватися лідеру інспекції . У процесі інспекції лідер керує сесією і перевіряє , що всі підготувалися до інспектування . Загальним інструментом , використовуваним при інспектуванні , є перевірочний лист (

checklist), що містить аномалії і питання, пов'язані з аспектами , що викликають інтерес. Результуючий лист часто класифікує аномалії і оцінюється командою з точки зору його завершеності і точності. Рішення про завершення інспекції приймається відповідно до одним (будь-яким) з трьох критеріїв:

1. Прийняття з відсутністю або малою необхідністю переробки
2. Прийняття з перевіркою перероблених фрагментів
3. Необхідність повторної інспекції.

Розробка тестів

Механізм варіантів використання (uses cases) , розглянутий в лекції 8 , дозволяє відповісти на питання: як буде використовуватися система . Щоб перевірити систему , використовується аналогічний механізм : тестових сценаріїв (test cases) .

Тестові сценарії (ТЗ) рекомендується створювати вже на ранніх стадіях роботи з вимогами , в ідеалі - після отримання запитів співвласників , паралельно з розробкою варіантів використання .

Тестові сценарії , як і варіанти використання , можуть підтримувати різні рівні абстракції. Розрізняються концептуальні і детальні ТЗ. Концептуальний рівень передбачає опрацювання процедури тестування , інваріантну до конкретної реалізації UI .

Як використовувати тестові сценарії для тестування вимог? Пропонується наступна процедура.

1 . Побудувати матрицю , де по вертикалі відзначені функціональні вимоги , а по горизонталі - тестові сценарії.

2 . Переконатися , що кожен з ТЗ здійснимо на існуючому наборі вимог.

3 . Переконатися , що для кожного вимоги представлений як мінімум один ТЗ.

4 . Прокреслити "шлях" кожного з ТЗ на мапі діалогів. Це дозволить : виявити некоректні або пропущені вимоги , виправити помилки на карті діалогів і відшліфувати варіанти тестування.

Як бути з тестуванням нефункціональних вимог? Згідно [12.6], процедура аналізу вимог вважається виконаним тільки тоді, коли всі вимоги, включені в специфікацію, володіють методами оцінки відповідності їм створюваного програмного продукту.

Для того, щоб нефункціональні вимоги були вимірні, кожному з них в ідеалі необхідно зіставити кількісну метрику. Якщо це не вдається - можливо, вимога слід переформулювати, або деталізувати.

Визначення критеріїв прийнятності

При формальної приймання продукту існують дві типові процедури : демонстрація продукту Розробником на тестових сценаріях і перевірка продукту Замовником . Далеко не кожного Замовника можна переконати , що він не повинен " тикати кнопки", що лежать за межами тестових сценаріїв . Однак , в період стабілізації продукту , для Замовника важливіше навіть не кількість виявлених дефектів , а можливість перевірки - чи годиться розроблена АІС для вирішення поставлених ним завдань.

Щоб не відкладати настільки важливе питання до моменту приймання системи , вкрай важливо , поряд з формуванням вимог , залучити Замовника на ранніх стадіях створення продукту в процес формування критеріїв

прийнятності. Критерії прийнятності (acceptance criteria) повинні відобразити точку зору Замовника на те , що він вважає правильною системою .

Делегування розробки тестів на прийнятність користувачам - ефективна стратегія розробки вимог. Це дозволяє вже на етапі збору інформації перейти від формулювання питання з " Що вам потрібно робити за допомогою системи? " до " Як ви робите висновок про те , що система задовольняє вашим потребам? " . Якщо клієнт не може описати , як він оцінить , що конкретну вимогу задоволено системою , значить , вимога сформульована недостатньо ясно.

Раннє формування тестів для перевірки прийнятності дозволяє виявити дефекти у вимогах .

Перевірка прийнятності базується на ключових (істотних) варіантах використання . При цьому слід абстрагуватися від альтернативних сценаріїв і виключень і зосередити увагу на основному потоці подій. Необхідно врахувати також і нефункціональні вимоги , такі , як продуктивністю , легкість і простота використання.