МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДНІПРОДЗЕРЖИНСЬКИЙ ДЕРЖАВНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

«АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

для студентів напряму підготовки

6.050103 «Програмна інженерія»

ЗАТВЕРДЖЕНО:

редакційно-видавничою секцією

науково-методичної ради ДДТУ

_____ 2015 р.

протокол № _____

Дніпродзержинськ -2015

Конспект лекцій з дисципліни «Аналіз вимог до програмного забезпечення» для студентів напряму підготовки 6.050103 «Програмна інженерія». Укладач: канд. фіз.-мат. наук, доцент Божуха Л.М.– Дніпродзержинськ, ДДТУ: 2015. – 94 с.

Укладач: Л.М. Божуха, доцент, кандидат фізико-математичних наук.

Затверджено на засіданні кафедри ПЗС

Протокол № ___ від ___ . ___ . 2015 р.

Рецензент: Рецензент: О.С. Косухіна, доцент, кандидат технічних наук

Коротка анотація видання. Конспект лекцій з аналізу вимог до програмного забезпеченнямістить основні теоретичні відомості дисципліни, контрольні питання, що сприяє подальшому самостійному поглибленню знань та навичок студентів.

Рекомендовані для студентів напряму підготовки 6.050103 «Програмна інженерія», можуть бути використані студентами інших спеціальностей.

ЗМІСТ

Змістовий модуль 1. *Вимоги та їх властивості*

Тема 1. Введення. Поняття і класифікація вимог

Визначення ІС Класифікація ІС. Класифікація за масштабом. Класифікація по архітектурі. Класифікація за характером використання інформації. Класифікація по системі подання даних. Класифікація за підтримуваним стандартам управління і технологій комунікації. Класифікація за ступенем автоматизації. Роль вимог в задачі впровадження АІС.

Визначення поняття вимоги. Класифікація вимог. Вимоги до продукту і процесу Рівні вимог. Системні вимоги та вимоги до програмного забезпечення. Функціональні, нефункціональні вимоги і характеристики продукту.

Класифікація RUP. Методології і стандарти, що регламентують роботу з вимогами.

Визначення ІС

Далі по тексту *інформаційною системою (ІС)*, або автоматизованої *ІС, АІС,* будемо називати програмно-апаратну систему, призначену для автоматизації цілеспрямованої діяльності кінцевих користувачів, що забезпечує, у відповідність із закладеної в неї логікою обробки, можливість одержання, модифікації й зберігання інформації.

Ключовим моментом у цьому визначенні є поняття «цілеспрямованої діяльності». Мова йде про діяльності, спрямованої на рішення конкретної задачі, що коштує перед користувачем (колективом користувачів).

Деякі дослідники визначають *ІС* трохи іншим способом. *ІС у широкому змісті* – взаємозалежна сукупність засобів, методів і персоналу, використовуваних для зберігання, обробки й видачі інформації в інтересах досягнення поставленої мети.

Основні відмінності такого підходу: 1) уведення користувачів системи «усередину» ІС, 2) необов'язковість використання засобів обчислювальної техніки. Такий підхід також має право на життя. Так, наприклад, у ньому зручно простежувати загальну історію виникнення й розвитку систематичних засобів обробки інформації в бізнесі, що почалася, мабуть, у доком‘ютерну епоху. Однак, тому що 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.

 Бухгалтерія 8.0 Використається з метою формування бухгалтерської звітності підприємства перед податковими органами. Є інформаційною системою.

---

[1] Маглинец Ю. А. Розробка інформаційних систем. Частина 1, Структурні методи. – Красноярськ.: Кларитеанум, 2004. – 120 з.

 MS Excel. Програмний засіб універсального характеру, призначене 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 й оснащена макросами, що дозволяють розраховувати заробітну плату й формувати платіжні відомості. Є інформаційною системою.

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

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

Класифікація ИС

Інформаційні системи можуть бути класифіковані по різних ознаках. Розглянемо найбільш важливі з них.

Класифікація за масштабом

За масштабом ІС будемо підрозділяти на однокористувальницькі, групові й корпоративні.

Однокористувальницькі ИС, як це ясно з назви, призначені для використання на одному робочому місці. У цей час на світовому й вітчизняному ринку представлена множина рішень, призначених для автоматизації діяльності окремо взятого користувача. Як правило, це – рішення, орієнтовані на фахівця в тій або іншій області, будь те складання специфікацій для зборки виробів з комплектуючих, планування ремонтів устаткування, облік витрат і доходів приватного підприємця оптової торгівлі, або складання розкладів занять у деканаті.

В теперішній час альтернативу таким узкоспециализированным системам склали табличні процесори, що не мають проблемної спеціалізації, у першу чергу – MS Excel. Системи цього класу важко віднести до класу ІС, але найчастіше вони дозволяють непрограмуючому фахівцеві створити й, що дуже важливо, самостійно розвивати власні рішення, що заміняють, а місцями й перекрывающие функціонала однокористувальницьких систем зразка 90-х років.

В основі більшості однокористувальницьких систем лежить стандарт X-Base (Clipper, FoxPro, dBase). Широко використаються також рішення на базі систем Paradox, Clarion, MS Access. Кожна з перерахованих конкуруючих систем володіє власної высокоуровневой інструментальним середовищем, що дозволяє спроектувати базу даних, логікові обробки, користувальницький інтерфейс, звіти за допомогою «помічників»-построителей. На рубежі тисячоріч з'явилися також і однокористувальницькі рішення на базі промислових реляционных СУБД. У цьому випадку ПО сервера інсталюється безпосередньо на робочу станцію користувача. Прикладом може служити Personal Oracle. Дані рішення висувають значні вимоги до ресурсів робочої станції, однак несуть у собі багато переваг промислових СУБД.

Групові системи призначені для автоматизації діяльності в робочій групі (відділі, кластері, групі проекту й т.д.). На відміну від однокористувальницьких ІС, групові системи, як правило, представляють спеціалізовані клієнтські рішення (їх часто називають автоматизованими робітниками місцями, АРМ) для різних учасників групи. Наприклад, для оптової фірми, ІС може представляти набір таких АРМ, як «Менеджер по продажах»,

«Комірник», «Постачальник», «Директор». Для навчального планування – «Викладач», «Працівник бюро планування», «Працівник навчального відділу», «Фахівець із планування на кафедрі», «Працівник деканату».

Групове використання рішень на базі табличних процесорів можливо, але має істотні обмеження, пов'язані з розмежуванням доступу, регламентацією й синхронізацією внесених змін. По суті єдиний режим їхнього використання, що забезпечує коректність даних – «файловий сервер, один автор, N читачів».

При створенні групових ІС у цілому використаються ті ж засоби й інструментальні середовища, що й при створенні однокористувальницьких ІС. Треба, однак, відзначити, що для використання в групі при виборі між системами з файловим і реляційним сервером варто віддавати перевагу реляційному серверу, причому доцільно використання виділеного сервера. Це може бути, наприклад, сервер Oracle, DB2, MS SQL, Sybase, Informix.

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

Серед вимог, пропонованим до сучасного КИЦЬ:

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

Приклади ERP-систем – SAP R3, «Галактика», MS Navision Axapta.

Класифікація по архітектурі

1. Архітектура «Файл-сервер». Історично перша архітектура інформаційних систем. модулі, що виконують Як, так і дані розміщаються в окремих файлах операційної системи. Доступ до даних здійснюється шляхом вказівки шляху (path) і використання файлових операцій (відкрити, уважати, записати). Для зберігання даних використається виділений сервер (окремий комп'ютер), що і є файловим сервером. Модулі, що виконують, зберігаються або на робочих станціях, або на файловому сервері. В останньому випадку спрощується процедура їхнього адміністрування, але при цьому зростають вимоги до надійності мережі.

с Архітектура «Клієнт-сервер». Клієнт-сервер – це не тільки архітектура, це – нова парадигма, що прийшла на зміну застарілим концепціям. Суть її полягає в тім, що клієнт (виконує модуль, що) запитує ті або інші сервіси відповідно до певного протоколу обміну даними. При цьому, у відмінності від ситуації з файловим сервером, немає

необхідності у використанні прямих шляхів операційної системи: клієнт їх «не знає», йому «відомі» лише ім'я джерела даних і інші спеціальні відомості, використовувані для авторизації клієнта на сервері. Сервер, що фізично може перебувати на тім же комп'ютері, а може - на іншому кінці земної кулі, обробляє запит клієнта й, зробивши відповідні маніпуляції даними, передає клієнтові запитувану порцію даних.

В рамках напрямку «клієнта-сервер» існують два основних «діалекти»: «тонкий» і «товстий» клієнт.

В системах на основі тонкого клієнта використається потужний сервер баз даних, це – високопродуктивний комп'ютер і бібліотека так званих збережених процедур, що дозволяють робити обчислення, що реалізують основну логіку обробки даних, безпосередньо на сервері. Клієнтський додаток, відповідно, висуває невисокі вимоги до апаратного забезпечення робочої станції. Основне достоїнство таких систем – відносна дешевина клієнтських станцій.

Системи з товстим клієнтом, навпроти, реалізують основну логіку обробки на клієнті, а сервер являє собою в чистому виді сервер баз даних, що забезпечує виконання тільки стандартизованих запитів на маніпуляцію з даними (як правило
– читання, запис, модифікацію даних у таблицях реляционной бази даних). У системах такого класу вимоги до робочої станції вище, а до сервера – нижче. Достоїнство архітектури – переносимость серверного компонента на сервери різних виробників: всі промислові сервери баз даних реляционного типу підтримують роботу зі стандартизованою мовою маніпулювання даними SQL, але внутрішня убудована мова обробки даних, необхідний для реалізації логіки обробки на сервері в кожного із серверів свій.

3. Тришарова архітектура. Базується на подальшій спеціалізації компонентів 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 бізнеси-логіки. Цей шар може являти собою або виділений сервер (сервер додатків), або розміщатися на клієнті як динамічна бібліотека. Дана архітектура дозволила з'єднати достоїнства тонкого й товстого клієнтів: гарна переносимість з'єднується в ній з невисокими вимогами до клієнта.

З розвитком інтранет-інтернет технологій з'явився різновид тришарової архітектури на підставі використання web-технологій. У цьому різновиді роль сервера додатків грає web-сервер, а як клієнт використається стандартний web-браузер. Достоїнства – у знижених вимогах до клієнта й у легкої вбудовуємості даної архітектури у світові інформаційні мережі. Основний недолік - відомі обмеження, що накладають на інтерфейс користувача web-браузерами.

Класифікація по характері використання інформації

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

Кінцеві користувачі інформаційно-пошукових систем (ІПС), як правило, мають доступ до збереженим даних тільки «по читанню» і використають дані системи для пошуку відповідей на ті або інші питання. Доступ по модифікації даних має адміністратор системи, у функції якого входить забезпечення актуальності інформації, усунення помилок.

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

Альтернатива ІПС – керуючі системи автоматизують (повністю або частково) діяльність, пов'язану із прийняттям рішень. Дії кінцевих користувачів таких систем приводять до модифікації інформації, що, звичайно, не виключає можливості просто одержувати інформацію, як в ІПС.

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

Класифікація по системі подання даних

Серед найпоширеніших засобів і моделей подання даних варто виділити:

- «саморобні» формати зберігання даних, що зберігаються у файлах (текстових, бінарних);
- спеціалізовані формати зберігання даних, що використалися в «дореляционный» період (наприклад, x-Base, paradox);
- мови структурованої розмітки на основі формату xml;
- реляционная модель; SQL сервер;
- об'єктна, объектно-реляционная модель;
- документоориентированное сховище (IBM Lotus/Domino).

Класифікація за підтримуваними стандартами керування й технологіям комунікації

Епоха стихійної розробки АИС закінчена. Сучасні автоматизовані інформаційні системи розробляються, виходячи зі сформованих реалій автоматизованого керування бізнесом. Існує значна кількість концепцій, технологій, підходів, що знайшли своє ефективне застосування в різних галузях промисловості по усьому світі. Деякі з них придбали статус міжнародних стандартів. У специфікації АИС, розроблювальної для масового продажу, як правило, вказується – які стандарти й технології керування вона підтримує. Менш строгі вимоги до АИС, створюваним під замовлення для конкретного підприємства. Однак і в цьому випадку не враховувати сформований у світі позитивний досвід просто нерозумно. Нижче перераховані деякі, найбільш важливі, технології й стандарти.

MRP (Material Requirements Planning) – планування поставок матеріалів, виходячи з даних про комплектації виробленої продукції й плану продажів.

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

MRPII (Manufacture Resource Planning) – планування матеріальних, мощностных і фінансових ресурсів, необхідних для виробництва. Стандартизовано ISO.

ERP (Enterprise Resource Planning) – финансово-ориентированное планування ресурсів підприємства, необхідних для одержання, виготовлення, відвантаження й обліку замовлень споживачів на основі інтеграції всіх відділів і підрозділів компанії.

SCM (Supple Chain Management) – керування ланцюжками поставок. Реалізація бізнесів-процесів на базі зовнішніх підприємств і торговельних площадок Засновано на

референтній моделі SCOR, стандартизованої Supply Chain Council.

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

ERPII (Enterprise Resource & Relationship Processing) - керування ресурсами й взаєминами підприємства. Поєднує в собі 3 перераховані вище технології.

Workflow – технологія, що управляє потоком робіт за допомогою програмного забезпечення, здатного інтерпретувати опис процесу, взаємодіяти з його учасниками й при необхідності викликати відповідні програмні додатки.

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

Project Management – керування 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. Поєднує в собі практично всі перераховані вище підходи й технології.

Настільки лаконічні визначення, звичайно ж, дозволяють лише ознайомитися із сучасною термінологією. Задача їхнього детального вивчення не входить у план занять. Тих, хто захоче одержати докладний матеріал із цих питань, можна порекомендувати наступні літературні джерела [2-6].

Класифікація по ступені автоматизації

Приводиться класифікація:

Ручні ІС характеризуються відсутністю сучасних технічних засобів переробки інформації й виконанням всіх операцій людиною. Наприклад, про деятель-ности менеджера у фірмі, де відсутні комп'ютери, можна говорити, що він працює з ручний ИС.

Автоматичні ІС виконують всі операції по переробці інформації без участі людини.

Автоматизовані ІС припускають участь у процесі обробки інформації й людини, і технічних засобів, причому головна роль приділяється комп'ютеру. У сучасному тлумаченні в термін "інформаційна система", як правило, вкладається поняття автоматизируемой системи.

Роль вимог у задачі впровадження АІС

Які можна зробити висновки з розглянутої класифікації АИС? Навіть не говорячи про різноманіття виробників АИС, не розглянутого в дійсній лекції, на підставі тільки сформульованих ознак стає очевидним, що існує значна кількість АИС і дані АИС істотно розрізняються між собою. Отже, вибір АИС для підприємства – досить нетривіальна задача. Для того, щоб успішно неї вирішити, необхідно добре знати об'єкт впровадження (автоматизируемое підприємство), особливості його діяльності, стратегію розвитку й багато інших аспектів, що визначають характеристики закуповуваної АИС. Зазначені знання в остаточному підсумку формалізуються в документі вимог до АИС, на основі якого й здійснюється вибір і наступне настроювання АИС. У ще більшому ступені вимоги до АИС важливі при розробці АИС на замовлення. Докладніше про це – у наступних лекціях.

Визначення поняття вимоги

У редакції нотації RUP приводиться наступне визначення: «Вимога – це умова або можливість, який повинна відповідати система».

В IEEE Standard Glossary of Software Engineering Terminology (1990) дане поняття трактується ширше. Вимога – це:

1. умови або можливості, необхідні користувачеві для рішення проблем або досягнення цілей;

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

3. документоване подання умов або можливостей для пунктів 1 і 2 (кінець цитати).

Уведемо ще одне визначення. Вимоги – це вихідні дані, на підставі яких проектуються й створюються автоматизовані інформаційні системи. Первинні дані надходять із різних джерел, характеризуються суперечливістю, неповнотою, нечіткістю, мінливістю. Вимоги потрібні в частковості для того, щоб Розроблювач міг визначити й погодити із Замовником тимчасові й фінансові перспективи проекту автоматизації. Тому значна частина вимог повинна бути зібрана й оброблена на ранніх етапах створення АИС. Однак зібрати на ранніх стадіях всі дані, необхідні для реалізації АИС, вдається тільки у виняткових випадках. На практиці процес збору, аналізу й обробки розтягнуть у часі протягом усього життєвого циклу АИС, найчастіше нетривіальний і містить множина підводних каменів; докладніше про процес – у лекціях

Класифікація вимог

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

Вимоги до продукту й процесу

Вимоги до продукту . У своїй основі вимоги – це те, що формулює замовник. Ціль, що він переслідує – одержати гарний кінцевий продукт: функціональний і зручний у використанні. Тому вимоги до продукту є основним класом вимог. Більш докладно вимоги до продукту деталізуються в наступним нижче класифікаціях.

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

Наскільки докладно Замовникові варто регламентувати вимоги до проекту – питання риторичний. Відповідь на нього залежить об множини факторів, таких, як цінність кінцевого продукту для Замовника, ступінь довіри Замовника до Розроблювача, сума підписаного контракту, ув'язування строку здачі продукту в експлуатацію з бізнес-планами Замовника й т.д. Однак, з усією визначеністю можна сказати наступне: 1)

регламентація процесу Замовником дозволяє знизити його ризики; 2) заходу Замовника по регламентації процесу приводять до додаткових накладних витрат. Потрібно знайти розумний компроміс між ступенем контролю ризиків і величиною витрат.

Як вимоги до проекту можуть бути внесені регламент звітів Розроблювача, спільних семінарів по оцінці проміжних результатів, визначені характеристики компетенций учасників робочої групи, що виконують проект, їхня кількість, зазначена методологія керування проектом. Нижче сформульований приклад формулювання вимоги до офшорного проекту (Замовник і Розроблювач фізично перебувають у різних державах) – у цій ситуації Замовникові потрібен твердий контроль над Розроблювачем.

1) Розроблювач представляє Замовникові погоджений план робіт с деталізацією (WorkBreakdownStructure - WBS) з точністю до конкретних виконавців.

2) Розроблювач здійснює щоденні зборки, регресійне тестування компонентів розроблювального продукту й тестування продукту в цілому.

3) Всі управлінські й проектні артефакти, вихідні коди й тестові приклади розміщаються в режимі online в інтегрованому середовищі розробки Rational ClearCase з можливістю для Замовника здійснення online-моніторингу на базі web-технологій.

Рівні вимог

Впровадження ИС на підприємстві завжди переслідує конкретні бізнеси-мети – такі, як, наприклад, підвищення прозорості бізнесу, скорочення строків обробки інформації, економія накладних витрат і т.д. Сучасні інформаційні системи – це великі програмні системи, що містять у собі множина модулів, функціональних, интерфейсных елементів, звітів і т.д. Як охопити єдиним поглядом такі різнорідні речі, як мети, переслідувані топ-менеджером підприємства Замовника, опис інтерфейсу користувача й характеристики модуля, що здійснює розрахунок собівартості виробу?

На щастя, людство вже давно винайшло прийоми боротьби зі складністю, широко застосовувані в моделюванні складних об'єктів – абстракцію й декомпозицію. Стосовно до дисципліни аналізу вимог до програмних систем ці принципи працюють у такий спосіб. Вимоги розділяються по рівнях. Рівні вимог зв'язані, з одного боку, з рівнем абстракції системи, з іншого боку - з рівнем керування на підприємстві.

Звичайно виділяють три рівні вимог.

На верхньому рівні представлені так звані бізнеси-вимоги (business requirements). Приклади бізнесу-вимоги: система повинна скоротити строк оборотності оброблюваних на підприємстві замовлень у три рази. Бізнеси-вимоги звичайно формулюються топ-менеджерами, або акціонерами підприємства.

Наступний рівень – рівень вимог користувачів (user requirements). Приклад вимоги користувача: система повинна представляти діалогові засоби для уведення вичерпної інформації про замовлення, наступну фіксацію інформації в базі даних і маршрутизації інформації про замовлення до співробітника, відповідальному за його планування й виконання. Вимоги користувачів часто бувають погано структурованими, що дублюються, суперечливими. Тому для створення системи важливий третій рівень, у якому здійснюється формалізація вимог.

Третій рівень – функціональний (functional requirements). Приклад функціональних вимог (або просто функцій) по роботі з електронним замовленням:

замовлення може бути *створений, відредагований, вилучений* і *переміщений* з ділянки на ділянку.

Існують об'єктивні протиріччя між вимогами різних рівнів. Так, очевидним бізне-вимогою є вимога про повноту інформації, що збирає на робочих місцях користувачів у єдину базу даних. Ніж повніше інформація – тим глибше база для аналізу діяльності й прийняття рішень. З іншого боку, конкретному користувачеві системи цілком може бути досить використання тільки тієї частини інформації, що впливає на виконання його основних функцій.

Важливі правила впровадження й використання АІС на підприємстві – «Одна крапка збору», «Дані збираються там, де вони з'являються». Використання цих правил дозволяє уникнути витрат на необґрунтоване дублювання інформації й, що важливіше – втрат від помилок обліку, що неминуче виникають при дублюванні крапок уведення.

Впровадження АІС на підприємстві приводить до необхідності оснащення всіх крапок уведення інформації автоматизованими робітниками місцями (АІС), навчанню персоналу й, найчастіше, оптимізації й підвищенню рівня формалізації робочих процесів, виконуваних персоналом. Тому впровадження АІС – непростий процес, що часто вимагає «перекроювання людського матеріалу» і зустрічаючий опір з боку користувачів, які не готові, або не хочуть працювати по-новому.

Системні вимоги й вимоги до програмного забезпечення

Існують різні трактування поняття «Системні вимоги» (system requirements).

К. Вігерс формулює даний термін, як «високорівневі вимоги до продукту, які містять багато підсистем, тобто системі». При цьому під системою розуміється програмна, програмно-апаратна, або людино-машинна система. Дана система є складною, структурованою системою й системними вимогами є підмножиною функціональних вимог до продукту. У дану підмножину доцільно відносити найбільш важливі, істотні вимоги, які ставляться в цілому до системи й не містять надлишкової деталізації.

INCOSE (International Council on Systems Engineering) дає більше детальне визначення системи: «комбінація взаємодіючих елементів, створена для досягнення певних цілей; може включати апаратні засоби, програмне забезпечення, убудоване ПЗ, інші засоби, людей, інформацію, техніки (підходи), служби й інші підтримуючі елементи». Таким чином, відбувається поділ між системними вимогами, як узагальнюючому поняттю й вимогами до програмного забезпечення, як виділеному підмножині системних вимог, спрямованих винятково на програмні компоненти системи. Цей же підхід простежується в стандарті ДЕРЖСТАНДАРТ Р ИСО/МЭК 12207/99 [14]: роботи, пов'язані із системою в цілому й із програмним забезпеченням виділяються в окремі групи з метою зручності оперирования.

У практиці комп'ютерної інженерії існує інший, більше вузький контекст використання даного поняття: під системними вимогами у вузькому змісті розуміється вимоги, висунуті прикладною програмною системою (зокрема – інформаційної) до середовища свого функціонування (системної, апаратної). Приклад таких вимог – тактова частота процесора, об'єм пам'яті, вимоги до вибору операційної системи.

Функціональні, нефункціональні вимоги й характеристики продукту

Функціональні вимоги регламентують функціонування або поводження системи

(behavioral requirements). Функціональні вимоги відповідають на запитання «що повинна робити система» у тих або інших ситуаціях. Функціональні вимоги визначають основний «фронт робіт» Розроблювача, і встановлюють мети, задачі й сервіси, надавані системою Замовникові.

Функціональні вимоги записуються, як правило, за допомогою правил, що пропонують: «система повинна дозволяти комірникові формувати прибуткові й видаткові накладні». Іншим способом є так звані варіанти використання (uses cases) – популярний і досить продуктивний спосіб подання вимог.

Це – основний, визначальний вид вимог, що буде розглядатися протягом усього лекційного курсу.

Нефункціональні вимоги, відповідно, регламентують внутрішні й зовнішні умови або атрибути функціонування системи. К. Вігерс виділяє наступні основні групи нефункціональних вимог:

- Зовнішні інтерфейси (External Interfaces),
- Атрибути якості (Quality Attributes),
- Обмеження (Constraints).

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

Основні *атрибути якості*:

- Застосовність,
- Надійність,
- Продуктивність,
- Експлуатаційна придатність, досить добре

розкриті в моделі FURPS (див. нижче).

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

Характеристики продукту. К. Вигерс формулює характеристику, «фичу» (feature), як набір логічно зв'язаних функціональних вимог, які забезпечують можливості користувача й задовольняють бізнесів-мети.

Існує й більше загальний погляд на дане поняття[2]: «features можуть бути як стосовним до функціональних, так і до нефункціональних вимог і можуть змінюватися від версії до версії продукту».

Відзначається, що «з погляду інженерії вимог, features є самостійним артефактом, що може бути співвіднесений як з функціональними вимогами, так і з нефункціональними».

Роль характеристик проявляється в першу чергу в області маркетингу: не всякий потенційний споживач продукту стане читати його функціональні описи, а набір ключових характеристик, 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 використається модель FURPS+ з посиланням на стандарт IEEE Std 610.12.1990.

Акронім FURPS позначає 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,

частина з яких уже була розглянута вище.

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

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

Методології й стандарти, що регламентують роботу з вимогами

Серед основних нормативних документів в області 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 ДЕРЖСТАНДАРТ:
   - ДЕРЖСТАНДАРТ 34.601-90. Інформаційна технологія. Автоматизовані системи. Стадії створення.
   - ДЕРЖСТАНДАРТ 34.602-89. Інформаційна технологія. Технічне завдання на створення автоматизованої системи
   - ДЕРЖСТАНДАРТ 19.201-78. Єдина система програмної документації. Технічне завдання. Вимоги до змісту й оформлення.

Тема 2. Вимоги та їх властивості. Процес аналізу вимог

Властивості вимог. Повнота. Ясність Коректність і узгодженість (несуперечність). Верифіцируємість. Необхідність і корисність при експлуатації. Здійсненність. Трасируємість. Впорядкованість за важливістю і стабільності. Наявність кількісної метрики. Яких вимог не повинно бути

Процес аналізу вимог. Робочий потік аналізу вимог. Хто створює та використовує вимоги. Організація роботи з вимогами на прикладі MSF.

Ф. Брукс у своєму, що тепер уже стали класичним, у такий спосіб охарактеризував роль вимог у розробці програмного забезпечення. Найсуворіше і єдине правило побудови систем програмного забезпечення (ПЗ) – вирішити точно, що ж будувати. Ніяка інша частина концептуальної роботи не є такий важкої, як з'ясування деталей технічних вимог, у тому числі й взаємодія з людьми, з механізмами й з іншими системами ПО. Ніяка інша частина роботи так не псує результат, якщо вона виконана погано. Помилки ніякого іншого етапу роботи не виправляються так важко (кінець цитати).

Наука добування й формалізації якісних (іноді говорять «гарних», «правильних») вимог носить багато в чому емпіричний характер. Однак, у практиці розробки програмних систем нагромадилися певні подання про те, якими властивостями повинні володіти вимоги до програмної системи. Це:

- повнота,
- ясність,
- коректність,
- погодженість,
- верифицируемость,
- необхідність,
- корисність при експлуатації,
- осуществимость,
- модифицируемость,
- трассируемость,
- упорядкованість по важливості й стабільності,
- наявність кількісної метрики.

Більшість із цих властивостей розкрито в першому розділі стандарту IEEE і широко обговорюється в роботах. 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 справа з характеристиками ще неіснуючої системи. Ідея про те, що необхідно сформулювати всі вимоги повністю, тобто вичерпним образом, до початку проектування, а тим більше – реалізації системи, зжила себе разом з так званим каскадним підходом, що підтримував послідовну модель реалізації системи. Спіральний підхід, на якому базується більшість сучасних методологій, передбачає поетапне виділення й деталізацію вимог на всьому протязі циклу розробки системи.

Проте, вимога повноти пред'являється до вимог, які формуються до системи. Треба розуміти, що дана вимога – це скоріше тенденція, мета, до якого потрібно постаратися максимально наблизитися на як можна більше ранніх стадіях проекту.

Вимога повноти можна розглядати у двох аспектах: повнота окремої вимоги й повнота системи вимог.

Повнота окремої вимоги – властивість, що означає, що текст вимоги не вимагає додаткової деталізації, тобто в ньому передбачені всі необхідні нюанси, особливості й деталі даної вимоги.

Повнота системи вимог – властивість, що означає, що сукупність артефактів, що описують вимоги, що вичерпує образом описує все те, що потрібно від розроблювальної системи.

Ясність (недвозначність, визначеність, однозначність специфікацій). Кожний зі співвласників[6] розроблювальної системи має свій особистий досвід сприйняття подій зовнішнього миру. Слово, вимовлене вголос, викликає індивідуальні асоціації в семантичному просторі кожного окремого сприймаючого суб'єкта. Те, що є ясним, допустимо, для кардиохирурга , зовсім необов'язково буде таким для фахівця в області програмної інженерії.

Відповідно, вимога має властивість ясності, якщо воно подібним образом сприймається всіма співвласниками системи. На практиці ясність вимог досягається в тому числі й у процесі консультацій, у ході яких відбувається «вирівнювання тезаурусів» співвласників системи. Гарною підмогою в цьому служить погоджений сторонами глоссарий ключових понять предметної області.

К. Вигерс дає наступну пораду по підвищенню ясності документів: «Пишіть документацію просто, коротко й точно, застосовуючи лексику, зрозумілу користувачам».

Ще однією стороною поняття «ясність вимоги» є його прослеживаемость (див. також поняття трассируемости нижче по тексту). Вимога, що сформульована ясно, може бути простежено, починаючи від того документа, де воно сформульовано вперше, аж до робочих специфікацій.

Коректність і погодженість (несуперечність).

Коректність – одне з найважливіших властивостей вимог. К. Вигерс в [8] уводить поняття коректності вимоги через точність опису функціональності . У цьому змісті коректність деякою мірою конкурує з повнотою. Але є й розходження: якщо властивість повноти носить скоріше якісний характер: абсолютна повнота представляє недосяжний ідеал, до якого можна наближатися, то властивість коректності носить оцінний характер і задає дихотомію: кожне з вимог або коректно, або немає. Крім того, можна міркувати про взаємну коректність вимог або погодженості (несуперечності): якщо дві вимоги вступають у конфлікт, значить – як мінімум одне з них некоректно. В ієрархії вимог (див. матеріали лекції 2) можна виділити вертикальну й горизонтальну погодженість. Іншими словами, вимоги не повинні суперечити, відповідно, вимогам свого рівня ієрархії й вимогам «батьківського» рівня. Так, вимоги користувачів не повинні суперечити бізнесам-вимогам, а функціональні вимоги – вимогам користувача.

Верифицируемость (придатність до перевірки). Ознаки (властивості) вимог, розглянуті в дійсній лекції, не можна вважати незалежними. У математичній статистиці

такі ознаки називаються кореллируемыми. Так, властивість верифицируемости істотно зв'язана із властивостями ясності й повноти: якщо вимога викладена мовою, зрозумілому й однаково сприйманим учасниками процесу створення інформаційної системи, причому воно є повним, тобто жодна з важливих для реалізації деталей не упущена – виходить, це вимога можна перевірити. При цьому в ході перевірки в сторін (приймаючої й сдающей роботу) не повинне виникнути нерозв'язних протиріч в оцінках. Методи верифікації вимог будуть розглянуті в лекції 6. Тому що добре сформульовані вимоги становлять основу успішного створення системи – роль верифицируемости важко переоцінити. Вимоги до системи представляють основу контракту між Замовником і Виконавцем і якщо дані вимоги не можна перевірити – значить і контракт не має ніякого змісту, отже, успіх або невдача проекту будуть залежати тільки від емоційних оцінок сторін і їхньої здатності домовитися, а це – занадто хибка основа для здійснення робіт.

Необхідність і корисність при експлуатації. Одні із самих суб'єктивних і властивостей, що перевіряють важко, вимог.

Возвращаясь до ієрархії вимог у лекції 2, найбільш безперечними вимогами варто вважати бізнесу-вимоги. Дані вимоги формулюють перші обличчя, що представляють Замовника, і навряд чи хто- нибудь краще їх зможе сказати, яким умовам повинна відповідати створювана інформаційна система, щоб відповідати бізнесам-метам підприємства. Проте, якщо в представника Виконавця виникають сумніву в необхідності тієї або іншої бізнесу-вимоги, викликані інтуїтивними міркуваннями, або досвідом впровадження інформаційних систем на аналогічних підприємствах, він повинен виявити ініціативу й зібрати спільну нараду сторін. Аргументи на користь відсутності необхідності вимоги безсумнівно будуть сприйняті, особливо якщо вони будуть мотивовані в бізнесі-термінології Замовника й підтверджені викладеннями, що прогнозують співвідношення витрат на виконання вимоги й очікуваної від нього ефективності.

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

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

Більше слабкої, чим «необхідність» формулюванням володіє властивість «корисність при експлуатації». Розмежування між даними властивостями можна провести в такий спосіб. Необхідними варто вважати властивості, без виконання яких неможливо, або утруднене виконання автоматизованих бізнесів-функцій користувачів; корисними при експлуатації варто вважати будь-які властивості, що підвищують ергономічні якості продукту.

Осуществимость (выполнимость). Є до деякої міри конкуруючої по уведеним вище двох властивостях.

У принципі ніхто не заважає сформулювати вимогу, выполнимость якого обмежується сьогоднішнім рівнем розвитку техніки й технології, хоча багато чого з того, що було нездійсненно десять років тому, цілком здійсненно сьогодні. Можна сформулювати вимога, выполнимость якого обмежена науковими поданнями про будову Вселеної, наприклад – вимога миттєвої передачі інформації із земної станції на Марса, хоча й фундаментальних поданнях іноді міняються, нехай і не так швидко, як розвиток IT-технологій.

Вертаючись із небес на землю, відринемо ті вимоги, які можна визнати абсурдними й зупинимося на тих, які здійсненні принципово. З погляду науки керування вимогами, далеко не все з них є здійсненними.

Виконуємість вимоги на практиці визначається розумним балансом між цінністю (ступенем необхідності й корисності) і потрібними ресурсами. Так, якщо вартість контракту на розробку інформаційної системи становить $10000, а витрати на виконання нової вимоги, що виникло в момент, коли проект виконаний наполовину, оцінюється в $4000, чи є воно нездійсненним? Швидше за все, так, якщо Виконавець доведе Замовникові новизну вимоги (вимога не входила в погоджені специфікації) і складність його виконання. Але, якщо вимога є критично важливим, необхідним, але випало з поля зору під час підписання контракту Замовник готовий виділити додатково фінансування, а Виконавець – трудові ресурси – виходить, вимога здійсненна. Таким чином, вимога існуємості в ряді випадків також варто вважати суб'єктивним, а критерії його оцінки лежать в області домовленостей між Замовником і Виконавцем.

Відмінною ілюстрацією балансування між цінністю та виконуємістю вимог є так званий трикутник компромісів.

Як пояснення до малюнка приведемо цитату з «білих сторінок», розміщених Microsoft у відкритому доступі. Добре відома взаємозалежність між ресурсами проекту (людським і фінансовими), його календарним графіком (часом) і реалізованими можливостями (рамками). Ці три змінні утворять трикутник, показаний на мал.. Після досягнення рівноваги в цьому трикутнику зміна на кожній з його сторін для підтримки балансу вимагає модифікацій на іншій (двох інших) сторонах і/або на споконвічно зміненій стороні.

Необхідно забезпечити можливість переробки вимог, якщо знадобиться, і підтримувати історію змін для кожного положення. Для цього всі вони повинні бути унікально позначені й позначені, щоб ви могли посилатися на них однозначно. Кожна вимога повинне бути записане в специфікації тільки один раз. Інакше ви легко одержите непогодженість, змінивши тільки одне положення із двох однакових.

Краще використайте посилання на первісні твердження, а не дублюйте положення. Модифікація специфікації стане набагато легше, якщо ви складете зміст документа й

покажчик. Збереження специфікації в базі даних комерційного інструмента керування вимогами зробить їх придатними для повторного використання (кінець цитати).

Трасируємість. Трасируємість вимоги визначається можливістю відстежити зв'язок між ним і іншими артефактами інформаційної системи (документами, моделями, текстами програм та ін.). Окрема трасу являє собою спрямоване бінарне відношення, задане на множині артефактів ИС, де перший елемент відносини представляє відповідну вимогу, а другий – артефакт, залежний від даної вимоги. На практиці трасування аналізуються за допомогою графовых, або табличних моделей.

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

Інша мета трасування – підвищити керованість проектом: при зміні окремо взятої вимоги стає зрозуміло – які із проектних, робочих і інших артефактів підлягають зміні (див. також матеріали лекції 6).

Упорядкованість по важливості й стабільності. *Пріоритет вимоги* являє собою кількісну оцінку ступеня значимості (важливості) вимоги. Пріоритети вимог звичайно призначає представник Замовника. Розроблювач, відштовхуючись від пріоритетності вимог, управляє процесом реалізації інформаційної системи.

Стабільність вимоги характеризує прогнозну оцінку незмінності вимог у часі.

Наявність кількісної метрики. Кількісні метрики відіграють важливу роль у верифікації й атестації інформаційних систем. У першу чергу це ставиться до нефункціональних вимог, які, як правило, повинні мати під собою кількісну основу (запит повинен спрацьовуватися не більш, ніж ___ секунд; середній наробіток на відмову повинна становити не менш, ніж ___ годин). Функціональні вимоги також можуть розширюватися кількісними мірами за допомогою так званих аспектів застосовності (див. матеріал лекції 5).

Яких вимог не повинне 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 – одне з типових «пасток», типових для недосвідчених команд розроблювачів. Варіантів реалізації завжди більше, ніж один, а для ухвалення зваженого рішення потрібна максимально більше повна інформація. Тому етапи роботи з вимогами, проектування й реалізації плануються по черзі, хоча й можуть бути частково запараллелены в рамках ітераційного підходу до створення програмних систем (див. матеріали заключної лекції).

Робочий потік аналізу вимог

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

Для його позначення в англомовній літературі, як правило, використається поняття «Requirement Process». У вітчизняній практиці, поряд з узагальнюючим терміном «аналіз вимог», прийнятим, зокрема, у ДЕРЖСТАНДАРТ Р ИСО/МЭК 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, який будемо дотримуватися протягом лекційного курсу.

SWEBOK пропонує виділити в Requirement Process наступні основні складові:

- Requirements Elicitation (Добування вимог),
- Requirements Analysis (Аналіз вимог у вузькому змісті),
- Requirements Specification (Специфицирование вимог),
- Requirements Validation (Перевірка вимог).

В якості приклада альтернативної декомпозиції потоку робіт можна розглянути погляд, запропонована в RUP [6]. 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 (Керування контекстом системи),
- Refine the System Definition (Уточнення визначення системи).

Узагальнюючи зазначені вище декомпозиції, а також підходи, описані в [9,10-12], далі будемо дотримуватися наступної, більше зручної в методичному плані схемою декомпозиції потоку робіт «Робота з вимогами»:

- Формування бачення;
- Виявлення вимог;
- Класифікація й специфікація вимог;
- Розширений аналіз вимог (моделювання й прототипирование);
- Документування вимог;
- Перевірка вимог;
- Керування вимогами;
- Удосконалювання 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 використатися командою розроблювачів і, по-друге, правильно їх організувати. У питання організації входить упорядкування робіт у часі, інтерфейси між ними, паралелізм, робота з ризиками й багато чого іншого.

Знайти відповідь на перше питання може допомогти загальна класифікація задач, робіт і операцій програмної інженерії, представлена в ДЕРЖСТАНДАРТ Р ИСО/МЭК 12207-99. Інша, більше пізня за часом класифікація, є присутнім в SWEBOK. Однак потрібно відзначити, 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 й сотні різних методологій (процесів), серед найбільш відомих – MSF, RUP, Oracle PJM, XP, FDD, SCRUM, PSP, Crystal, DSDM.

Методології підрозділяються на 3 «хвилі»: каскадні (історично перші), прогнозуючі (наприклад, RUP) і «швидкі» (agile), що ввійшли в широку практику на рубежі тисячоріч.

Опису методологій істотно рознятися об'ємом (від десятків до тисяч сторінок тексту), наборами базових робіт і робочих кваліфікацій, акцентами (робота з моделями, керування ризиками, побудова команди та ін.). Але автори їхніх описів звичайно сходяться в одному: краща з методологій, який потрібно випливати, щоб домогтися успіху – саме та, котру пропонує (описує, рекламує) автор. Рідкісним винятком є роботи А. Коберна, автора групи методологій Crystal, де він пропонує брати за основу не «найкращий» із процесів, а той, котрий, по-перше, щонайкраще відповідає проектній задачі, а в других – команді, що буде його реалізовувати. Вводиться кілька метрик, що дозволяють частково формалізувати процес підбора методології.

Чому потрібно аналізувати вимоги?

Із сказаного вище випливає, що не всі роботи й операції, відомі в програмній інженерії, використаються в тій або іншій методології й, 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 об'єм результатів, очікуваних від АТ?

З усією очевидністю можна затверджувати: так, АТ, як етап розробки ИС, неможливо пропустити: цей етап закладає фундамент усього процесу проектування й реалізації системи. Ступінь пророблення АТ може бути різної: від зовсім неформальної записки, представленої на одній сторінці, до розгорнутої системи документів, моделей і прототипів, побудованої відповідно до принципів однієї із прогнозуючих методологій, наприклад, RUP. Вона залежить від наступних основних факторів: розмірів проекту, величини наявних ресурсів і ступеня ризиків. Невисока глибина пророблення прийнятна для невеликих проектів, що характеризуються невеликим ресурсом і невисокими ризиками. Добре пророблені вимоги дозволяють:

- выработать загальне розуміння між Замовником і Розроблювачем;
- визначити рамки проекту;
- більш точно визначити фінансові й тимчасові характеристики проекту;
- убезпечити Замовника від ризику одержати продукт, у якому він не зможе працювати,
- убезпечити Розроблювача від ризику потрапити в ситуацію «неконтрольованого розмиття границь », що може привести до непередбачених витрат ресурсів понад початкові очікування.

Аналіз вимог – це процес (процес-процес-бізнес-процес) і, отже, до нього підходять методи й засоби процесного підходу до керування.

Одне із 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 *мети* переслідує процес.

RUP пропонує наступні цілі для потоку робіт АТ:

- домогтися однакового розуміння із замовником і користувачами про те, що повинна робити система;
- дати розроблювачам найкраще розуміння вимог до системи;
- визначити границі системи;
- визначити інтерфейс користувача й системи.

Хто створює й використає вимоги

Як і ким використаються вимоги?

Фахівець із АТ – постановка задачі, визначення рамок проекту

Представник замовника – постановка задачі, визначення рамок проекту, контроль роботи виконавця, приймання результатів роботи.

Архітектор системи – розробка архітектури, проектування підсистем

Програміст – розробка програмного коду.

Тестировщик – складання тест-плану, тестових сценаріїв.

Менеджер проекту – планування й контроль виконання робіт.

У рамках курсу лекцій для всіх згаданих вище осіб будемо використати узагальнюючий термін «Співвласники (зацікавлені сторони)» (stakeholders). Співвласниками, слідом за розроблювачами RUP і MSF (див., наприклад, [9,13]), будемо називати всіх учасників проекту створення програмної системи, прямо або побічно зацікавлених у його успіху. Автори більшості сучасних методологій розробки програмних систем сходяться тім, що в групі співвласників ключову роль грають дві групи представників Замовника – ті, хто ставить стратегічні цілі й виділяє фінансування й ті, хто буде безпосередньо використати розроблений продукт. Причому, на відміну від каскадних методів, де Замовник підключався в початковій фазі – складанні технічного завдання й кінцевої – прийманню готової роботи, у сучасних методологіях Замовник, дійсно зацікавлений в успіху проекту автоматизації, повинен брати участь у ньому **безупинно.**

Організація роботи з вимогами на прикладі MSF

В MSF для позначення ролі учасників команди софтверного проекту використається поняття рольових кластерів.

MSF заснованийий на постулатіі про шіст якісних цілях, досягнення яких визначає успішність проекту. Ці мети спричиняються модель проектної групи. У той час як за успіх проекту відповідальна вся команда, кожний з її рольових кластерів, обумовлених моделлю, асоційований з однієї зі згаданих шести цілей і працює над її досягненням.

Шість рольових кластерів моделі проектної групи – це “Керування продуктом” (product management), “Керування програмою” (program management), “Розробка” (development), “Тестування” (test), “Задоволення споживача” (user experience) і “Керування випуском” (release management). Вони відповідальні за різні області компетенції (functional areas) і пов'язані з ними мети й задачі.

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

- Envisioning (вироблення концепції),
- Planning (планування),
- Developing (розробка),

- Stabilizing (стабілізація),
- Deploying (впровадження).

В фазі вироблення концепції робота з вимогами найбільш інтенсивна.

Табл. 1.

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

Як видно з таблиці, всіх 6 кластерів працюють зі своїми групами вимог.

Триває щільна робота з вимогами й на наступній фазі – фазі планування, див. табл. 2.

Табл. 2.

Рольовий кластер	Фокус
Керування продуктом	Аналіз вимог-бізнесів-вимог
Керування програмою	Функціональна специфікація
Задоволення споживача	Сценарії/приклади використання, користувальницькі вимоги, вимоги локалізації й загальнодоступності (accessibility).
Тестування	Вимоги тестування.
Керування випуском	Експлуатаційні вимоги.

У фазах розробки й впровадження робота з вимогами зосереджує в кластерах керування продуктом і програмою, див., відповідно, табл. 3,4.

Табл. 3.

Рольовий кластер	Фокус
Керування продуктом	Очікування замовника.
Керування програмою	Керування функціональною специфікацією.

Табл. 4.

Рольовий кластер	Фокус
Керування продуктом	Одержання відкликань і оцінок замовника; акт про прийом виконаної роботи.
Керування програмою	Зіставлення рамок проекту з поставленим рішенням; керування стабілізацією.

Тема 3. Контекст завдання аналізу вимог. Виявлення вимог

Властивості вимог Аналіз вимог, бізнес-аналіз, аналіз проблемної області. Методології бізнес-аналізу. Вимоги та архітектура АІС. Аналіз вимог і інші робочі потоки програмної інженерії. Аналіз вимог, бізнес-аналіз, аналіз проблемної області. Джерела вимог. Стратегії виявлення вимог. Інтерв'ю, Анкетування. Спостереження. Самостійний опис вимог. Спільні семінари. Прототипування.

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

Автори, «батьки-засновники» RUP і UML, у цьому питанні дають певну волю : можна створювати модель^-моделі-бізнесів-моделі за допомогою відповідних розширень UML і рекомендацій RUP, а можна обмежитися виробленням глоссария об'єктів предметної області. Як і в питанні вибору глибини пророблення артефактів АТ, питання – проводити або не проводити аналіз^-аналіз-бізнес-аналіз (або, точніше кажучи, аналіз проблемної області), вирішується залежно від конкретної задачі.

Роль глоссария при АТ. Трохи перебільшуючи, можна сказати, що Замовник і Розроблювач завжди розмовляють різними мовами. Загальне розуміння виробляється із працею, цей процес забирає час, але важливість його важко переоцінити: адже успішна реалізація проекту в області й впровадження АИС багато в чому залежить від того, чи вдасться виробити й документувати їхнє загальне подання про предмет розробки. Якщо ж Розроблювач іде ще далі й вникає особливо ведення справ на підприємстві Замовника – він, по-перше, зможе домогтися кращого розуміння вимог до АИС і, по-друге, брати участь поряд із Замовником у формулюванні цих вимог, аналізі пропущених вимог та ін. Глосарій (докладніше див. в. лекції 4) можна розглядати, як документ, що засвідчує загальне розуміння основної термінології Замовником і Розроблювачем.

Задачу аналізу бізнесів-процесів (ділове моделювання), настільки популярну в останні десятиліття через стійку кон'юнктуру, варто розглядати, як частина більше загальної задачі, аналізу проблемної області. Роботи, присвячені аналізу проблемної області, з'явилися у вітчизняній літературі в середині минулого століття; дана тематика нерозривно пов'язане із задачним підходом і інженерією експертних систем. Чи застосовні методи, прийняті при побудові інтелектуальних систем для такої «більше приземленої» задачі, як задача побудови АІС – безумовно, так. Так, стратегії добування знань багато в чому перетинаються з рекомендаціями з роботи аналітика, методи рішення задачі шляхом редукції на підзадачі й пошуку в просторі станів знайшли своє відбиття в множині методик бізнесу-аналізу, аналізу й синтезу програмних систем і цей список можна продовжувати. Інше питання – наскільки результативне застосування тих або

інших моделей і методів при описі організаційних систем.

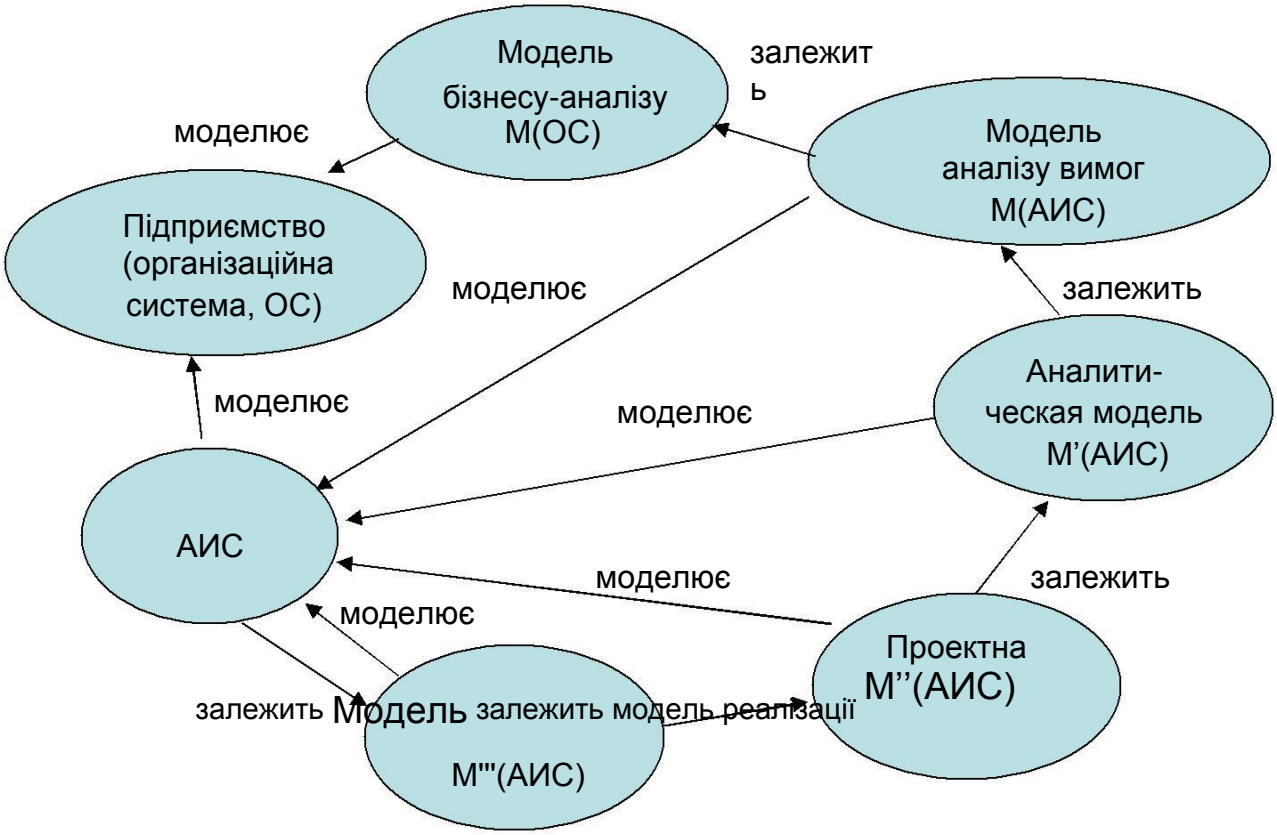
Ключ до рішення цього питання лежить у наступному: спочатку треба визначити мети задачі самого бізнесу-аналізу, як етапу побудови КИСНУВ.

З позицій моделювання, аналіз вимог (АТ) і аналіз проблемної області (АПО) – принципово різні процеси.

АПО переслідує класичні цілі створення моделі: у наявності об'єкт (автоматизируемое підприємство або організаційна система, ОС) і задача аналітика – відбити цей об'єкт у створюваній моделі з необхідним ступенем точності (мал.1).

Аналіз вимог, навпроти, спрямований на моделювання уявлюваного, що ще не існує об'єкта (АІС) (мал. 2). Т. е. спочатку створюється модель, а потім, на її підставі, синтезується об'єкт.

Для того, щоб прояснити зв'язок між цими процесами, необхідно помітити, що створювана АІС також є моделлю, по відношенні до 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 маючи модель АПО, ми, звичайно, можемо створити модель АТ. Але при цьому ми ризикуємо тим, що при синтезі оригіналу моделі (тобто АИС), не маючи знання про ОС, ми можемо потрапити в ситуацію неузгодженості: результуюча АИС не буде ингерентна (погоджена с) ОС і, тим самим, не стане життєздатної.



Чи треба із цього, що етап АПО є необхідною ланкою створення КИСНУЛА? Ні, не завжди. Тут доречно звернутися до класифікації задач і методологій А. Коберна. Крім того, це залежить від складу третього компонента «трикутника моделювання» – моделюючого суб'єкта, у нашому випадку – колективу Розроблювача.

Якщо моделюючий суб'єкт має неявні знання про ОС у достатньому об'ємі – виходить, АПО можна виключити. На практиці це можливо в наступних випадках: а)

Розроблювач є частиною (структурним підрозділом, дочірнім підприємством і т.д.) ОС, у колектив Розроблювача входять експерти, що добре знають предметна область; б) Замовник нарівні й Розроблювачем бере участь у створенні документа АТ і розділяє з ним відповідальність за прийняття рішень. Це – шлях «agile методологій» (див. матеріали заключної лекції).

Розглянемо тепер узагальнену «формулу» створення АИС.

ОС → М(ОС) → М(АІС) →М’ (АІС) → М’’(АІС) → М’’’(АІС) → АІС

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

Аналізуючи модель проблемної області, у ній можна вичленувати, з одного боку, задачі й функції, реалізовані усередині ОС і функції комунікації ОС і середовища, з іншого боку - пристрій предметної області (на початку – на рівні концептуальної моделі), із третьої – вимоги до інформації і її обробці. Виділивши серед функцій ті, які підлягають автоматизації, ми одержуємо основу для виявлення функціональних вимог до системи. Інша, зібрана на етапі АПО, інформація служить для пошуку нефункціональних вимог. У результаті одержуємо модель АТ, як перше наближення моделі АІС, М(АІС).

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

Модель рівня реалізації дозволяє синтезувати властиво АІС, як сукупність програмних, інформаційних, організаційних і ін. артефактів.

АІС у свою чергу являє собою модель організаційної системи М’(ОС), замикаючи цикл моделювання.

Методології бізнесу-аналізу

Методології бізнес аналізу можна розділити на три категорії по типах моделей:

- моделі, що переслідують ціль аналізу й поліпшення організаційної системи (наприклад, SWOT , VCM, BPR, CPI/TQM/ISO9000, BSC),
- моделі загального призначення, такі, як SADT, DFD, IDEF1, IDEF3, IDEF5 і інших.
- моделі, спеціально розроблені для використання при автоматизації (наприклад, ISA, BSP, ARIS, RUP).

Найбільш розвинена модель опису проблемної області пропонується в методології ARIS.

Архітектура ARIS [5] 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 використовуваних і вироблених продуктів і послуг.
- Інформаційна (підсистема даних). Описує одержання, поширення й доступ до інформації (даним).
- Підсистема процесів керування. Визначає логічну послідовність виконання функцій за допомогою подій і повідомлень. Можна сказати, що підсистема керування — це сукупність рознесених у часі повідомлень різного роду.
- Підсистема цілей організації. Описує ієрархію цілей, що досягають у ході

виконання того або іншого процесу.

- Підсистема засобів виробництва. Описує життєвий цикл основних і допоміжних засобів виробництва.
- Підсистема людських ресурсів. Описує прийом на роботу, навчання й просування по службі персоналу організації.
- Підсистема розташування організаційних структур. Описує територіальне розташування організаційних одиниць (кінець цитати). Даний

поділ є в певній мері умовним; виділені «підсистеми» не є підсистемами в змісті системного аналізу, тому що взаимопроникают і перетинаються. Вони представляють скоріше сукупність предметів дослідження, різних поглядів на досліджуваний об'єкт.

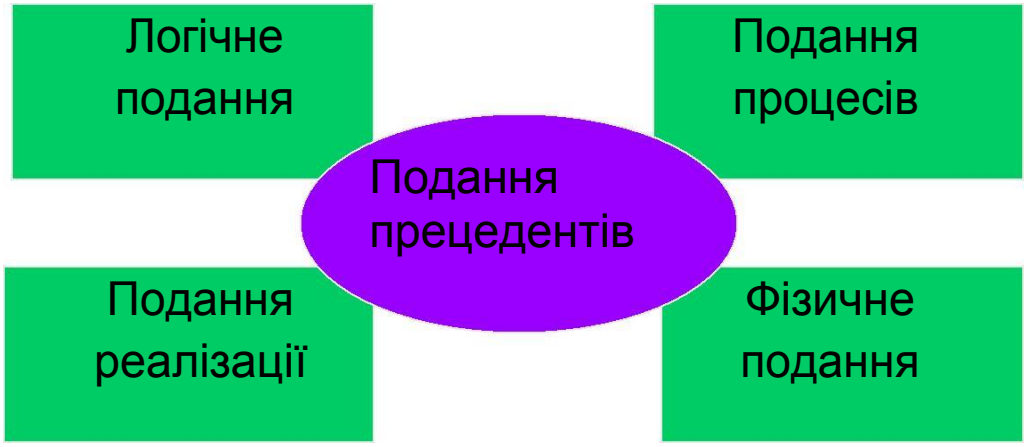
Слухачеві курсу пропонується самостійно проаналізувати, які групи й категорії вимог до системи дозволяє прояснити той або інший компонент прийнятої в ARIS структуризації об'єкта дослідження.

Вимоги й архітектура АИС

Говорячи про архітектуру АИС, звичайно підкреслюють ділення на апаратні, програмні , інформаційні, організаційні компоненти, їхній связность і деталізацію.

Метафора архітектури RUP описується у вигляді 4+1 подань: логічне, подання процесів, подання реалізації й фізичне подання зв'язуються між собою поданням варіантів використання (use case), що відіграє центральну роль у виробленні архітектури системи (мал. 2).

Вимоги первинні стосовно архітектури. Але це не виходить, що вимоги й архітектура повинні розроблятися послідовно.



Навпроти, ці процеси взаємопов'язані й повинні бути істотно запаралелені. Як тільки зібраний мінімальний набір ключових вимог, що дають розуміння про те, що потрібно робити – повинна бути знайдена архітектура, що пояснює, як це може бути реалізовано. У великих, відповідальних проектах звичайно розглядається трохи альтернативних архітектур, їхні достоїнства й недоліки стосовно до вихідних вимог.

У процесі роботи з вимогами вони деталізується, деталізується й архітектура. У випадку множинності альтернативних архитектур 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 й деталі архітектури. Стійкість архітектури проявляється в незначних її змінах при додаванні, деталізації й зміні

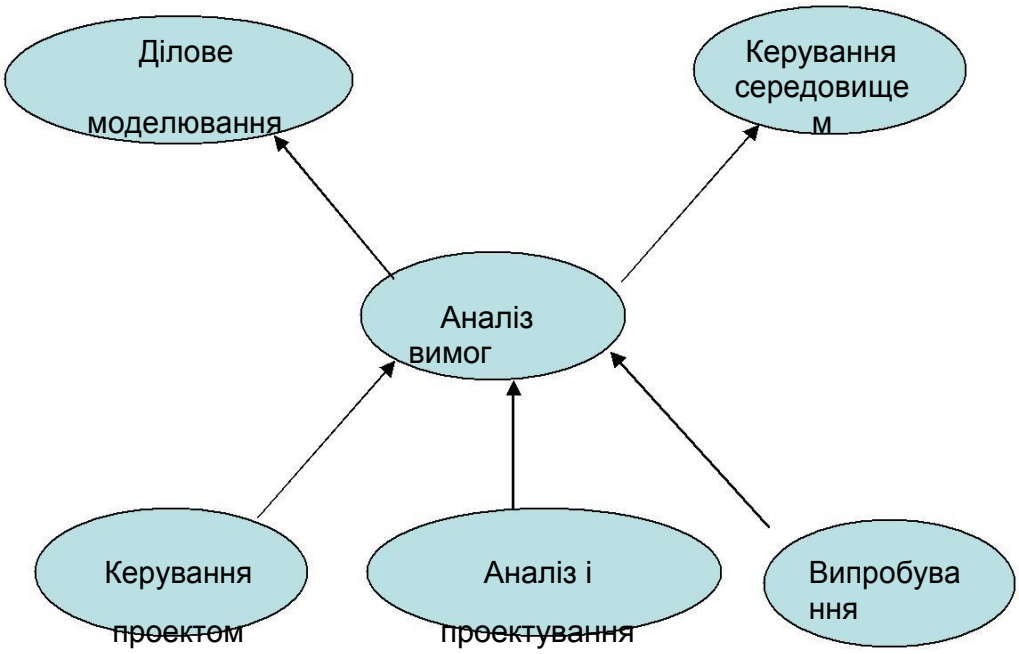
вимог. Якщо наступив момент, при якому поява нової інформації про вимоги перестала впливати на архітектуру – виходить, архітектура стабілізувалася.

Эте – нормальний, природний шлях розвитку вимог і архітектури. Але що робити, якщо вимоги змінилися настільки, що архітектура перестала їм відповідати? Причин тому можуть бути різні, наприклад: недосвідчена архітектурна група не виявила досить далекоглядності ; група по зборі вимог пропустила на ранніх стадіях критичні, архітектурно значимі вимоги; у бізнесі-сфері Замовника відбулися більші зміни, що викликали корінну зміну вимог до системи. Наслідки також можуть бути різними: домовленість про збільшення строків і сум за контрактом; виправлення ситуації за рахунок Розроблювача; розрив контракту.

Альтернативний вихід пропонується в методології XP: архітектура – не догма, а всього лише метафора. Якщо вимоги ввійшли врозріз із існуючою архітектурою – виходить, архітектуру потрібно просто змінити. Варто розуміти, що шлях і рецепти XP при гаданій простоті орієнтовані далеко не на будь-який колектив. Команда XP складається їхніх професіоналів, що мають позитивний досвід роботи в цій методології.

Аналіз вимог і інші робочі потоки програмної інженерії

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



Потік робіт «ділове моделювання» є основою для аналізу й формування вимог до АІС, дозволяє уникнути помилок.

Потік робіт «керування середовищем» надає вихідну інформацію для робочої групи АТ, що регламентує формати оформлення, CASE-засобу, регламенти роботи.

Потік робіт «керування» ґрунтується на специфікації вимог. Стратегічне й тактичне планування, формування проміжних віх (очікуваних результатів) тісно вв'язано з вимогами до системи.

Потік робіт «аналіз і проектування» здійснюється на основі вихідних даних, наданих АТ. У певній мері ці потоки робіт проводяться паралельно. При виявленні

проблем, пов'язаних з вимогами, виникає зворотний зв'язок від цього потоку робіт до потоку робіт АТ.

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

Для потоку робіт «реалізація» зв'язок з вимогами не зазначена. Тим часом автор вважає, що вимоги повинні аналізуватися й ураховуватися у ВСІХ робітників потоках проекту, навіть якщо це формально не передбачено обраним групою процесом. Людям властиво помилятися й помилки, зроблені на ранніх стадіях проекту, при русі від етапу до етапу наростають, як сніжний кому. Тому будь-якому учасникові команди, зацікавленому в успіху проекту, нелишне заглянути в специфікацію вимог і переконатися в тім, що та робота, що йому доручена, відповідає тій або іншій вимозі. Це дозволяє організувати зворотний зв'язок, що дозволяє відстежити помилки в специфікаціях. Багато проектів зайшли в тупик саме через відірваність групи, відповідальної за реалізацію від групи збору й аналізу вимог.

Джерела вимог.

Основним джерелом вимог до інформаційної системи, безумовно, є міркування, висловлені представниками Замовника. У відповідність із ієрархічною моделлю вимог дана інформація структурируется як мінімум на 2 рівні: бізнесу-вимоги й вимоги користувачів. Проблема полягає в тому , що вимоги формулюються до створюваного, що ще не існує системі, тобто по суті вирішується початкова подзадача задачі проектування АИС, а представники Замовника далеко не завжди бувають компетентні в даному питанні. Тому, поряд з вимогами, висловленими Замовником, доцільно збирати й вимоги від інших співвласників системи: співробітників аналітичної групи виконавця, зовнішніх експертів і т.д.

Результуючий, часто досить сирий матеріал розглядається, як документ «Вимоги співвласників»[7]. На вимоги співвласників звичайно не накладається ніяких спеціальних обмежень.

Продовжуючи міркування, початі в попередній лекції, модель створюваної інформаційної системи в певній мері повинна відбивати модель ОС.

Тому іншим важливим джерелом інформації, крім виявлення вимог, є артефакти, що описують предметну область. Це можуть бути документи з описом бізнесів-процесів підприємства, виконані консалтинговим агентством, або просто документи (посадові інструкції, розпорядження, зводи бізнесів-правил), прийняті на підприємстві. Однієї з деяких методологій, у яких спеціально виділяються робочий потік ділового моделювання, є Rational Unified Process.

Ще одна альтернатива, використовувана при виявленні вимог – так звані «кращі практики», широко використовувані в цей час у бізнесі-консалтингу й при впровадженні корпоративних інформаційних систем. Кращі практики являють собою описи моделей діяльності успішних компаній галузі, використовувані тривалий час у сотнях і тисячах компаній по усьому світі.

Підсумовуючи сказане, відзначимо, що основними джерелами, що утворять «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 проробити чималу роботу, пов'язану з підбором респондентів і інформаційних матеріалів, організацією інтерв'ю й т.д.

Стратегії виявлення вимог

Інтерв'ю

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

У ставшей вужі класичної, але нітрохи що не загубила актуальність монографії Д. Марко [3] у процесі проведення інтерв'ю пропонується виділити три підлеглі процесу: підготовку, проведення інтерв'ю (опитування) і завершення. Нижче приводиться короткий огляд рекомендацій Д. Марко з акцентом на виявлення вимог (у монографії дані рекомендації з інтерв'ювання з метою формування моделі об'єкта дослідження).

1. Підготовка

Підготовка дозволяє спланувати процес опитування й виробити стратегію керування 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;
- домовитеся про зустріч;
- установите попередню програму зустрічі;
- вивчите супутню інформацію;
- погодьте свої дії із групою проектування[8].

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

- Він дійсно є експертом по даному питанню;
- Його думка дійсно є коштовним при формуванні цільового набору вимог[9].

Важливо заздалегідь обмовити мета зустрічі й обмежити бесіду в межах години або менш. Практика показує, що активне спілкування в процесі інтерв'ю, як правило,

2. Проведення опитування

Нижче наведена цитата, з деякими скороченнями й виключенням матеріалу, що не ставиться до АТ.

У проведенні опитування найважливіше – правильно організувати й підтримувати потік інформації від експерта до вас. Рекомендується витратити час на обмірковування вірного початку опитування, при зборі інформації з можливості використати записи, закінчувати розмову плавно. Обговоримо докладніше кожного із цих пунктів.

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

Потім сформулюйте перше питання. Помнете, що перше питання часто задає тон всій розмові, тому добре продумайте його.

Збирайте інформацію, роблячи запису про усім (про спеціальні терміни, взаємозв'язки між частинами системи й т.п.) і обмежуючи час бесіди. Запишіть SADT-функції й дані, спробуйте накидати діаграму. Підтримуйте потік інформації, задаючи питання, які уточнюють і підтверджують відповіді.

Насамперед, не заперечуйте.

Ніколи не задавайте навідних запитань або питань із короткими відповідями "так" або "ні". Замість цього записуйте те, що вам говорять, і просите підбити підсумок або дати

пояснення.

Ви одержите від опитування більше, якщо ви дасте експертові можливість говорити те, що він хоче сказати, а то, що ви хочете почути (кінець цитати).

3. Завершення

Стежите за виникненням наступних ситуацій:

- ви вже одержали досить інформації;
- ви одержуєте великий об'єм невідповідної інформації;
- велика кількість інформації вас придушує;
- експерт починає утомлюватися;
- у вас із експертом часто виникають конфлікти.

Кожна із цих причин - достатня підстава для завершення бесіди.

Коли ви вважаєте потрібним закінчити опитування, завершуйте бесіду плавно. Коротко підсумуйте основні пункти й зробіть огляд отриманих відомостей, які можуть бути опущені або невірно витлумачені. Домовитеся про час наступної зустрічі, якщо вона потрібна, і одержите рекомендації для найближчих опитувань. Поставте експерта в популярність, коли і як ви збираєтеся використати отриману інформацію й коли ви надішлете йому матеріал на рецензування.

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

Що потрібно пам'ятати при опитуванні

Наступні рекомендації допомагають підтримувати безперервність потоку й вірогідність інформації, що надходить від експерта:

- робіть паузи, поки експерт думає. Дайте експертові можливість вирішувати, що сказати далі. Ніколи не перебивайте, підказуючи відповідь або задаючи інше питання;
- намагайтеся не задавати навідних запитань, питань-підказок, питань, що містять відповідь, тому що це не дозволяє експертові ділитися своїми знаннями. Намагайтеся не задавати контрольних питань, тому що це перериває потік інформації;
- робіть запису, щоб зосередитися на предметі розмови й щоб підготуватися до наступного питання, але не стаєте стенографом, інакше ви можете втратити контроль над опитуванням (кінець цитати).

Анкетування

Анкетування – самий малозатратный для аналітика спосіб добування інформації, він же – і найменш ефективний. Звичайно застосовується як доповнення до інших стратегій виявлення вимог.

Недоліки анкетування очевидні: респонденти часто бувають нездатні, або слабко мотивовані в тім, щоб добре й інформативно заповнити анкету . Велика ймовірність одержати неповну або зовсім помилкову інформацію. Перевага – у тім, що підготовка й аналіз анкет вимагають невеликий ресурс.

Л. Мацяшек рекомендує формулювати в анкетах *питання із замкнутим циклом відповідей* в одній з наступних трьох форм.

Многоальтернативные питання. Ця форма анкети відома всім, хто коли або

проходив тестування; може розширюватися коментарями респондента у вільній формі.

Рейтингові питання. Представляють визначений набір відповідей на сформульовані питання. Використаються такі значення, як «абсолютно згодний», «згодний», «ставлюся нейтрально», «не згодний», «абсолютно не згодний», «не знаю».

Питання з *ранжируванням*. Передбачає ранжирування (упорядочивание) відповідей шляхом присвоювання їм порядкових номерів, процентних значень і т.п.

Спостереження

Спостереження за роботою моделируемой організаційної системи - корисна стратегія одержання інформації (хоча, строго говорячи, за результатами спостереження можна одержати модель ОС, а не модель АТ).

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

Через спостереження, а можливо, і участь аналитики одержують інформацію об происходящих день за вдень 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 вимірів: співробітники організації, перебуваючи «під ковпаком» можуть почати поводитися принципово по іншому, чим звичайно.

Самостійний опис вимог

Документи – гарне джерело інформації, тому що вони найчастіше доступні і їх можна "опитувати" у зручному для себе темпі. Читання документів - прекрасний спосіб одержати первісне подання про систему й сформулювати питання до експертів.

Якщо досвідчений аналітик уже досліджував велику кількість систем такого ж типу, що й на підприємстві впровадження, він має фундаментальні знання в відповідної предметної області, щодо певного класу систем. Автори методології SADT рекомендують проводити *самоопитування* для того, щоб одержати максимальну користь від своїх знань.

За результатами аналізу документів і власних знань аналітик може скласти *опис* вимог і запропонувати його представникам Замовника як інформація до міркування, або – основи для формування технічного завдання.

Недолік цієї стратегії – небезпека пропуску знань, специфічних для об'єкта дослідження (у випадку самоопитування), або – неформалізованих знань, емпіричних правил і процедур, широко використовуваних на практиці, але не ввійшли в документи.

Спільні семінари

Крім класичного інтерв'ю «тет а тет», існує значна кількість методик, що припускають широку участь представників Замовника й Виконавця.

Правила *мозкового штурму* припускають повну розкутість і волю думок, навіть самих вигадливих і на перший погляд «маревних». Перше правило мозкового штурму – «повна заборона на будь-яку критику». Усяка висловлена думка являє цінність, а повна відсутність заборон дозволяє повноцінним образом підключити творчу фантазію.

Потім, на другому етапі, всі висловлені думки ретельним образом обговорюються, свідомо неприйнятні варіанти відсіваються, формуються колективні пропозиції.

Правила *JAD-методу* , що вважається одним із сучасних способів добування вимог,

були вперше сформульовані наприкінці 1970-х років компанією IBM. Учасники JAD-наради:

Вїдучий – 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, що обговорюють вимоги й приймають рішення.

Розроблювачі – аналитики й інші учасники проектної команди. Працюють у 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 більше продуктивна, групи швидше навчаються, більше схильні до кваліфікованих висновків, дозволяють виключити багато помилок.

Ця стратегія, мабуть, одна із самих витратних, однак вона окупається за рахунок меншої кількості помилок і відмові від формалізації на користь живого спілкування, виробленню загальної мови та ін. Деякі методології (наприклад, XP) ґрунтуються на постійному тісному контакті між Замовником і Виконавцем і, якщо такої можливості немає – XP-проект просто не зможе відбутися.

“ зустрічі, ЩоРоз'ясняють, або «запланований мозковий штурм» - термін, що прийшов із загальної практики менеджменту й базований на ідеях співробітництва зацікавлених осіб для спільного аналізу шляхів рішення проблем, визначення й попередження ризиків і т.п.

Прототипирование

Прототипирование – ключова стратегія виявлення вимог у більшості сучасних методологій (докладніше див. у лекції 5). Програмний прототип – «дзеркало», у якому видно відбиття того, як зрозумів Виконавець вимоги Замовника. Процес виявлення вимог шляхом прототипирования тим більше інтенсивний, чим це дзеркало кривее. Документальний спосіб виявлення вимог завжди уступає живому спілкуванню. Аналіз того, що зроблено у вигляді інтерфейсів користувача дає ще більший ефект. Підключається правополушарный канал сприйняття, що, як відомо, працює в більшості людей на порядок ефективніше, ніж вербальний.

Метод *RAD* – один з найбільш відомих способів швидко створювати прототипи[10].

RAD базується на наступних базових принципах:

- Еволюційне прототипирование;
- CASE-засоби, як основний інструмент, включаючи можливості прямого й зворотного проектування й автоматичної генерації коду;
- Висококваліфіковані фахівці, що добре володіють розвиненими інструментальними засобами;
- Інтерактивний JAD-метод, у якому спілкування сполучається з розробкою в режимі online;
- Тверді тимчасові рамки, як протиотрута від «розповзання границь» проекту: якщо команда не укладається в строк – функціонал звужується.

Тема 4. Формування бачення. Специфікація вимог

Бачення продукту і межі проекту. Концепція в стандартах. Бачення в RUP. Бачення / рамки MSF. Бачення продукту і межі проекту. Актори і варіанти використання. Глосарій.

Специфікація варіантів використання. Вільний формат. Шаблон повного опису варіанти використання за А. Коберну. Табличні уявлення варіанти використання. Шаблон варіанти використання RUP. Вибір форми опису варіанти використання. Специфікація функціональних вимог. Атрибути вимог.

Роботи з формування бачення продукту й границь проекту звичайно починаються на самій ранній фазі проекту, до початку широкомасштабних консультацій по виявленню докладних вимог, хоча в цілому наявність і послідовність даних кроків залежить від обраної методології. На практиці дані роботи найчастіше сполучаються. Правила добування вимог, розглянуті в лекції 3, можуть бути використані й при формуванні бачення.

Аналізуючи літературу по розглянутій тематиці, можна виділити наступні широко употребимые ключові слова: з одного боку – концепція, бачення, образ, з іншого боку - рамки, границі, контекст.

У першому випадку мова йде про бачення того, який повинна бути система. Обговорюються высокоуровневые вимоги (можливості, властивості) продукту й найбільш істотні *обмеження*. Ряд авторів, навпроти, наполягає на тому, що бачення повинне бути «*нічим не обмеженим*».

Поняття бачення широко употребимо в бізнесі-аналізі. Якщо в топ-менежмента компанії є подання про те, які ключові цілі, сегменти ринку , товарні позиції, прибуток повинні бути досягнуті, допустимо, через 5 років – виходить, 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 не від поточних ресурсів і обмежень, а від стратегічних цілей, застосовуючи інновації, ноу-хау й т.п.

Даний досвід формування бачення багато в чому переносимо й на процес розробки інформаційних систем: потрібно «побачити» в обрії середньо- і (або) довгострокового планування, як АИС впише в організаційні процеси підприємства, які ключові вигоди вона дасть, які проблеми дозволить дозволити. При пошуку нових методів і засобів керування підприємством на основі інформаційних технологій найчастіше доводиться «перекроювати» існуючі бізнеси-процеси; по суті впровадження АИС, що зачіпає істотний відсоток процесів підприємства, неминуче приводить до перебудови цих процесів з метою оптимізації діяльності підприємства, досягнення ключових факторів ефективності та ін.

У другому випадку (рамки, границі, контекст) обговорюються такі питання, як границя системи й середовища, необхідні ресурси на створення системи, строки. Побудувавши «нічим не обмежене бачення», рано або пізно доводиться повернутися до таким

прозаїчним речам, як бюджет, календарне планування, підбор персоналу, віхи проекту.

Чи завжди потрібно створювати документ «Концепція»? Чи варто розділяти бачення й границі?

Зачасту Замовник усвідомлює необхідність автоматизації, як спосіб рішення проблем, що нагромадилися. Сформулювавши для себе проблему, Замовник часто бачить і варіант її рішення, з яким приходить до Виконавця («мені потрібний сайт», «потрібна CRM-система» і т.п.). Кваліфікований Виконавець не повинен, сломя голову, поспішати вирішувати задачу у формулюванні Замовника. По образному вислові Г. Калянова[11] автоматизувати процеси «як є» – однаково, що асфальтувати доріжки, по яких ходять корови.

У нотації RUP є присутнім важлива метафора: «Побачити проблему за проблемою». Концепція саме й служить для того, щоб допомогти Замовникові виявити саме ті вимоги до системи, які допоможуть йому оптимизировать роботу свого підприємства в довгостроковій перспективі.

Поэтому етап формування концепції важливий, але він пред'являє й до Замовника й до Виконавця досить високі вимоги: Замовник повинен виділити ресурси й бути готовим до трудозатратам на спільний пошук рішень; Виконавець повинен мати достатню кваліфікацію як у сфері IT-, так і в сфері керування підприємствами, щоб розроблювальний засіб автоматизації дійсно принесло користь. Все вищесказане нітрохи не виключає можливість роботи без концепції: або мова йде про невеликий проект, закладати в бюджет якого етап вироблення концепції просто нерентабельно, або Замовник сам має достатню кваліфікацію, щоб сформулювати вимоги до АИС, маючи «концепцію в голові» і час для консультування Розроблювача.

Деякі аргументи за поділ бачення й границь були наведені вище. Провести чітку границю між цими поняттями пропонує, зокрема, процес MSF. В остаточному підсумку, питання «розділяти або не розділяти» визначається обраною методологією.

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

Концепція в ДЕРЖСТАНДАРТ

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

Основные роботи етапу:

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

Тому що даний етап хронологічно коштує на другому місці, до його початку в Розроблювача на руках уже є документ, у якому зібрані основні вимоги користувачів.

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

ДЕРЖСТАНДАРТ, на відміну від більшості сучасних методологій, у загальному випадку закладає многоальтернативность варіантів концепції системи й планів їхньої

реалізації. Кожний із пророблених варіантів оцінюється з позицій необхідних ресурсів і функціональності. Для варіантів повинні бути представлені оцінки переваг і недоліків. Корисність пророблення декількох варіантів концепції полягає в тім, що Замовникові важко сформулювати самостійно бачення системи, у той час, як вибір з набору варіантів, представлених Розроблювачем – цілком посильна задача.

Крім того, концепція повинна відбивати оцінки якості, умови приймання системи, оцінку ефекту, очікуваного від реалізації. При оформленні звіту необхідно привести обґрунтування пропонованого варіанта.

Бачення в RUP

Кроки, які необхідно пройти для формування документа «Бачення»:

- Формулювання проблем.
- Ідентифікація співвласників
- Визначення границь системи
- Ідентифікація обмежень
- Формулювання постановки задач
- Визначення можливостей системи
- Оцінка результатів

Для опису *проблем* пропонується шаблон, показаний у табл.

Проблема	(опис проблеми)
Зачіпає	(співвласники, що зачіпають проблемою).
Її наслідком є	(яке вплив проблеми).
Успішне рішення	(список деяких ключових переваг від успішного рішення).

Ідентифікація *співвласників* припускає пошук і фіксацію интересантов проекту – представників Замовника й Виконавця, інвесторів, зовнішніх експертів та ін.

Визначення *границь системи* являє собою нетривіальний процес. Для цього використають контекстні діаграми. RUP у пошуку границь пропонує відштовхуватися від акторов і варіантів використання.

Серед джерел *обмежень* звичайно виділяють:

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

Опис *можливостей* системи являє собою формулювання высокоуровневых вимог.

Шаблон документа «Vision» RUP містить наступні основні розділи:

1. Введення
2. Позиціювання
3. Опису співвласників і користувачів
4. Короткий огляд виробу
5. Можливості продукту
6. Обмеження
7. Показники якості

8. Старшинство й пріоритети
9. Інші вимоги до виробу
10. Вимоги до документації
11. Додаток.

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

В розділі «позиціювання» міститься визначення розв'язуваної проблеми (проблем), вказується цільовий замовник і досліджуються ділові переваги виробу перед аналогічними на ринку.

В описі співвласників і користувачів, крім властиво опису цих двох груп, досліджується демографія ринку: цільові ринкові сегменти; розмір і темпи росту ринку; існуючі конкурентні пропозиції на ринку; репутація Розроблювача на ринку;

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

В розділі, присвяченому можливостям продукту, вони описуються більш докладно, кожна – в окремому параграфі.

В розділ «Обмеження» варто виносити існуючі технічні, технологічні й ін. обставини, які необхідно враховувати на даній стадії.

Розділ «Показники якості» містить опис найбільш істотних нефункціональних вимог до системи (ефективності, надійності, отказоустойчивости й ін.).

Розділ «Старшинство й пріоритети» ранжирует сформульовані раніше вимоги й можливості системи по ступені важливості, черговості реалізації й т.п.

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

В вимогах до документації приводяться ключові характеристики посібника користувача, інтерактивної довідки, посібника з установки й конфігурування,
файлу Read Me.

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

Бачення / рамки в MSF

Відповідно до білої книги MSF [25], на фазі вироблення концепції (envisioning phase) заставляється одна з фундаментальних основ успіху проекту – **створення й зімкнення проектної групи** на основі вироблення єдиного бачення. Проектна група повинна **чітко уявити собі**, що вона хоче зробити для замовника й сформулювати свою мету таким чином, щоб максимально мотивувати як замовника, так і саму проектну команду. Вироблення высокоуровневого погляду на мети й умови проекту можуть розглядатися як рання форма планування; вона підготовляє ґрунт для процесів створення детальних планів, які будуть здійснені безпосередньо під час фази планування.

Основними задачами фази вироблення концепції є створення ядра проектної групи (див. нижче) і підготовка **документа загального опису й рамок проекту**

(vision/scope document). Формування бачення проекту й специфицирование його рамок – не одне й теж, хоча для успіху проекту необхідно й те, і інше. *Бачення (vision)* – це нічим подання, що обмежує не, про те, яким повинне бути рішення[12] . *Рамки (scope)* же дають чіткі границі того, що із запропонованого цим баченням буде реалізовано в умовах існуючих проектних обмежень.

Керування ризиками являє собою ітеративний процес, здійснюваний протягом усього життєвого циклу проекту. Під час фази вироблення концепції проектна група готовить документ оцінки ризиків і представляє головні ризики проекту разом із загальним описом і рамками проекту. Для одержання подальшої інформації про керування ризиками, див. "Білу книгу" дисципліни керування ризиками MSF.

Также під час фази вироблення концепції виробляється виявлення й аналіз **бізнесів-вимог.** Більш детально ці вимоги розглядаються під час фази планування.

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

Шаблон MSF містить наступні розділи:

- Бізнесу-переваги,
  - Опис переваг
  - Формулювання бачення
  - Аналіз вигід
- Концепція рішення
  - Мети, задачі, припущення й 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

Акторы й варіанти використання

Результатом виявлення вимог, див. матеріали лекції 3 є реєстр вимог. Вимоги співвласників звичайно оформляються в простій письмовій формі, без якої-небудь особливої регламентації. Типовий приклад оформлення вимоги до програми електронної пошти – «Система повинна дозволяти набирати текст повідомлення з можливістю форматування тексту й вставки смайликов». Дані вимоги далеко не у всьому можуть задовольняти критеріям, сформульованим у лекції 2: вони можуть суперечити один одному, бути неясними, неточними й т.д. Проте, документ «Вимоги співвласників», незважаючи на невисокий рівень формалізації, грає дуже важливу роль: у ньому зібрані думки всіх зацікавлених сторін і головна мета збору початкових вимог полягала в тім, щоб одержати по можливості як можна більше повний набір вимог, не пропустивши чогось

важливого.

Для того, щоб підвищити рівень інформативності вимог, усунути взаємні протиріччя й домогтися виконання їх інших основних характеристик, здійснюється перехід від повністю неформалізованих текстів до частково регламентованого (наприклад, шаблонами MS Word) текстам, класифікація, присвоєння наборів атрибутів, побудова моделей, прототипирование.

Самим популярним і досить ефективним способом підвищення інформативності вимог є оформлення їх у вигляді варіантів використання (use case), запропонований И. Якобсоном (див., наприклад, [26]).

Перш, ніж приступитися властиво до специфицированию вимог у формі варіантів використання, RUP рекомендує виявити реєстр акторов[13] (actors) і варіантів використання.

Актор – це хтось або щось, що володіє активністю стосовно програмної системи. Якщо ви розробляєте простий текстовий редактор, те, швидше за все, вибір актора не складе особливої праці: це буде користувач, що набирає текст. Однак не завжди всі так просто. Крім користувача в якості актора може розглядатися інша програмна система, апаратний пристрій, у ряді випадків – активний компонент самої системи. Пошук акторов корпоративної інформаційної системи звичайно зводиться до аналізу ролей різних користувачів. Менеджер по продажах, старший менеджер і начальник відділу продажів – один актор, два або три? Це залежить від їхніх функціональних обов'язків, розмежування доступу, способів використання інформаційної системи. Пошук акторов може здійснюватися, наприклад методом мозкового штурму. Надалі при необхідності знайдені акторы можуть узагальнюватися, переглядатися й поєднуватися.

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 варіанта використання припускає виділення серед усього функціонала системи підмножини, корисного конкретному кінцевому користувачеві (точніше кажучи, типу кінцевого користувача). Інша сторона – варіант використання повинен не тільки бути корисний, а ще й дозволяти одержувати КП конкретні закінчені результати. Так, однієї з функцій текстового редактора, мабуть, є створення порожнього файлу. Але навряд чи КП буде використати 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.

Глоссарий

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

Крім того, глоссарий є відправною крапкою для побудови більше розгорнутих моделей проблемної області, які, на стадії реалізації інформаційної системи, лягають в

основу об'єктної моделі (для объектно-ориентированных додатків) і моделі даних (для генерації схеми бази даних).

Глоссарий оформляється , як текст, що складається з абзаців, кожний з яких визначає значення одного з термінів проблемної області. Термін звичайно виділяють напівжирним кеглем. Іноді проблемну область доцільно сегментировать на ряд

У різних російськомовних текстах авторові доводилося бачити наступні переклади терміна «actor»: актор, актор, актант, агент, суб'єкт, користувач. Тому що всі вони або неблагозвучні, або неточні, через брак кращого далі по тексту будемо користуватися перекладом, найбільш близьким до транскрипції «подобластей» (subject areas). Тоді кожної з них у глоссарии виділяється окремий параграф.

Специфікація варіанта використання

Існують різні шаблони опису варіантів використання. Так, у монографії [27] розглядаються наступні основні стилі опису:

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

Крім того, іноді доцільно використати:

- Псевдокод,
- Діаграму активності UML (див. лекції 5),
- Інші графічні моделі.

Вільний формат

Вільний формат припускає опис дій користувача й системи в оповідальному стилі, наприклад: «Менеджер запитує в Системи список замовлень за період . Система відображає на екрані знайдені замовлення даного Менеджера із вказівкою їхніх основних атрибутів». Вільний стиль допускає використання конструкцій «Якщо те». «Якщо Менеджер має повноваження Начальника Відділу, то Система надає можливість перегляду замовлень всіх менеджерів цього відділу».

Шаблон повного опису варіанта використання по А. Коберну

Назва <коротка фраза у вигляді дієслова в невизначеній формі зробленого виду, що відбиває мета>

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

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

Рівень <один із трьох: узагальнений, мети користувача, подфункции>. Автор задає визначену трёхуровневую класифікацію вимог, у цілому відповідної класифікації вимог на бізнеси-вимоги, вимоги користувачів і функціональні вимоги, див. матеріали лекції 1.

Основна діюча особа <ім'я ролі основного актора або його опис>. **Учасники й інтереси** <список інших акторов-участников прецеденту з вказівкою їхніх інтересів>.

Предусловие <те, що очікується, уже має місце>.

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

Гарантії успіху <що одержать акторы-участники у випадку успішного досягнення мети>.

Тригер <те, що «запускає» варіант використання, звичайно – подія в часі>.

Основний сценарій <тут перераховуються кроки основного сценарію, починаючи від тригера й 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 кроку.Номер розширення> <Умова>:<Дія або посилання на підлеглий варіант використання>.

Кожної із кроків основного сценарію може мати 1 або більше розгалужень. Кожне розгалуження оформляється у вигляді розширення. У блоці «Розширення» всі розширення описуються послідовно.

У випадку, якщо альтернативний сценарій не вдається описати одним рядком – застосовується наступний формат.

Починаючи з рядка, що випливає після опису розширення, іде опис його дій у форматі основного сценарію:

<Номер кроку.Номер розширення.Номер кроку розширення> <Дія> Опис розширення закінчується описом виходу з розширення. Основні варіанти виходу з розширення: повернення до чергового по номері кроку основного сценарію, закінчення прецеденту, перехід до іншого кроку основного сценарію.

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

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

Табличні подання варіанта використання

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

Таблиця в 2 стовпчики:

Актор	*Дія*
Користувач	Формує запит на пошук замовлень
Система	Відображає список замовлень
Користувач	Вибирає необхідне замовлення
Система	Показує докладну інформацію із замовлення

Таблиця в 3 стовпчики:

№ кроку	*Користувач*	*Система*

1	Робить запит на пошук замовлень	Відображає список замовлень
2	Вибирає необхідне замовлення	Показує докладну інформацію з замовленню

Шаблон варіанта використання RUP

Із шаблоном опису варіанта використання RUP і прикладами можна ознайомитися в інтерактивній версії RUP.

Нижче наведений короткий огляд його розділів.

1. **Найменування й короткий опис.** У цьому розділі вказується: найменування варіанта використання, акторы варіанта використання, коротке (в один абзац) опис варіанта використання.

2. **Потік подій**

2.1. **Основний потік подій**

Так само, як в «Основний сценарій».

2.2. **Альтернативні потоки подій**

Кожний з альтернативних сценаріїв описується в окремому параграфі, у тім же стилі , що й основний потік подій. Альтернативні сценарії описують поводження системи при будь-яких відхиленнях від основного сценарію, а також поводження у виняткових ситуаціях.

3. **Спеціальні вимоги**

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

4. **Предусловия**

Події, описувані предусловиями або постусловиями, повинні бути станами, які користувач може спостерігати [15**Помилка ! Джерело посилання не знайдений.**]. Предусловие описує стан, у якому система повинна перебувати до початку виконання прецеденту.

5. **Постусловия**

Постусловие RUP по суті описує те ж, що й мінімальна гарантія в Коберна. Л. Новиков [15**Помилка! Джерело посилання не знайдений.**] акцентує увагу на тім, що коректно сформульоване постусловие повинне бути щирим при будь-якому можливому сценарії прецеденту, а не описаному в основному потоці.

6. **Крапки розширення**

Даний параграф визначає положення крапок, що розширюють потік подій.

Вибір форми опису варіанта використання

При виборі форми й ступеня подробиці варіанта використання варто враховувати такі фактори, як:

- Розміри проекту,
- Важливість проекту й варіанта використання,
- Традиції, що зложилися в колективі «Замовник-Розроблювач».

Для невеликого проекту навряд чи буде доцільним застосовувати опису варіантів використання в розгорнутому форматі, досить використати коротку форму вільного стилю. Для проекту, у якому задіяно більше десяти учасників, у якому виникають

проблеми розбивки на мікро-колективи, координації учасників, варто вибрати більше формалізований і більше докладний варіант.

Ступінь подробиці залежить також від критичності проекту в цілому й критичності варіанта використання в даному проекті. А. Коберн ділить всі програмні проекти по ступені критичності на 4 категорії: виходячи із ціни помилок: «проекти, помилки в які можуть привести к..…»:

- небезпеки для життя людей,
- непоправним фінансовим втратам,
- фінансовим втратам в обмеженому об'ємі,
- зниженню комфортності кінцевого користувача.

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

Крім того, у тому самому проекті можуть зустрічатися більше важливі – з позицій частоти й масовості використання, складності для розуміння, технічних ризиків і т.д. і менш важливі прецеденти. У цьому випадку для різних прецедентів того самого проекту цілком припустимий опис із різним ступенем подробиці.

Нарешті, специфікація варіантів у стилі Коберна, стилі RUP, у табличній формі, з використанням псевдокодів або графічних конструкцій (Моделювання вимог) багато в чому визначається суб'єктивним вибором автора прецедентів і сформованим досвідом роботи із замовником проекту.

Специфікація нефункціональних вимог

Опис нефункціональних вимог звичайно здійснюється у формі, близької до вільного формату опису варіанта використання. RUP рекомендує концентрувати нефункціональні вимоги в документі, що описує варіант використання у всіх випадках, коли це можливо. У випадку, якщо нефункціональні вимоги носять загальний характер і не можуть бути прив'язані до конкретного прецеденту – вони виносяться в документ «Додаткова специфікація».

Атрибути вимог

Опису вимог повинні бути операбельны. Для цього всі вимоги повинні враховуватися в тієї або іншій обліковій системі, будь те електронна таблиця MS Excel, спеціалізована база даних, або інтегроване середовище керування змінами. При реєстрації вимоги воно проходить класифікацію відповідно до певної системи ознак. Основні ознаки (атрибути) вимог були розглянуті в лекції 1. Крім того, для оперативного керування вимогами буває корисно призначити їм такі властивості , як проект, відповідальна особа, статус, ризик, ступінь закінченості й т.п. В RUP для керування атрибутами вимог передбачений артефакт «Атрибути вимог».

Артефакт «Атрибути вимог», пропонований RUP, являє собою репозиторий текстів вимог, їхніх атрибутів і трассируемости.

Атрибути вимог представлені матрицею атрибутів вимог, де для кожного типу вимог перераховуються вимоги по однієї осі й атрибути вимог цього типу по іншійій. Для кожної вимоги вказуються значення його відповідних атрибутів. Приклади атрибутів:

статус у часі, пріоритет, важливість, ризик, № ітерації (етапу) у плані.

Трассируемость описується у вигляді дерева , що показує в графічному виді вхідні й (або) вихідні зв'язки трассируемости (див. матеріали лекції 7).

Питання та завдання до змістового модулю № 1

1. Місце аналізу вимог у життєвому циклі ПЗ.
2. У чому складність процесу збору вимог до ПЗ?
3. Навіщо потрібен аналіз зібраних вимог до ПЗ?
4. Коли починається процес збору вимог?
5. Перелічіть основні типи вимог та назвіть їх характеристики.
6. Залежність вимог до ПЗ від цільового сегмента ринку.
7. Для чого і як формуються профілі користувачів?
8. Техніки вилучення вимог.
9. Що повинні включати сценарії використання програмного забезпечення?

Змістовий модуль 2. *Документування та перевірка вимог*

Тема 5. Розширений аналіз вимог. Моделювання та прототипування

Які моделі використовувати. Моделі UML, що пояснюють функціональність системи. Діаграма варіантів використання. Діаграма дій, діаграма станів. Діаграми UML, що пояснюють внутрішній устрій системи. Альтернативні мови моделювання. Діаграма потоків даних. Інші види моделей.

Мети прототипування. Класифікація прототипів Горизонтальний і вертикальний прототипи. Одноразовий і еволюційні прототипи. Паперовий прототип.

Розкадровка. Ілюстровані сценарії прецедентів. Орієнтири. Середні значення атрибутів об'єктів та обсяги. Середня інтенсивність використання.

Які моделі використати

Вербальні описи варіантів використання системи, розглянуті в попередній лекції, на сьогодні є стандартом де-факто для формулювання угоди між Замовником і Виконавцем. Якщо обидві сторони готові виділити достатню кількість часу на уважний і

всебічний аналіз вимог до системи й на початковій фазі її створення сформулювали 80% всіх можливих сценаріїв використання системи на зрозумілому сторонами мові – виходить, ключові ризики створення системи – ризик різного розуміння проблеми й ризик розмиття границь багато в чому переборені.

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 й розуміння вимог для Замовника, існує ряд прийомів. По-перше, вимоги можна формулювати на різних рівнях абстракції. Так, рівень опису вимог, підтримуваний у документі «Бачення», є досить збалансованим . Те ж можна сказати й про короткі (в один абзац) опису ключової функціональності системи. Діючи таким чином, ми, мабуть, вирішимо проблему залучення Замовника в аналіз задач, однак зазначені вище ризики будуть знижені недостатньо: концептуальні описи функціональності залишають багато волі для тлумачення в ту або іншу сторону.

Гарною підмогою в рішенні задачі є застосування візуальних засобів опису вимог. Як відомо, у більшості людей правополушарное (образне) мислення дозволяє сприймати інформацію в рази й порядки більше прискореному темпі, чим левополушарное (вербальне).

На сьогодні в арсеналі аналітика існують десятки методик, мов, візуальних подань, що дозволяють моделювати вимоги до системи. При створенні інформаційних систем стандартом де-факто є універсальна мова моделювання, UML [14,15]. Деякі інші нотації були згадані в лекції 3.

Як ми вже відзначали в лекції, присвяченої аналізу контексту АТ, процес аналізу вимог тісно зв'язаний, з одного боку, з аналізом проблемної області, з іншого боку - з архітектурним аналізом і проектуванням. Часто на практиці буває важко вичленувати границі компетенций цих потоків робіт. Так, модель аналіз потоків даних, широко використовувана в аналізі проблемної області, згадується багатьма авторами, як модель, корисна в аналізі вимог. Ряд дослідників вважає за доцільне ілюструвати опису вимог діаграмами класів, хоча, строго говорячи, виділення класів ставиться не до аналізу вимог, а до архітектурного аналізу.

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

По-перше, аналіз вимог покликаний вивчати взаємодії автоматизованої інформаційної системи і її середовища, тобто користувачів, мережних і системних компонентів, що перебувають поза системою. Отже, модель^-моделі-бізнеси-моделі, що описують взаємодії між компонентами організаційної системи, строго говорячи, можна розглядати лише як «сировину» для добування 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, допустимо, діаграма взаємодії об'єктів, що реалізують той або інший варіант використання, можна розглядати скоріше, як ілюстрацію внутрішнього пристрою системи, корисну для програміста, чим модель для замовника.

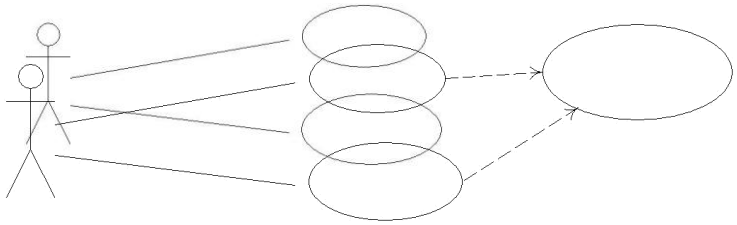
Однак, найбільш важливим є третє міркування, у чомусь «опозиційне» стосовно

перших двох. Для моделювання аналізу вимог варто застосовувати моделі, що найбільше адекватно проясняють функціональність системи й особливості її використання. Однак, аналітик вільний вибирати ті мови й методики, які дозволять домогтися цільової функції: зниження ризиків нерозуміння між Виконавцем і Замовником і размытия границь. Тому, ілюструючи варіанти використання, починайте з «канонічних» способів, які будуть розглянуті трохи нижче, але, якщо порахуєте доцільним відхилитися від них – експериментуйте сміло.

Моделі UML, що пояснюють функціональність системи

Діаграма варіантів використання

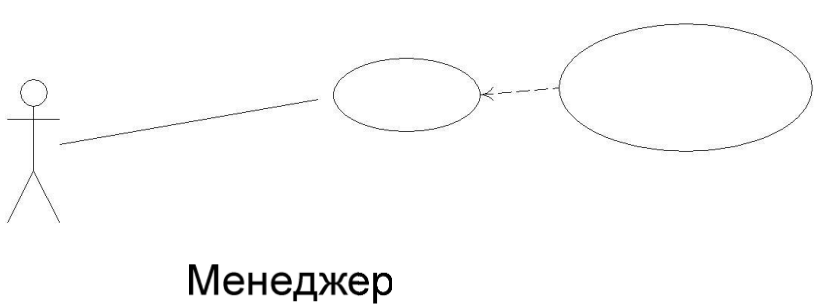
Діаграма варіантів використання UML, Use Case Diagram – одне з найпростіших подань системи. Її базові «будівельні елементи» – акторы й варіанти використання. Діаграма задумана так, щоб дати найбільш загальне подання про функціональність системи (її компоненти), не вдаючись у деталі взаємозв'язків функцій. Тому основний вид відносини, використовуваний у діаграмі – асоціація між актором і варіантом використання.



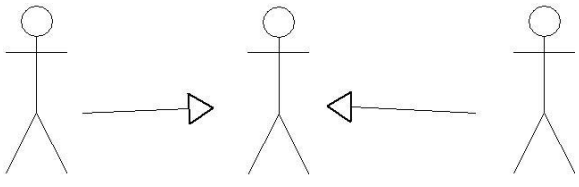
Інші види відносин – відношення включення (include), розширення (extend) і узагальнення/генералізації.

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

Розширення в точності відповідає крапці розширення, використовуваної при описі варіанта використання, див. матеріали лекції 4.



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

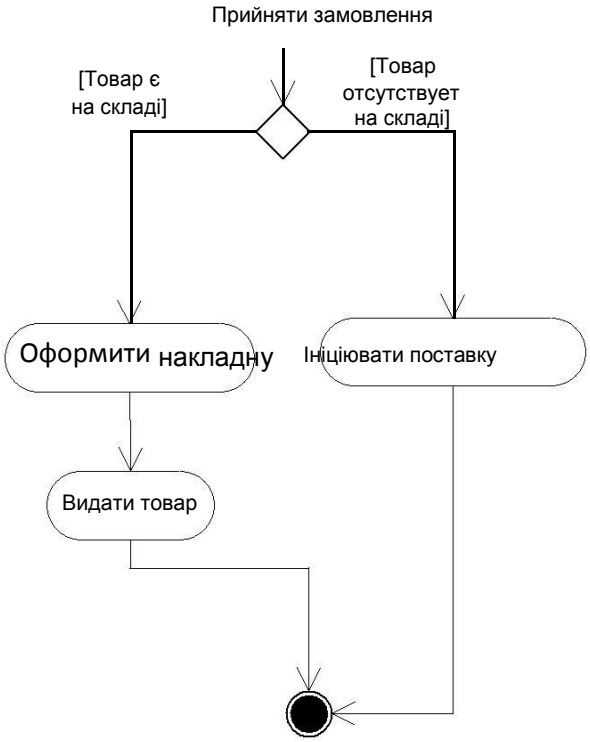


Діаграма дій

Якщо діаграма варіантів використання дає «вид зверху» на функціональність системи, діаграма дій UML, навпроти, дозволяє докладно ілюструвати окремий варіант використання і його сценарії.

Основні компоненти опису системи:

- Функції (дії),
- Символи «старт» і «стіп»,
- Потоки керування,
- Разветвители,
- Лінійки синхронізації.



Діаграма дій дозволяє проілюструвати варіант використання з необхідним ступенем подробиці. Лінійний варіант використання приводить до діаграми дій з лінійним потоком керування між діями. Дії варіанта використання з альтернативними сценаріями реалізується через разветвители. Лінійки синхронізації дозволяють описувати такі складні конструкції, як синхронізацію початку (закінчення) паралельних у часі процесів.

Крім стандартного формату опису, UML пропонує варіант із «плавальними доріжками». Цей формат зручний для опису случаючи, коли у варіанті використання беруть участь трохи акторов.

Діаграма станів

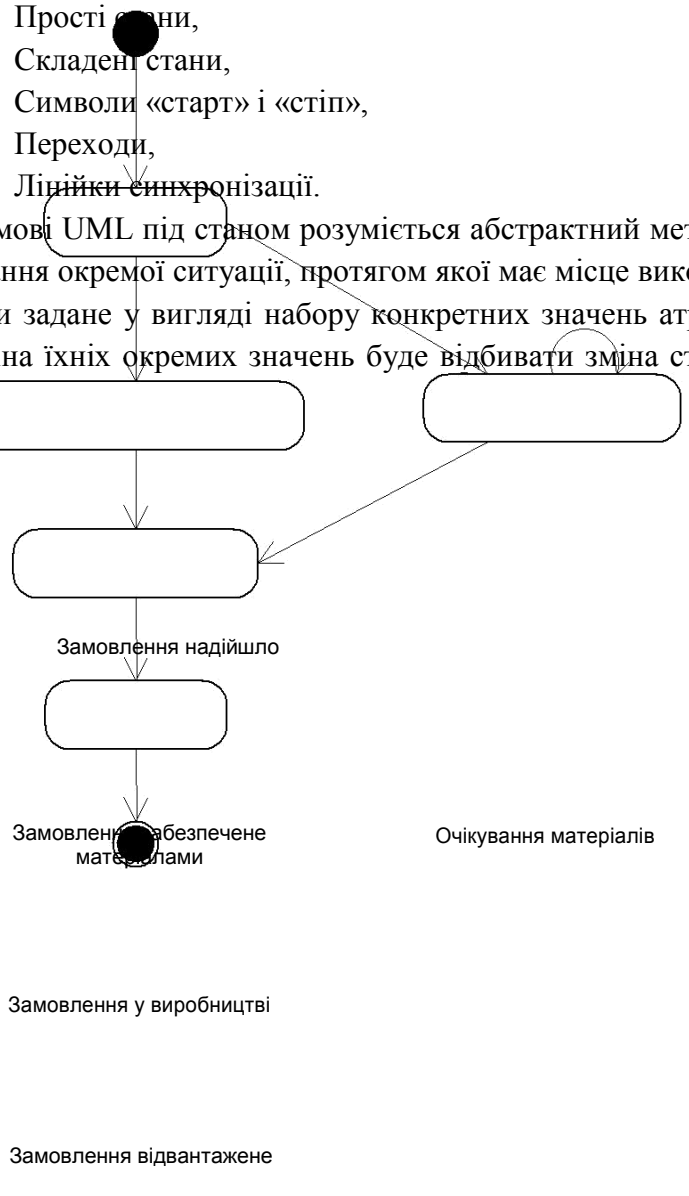
Діаграма станів в аналізі вимог використається, коли потрібно досліджувати поводження системи, як кінцевого автомата. Це подання прийшло в UML з теорії систем.

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

Основные компоненти опису системи:

- Прості стани,
- Складені стани,
- Символи «старт» і «стіп»,
- Переходи,
- Лінійки синхронізації.

У мові UML під станом розуміється абстрактний метакласс, використовуваний для моделювання окремої ситуації, протягом якої має місце виконання деякої умови [15]. Стан може бути задане у вигляді набору конкретних значень атрибутів класу або об'єкта, при цьому зміна їхніх окремих значень буде відбивати зміна стану моделируемого класу або об'єкта.



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

Подія [сторожова умова] / вираження дії Іноді буває корисним

об'єднати частина станів в один багатство-майно-становище-стан-позначка-стан.

Графічно це виглядає, як символ стану (прямокутник з округленими кутами), що містить усередині себе кілька символів станів. При цьому можливі переходи між підлеглими станами, переходи між підлеглим і зовнішнім станами й переходи між складеним і зовнішнім станом.

Діаграми UML, що пояснюють внутрішній пристрій системи

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

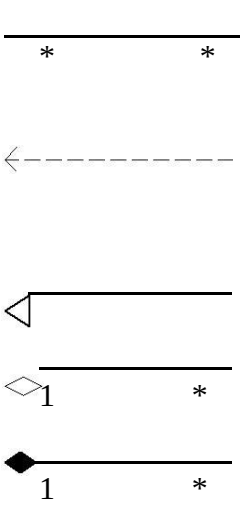
Діаграма класів. Для створення діаграми класів необхідно:

1. Здійснити пошук класів (ключових компонентів проблемної області).
2. Для кожного знайденого класу визначити його ім'я, основні атрибути, операції й (або) відповідальності.
3. Досліджувати відносини знайдених класів.

Класи на діаграмі представляються у вигляді прямокутників, відносини – у вигляді ліній, що зв'язують прямокутники. Лінії різного типу графічно відрізняються різним штрихуванням і стрілками.

Прийнято виділяти 3 рівні 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 їхнього найменування й відповідальності. Атрибути й операції можна не вказувати, або ввести тільки самі основні, відклавши їхнє дослідження на більше пізні стадії деталізації.

Відносини, що підлягають аналізу на концептуальному рівні – це:



асоціація (іменований зв'язок),
залежність (зміни в одному класі приводять до змін в іншому),
узагальнення / генералізація (родовидовое відношення),
агрегація (відношення «частина-ціле»),
композиція (відношення «частина-ціле», однозначно регламентуюча кількість і склад частин цілого).

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

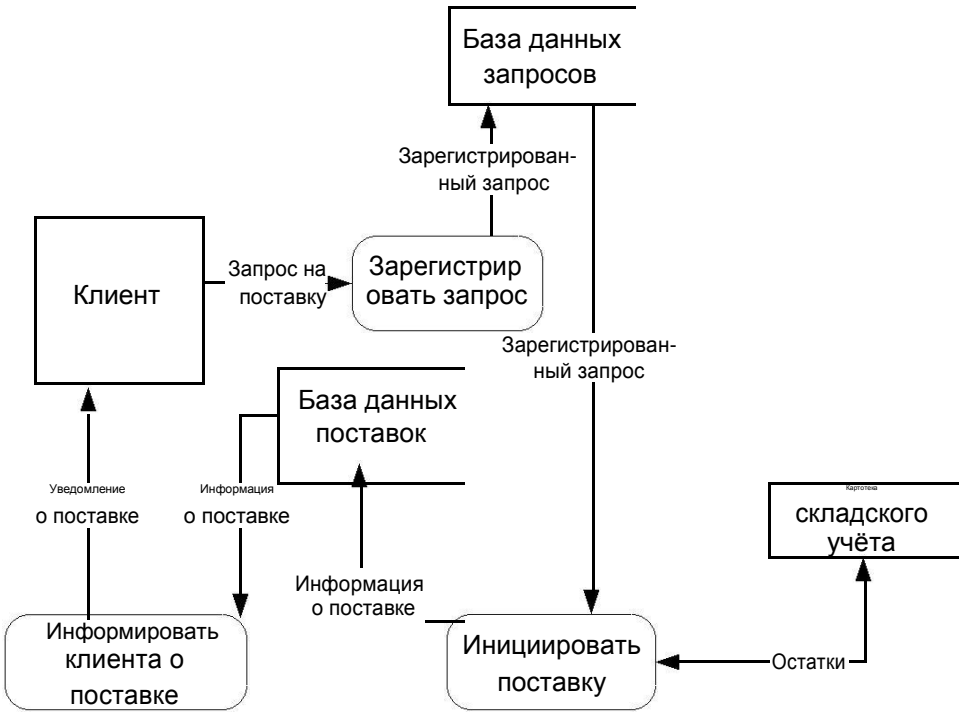
На думку автора, якщо діаграма класів у ряді випадків і може розглядатися, як артефакт, що пояснює структуру проблемної області, то діаграми взаємодії навряд чи варто розглядати в якості виразного язикового засобу, що ілюструє вимоги до системи в діалозі «Замовник-Виконавець». Проте, у відповідність із Принципом 3 волі вибору язикових засобів, аналітик, що вирішив використати 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 й проектування інформаційних систем, що існували в «доюмээльную» епоху. Незважаючи на місце, що має, у сучасних умовах зсув акцентів від структурного до объектно-ориентированному підходу до аналізу й проектування систем, «стародавні » структурні нотації по- колишньому широко й ефективно використаються як у бізнесі-аналізі, так і в аналізі інформаційних систем.

Історично зложилося так, що для опису діаграм DFD використаються 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 використається поняття зовнішньої сутності. Усередині системи існують процеси перетворення інформації, що породжують нові потоки даних. Потоки даних можуть надходити на вхід до інших процесів, міститися (і витягати) у накопичувачі даних, передаватися до зовнішніх сутностей.

Модель DFD, як і більшість інших структурних моделей – иереархическая модель. Кожний процес може бути піддадуть декомпозиції, тобто розбивці 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 – структурного компонента розроблювальної системи.

Нотація DFD – зручний засіб для формування контекстної діаграми, тобто діаграми, що показує розроблювальну АИС у комунікації із зовнішнім середовищем. Це - діаграма верхнього рівня в ієрархії діаграм DFD. Її призначення – обмежити рамки системи, визначити, де закінчується розроблювальна система й починається середовище. Інші нотації, часто використовувані при формуванні контекстної діаграми – діаграма SADT15, діаграма Діаграма варіантів використання.

Інші види моделей

Серед різноманіття моделей, що використаються в аналізі систем, хочеться особливо відзначити ще дві нотації, що дозволяють описати складні многоальтернативные взаємодії компонентів інформаційної системи – нотацію IDEF3 і eePC-діаграму ARIS.

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

Мети прототипування

Все те, що говорилося в попередній лекції про особливості сприйняття людиною

вербальної й невербальної інформації з відношення до моделей, у ще більшому ступені варто віднести й до візуальних прототипів.

Розглянемо основні цілі, що вимагають застосування прототипів [8-21]:

- прояснити неясні вимоги до системи;
- вибрати одне з різних концептуальних рішень;
- проаналізувати осуществимость.

1. **Неясні вимоги.** Часто Замовникові 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 – у цьому випадку корисний будь-який результат прототипирования: якщо Виконавець зрозумів вимоги добре – користь очевидна; якщо не дуже – користь полягає в тім, що Замовник може вказати, у чому полягає нерозуміння, тим самим вирішивши основну задачу – зробити неясне ясним.

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

Розглянемо приклад. Постачальникові надходить вхідний потік вимог на комплектацію замовлень матеріалами. Різні позиції того самого вимоги можуть бути закуплені в різних постачальників. Постачальник повинен зіставити постачальника кожної позиції кожного з вимог. Є як мінімум два сценарії рішення цієї задачі.

А) Сценарій послідовної обробки вимог.

А1. Система відображає реєстр вимог, наявних у вхідній черзі. А2. Користувач вибирає чергову вимогу.

А3. Система відображає перелік матеріалів вимоги й довідник постачальників.

А4. Користувач зіставляє кожної з позицій вимоги постачальника з довідника постачальників.

А5. Система надає вимозі статус «оброблене», висилає по електронній пошті авторові вимоги повідомлення.

А6. Продовжувати із кроку А1, поки черга не спорожніє. Б) Сценарій угруповання за матеріалами.

Б1. Система відображає позиції всіх вимог і довідник постачальників.

Б2. Користувач групує позиції по типі (так, щоб однотипні позиції, що поставляють тим самим постачальником, перебували поруч).

Б3. Користувач вибирає групу позицій і зіставляє їй постачальника.

Б4. Система перевіряє – чи не з'явилися повністю оброблені вимоги. При позитивному результаті перевірки привласнює цим вимогам статус «оброблене» и высылает по електронній пошті авторові вимоги повідомлення.

Б5. Продовжувати із кроку Б1, поки черга не спорожніє.

Перший сценарій зручний тим, що дозволяє постачальникові працювати в розрізі авторів вимог, почати із самих критичних за часом вимог, контролювати процес їхньої обробки. Другий сценарій зручний тим, що дозволяє одночасно спостерігати на екрані рядка різних вимог, поєднуючи їх у єдине замовлення.

Після реалізації прототипів UI по першому й другому сценаріях Замовник, оцінивши їхні достоїнства й недоліки, зміг у діалозі з Розроблювачем сформулювати третій, комбінований сценарій, що сполучить достоїнства перших двох.

3. **Аналіз осуществимости.** Часто буває так, що комбінація функціональних, нефункціональних вимог і обмежень така, що виникає ризик неможливості їхньої 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), що реалізують відповідну частину системи, що імітують потоки даних, що надходять на її вхід і їхню обробку.

Класифікація прототипів

Слідом за К. Вігерсом розглянемо наступні класифікації прототипів:

- горизонтальні й вертикальні;
- одноразові й еволюційні;
- паперового й електронні, розкадрування[16].

Горизонтальний прототип

Горизонтальний або поведінковий прототип (horizontal prototype, behavioral prototype) моделює інтерфейс користувача додатка, не зачіпаючи логікові обробки й базу даних.

Якщо в розробленому інтерфейсі використаються фрагменти бази даних – вони імітуються в програмному коді. При цьому тексти, відображувані на екрані, повинні відбивати реальну специфіку проблемної області, інакше користувачеві буде важко зосередитися. Якщо при натисканні на елемент керування повинні вироблятися якісь розрахунки або запити в зовнішні системи – результати запитів також імітуються. Бажано реалізувати ту частину коду, що 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 варто використати для досягнення мети прояснення неясних, або многоальтернативных вимог.

Вертикальний прототип

Вертикальний або структурний прототип (vertical prototype, structural prototype) не обмежується інтерфейсом користувача. Він реалізує вертикальний «зріз» системи, торкаючись всіх рівнів її реалізації . При створенні такого роду прототипів рекомендується використати ті мови й середовища реалізації, що й при виготовленні цільової системи (що, загалом кажучи, зовсім не обов'язково для горизонтальних прототипів).

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

Одноразовий прототип

Одноразовий або дослідницький прототип (throwaway prototype, exploratory prototype) створюється, коли потрібно швидко промакетировать ті або інші аспекти й компоненти системи.

Цілям створення дослідницьких прототипів служить технологія RAD (rapid application development) – швидка розробка додатків[17], див. матеріали лекції 3.

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

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

К. Вігерс приводить наступну схему переходу від одноразового прототипу до детально пророблено UI:

На малюнку присутній нове, не розкрите раніше не поняття: «карта діалогу», говорять також «схема діалогу». Перш, ніж створювати горизонтальний прототип, необхідно визначитися – які основні екрани будуть присутні, які вікна будуть відкриватися, які правила переходу між ними будуть підтримуватися. Інформація такого роду добре лягає на модель діаграми станів, див. матеріали лекції 5, де різним екранам (вікнам) зіставляються стани, а активним елементам керування, що викликає закриття одних інтерфейсних елементів і відкриття інших – переходи.

Еволюційний прототип

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

Код еволюційного прототипу повинен послідовно, у плині однієї або більше ітерацій, перерости в код цільового додатка. Тому даний вид прототипів вимагає всього того, від чого варто відмовитися при створенні одноразових прототипів: скрупульозної розробки, застосування технологічних методів і прийомів, тестування результатів і т.п.

У таблиці наведене співвідношення між розглянутими вище 4 видами прототипів.

	Одноразовые	*Эволюционные*
Горизонтальные	▪ Прояснение и уточнение примеров использования и функциональных требований ▪ Выявление пропущенных требований ▪ Исследование возможных вариантов интерфейса пользователя	▪ Реализация базовых вариантов использования ▪ Реализация дополнительных вариантов использования по приоритетам ▪ Реализация и доработка web-сайтов ▪ Адаптация системы к быстро меняющимся требованиям бизнеса
Вертикальные	▪ Демонстрация технической осуществимости	▪ Реализация и наращивание ключевой клиент-серверной функциональности и уровней коммуникации ▪ Реализация и оптимизация основных алгоритмов ▪ Тестирование и настройка

Паперовий прототип

Паперовий прототип (paper prototype) – відмінна альтернатива розглянутим вище різновидам електронних прототипів у випадку, коли Розроблювач обмежений у ресурсах. Начерки інтерфейсів на папері, звичайно, не замінять інтерфейс, створений у середовищі розробки. Однак , при всіх недоліках, у таких прототипів є два істотних достоїнства.

1. Замовник не стане акцентувати увагу на колірному рішенні, формі кнопок и т. п., відволікаючись від аналізу функціональності.

2. Замовник ніколи не скаже, дивлячись на паперовий інтерфейс: «Так ви, я бачу, уже створили систему на 85%! Давайте закінчимо її в плині тижня».

Розкадрування

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 волі вибору, надаваної йому поведінковим прототипом. Але ідею покрокової зміни екранів у процесі реалізації сценарію варіанта використання цілком можна реалізувати. Даний вид рішення визначається, як *пасивне розкадрування*. *Активне розкадрування* є подальшим розвитком поняття пасивного розкадрування, із застосуванням засобів анімації й т.п. Третій вид розкадрування, – *інтерактивна* являє собою електронний одноразовий горизонтальний прототип.

Ілюстровані сценарії прецедентів

Ілюстровані сценарії прецедентів, ИСП [15], поряд із прототипами дозволяють досягти кращого розуміння між Замовником і Розроблювачем. Але якщо прототипи адресовані скоріше Замовникові, ніж Розроблювачеві, то з ИСП ситуація обстоит навпаки: вони містять додаткові відомості, що допомагають Розроблювачеві краще зрозуміти специфіку проблемної області й, тим самим, краще відбити її в інтерфейсі користувача.

Основна ідея ИСП – «розбавити» текст опису сценарію варіанта використання *аспектами застосовності.*

Аспект застосовності – інформація , що дозволяє розширити опис прецеденту описами, що конкретизують тієї або іншої його особливості й, в остаточному підсумку, підвищити ступінь комфортності користувача.

Розрізняють [15] 3 різновиду аспектів застосовності: орієнтири, середні значення атрибутів і об'єми об'єктів, середня інтенсивність використання.

Орієнтири

Орієнтири – це опис опциональных функціональних можливостей системи. Відсутність таких можливостей не приводить до фатальної невдачі. Присутність – поліпшує застосовність, постачаючи корисною інформацією. Орієнтири варто розцінювати не як вимоги, а як побажання або рекомендації.

Приклад. Опис потоку подій ИСП для прецеденту «Оформити замовлення», розширеного орієнтирами (текст у квадратних дужках).

У процесі виконання прецеденту менеджер по прийому замовлень вибирає замовника із клієнтської бази, визначає товарні позиції з довідника й указує їхню кількість. Система відображає на моніторі найменування позицій, ціну, суму й

кількість на складі. Менеджер призначає знижку й визначає порядок оплати. Система розраховує підсумкову суму . [Менеджер повинен мати можливість бачити поточне сальдо розрахунків із клієнтом і дані по останнім десятьох угодах зі статистикою по дисципліні дотримання договірних зобов'язань].

Середні значення атрибутів і об'єми об'єктів

Дана інформація дозволяє оптимальнее 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 керування checkbox, при виборі, обмеженому 2-3 десятками позицій – список, що випадає, при різноманітті, вимірюваному тисячами варіантів, будуть потрібні додаткові засоби фільтрації й пошуку.

Пример. Опис потоку подій ИСП для прецеденту «Оформити замовлення», розширеного об'ємами й середніми значеннями об'єктів (текст у фігурних дужках).

У процесі виконання прецеденту менеджер по прийому замовлень вибирає замовника із клієнтської бази {до 10000 клієнтів}, визначає товарні позиції з довідника {товари розбиті на 10 категорій, кількість позицій у категорії не перевищує 500} і вказує їхня кількість {до 100 позицій, середня закупівля – 8 позицій}. Система відображає на моніторі найменування позицій, ціну, суму й кількість на складі. Менеджер призначає знижку й 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, у яких все повинне бути ідеально (швидкодію, зручність користування, мінімум дій на виконання операцій). Наприклад – інтерфейс касира в супермаркеті. Інша крайність – сценарії, виконувані час від часу, не щодня й не потребуючої особливої оперативності (наприклад, розрахунок заробітної плати за місяць). Ці дані дозволяють, структурировать подачу інформації, забрати з «головних» інтерфейсів рідко використовувані опції й т.п.

Приклад. Фрагмент опису потоку подій ИСП для прецеденту «Оформити замовлення для нового клієнта », розширеного об'ємами й середніми значеннями об'єктів (текст у круглих дужках).

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

Тема 6. Документування та перевірка вимог

Документування вимог у відповідність з стандартами. Опис вимог до системи у відповідність з стандартами. Документування вимог в RUP. Документування вимог на основі IEEE Standard 830-1998. Документування вимог MSF.

Верифікація і валідація. Двозначність вимог. "Золочення" продукту. Мінімальна специфікацій. Пропуск типів користувачів. Методи і засоби перевірки вимог. Неофіційні перегляди вимог. Інспекції. Розробка тестів. Визначення критеріїв прийнятності.

Щоб вимоги, виявлені й описані (див. матеріали лекції 3 і лекції 4) прийняли силу угоди між Замовником і Розроблювачем, їх необхідно оформити у вигляді документа. У практиці для цього звичайно використається документ «Технічне завдання», ТЗ, у західної – «Software Requirements Specification», SRS (специфікація програмних вимог). По суті це – той самий документ, тому далі по тексту будемо вживати термін «ТЗ», розглядаючи різні шаблони його побудови – як на основі російських ДЕРЖСТАНДАРТ, так і західних методологій і стандартів.

Документування вимог у відповідність із ДЕРЖСТАНДАРТ

Документування вимог регламентоване ДЕРЖСТАНДАРТ 19.201-78 «Технічне завдання, вимоги до змісту й оформлення» і ДЕРЖСТАНДАРТ 34.602-89 «Технічне завдання на створення автоматизованої системи» (ТЗ на АС) [27-28].

Другий документ, по суті, є більше проробленою версією першого, адаптованої до створення автоматизованих інформаційних систем, тому далі розглянута структура ТЗ у відповідність із ДЕРЖСТАНДАРТ 34.602-89.

Незважаючи на те, що для сфери IT 17 років – це ціла епоха, даний документ практично не застарів: його авторам удалося розробити збалансовані рекомендації, абстрагуючись від конкретних технічних і технологічних рішень. Крім того, він як і раніше відіграє роль державного стандарту і при висновку контрактів з державними підприємствами Розроблювача можуть зобов'язати оформити ТЗ відповідно до духу й буквою цього документа.

Структура ТЗ у відповідність із ДЕРЖСТАНДАРТ 34.602-89

У задачі лекції не входить перерахування всіх правил і рекомендацій даного ДЕРЖСТАНДАРТ, що бажають можуть ознайомитися з ним безпосередньо. Нижче будуть перераховані розділи, передбачені ДЕРЖСТАНДАРТ і розглянуті основні моменти, на які варто звернути увагу.

Загальні відомості – у цьому розділі , крім юридичних реквізитів сторін та ін. ділової інформації ДЕРЖСТАНДАРТ рекомендує вказати джерела й порядок фінансування робіт.

Призначення й мети створення (розвитку) системи – тут необхідно вказати показники об'єкта автоматизації, які повинні бути досягнуті й критерії оцінки досягнення цих показників. Даним розділом на практиці часто зневажають і зовсім дарма – адже саме в цьому розділі заставляються високрівневі бізнесу-вимоги й формулюються критерії їхнього досягнення.

Характеристика об'єктів автоматизації – досить важливий розділ. Його основні «розрізи» - організаційна структура, структура керування, структура розташування підприємства і його філій. Гарний опис об'єкта автоматизації дозволяє заощадити час на визначення класів користувачів, для великих територіально-розподілених систем – закласти структуру й топологію мережних комунікацій.

Вимоги до системи – ключовий розділ дійсного документа, тому він буде розглянутий нижче, більш докладно.

Розділ **«Склад і зміст робіт зі створення системи**», говорячи 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, їхній основний зміст).

Порядок контролю й приймання системи – також один із ключових компонентів ТЗ. Він розподіляє ролі Замовника й Розроблювача в підготовці системи до випробувань і проведення випробувань. Тут доречно обмовити правила проведення випробувань, сформулювати основні тестові сценарії й критерії приймання.

Вимоги до складу й змісту робіт по підготовці об'єкта автоматизації до уведення системи в дію, знову ж, апелируя до сучасної термінології, обмовляють порядок проведення реінжинірингу підприємства, якому необхідно здійснити для того, щоб домогтися від впровадження АИС належного ефекту (підбор і навчання персоналу, зміни в організаційній структурі й т.п.).

Документ закінчується розділами «**вимоги до документування**» і «**джерела розробки**», що визначають, відповідно, перелік і форми документації, що підлягає розробці й перелік уже наявних документів, що містять передумови для розробки.

Як додатки ДЕРЖСТАНДАРТ рекомендує використати **розрахунок очікуваної ефективності системи** й **оцінку науково-технічного рівня системи**.

Опис вимог до системи у відповідність із ДЕРЖСТАНДАРТ 34.602-89

ДЕРЖСТАНДАРТ розділяє всі **вимоги до системи** на три класи:

- вимоги до системи в цілому;
- вимоги до функцій (задачам), виконуваним системою;
- вимоги до видів забезпечення.

Серед вимог *до системи в цілому* (системні вимоги) вказуються вимоги до:

- структурі системи (тут заставляються высокоуровневые архітектурні рішення, або структурні обмеження, уводиться ділення на підсистеми, комплекси й модулі, вирішуються питання комунікації компонент системи й системи із зовнішнім миром),
- режимам функціонування системи;
- персоналу (вказується чисельність, необхідна кваліфікація й режим роботи);
- надійності;
- безпеки;
- ергономіці й технічній естетиці;
- транспортабельності для рухливих АС;

- експлуатації, технічному обслуговуванню, ремонту й зберіганню компонентів системи;
- захисту інформації від несанкціонованого доступу;
- схоронності інформації при аваріях;
- захисту від впливу зовнішніх впливів;
- патентній чистоті;
- стандартизації й уніфікації,

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

Вимоги ДЕРЖСТАНДАРТ *до функцій (задачам)*, у перекладі на сучасну мову, підрозділяються на:

- перелік функціональних вимог у прив'язці до підсистем і черг автоматизації;
- часовий регламент реалізації функціональних вимог;
- вимоги до якості реалізації кожного з функціональних вимог (у тому числі - формі подання вихідної інформації, характеристики необхідної точності й часу виконання, вимоги одночасності виконання групи функцій, вірогідності видачі результатів);
- перелік і критерії відмов для кожної функціональної вимоги, по якому були задані вимоги по надійності.

Вимоги до *видів забезпечення*. Серед видів забезпечення ДЕРЖСТАНДАРТ указує математичне, інформаційне, лінгвістичне, програмне, технічне, метрологічне, організаційне, методичне.

Документування вимог в RUP

Шаблон SRS, запропонований в RUP[18], по суті являє собою контейнер, у який необхідно «упакувати» артефакти, отримані в процесі специфицирования вимог. Крім того, SRS частково перегукується з документом «Бачення» (див. матеріали «лекції 4»). Шаблон зручний своєю компактністю й лаконізмом.

1. Введення.

   1.1. *Ціль*. Документ повинен вичерпним образом описувати зовнішнє поводження системи, а також нефункціональні вимоги й обмеження.

   1.2. *Коротке зведення можливостей*.

   1.3. *Визначення, акроніми й скорочення*.

   1.4. *Посилання*.

   1.5. *Короткий зміст.*.

2. Огляд системи

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

   2.2. *Припущення й залежності*. Дана секція описує ключові технічні можливості, компоненти, підсистеми, зв'язані проекти, які можуть впливати на життєздатність розроблювальної системи.

Припущенням (assumption) називається положення, що вважається щирим при відсутності доказу або визначальної інформації. При визначенні залежностей (dependencies) проекту від зовнішніх факторів, необхідно проаналізувати, які нові

операційні системи, регламенти бізнесів-процесів, стандарти якості, інформаційні системи можуть з'явитися на підприємстві впровадження і як це може вплинути на функціонування виготовленої АИС.

3. Опис вимог

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

3.2. *Спеціальні вимоги*. Параграф містить опис функціональних вимог (не описаних, як варіанти використання), а також опис нефункціональних 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, що описують прототипи користувальницького інтерфейсу.

Документування вимог на основі IEEE Standard 830-1998

Розглянемо шаблон документа опису вимог, складений К. Вигерсом на основі стандарту. Даний стандарт містить розгорнутий опис вимог, що може бути оптимизировано для потреб конкретної організації.

1. Введення

1.1 Призначення документа.

1.2. Підтримувані угоди.

1.3. Передбачувана аудиторія й рекомендації з послідовності роботи з документом для кожного класу читачів.

1.4. Границі проекту. Тут утримується 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 більшої системи, укажіть, як це ПО співвідноситься з усією системою й визначите основні інтерфейси між ними.

2.2. ПРОсобенности 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 й характеристики користувачів. Документується процес пошуку акторов, у якому виявляються всі користувачі системи й здійснюється узагальнення (виділення класів) користувачів. Знайдені класи описуються (наприклад – рівень кваліфікації, доступний функціонал і т.д.).

2.4. Операційне середовище. Розглядається середовище функціонування АИС, включаючи апаратні засоби, операційні системи, для розподілених систем – географічне розташування користувачів і серверів, топологія мережі.

2.5. Обмеження проектування й реалізації. Розглянемо класифікацію обмежень:

- певні технології, засоби, мови програмування й бази даних, які

варто використати або уникати;
- обмеження, що накладають операційним середовищем продукту;
- обов'язкові угоди або стандарти розробки;
- зворотна сумісність із продуктами, випущеними раніше;
- обмеження, що накладають бізнесами-правилами;
- обмеження, пов'язані з устаткуванням, наприклад вимоги до швидкодії, обмеження пам'яті або процесора;
- угоди, пов'язані з користувальницьким інтерфейсом існуючого продукту, які необхідно дотримувати при його поліпшенні
- формати й протоколи обміну даними.

2.6 Документація для користувачів.

2.7 Припущення й 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. Приводиться короткий опис функції й вказується її пріоритет (ступінь важливості/черговості реалізації).

3. і.2 Послідовності «вплив - реакція». Необхідно перелічити послідовність впливів, надаваних на систему (дії користувачів, сигнали зовнішніх пристроїв і ін.), і відгуки системи, що визначають реакцію конкретної функції.

3. і.3 Функціональні вимоги. Необхідно дати деталізацію і-й функції, перелічити деталізовані функціональні вимоги, включаючи реакцію на очікувані помилки й невірні дії. Кожній детальній функціональній вимозі привласнюється унікальний ідентифікатор.

4. Вимоги до зовнішнього інтерфейсу

Нижче розглянуті конкретні рекомендації з написання розділів цього параграфа:

4.1 Інтерфейси користувача

Основні характеристики UI:

- посилання на стандарти графічного інтерфейсу користувачів або стильових рекомендацій для сімейства продукту, які необхідно дотримувати;
- стандарти шрифтів, значків, назв кнопок, зображень, колірних схем, послідовностей полів вкладок, часто використовуваних елементів керування й т.п.;
- конфігурація екрана 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 із проблемами із зором.

4.2 Інтерфейси встаткування

Опишіть характеристики кожного інтерфейсу між компонентами ПЗ й устаткування системи. В опис можуть входити типи підтримуваних пристроїв, взаємодії даних і елементів керувань між ПЗ й устаткуванням, а також протоколи взаємодії, які будуть використатися,

4.3 Інтерфейси ПО

Опишіть з'єднання продукту й інших компонентів ПО (ідентифіковані по імені й версії), у тому числі бази даних, операційні системи, засоби, бібліотеки й інтегровані комерційні компоненти. Укажіть призначення елементів повідомлень, даних і елементів керування, обмін якими відбувається між компонентами ПО. Опишіть служби, необхідні зовнішнім компонентам ПО, і природу взаємодії між компонентами. Визначите дані, до яких будуть мати доступ компонента ПО.

4.4 Інтерфейси передачі інформації

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

5. Інші нефункціональні вимоги

В цьому розділі описуються інші нефункціональні вимоги, що не ставляться до вимог до інтерфейсу, які представлені в розділі 4, і до обмежень, описуваним у розділі 2.5.

5.1 Вимоги до продуктивності

Укажіть спеціальні вимоги до продуктивності для різних системних операцій. Обґрунтуйте їхню необхідність для того, щоб допомогти розроблювачам прийняти правильні рішення, що стосуються дизайну. Наприклад, через тверді вимоги до часу відгуку бази даних розроблювачі можуть зеркализовать базу даних у декількох географічних метаположеніях або денормализовать зв'язані таблиць баз даних для одержання більше швидкої відповіді на запит.

Додаток А. Словник термінів (глоссарий).

Додаток Б. Моделі аналізу. У цей розділ містяться всі моделі, побудовані в процесі аналізу вимог (див. матеріали лекції 09-моделювання вимог).

Додаток В. Список питань

Це динамічний список ще не дозволених проблем, пов'язаних з вимогами. Це можуть бути елементи, позначені як «ТВ» (to be determined — необхідно визначити), відкладені рішення, необхідна інформація, недозволені конфлікти й т.п.

Документування вимог в MSF

На початку фази проектування проектна група працює із проектними вимогами. Вони підрозділяються на:

- бізнесу-вимоги,
- вимоги до експлуатації,
- системні вимоги,
- вимоги користувача.

Одним з основних результатів фази проектування є

- функціональна специфікація, що служить:
  - інструкцією команді розроблювачів про те, що вони повинні будуть створити;
  - основою для оцінювання об'єму роботи;
  - чітка угода із Замовником про те, що повинне бути зроблене;
  - основою для синхронізації роботи всієї проектної команди.

С шаблонами відповідних документів можна ознайомитися на сайті Microsoft.

Верифікація й валидация

Термін «верифікація» (verification) у російськомовній літературі звичайно переводять, як «перевірка». Термін «валидация» - як «перевірка правильності», «атестація», «твердження».

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

Важко очікувати від читача, що вперше зштовхнувся із цією термінологією і її визначеннями в цьому й іншому стандартах, ясності розуміння. Принаймні, в автора дійсного курсу лекцій при першому знайомстві з визначеннями тих же понять в ISO IEC 12207 виникло чітке відчуття, що автори стандарту явно чогось не договорюють. Поміркуєте самі: відповідно до даного стандарту, *верифікація* – це підтвердження експертизою й поданням об'єктивних доказів того, що конкретні вимоги повністю реалізовані. З іншого боку, *валидация* - це підтвердження експертизою й поданням об'єктивних доказів того, що конкретні вимоги до конкретних об'єктів повністю реалізовані. Правда, до честі авторів останнього стандарту, вони приводять примітку, що трохи наближає читача до розуміння: «валидация пов'язана з експертизою продукту з метою визначення його відповідності *потребам* користувача». У цьому й полягає суть відмінності: якщо верифікація пов'язана із з'ясуванням того, чи задовольняє розроблювальний об'єкт, або процес його створення *сформульованим вимогам*, те валидация відповідає на запитання – чи правильно розроблений цільовий об'єкт (продукт), чи задовольняє він *потребам* замовника. Інший аспект валидации полягає в тім, що вона звичайно погоджується з формальним прийманням (атестацією) системи.

Деякі стандарти, наприклад SWEBOK, IEEE 1059-93 “IEEE Guide for Software Verification and Validation Plans”, уводять для цих двох процесів узагальнююче поняття V&V (Validation and Verification). Згідно IEEE 1059-93, верифікація й валидация програмного забезпечення – упорядкований підхід в оцінці програмних продуктів, застосовуваний протягом усього життєвого циклу. Зусилля, прикладені в рамках робіт з верифікації й валидации, спрямовані на забезпечення якості як невід'ємної характеристики програмного забезпечення й задоволення користувальницьких вимог (кінець цитати).

З вищесказаного ясно, як здійснити верифікацію й валидацию АИС і (або) процесу її створення: у першому випадку необхідно переконатися, що АИС (компонента, процес) відповідає сформульованим вимогам, у другому – що АИС дійсно працює! Але якщо критерієм перевірки АИС служать вимоги, те що може послужити критерієм перевірки самих вимог? Відповідь полягає в тім, що вимоги повинні задовольняти властивостям, сформульованим у лекції 2, Крім того, варто переконатися в тім, що [8]:

- у специфікації вимог до ПО належним чином описані передбачувані можливості й характеристики системи, які задовольнять потреби різних зацікавлених у проекті осіб;
- вимоги до ПО точно відбивають системні вимоги, бізнесу-правила й ін.;
- вимоги забезпечують якісну основу для проектування й зборки ПО.

Деякі типові проблемні ситуації процесу формування й оцінки вимог

Двозначність вимог

У ряді проблем і недоліків вимог двозначність, є, мабуть, найбільш критичним фактором ризику проекту, що закладає у фазі формування вимог. Двозначність (невідповідність властивості ясності, визначеності) закладає під проект «бомбу вповільненої дії». На практиці вимога, сформульована двозначним образом , може привести до різних його інтерпретацій представниками Розроблювача й Замовника. Розроблювач, керуючись своєю інтерпретацією, визначить на її основі архітектурну основу, створить аналітичні й проектні моделі й в остаточному 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 й т.д.).

Тим самим, якщо не закласти засобу на перевірку вимог на предмет двозначності в момент їхнього формування – існує ризик неприйняття готової системи в момент приёмо-здавальних випробувань, тому що кожна зі сторін буде дотримуватися своєю версією інтерпретації вимог, що веде до збитків, судових процесів і т.п. і тому є маса прикладів.

«Золочення» продукту

Під «золоченням» [8] розуміють такі ситуації, коли розроблювачі 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 впустую (кінець цитати).

Ця ситуація виникає у випадку, коли, по-перше, у колективі Розроблювача присутні творчих особистостей (адже далеко не всяка команда стане проявляти ініціативу й робити поверх того, про що її просили), у других – існує розрив у проходженні інформації від Замовника до Розроблювача. Ініціативний розроблювач «золотить» продукт із найкращих спонукань, але, можливо, він погано знаком з бізнесом-процесом Замовника й закладені їм «фичи» попросту не будуть затребувані.

Інша сторона «золочення » полягає в тім, що група представників Замовника неоднорідна по своїй структурі й може виникнути ситуація, коли представник Замовника, що формулює «дорогі» вимоги, не має відповідні повноваження. Це – специфіка російських підприємств, де часто все буває влаштовано істотно неформально.

Мінімальна специфікація

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

Однак, для роботи «не за правилами», по-перше, повинні бути об'єктивні передумови, по-друге – варто усвідомлювати вигоди й ризики цього вибору.

Мінімальна специфікація доречна, якщо має місце наявність одночасно трьох обставин (об'єднання по «И»):

а) ціна контракту й розміри проекту такі, що розробка розгорнутого ТЗ економічно недоцільна; б) колектив Розроблювача має достатній ступінь професіоналізму й досвіду виконання проектів у суміжних областях, щоб уміти створювати по короткій специфікації продукт, що пройде приймання Замовником;

в) між Замовником і Розроблювачем існують конструктивні відносини й обидві сторони розуміють і приймають ризики міні-специфікації.

Інший варіант роботи з міні-специфікації: Замовник і Розроблювач розуміють, що створення розгорнутої специфікації відтягає закінчення випуску готового продукту, що головна мета проекту – продукт, а не документація й готові до щільного співробітництва в процесі його створення. Це – шлях так званих agile-методологій розробки ПО, докладніше див. в http://www.agile.org.

Основний ризик застосування міні-специфікації полягають у тім, що вони базуються на людському факторі. Гарні й конструктивні відносини між сторонами «на березі» повинні зберегтися на всьому протязі проекту, у противному випадку в сторін виникнуть істотні проблеми у формальному доказі того – що повинна робити програма, тому що міні-специфікація для цього недостатньо повна.

Пропуск типів користувачів

Корпоративні АИС створюються для того, щоб бути використаними різними групами користувачів. Може скластися ситуація, у якій у групу представників Замовника, що беруть участь у формуванні вимог, потраплять найбільш ініціативні персони підприємства, які, як видно, зможуть донести свій голос до представників Розроблювача. Ті ж категорії користувачів, у яких не найдеться активних представників, можуть виявитися «за бортом» автоматизації. Саме ця помилка формування вимог називається «пропуск класів користувачів». Щоб її уникнути, представник Розроблювача повинен об'єктивно оцінити організаційну структуру підприємства і його бізнесів-процесів і вдумливо підійти до вибору ключових персон, проведення інтерв'ю з якими допоможе сформувати цілісну картину вимог до створюваного АИС.

Методи й засоби перевірки вимог

Напрацьовано значну кількість методів і засобів перевірки вимог [8,21-31]. Вони відрізняються по ряду параметрів. Так, розрізняють:

- по широті аналізу – перегляд (вибіркова перевірка) і наскрізний контроль (тотальна перевірка);
- по ступені формалізації – неофіційні процедури, процедури, проведені за формальними правилами (інспекції, експертизи);
- по складу групи перевірки – з (без) участю автора, з (без) участю менеджера проекту, з (без) участю представників зовнішніх організацій;
- по використовуваних засобах – тексти вимог, тестові сценарії, критерії прийнятності, прототипи.

Поняття й методи прототипирования були розглянуті в лекції 5 Деякі інші, найбільш важливі з перерахованого вище, методи й засоби, розглянуті далі по тексту.

Неофіційні перегляди вимог

Розрізняють [8] кілька способів неофіційних переглядів вимог:

- перегляд «за столом»,
- колективна перевірка,
- критичний аналіз.

В перших двох випадках автор вимог звертається по допомогу до колег (відповідно, до одному, або до декількох) з метою видачі практичних рекомендацій з поліпшення продукту. У третьому випадку автор здійснює презентацію розроблені їм

вимоги на нараді з наступним обговоренням.

Неофіційні перегляди використають для знайомства з розробкою, збору відкликань, формування зворотного зв'язка. По статистиці, наведеної в [31], неофіційні перегляди дозволяють виявити до 60% помилок у вимогах.

Інспекції

Поняття інспекції, стосовно до IT-індустрії, уперше було сформульовано Майклом Фэганом (Michael Pagan) з IBM у середині 70-х р. [19].

Відповідно до стандарту IEEE[20] , проведення інспекцій, на відміну від неформальних переглядів, базується на зводі формальних вимог і правил. Представлений нижче огляд правил наведений, ґрунтуючись на роботі [6]. Крім того, слухачам варто порекомендувати ознайомитися з параграфом «Проведення експертизи» глави 15 монографії, де представлений детальний опис процедури експертизи.

Особи, що займають управлінські позиції (менеджери) у відношенні до будь-яких членів 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 утягує авторів проміжного або кінцевого продукту.

У групу інспекції входять лідер, реєстратор, рецензент і трохи (від 2 до 5) інспекторів. Члени команди інспектування можуть спеціалізуватися в різних областями експертизи (мати різні області компетенції), наприклад, предметної області, методах проектування, мові й т.п. У заданий момент (проміжок) часу інспекції проводяться у відношенні окремого невеликого фрагмента продукту (у більшості випадків, фокусируясь на окремих функціональних або інших характеристиках; часто, відштовхуючись від окремих бізнесів-правил, функціональних вимог або атрибутів якості, прим. автора). Кожний член команди повинен досліджувати оцінюваний продукт і інші вхідні дані до проведення інспекційної зустрічі, застосовуючи, можливо, ті або інші аналітичні техніки в невеликим фрагментам продукту або до продукту, у цілому, розглядаючи в останньому випадку тільки один його аспект, наприклад, інтерфейси. Будь-яка знайдена аномалія повинна документуватися, а інформація передаватися лідерові інспекції. У процесі інспекції лідер керує сесією й перевіряє, що все підготувалися до інспектування. Загальним інструментом, використовуваним при інспектуванні, є перевірочний аркуш (checklist), що містить аномалії й питання, пов'язані з аспектами, що викликають інтерес. Результуючий аркуш часто класифікує аномалії й оцінюється командою з погляду його завершенности й точності. Рішення про завершення інспекції приймається відповідно д одним (кожного) із 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 у лекції 4, дозволяє відповістити на запитання: як буде використатися система. Щоб перевірити систему, використається аналогічний механізм: тестових сценаріїв (test cases).

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

Тестові сценарії, як і варіанти використання, можуть підтримувати різні рівні абстракції. Розрізняються концептуальні й детальні ТС. Концептуальний рівень припускає пророблення процедури тестування, інваріантну до конкретної реалізації UI.

Як використати тестові сценарії для тестування вимог? В [8] пропонується наступна процедура.

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

2. Переконатися, що кожний із ТС здійснимо на існуючому наборі вимог.

3. Переконатися, що для кожної вимоги представлений як мінімум один ТС.

4. Прочертити «шлях» кожного із ТС на карті діалогів. Це дозволить: виявити некоректні або пропущені вимоги, виправити помилки на карті діалогів і відшліфувати варіанти тестування.

Як бути з тестуванням нефункціональних вимог? Згідно [33], процедура аналізу вимог уважається виконаної тільки тоді, коли всі вимоги, включені в специфікацію, мають методи 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 варто переформулювати, або деталізувати.

Визначення критеріїв прийнятності

При формальному прийманні продукту існують дві типові процедури: демонстрація продукту Розроблювачем 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, у період стабілізації продукту, для Замовника важливіше навіть не кількість виявлених дефектів, а можливість перевірки – чи годиться розроблена АИС для рішення поставлених їм задач.

Щоб не відкладати настільки важливе питання до моменту приймання системи, надто важливо, поряд з формуванням вимог, утягнути Замовника на ранніх стадіях створення продукту в процес формування *критеріїв прийнятності*. Критерії прийнятності (acceptance criteria) повинні відбити точку зору Замовника на те, що він уважає *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 ваші потреби?». Якщо клієнт не може описати, як він оцінить, що конкретна вимога задоволена системою, виходить, вимога сформульована недостатньо ясно.

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

Перевірка прийнятності базується на ключові (істотних) варіантах використання . При цьому варто 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.

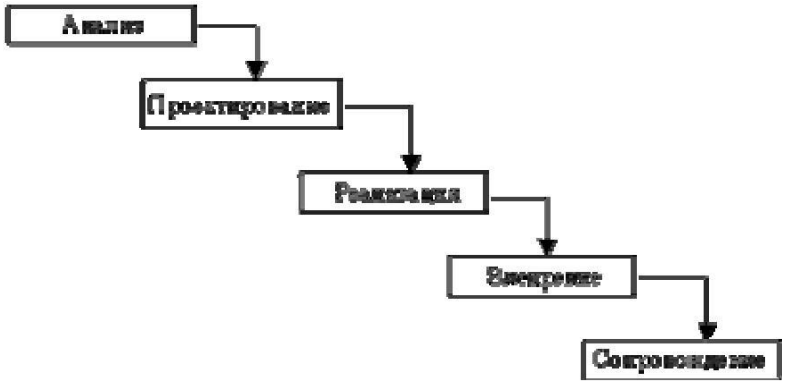
Тема 7. Вступ в управління вимогами

Принципи і прийоми управління вимогами. Базова версія вимог. Процедури керування вимогами Контроль версій.

Атрибути вимог. Контроль статусу вимог. Вимір трудовитрат, необхідних для управління вимогами. Управління змінами. Управління незапланованим зростанням обсягу.

Процес контролю змін. Аналіз впливу зміни. Трасируємість вимог.

Пройшовши етапи виявлення, всебічного аналізу, формалізації, специфікації, перевірки, вимоги до АИС здобуваю статус документа. Сторони ставлять на документі свої підписи, тим самим, засвідчуючи, що саме цей (представлений в SRS) набір вимог представляє звід законів, по якому створюється система. Потім здійснюється проектування й реалізація системи. Готова АИС передається Замовникові, що, разом з Розроблювачем здійснює її приймання й уведення в експлуатацію. Така схема була закладена в підході, що відомий у літературі, як «каскадний» або «водоспадний» [22] (див. мал.). У цій схемі немає місця керуванню вимогами, тому що вони статичні, сформульовані на початку проекту й незмінні в часі.



Каскадний підхід являв собою одну з перших систематизаций потоків робіт програмної інженерії й на момент своєї появи являв безумовну цінність. Однак практика виконання проектів автоматизації в рамках даного підходу показала низький (порядку 20%) відсоток успішних проектів. Перша причина – лавинообразное розростання ціни виправлення помилок, що виникли на ранніх етапах створення системи від етапу до етапу (схема не мала зворотних зв'язків і, відповідно, помилки збирали аж до етапу впровадження). Друга – статичність схеми. Великий проект автоматизації може тривати 2, 3 роки, а вимоги, заморожені в SRS, перестають відповідати бізнесам-реаліям підприємства впровадження, що за настільки довгий період може істотно змінитися.

Переважна більшість сучасних методологій керування проектами розробки програмного забезпечення при всій своїй розмаїтості сходяться в одному: вимоги можуть мінятися! Причому практично на будь-якій фазі виробництва АІС. Ця нова парадигма роботи з вимогами, безумовно, імпонує Замовникам. Тепер вони мають право помилитися й виправити свою помилку. Вони можуть дати волю своєму креативу й постійно винаходити нові можливості й форми реалізації продукту. Але яке Розроблювачеві? Якщо «двозначність – страшилка будь-якої специфікації вимог», то неконтрольована зміна й розростання вимог – ходячий кошмар Розроблювача. Питання контролю процесу змін вимог і його впливу на інші робочі потоки програмної індустрії настільки серйозний, що

породив окрему інженерну дисципліну – керування вимогами. Докладно ознайомитися з усіма етапами, артефактами, прийомами й методами даної дисципліни можна, вивчивши третій розділ монографії, короткий виклад якої лягло в основу цієї лекції.

Згідно RUP, керування вимогами – це систематичний підхід до виявлення, організації й документуванню вимог до системи, а також установка й підтримка угоди між клієнтом і групою розробки із приводу змін вимог до системи. Дана угода, як і тексти вихідних вимог, підлягає документальному оформленню.

До дій по керуванню вимогами ставляться:

- визначення основної версії вимог (моментальний зріз вимог для конкретної версії продукту);
- перегляд пропонованих змін вимог і оцінка ймовірності впливу кожної зміни до його прийняття;
- включення схвалених змін вимог у проект установленим способом;
- узгодження плану проекту з вимогами;
- обговорення нових зобов'язань, заснованих на оціненому впливі зміни вимог;
- відстеження окремих вимог до проектування, вихідного коду й варіантів тестування;
- відстеження статусу вимог і дій по зміні протягом усього проекту.

Принципи й прийоми керування вимогами

Базова версія вимог

Щоб домовитися про зміну вимог, спочатку потрібно їх зафіксувати в «первозданному виді».

Базова версія (baseline) – це набір функціональних і нефункціональних вимог, які розроблювачі зобов'язалися реалізувати в певній версії (ітерації).

Керування вимогами – це робочий процес, отже, він повинен підкорятися певним правилам і процедурам.

Процедури керування вимогами

Процедури керування вимогами базуються на:

- інструментах, прийомах і угодах по керуванню версіями різних документів вимог і окремих вимог;
- правилах складання базової версії вимог;
- статусах вимог, які будуть використатися, і категоріях осіб, які мають право змінювати їх;
- процедурах контролю за статусом вимоги;
- способах, за допомогою яких нові вимоги й зміни існуючих вимог пропонуються, обробляються, обговорюються й передаються всім зацікавленим особам;
- методах аналізу впливу запропонованої зміни;
- відстеженні зв'язків планів і зобов'язань проекту зі зміною вимог.

Контроль версій

Кожна версія документа вимог повинна містити історію переробки, де вказуються внесені зміни, дата кожного з них, особа, внесшее зміна, а також причина. Доцільно додавати номер версії до назви кожної окремої вимоги, якому можна послідовно збільшувати при модифікації вимог.

Для документування версій використаються текстові процесори, електронні таблиці. Існують спеціалізовані засоби для контролю версій і конфігурацій[26]

Атрибути вимог

З позицій керування, кожне з вимог являє собою самостійний об'єкт. Зміни здійснюються в описовій частині даного об'єкта. Контроль змін зручніше здійснювати за допомогою *атрибутів вимог*. Набір атрибутів підбирається для кожного проекту індивідуально, виходячи з максимальної результативності для команди проекту. При першому впровадженні засобів керування змінами рекомендується використати не більше п'яти атрибутів. Це кількість можна буде розширити згодом, коли команда «добере смаку» процесу керування змінами й у тому випадку, якщо додавання нових атрибутів виправдано практичними міркуваннями.

Як шаблон опису атрибутів вимог К. Вигерс [8] пропонує наступний набір:

- дата створення вимоги;
- номер його поточної версії;
- автор вимоги;
- особа, відповідальне за задоволення вимоги;
- відповідальний за вимогу або список зацікавлених осіб (щоб ухвалювати рішення щодо запропонованих змінах);
- стан вимоги;
- походження або джерело вимоги;
- логічне обґрунтування вимоги;
- підсистема (або підсистеми), для яких призначене вимога;
- номер версії продукту, для якого призначена вимога;
- використовуваний метод перевірки або критерій тестування прийнятності;
- пріоритет реалізації;
- стабільність вимоги

Контроль статусу вимог

В автоматизованих засобах керування проектами, наприклад MS Project, для контролю ступеня виконання тієї або іншої роботи використається поняття ступеня виконання (progress), що виражає у відсотках. Даний спосіб слабко застосуємо в програмістських розробках, де, у силу їх слабкої формализованности, важко оцінити роботу у відсотках. При керуванні вимогами рекомендується оперувати не відсотком, а статусом. К. Вигерс пропонує наступний шаблон для визначення статусу вимоги:

Стан	***Визначення***
Proposed (Запропоновано)	Вимога запитана авторизированим джерелом
Approved (Схвалено)	Вимога проаналізована, його вплив на проект перелічена, і воно було розміщено в базовій версії певної версії. Ключові зацікавлені в проекті особи погодилися із цим вимогою, а розроблювачі ПО зобов'язалися реалізувати його
Implemented (Реалізовано)	Код, що реалізує вимога, розроблений, написаний і протестований. Вимога відслідкована до відповідних елементів дизайну й коду
Verified (Перевірено)	Коректне функціонування реалізованої вимоги підтверджено у відповідному продукті. Вимога відслідкована

	до відповідних варіантів тестування. Тепер вимога уважається завершеним
Deleted (Вилучено)	Затверджена вимога вилучена з базової версії. Опишіть причини видалення й назвіть того, хто прийняв це рішення
Rejected (Відхилено]	Вимога запропонована, але не запланово для реалізації ні в однієї майбутніх версій. Опишіть причини відхилення вимоги й назвіть того, хто прийняв це рішення

Вимір трудозатрат, необхідних для керування вимогами

Керування вимогами, як і всякий інший процес, вимагає ресурсів. Контроль зусиль також дозволяє з'ясувати, чи виконують розроблювачі передбачувані задачі для керування вимогами. Основні трудозатраты по керуванню вимогами:

- пропозиція зміни вимог і нових вимог;
- оцінка запропонованих змін, включаючи оцінку впливу зміни;
- зміна роботи;
- відновлення документації вимог або бази даних;
- повідомлення про зміни вимог зацікавленим групам і окремим особам;
- контроль і звіт про стан вимоги;
- збір інформації про трассируемости вимог.

Керування змінами

Керування незапланованим ростом об'єму

Незапланований ріст об'єму ставить під удар:

- 80% проектів по розробці систем керування інформацією;
- 70% проектів по розробці військових систем ПЗ;
- 45% проектів по створенню ПЗ, виконуваних за контрактом.

Незапланованою зміною вимог уважається пропозиція нової функціональності й істотної модифікації після твердження базової версії вимог до проекту. Ніж довше триває робота над проектами, тим більше їхній об'єм.

Проблема полягає не в зміні вимог, а в тім, що запізнілі зміни впливають на вже пророблену роботу. Якщо кожний запит на зміну буде задовольнятися – проект, можливо, ніколи не буде завершений.

Ключова стратегія обмеження росту незапланованих вимог – розробка *гарних* вимог, керуючись прийомами й методами, розглянутими в попередніх лекціях, у максимальному контакті із Замовником.

Інша стратегія – це вміння сказати: «Ні». Психологія більшості людей улаштована так, що їм важко відмовляти, що в цьому випадку може привести до стану постійного пресингу. К. Вигерс пропонує «зм'якшити» цей підхід, замінивши «Ні», на «Не зараз» (вимога обов'язково буде виконано, але не в поточній версії). Автор лекційного курсу, відповідно, хотів би додати в цю скарбничку фразу «Навіщо?». Досвід показує, що дана фраза значно спрощує спілкування й дозволяє сподвигнуть представника Замовника на міркування: а чи дійсно його ідея гарна, а яку конкретну користь вона принесе бізнесу його підприємства?

Однак, не слід робити висновок із усього вищесказаного, що зміни не потрібні.

Зміни неминучі, прийнятні й у ряді випадків сприятливі. Бізнеси-процеси, ринкові можливості, що конкурують продукти, і технології — всі вони можуть мінятися в ході розробки продукту, і менеджери можуть визначити, як у відповідь на ці зміни необхідно відкоригувати напрямок роботи.

Процес контролю змін

Процес контролю змін дозволяє керівникові проекту приймати інформоване бізнес-рішення, що задовольняє клієнта й має комерційну цінність, при цьому контролюються витрати на життєвий цикл продукту. Ви можете відслідковувати статус всіх запропонованих змін і переконатися, що запропоновані вимоги не були загублені або упущені. Після твердження базової версії набору вимог дотримуйтеся цього порядку для внесення всіх пропонованих змін у неї.

Клієнти й інші зацікавлені особи часто ухиляються від нового процесу, однак процес контролю змін — не перешкода для модифікації. Це механізм підсумовування й фільтрації, що дає впевненість, що в проект включена найцінніша зміна.

Если запропонована зміна не настільки важливо, щоб зацікавлене в проекті особа витратила пари хвилин на передачу його через стандартні, прості канали, то варто задуматися про його цінності взагалі. Процес керування повинен бути ретельно задокументований, максимально простий і, що важливіше всього, ефективний.

Після реєстрації запиту на зміну необхідно ухвалити рішення щодо його подальшій долі. *Рада по керуванню змінами* (change control board, CCB) (іноді його називають радою по керуванню конфігурацією) був визнаний кращим практичним рішенням при розробці ПО[27]. На практиці функції Ради можуть бути делеговані й одній людині (залежно від розмірів проекту).

Запрос на зміну може бути прийнятий, або відхилений. У першому випадку його необхідно включити в план робіт над проектом, у другому – сформулювати мотивована відмова. При ухваленні рішення по запиті необхідно виходити: а) зі ступеня важливості запиту для Замовника й б) з його вартості для Розроблювача. Вартість визначається на підставі *аналізу впливу зміни***.**

Аналіз впливу зміни

Аналіз впливу забезпечує точне розуміння підтексту запропонованої зміни, що допомагає команді приймати інформовані бізнеси-рішення про те, яке зміна схвалити. Аналіз дозволяє виявити компоненти, які може знадобитися створити, змінити або відхилити, і оцінити витрати, пов'язані з реалізацією зміни. До того, як розроблювач відповість: «Звичайно, без проблем», він повинен витратити час на аналіз результату зміни.

Голова ради по керуванню змінами звичайно просить досвідченого розроблювача виконати аналіз результату певного.

Аналізу результатів змін зачіпає три аспекти.

1. Визначите можливі наслідки зміни. Часто вони викликають значний хвильовий ефект. Включення множини функцій у продукт може знизити його продуктивність до неприйнятного рівня, наприклад, коли системі, що запускає щодня, буде потрібно 24 години для завершення одного запуску.

2. Визначите всі файли, моделі й документи, які, можливо, прийде змінити, якщо

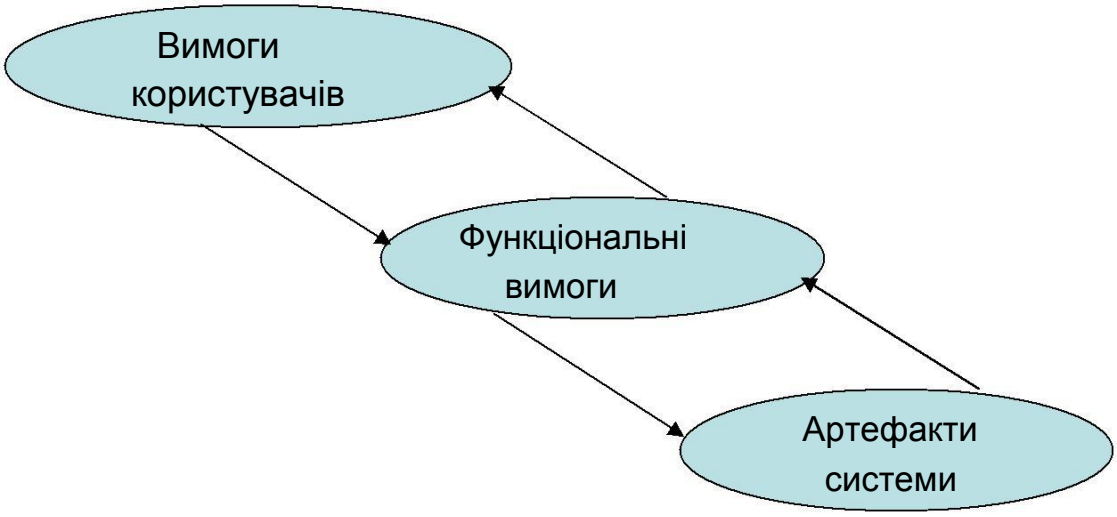
команда включить всі запитані зміни.

3. Визначите задачі, необхідні для реалізації зміни, і оціните зусилля, необхідні для виконання цих задач.

Трасируємість вимог

Зв'язку трассируемости допомагають стежити за розвитком вимоги в обох напрямках— від першоджерела до реалізації й навпаки. Трассируемость являє собою одну з якостей гарних вимог, див. матеріали лекції 2

Для здійснення аналізу трассируемости кожна вимога повинне бути унікально ідентифіковано.



Мал. Основні типи трассируемости вимог

Вимоги користувачів відслідковуються в напрямку до формально специфікованих функцій системи, щоб зрозуміти, які вимоги будуть порушені, якщо потреби клієнтів зміняться. Це також дозволяє переконатися в тім, що в специфікації вимог відбиті всі потреби клієнта. Можна здійснити аналіз і у зворотному напрямку, щоб визначити походження кожної вимоги до ПО.

У процесі аналізу, проектування й реалізації компонентів системи можна відслідковувати зв'язки, що ведуть від вимог до артефактів (документам, моделям, програмним модулям і т.п.) системи. Цей тип зв'язку гарантує, що кожна вимога задоволена, оскільки ви знаєте, який компонент відповідає кожній вимозі.

Четвертий тип зв'язку контролює окремі артефакти в напрямку до вимог для того, щоб ви знали причину створенню кожного з них.

Припустимо, тестеровщик виявить незапланована функціональність при відсутності відповідної вимоги. Цей фрагмент коду може свідчити, що розроблювач реалізував офіційну вимоггу, що аналітик тепер може додати до специфікації. Або ж це може бути код-«сирота», що прикрашає фрагмент, що не ставиться до продукту. Зв'язку трассируемости допоможуть вам відсортувати подібні ситуації й одержати більше повне подання про те, як саме фрагменти вашої системи становлять одне ціле. І навпаки, варіанти тестування, які створені на основі окремих вимог і які можна простежити до цих вимог, також являють собою механізм виявлення нереалізованих вимог, оскільки

очікуваної функціональності не буде.

Найбільш типовий спосіб подання зв'язків між вимогами й іншими елементами системами — матриця трасируємості вимог, що також називають матрицею відстеження вимог або таблицею трасируємості (requirements traceability matrix). У табл. показана ілюстрація частини такої матриці.

Табл.

Пользова-тельское требование	Функциональное требование	Элемент дизайна	Модуль кода	Вариант тестирования
UC-28	catalog.query.sort	Каталог класса	catalog.sort()	search.7 search. 8
UC-29	catalog.query.import	Каталог класса	catalog.import() catalog.validate()	search. 12 search. 13 search. 14

Інша форма подання зв'язків трассируемости – дерево трасувань.

Тема 8. Удосконалення процесів роботи з вимогами

Моделі вдосконалення ISO9000.

SEI-CMM, SEI-CMMI. Принципи вдосконалення Процес вдосконалення.

Оцінка поточних прийомів. Планування. Створення та апробація нових процесів. Оцінювання результатів і прийняття рішень.

Парадигма керування якістю, як спосіб організації виробництва, з'явилася давно. Ідеї, закладені в групі стандартів ISO9000[28], ідуть корінням, зокрема, і в такі «радянські» винаходи, як підтримка раціоналізаторських пропозицій, наставництва й ін.

У сучасній концепції процессно-ориентированной організації бізнесу процес безперервного вдосконалювання якості займає одну із ключових позицій.

Стосовно до софтверной індустрії, крім серії ISO9000, найбільше успішно себе стандартами, що зарекомендували, якості є SEI CMM, SEI CMMI, ISO/IEC 15504 (SPICE) , Bootstrap, TickIT.

Моделі вдосконалювання

ISO9000

Активне впровадження методів керування якістю на Заході почалося на початку 1960-х років. В основу стандартів серії ISO9000 лягла філософія підходів CPI (Continuous Process Improvement) і TQM (Total Quality Management) [39]. Підйом економіки післявоєнної Японії багато в чому був обумовлений ідеям, закладеним в TQM.

Якість – термін, що для одних означає необхідність робити те, що бажає споживач, для інших — те, що відповідає його потребам. Менеджмент якості, як він визначений в ИСО 9001:2000, виходить насамперед з того, що люди працюють краще, якщо їм відомо

те, чим вони займаються.

Тому перш, ніж приступитися до якої-небудь дії, варто одержати відповіді на питання: що потрібно робити, хто буде перевіряти зроблене, як це потрібно робити і як визначити, що робота завершена? Необхідно також продумати, як ви збираєтеся управляти даним процесом і як його можна вдосконалити.

Основні принципи ISO9000:

- Концентрація на потребах замовника;
- Активна лідируюча роль керівництва;
- Залучення виконавців у процеси вдосконалювання;
- Реалізація процессного підходу;
- Системний підхід до керування;
- Забезпечення безперервних поліпшень;
- Прийняття рішень на основі фактів;
- Взаємовигідні відносини з постачальниками.

Безумовне достоїнство стандартів цієї групи пов'язане з тим, що вони апробовані на множині підприємств миру. Однак рекомендації даних стандартів носять досить загальний характер і не враховують специфіку підприємств галузі IT. Для цього минулого розроблені спеціалізовані підходи, розглянуті нижче.

SEI-CMM, SEI-CMMI

Стандарт CMM (the Capability Maturity Model) розроблений інститутом інженерії програмного забезпечення (SEI) при університеті Карнегі^-Меллон.

Призначення стандарту – оцінка рівня «зрілості» (maturity levels) організації – розроблювача програмного забезпечення. Виділяються п'ять рівнів: початковий, повторюваний, певний, керований і оптимизирующий (докладніше див. в [22-8]). Даний стандарт одержав широку популярність, значну кількість західних IT-компаній сертифіковане по CMM.

В 2000 р. SEI випустив CMMI-SE/SW, інтегровану модель удосконалювання як ПО, так і можливостей конструювання систем.

CMMI-SE/SW має дві форми. Східчасте подання (the staged representation) відповідає структурі SW-CMM з невеликими уточненнями найменувань рівнів. П'ять рівнів зрілості містять 22 області технологічних процесів, показаних у таблиці 1. (CMU/SEI, 2000a). Безперервне подання (continuous representation), містить інший погляд: ті ж 22 області структурируются по 4 категоріям: керування процесами, керування проектами, конструювання й підтримка (CMU/SEI, 2000b).

В безперервному поданні замість рівнів зрілості визначаються шість рівнів здатностей (capability levels) для кожної області технологічних процесів. Це подання дозволяє кожній організації вирішувати, який рівень здатностей їй відповідає в кожній з 22 областей технологічних процесів.

Як і в CMM, у розглянутому стандарті на рівні 2 є область, іменована «Керування вимогами», але, на відміну від попереднього стандарту, на рівні 3 є й окрема область «Розробка вимог». Розміщення цієї області на рівні 3 не має на увазі, що вимоги для проектів організації, що не досягли рівня 2, збирати й документувати не потрібно. Керування вимогами розглядається як спосіб, що допомагає створювати більше передбачувані й менш хаотичні проекти, що становить сутність рівня 2 CMM. Прийнявши

порядок керування змінами й перевірки статусу вимог, організація може більше уваги приділяти розробці високоякісних вимог.

Табл. 1

Рівень зрілості	*Назва*	*Області процесів*
1	Початковий	(немає)
2	Керований	**Керування вимогами** Планування проекту Моніторинг і контроль проекту Керування угодами з постачальниками Виміри й аналіз Забезпечення якості процесів і продуктів Керування конфігурацією
3	Певний	**Розробка вимог** Технічне рішення Інтеграція продуктів Верифікація Валидация Концентрація уваги на процесі Визначення процесу організацією Організаційне навчання Інтегроване керування проектом Керування ризиком Аналіз і дозвіл питань
4	Кількісно керований	Продуктивність організаційних процесів Кількісне керування проектом
5	Оптимизирующий	Організаційні нововведення і їхнє розгортання Випадковий аналіз і дозвіл

Область процесів «Керування вимогами». Ключові теми містять у собі те, як команда розроблювачів повинна здобувати розуміння вимог і розв'язувати питання із клієнтами, утягувати учасників проекту в роботу з вимогами й управляти змінами. На відміну від SW-CMM, трасування (одне із ключових властивостей вимог) включено в розглянуту область процесів. У стандарті обговорюються наступні якості трасування:

- забезпечення запису джерел низкоуровневых або вторинних вимог;
- трасування кожної вимоги вниз, до вторинних вимог, і його розміщення по функціях, об'єктах, процесах і виконавцях;
- установка горизонтальних зв'язків між вимогами, що належать до одного типу.

Область процесів «Розробка вимог».

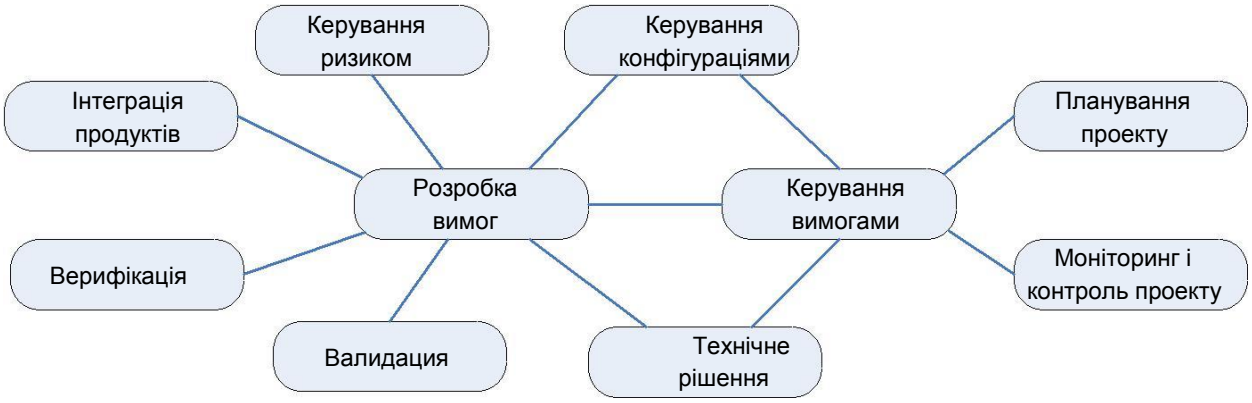
В CMMI-SE/SW описані три набори прийомів розробки вимог:

- прийоми, що визначають повний набір вимог клієнтів, які потім використаються для розробки вимог для продукту (виявити потреби зацікавлених у проекті осіб; перетворити потреби й обмеження у вимоги клієнтів);
- прийоми, що визначають повний набір вимог для продукту (установити компоненти продукту; розмістити вимоги по компонентах продукту; визначити вимоги до інтерфейсу);

- прийоми одержання вторинних вимог, розуміння вимог і їхнього підтвердження (установити оперативні концепції й сценарії; визначити необхідну функціональність системи; проаналізувати вторинні вимоги; оцінити вартість, строки й ризик створення продукту; затвердити вимоги).

Як у CMM, так і в CMMI формулюються мети, до яких проект або організація по розробці ПО повинні прагнути в різних областях процесів. У них також рекомендуються технічні прийоми, що допомагають досягти цих цілей.

CMMI-SE/SW регламентує взаємозв'язку між керуванням вимогами, розробкою вимог і інших областей процесів:



Принципи вдосконалювання

Сформульовані наступні принципи вдосконалювання якості програмних систем:

- поэтапность,
- безперервність,
- циклічність,
- наявність стимулу,
- орієнтація на меті,
- проектний підхід.

Заходу щодо вдосконалювання процесів варто вводити *поетапно*. Людям нерідко буває важко відмовлятися від старих звичок, звикати до нового. Запропоновані зміни повинні пройти перевірку досвідом; не все із запропонованого обов'язково дасть ефект, якісь новації прийде переглянути, від чогось – відмовитися.

Безперервність – один із ключових принципів керування якістю: часто заходи проводяться у вигляді кампанії, що загасає після одержання сертифіката. Організація, що прагне до лідерства, повинна підтримувати «дух якісної роботи» постійно.

Бізнес-процес поліпшення вимог характеризується *циклічністю* (див. Процес удосконалювання): його основні етапи повторюються на усе більше високому рівні. Цикли оптимізації в софтверных організаціях зручно пристосовувати до проектів, виконуваних у робочих групах. Аналіз недоліків доцільно робити тоді, коли вони в «оперативній пам'яті» групи проекту, наприклад – один раз у середині проекту й один – відразу після його закінчення. Кожний проект по-своєму унікальний і несе в собі потенціал для поліпшення процесів.

Основним *стимулом* до змін К. Вигерс уважає труднощі, з якими зштовхнулася команда проекту, наприклад:

- розроблювачі не уклалися в графік, тому що незрозумілі й неоднозначні вимоги

потрапили до них пізно;

- розроблювачам довелося багато працювати надурочно, тому що незрозумілі або розпливчасті вимоги були уточнені занадто пізно в процесі розробки;
- спроба тестування системи не вдалася, тому що тестировщики не розуміли, що продукт повинен робити;
- потрібна функціональність була реалізована, але користувачі не задоволені млявою продуктивністю, незручністю роботи або інших недоліків якості продукту;
- організації довелося піти на високі витрати на супровід, тому що клієнтам потрібна була маса додаткових функцій, які випливало визначити під час складання вимог;
- організація-розроблювач ПО придбала погану репутацію постачальника продуктів, які не подобаються клієнтам.

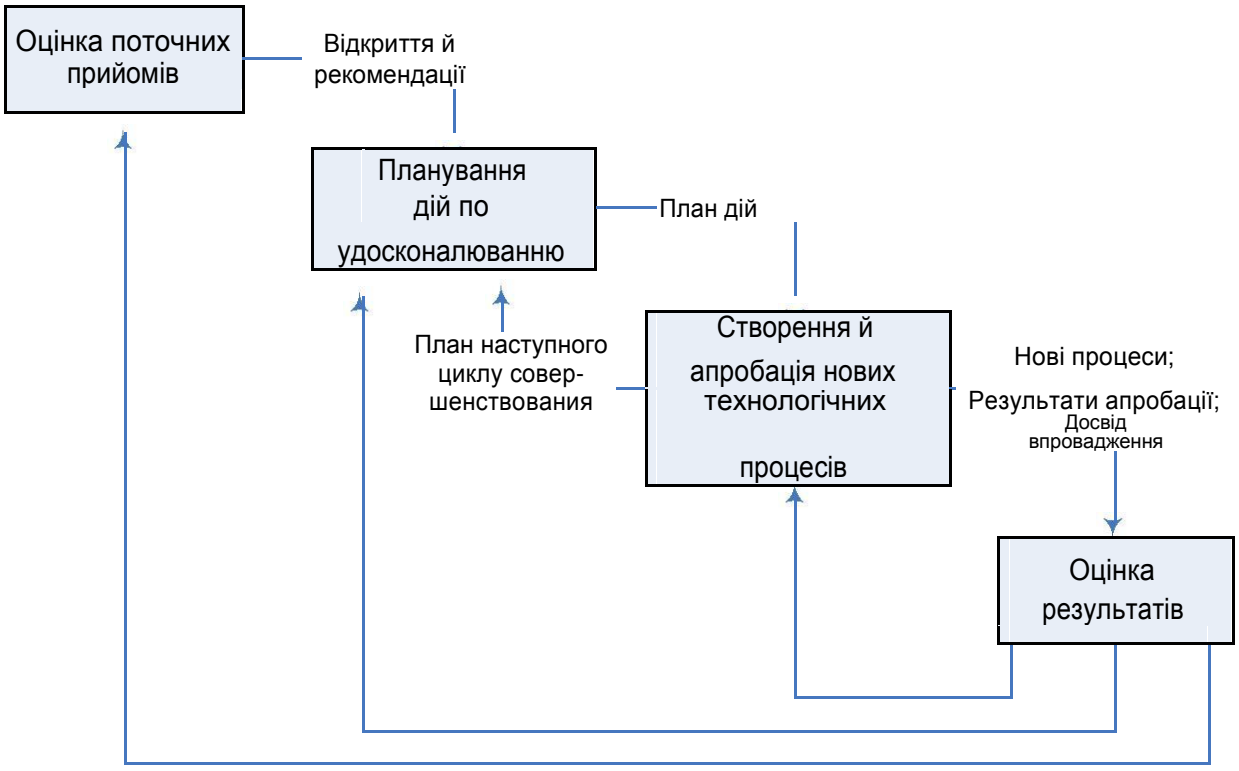
Зміни технологічних процесів повинні бути *целеориентированы*. Приклади цілей:

- зменшення об'єму роботи, викликаного проблемами з вимогами;
- підвищення точності планування й реалістичності планів;
- зниження (виключення) числа ситуацій появи нових вимог на фінішних етапах проекту.

Дії по вдосконалюванню процесів повинні плануватися, контролюватися й управлятися, як *міні-проекти*. Це спрощує виділення необхідних ресурсів, відстеження змін і оцінки результативності змін.

Процес удосконалювання

На мал. показаний типовий цикл удосконалювання процесів при створенні програмного забезпечення.



Оцінка поточних прийомів

У відповідність із принципом цілеспрямованості, у роботи з удосконалювання необхідно почати з формулювання цілей і оцінкою, наскільки існуючі процеси відповідають даним цілям. Для цілей оцінки застосовні відомі в бізнесі-моделюванні й аналізі вимог методи й прийоми: від проведення інтерв'ю й постановочних семінарів до фіксації моделі «Як є».

Ефективним способом є залучення зовнішніх консультантів, які можуть скласти неупереджений погляд на положення у вашій компанії.

Результатами оцінки є список виявлених сильних і слабких сторін у поточних процесах, а також, початкові рекомендації з удосконалювання (переходу до моделі «Як треба»).

Планування

В відповідності із принципом проектного підходу до проведення заходів щодо вдосконалювання, для міні-проекту вдосконалювання необхідно проробити все те, що звичайно робиться при ініціації проектів: здійснити декомпозицію робіт; виділити ресурси, призначити виконавців; створити плани робіт.

В Стратегічний план описує загальну програму вдосконалювання процесів вашої організації. Тактичні плани дій зачіпають конкретні області вдосконалювання, наприклад процес збору вимог або процедуру призначення пріоритетів. У кожному плані дій повинні бути зазначені мети дій по вдосконалюванню, учасники й окремі задачі. План також дає можливість відслідковувати виконання процесу вдосконалювання, відзначаючи виконання окремих задач.

В плані дій не повинне бути більше 10 пунктів; строк його реалізації не повинен перевищувати 2-3 місяця.

Нижче приведе шаблон декомпозиції задачі керування вимогами .

1. скласти проект процедури керування змінами; 2. перевірити й модифікувати процедуру керування змінами;

3. провести пробне випробування процедури керування змінами для проекту;

4. модифікувати процедуру керування змінами на основі зворотної реакції по пробному випробуванню;

5. оцінити інструментальні засоби виявлення проблем і вибрати одне з них для підтримки процедури керування змінами;

6. придбати обраний інструментальний засіб виявлення проблем і настроїти його для підтримки конкретної процедури;

7. впровадити нову процедуру керування змінами й інструментальний засіб в організації.

Створення й апробація нових процесів

Принцип поетапності призиває не робити «революцій» в удосконалюванні процесів. Будь-яка новація, опис якої знайдене в літературі, запозичена з досвіду колег або розроблена особисто вами, повинна пройти випробування на *вашій команді* й *ваших проектах*. Відомий неполіткоректний принцип «що росіянинові добре – те німцеві смерть» мовою сучасного менеджменту ІТ-проектів звучить, як «облік системи цінностей, прийнятих у команді розроблювачів».

Апробація на реальних задачах – єдиний гарантований спосіб перевірити – чи

годиться той або інший інструмент для вашої команди. Щоб не втягувати в масштабні експерименти значні ресурси існує спосіб пілотних (пробних) проектів.

К. Вігерс пропонує наступні методичні прийоми при апробації нових процесів:

- вибирайте для участі в пробних проектах людей, які будуть ставитися до нових прийомів неупереджено й зможуть дати їм оцінку. Це можуть бути як прихильники, так і скептики, але вони не повинні бути затятими супротивниками пропонованих прийомів;
- щоб результати було легко витлумачити, визначите кількість критеріїв, по яких команда буде оцінювати пробний проект
- визначите зацікавлених осіб, яких варто проінформувати про те, що являє собою пробний проект і чому він виконується;
- подумайте про можливості випробування нових процесів вроздріб у різних пробних проектах. Так вам удасться втягнути у випробування більше людей, що збільшує поінформованість, кількість відгуків і прихильників;
- для більше повної оцінки поцікавтеся в учасників пилотных проектів, що б вони відчули, якби їм довелося повернутися до існуючих прийомів роботи.

Оцінка результатів і прийняття рішень.

Оцінка результатів апробації нових процесів повинна показати – чи досягнуті поставлені цілі, чи варто здійснювати широкомасштабне впровадження новацій, апробованих на пілотному проекті, або «овчинка вироблення не коштує».

Серед ключових питань в області оцінки результатів можна виділити наступні.

- Наскільки гладко пройшли пробні проекти і як ефективно вони дозволили невизначеності відносно нових процесів?
- З чи оббираєтеся ви міняти що-небудь у наступних пилотных проектах?
- Як пройшло загальне впровадження нових технологічних процесів?
- Чи вдалося вам довести до відома кожного інформацію про користь нових процесів або шаблонів?
- Чи змогли учасники зрозуміти й ефективно застосувати нові процеси?
- Чи збираєтеся ви міняти що-небудь при проведенні наступного впровадження?

При оцінюванні результативності досягнення поставлених цілей треба различать заходу, користь від яких проявляється відразу й ті, вигода від яких виявиться через значний час. Необхідно враховувати ефект «кривої навчання» (learning curve): продуктивність падає, поки люди пристосовуються до нових способів роботи. Короткочасне падіння продуктивності, іноді називане «лощиною розпачу» - в — це частину необхідного внеску, що ваша організація вносить в удосконалювання процесів.

Незважаючи на всі можливі труднощі, заходи щодо поліпшення якості при уважному до них відношенні неминуче приведуть до підвищення ефективності роботи команди.

Тема 9. Вимоги в управлінні проектом. Висновок

Від рамок проекту до експрес-планування. Планування проекту на основі вимог, шлях RUP.

Вимоги до гнучких методологіях. Артефакти для роботи з вимогами до гнучких методологіях планування версій і ітерацій

Аналіз вимог та управління ризиками. Сучасні тенденції в розвитку АІС і технологій їх створення. Покупне або замовне ПЗ - критерії вибору. Стратегії вибору рішення. Аналіз вимог.

Аналіз невідповідності. Підхід на основі найкращих практик. Процес вибору рішення.

Щоб визначити кошторисну вартість і тривалість робіт із проекту автоматизації без грубих помилок, необхідно виявити й проаналізувати вимоги, а також сформувати архітектурну основу, украй бажано створити прототипи. І це – як мінімум. Іншими словами, для того, щоб створити АИС, спочатку потрібно проаналізувати можливість створення.

Така постановка питання цілком логічна й обґрунтована, це підтвердить будь-який Розроблювач. Однак, вона викликає багато питань, наприклад:

- Де знайти Замовника, що погодиться чекати 2-3 місяця, поки Розроблювач складе для нього комерційна пропозиція?
- Хто оплатить роботи з аналізу вимог? (очевидно, Замовник)
- Як бути, якщо ціна питання виявиться непомірної й від проекту прийде відмовитися – хто відшкодує Замовникові збитки на проведення досліджень?

Розумний Замовник зможе знайти вихід із цього непростого положення, наприклад:

- підшукавши Розроблювача, що володіє багатим досвідом виконання подібних проектів, що зможе дати необхідну оцінку значно швидше (але ризик помилки при цьому залишається);
- взявши на себе ризики можливого припинення проекту на ранніх стадіях, у випадку, якщо виявиться його невідповідність бюджетним обмеженням (у складних ризикованих проектах краще втратити 5% або 10% від бюджету, що закладає, чим всі 100%, як це було в «каскадних» схемах розробки) - шлях прогнозуючих методологій
- разделив з Розроблювачам відповідальність за кінцевий продукт, приготувавшись день за вдень працювати з ним рука об руку аж до появи результату – шлях гнучких (agile) методологій.

Від рамок проекту до експрес-планування

Початкову, саму грубу оцінку проекту можна зробити на підставі документа «Бачення». Так, шаблон Vision/Scope MSF містить список ключових характеристик/функцій, критерії прийнятності й (що дуже важливо) перелік характеристик/функцій, які лежать «поза рамками» проекту. Паралельно із проробленням концепції, перша фаза MSF містить роботи з аналізу ризиків: виявляються й оцінюються

головні ризики проекту.

Щоб зробити перше наближення плану й кошторису проекту на ранніх етапах аналізу, пропонується наступний підхід:

- Виділити 25 – 99 функцій, що характеризують систему (спільно, Замовник і Розроблювач);
- Установити пріоритети для кожної з функцій (Замовник);
- Оцінити трудозатраты (Розроблювач);
- Оцінити ризики (Розроблювач, можливе залучення Замовника);

Всі оцінки робляться по 3-бальній шкалі (високий, низький, середній; критичний, важливий, корисний і т.п.) і зводяться в таблицю:

№ пп.	*Функція*	*Пріоритет*	*Трудомісткість*	*Ризик*

Потім , у процесі консультацій Замовника й Розроблювача на основі отриманої інформації визначається набір функцій, що ввійдуть у базову версію проекту.

Результати планування на концептуальному рівні, пророблені , наприклад, на основі вищевказаного шаблона, дозволяють дати першу оцінку розмірам проекту. Така оцінка не відрізняється особливою точністю, але за певних умов (досвід рішення Розроблювачем аналогічних задач; довірчі відносини між Замовником і Розроблювачем) може служити відправною крапкою для висновку контракту на всю систему.

Планування проекту на основі вимог, шлях RUP

RUP підтримує дворівневу схему планування робіт над проектом, розділяючи план проекту й план ітерації. Виходячи з базової концепції планування ітераційних проектів, план проекту розбивається на фази:

- Початок,
- Уточнення,
- Конструювання,
- Перехід.

Виходячи з рекомендацій методології по декомпозиції робіт проекту залежно від ступеня складності проекту й кваліфікації команди, у кожній фазі виділяється одна або більше ітерацій.

Назначаются дати головних віх (закінчення фаз):

- цілей життєвого циклу (кінець фази початку, рамки проекту і його бюджет);
- архітектури життєвого циклу (кінець фази уточнення, закінчена архітектура);
- початкової працездатності (кінець фази конструювання, бета-версія продукту);
- випуску виробу (кінець фази переходу й повного циклу розробки). Призначаються дати малих віх, присвячені до закінчення ітерацій і їх головні мети. Основні цілі ітерацій – випуск релізів, демонструємих, або переданих Замовникові, однак не всяка ітерація приводить до випуску релізу.

План проекту створюється якомога раніше в початковій фазі й модифікується в міру необхідності.

Що це означає на практиці:

1) укрупнений план робіт складається «якомога раніше», наприклад – через місяць після початку робіт;
2) бюджет з'являється лише до закінчення першої фази (а вона, залежно від складності проекту може тривати місяць, а може й півроку);
3) як план, так і бюджет проекту являють собою лише прогноз, що може коректуватися протягом робіт над проектом;
4) на момент появи плану й бюджету повинне з'явитися докладний опис лише 20% ключової функціональності системи й «широке, але неглибоке» [2] опис 80% функціональності.

Таким чином, концепція укрупненого планування в RUP, як типопредставителе класу прогнозуючих методологій, припускає базувати відносини між Замовником і Розроблювачем на прогнозах, ступінь вірогідності яких залежить від таких факторів, як якість пророблення вимог, кваліфікація команди Розроблювача, складність і новизна проекту.

Більше конкретна інформація представлена в плані ітерації. Основні його особливості:

1) План ітерації базується на функціональних вимогах. До моменту початку ітерації зовсім точно відомо, які вимоги повинні бути оброблені (деталізовані, спроектовані, запрограмовані) у даній ітерації.

План ітерації складається, виходячи зі сформульованих вище оцінок вимог – пріоритетності, ступеня ризику, трудомісткості.

2) План ітерації має тверді строки. У випадку прояву незапланованих ризиків задовільним варіантом є досягнення домовленості про реалізації вимог даної ітерації не в повному об'ємі, або переносі вимог у наступну ітерацію; переносити строки поточної ітерації не рекомендується;

3) Точний план складається на одну, чергову ітерацію. До моменту закінчення поточної ітерації повинен бути зверстаний план чергової ітерації. Такий підхід дозволяє більш гнучко працювати з ризиками.

Таким чином, слід зазначити, що вимоги є вирішальним чинником у плануванні ітераційного проекту.

Вимоги в гнучких методологіях

В противага прогнозуючим методологіям створення програмного забезпечення, відносно недавно сформувалася парадигма гнучких (agile) методологій. У лютому 2001 р. ініціативна група з 17 фахівців об'єдналася в Альянс гнучкої розробки програмного забезпечення. Ця група розробила й прийняла Маніфест гнучкої розробки:

□□Індивідуальності й взаємодії ВИЩЕ процесів і інструментів

□□□Працююче програмне забезпечення ВИЩЕ всебічної документації

□□□Співробітництво із клієнтами ВИЩЕ переговорів за контрактом

□□□Реакція на зміни ВИЩЕ проходження плану і 12 додатків (у настільки ж лаконічній формі) до нього.

В певного ступеня на противагу всьому той, що було сказано в попередніх лекціях, члени Альянсу ставлять під сумнів необхідність всебічного моделювання й документування вимог і навіть зазіхають на святе святих – плани й контракти.

На сьогодні «бути гнучким» стало модним. Апологети методологій, затаврованих

членами Альянсу, як «прогнозуючі» і навіть «великовагові» вступають у дискусії – чи можна вважати адаптувати ту або іншу методологію на «гнучкі рейки». Так, опубліковані як мінімум два варіанти гнучкої трансформації для RUP; MSF опублікувало нотацію agile MSF.

Артефакти для роботи з вимогами в гнучких методологіях

З позицій роботи з вимогами основними засобами, якими оперують гнучкі методології, є карти подання системи, історії користувачів, приймальні тести й CRC-карти.

Карта подання деякою мірою заміняє документ «концепція», прийнятий у прогнозуючих методологіях. На відміну від концепції, подання – це текст розміром в 20-30 слів, що вміщається на невеликий (розміром з візитну) картці.

Історії користувачів (user story)дуже сильно нагадують короткі описи варіантів використання. Особливості історій користувача – у тім, що вони, по-перше, повинні бути дійсно короткими (також уміщатися на картці), в- других – у тім, що це – дійсно історії *користувачів*, тобто розповіді про те, як вони планують використати систему. Використання історій користувача виключає ситуацію, коли аналітик щось придумав (домыслил) за користувача – аж ніяк, ці артефакти створюють самі користувачі. Історії користувача повинні мати осмислені найменування й номери.

Історії користувача, крім базового функціонала, можуть містити *декорації*, дуже напоминающие *орієнтири* RUP (див. лекції 5).

Приймальні тести звичайно пишуть на звороті карти з відповідною історією користувача. Шаблон, використовуваний у методології XP, містить 3 пропозиції:

- Установка (контекст; ініціююча подія),
- Операція (функція з кількісними характеристиками),
- Підтвердження (результати виконання функції).

CRC-карти (Класс-Ответсвенность-Кооперация), як і попередні 3 артефакти, являють собою невеликі картки, у заголовку яких представлене назва класу, а нижче – таблиця у два стовпчики. У лівому стовпчику перераховані *відповідальності* (тобто высокоуровневый погляд на його методи) класу, у правої – класи, що складаються в кооперації з розглянутим класом.

Планування на основі вимог на прикладі XP

Планування включає наступні роботи:

- оцінювання,
- планування версій і ітерацій.

Оцінювання являє собою визначення обсягу робіт у розрізі історій користувача. Кожна історія оцінюється в *пунктах*. Один пункт дорівнює «ідеальної» (сорокачасовой) тижню, цілком присвяченої програмуванню. Якщо оцінка лежить у межах від 1 до 3 пунктів – то він ставиться на картці історії. Якщо оцінка менш 1 – на картці ставиться 0. Це – так званий «*пісок*». Якщо оцінка перевищує 3 пункти – ми маємо справу з «*епопеєю*». У цьому випадку картка позначається, як «split» і підлягає процедурі *поділу*. Інша стратегія роботи з такою карткою -спробувати вмістити її в оптимальний строк шляхом виключення декорацій. У випадку, якщо історія користувача складна для експрес-оцінки – необхідно провести дослідження або «*цвях*» планування.

Планування версій і ітерацій.

Планування в XP базується на наступних основних поняттях: продуктивність, пріоритети, вартість версії, складання плану версій, складання плану ітерацій.

Продуктивність або швидкодія команди базується на оцінках пунктів історії. Однак необхідно враховувати, що пункти представляють ідеальні оцінки, крім того істотну роль має досвід команди в оцінюванні (для починаючих команд можлива значна погрішність).

Ключову роль у призначенні *пріоритетів* грає, безумовно, замовник. Однак і Розроблювач має право голосу при відборі історій, які повинні потрапити у версію (питання архітектури, ключової функціональності й т.п.).

Вартість версії визначається, базуючись на продуктивності, пріоритетах і строках.

План версій дає Замовникові початкове розуміння вартості проекту. Ця оцінка дає йому можливість відмовитися від проекту в початковій його стадії, якщо строки й (або) ціна є неприйнятними.

У випадку, якщо план версій прийнятий – складається *план ітерацій*, що відбиває кроки (ітерації), які необхідно проробити, щоб домогтися необхідної функціональності продукту.

Аналіз вимог і керування ризиками

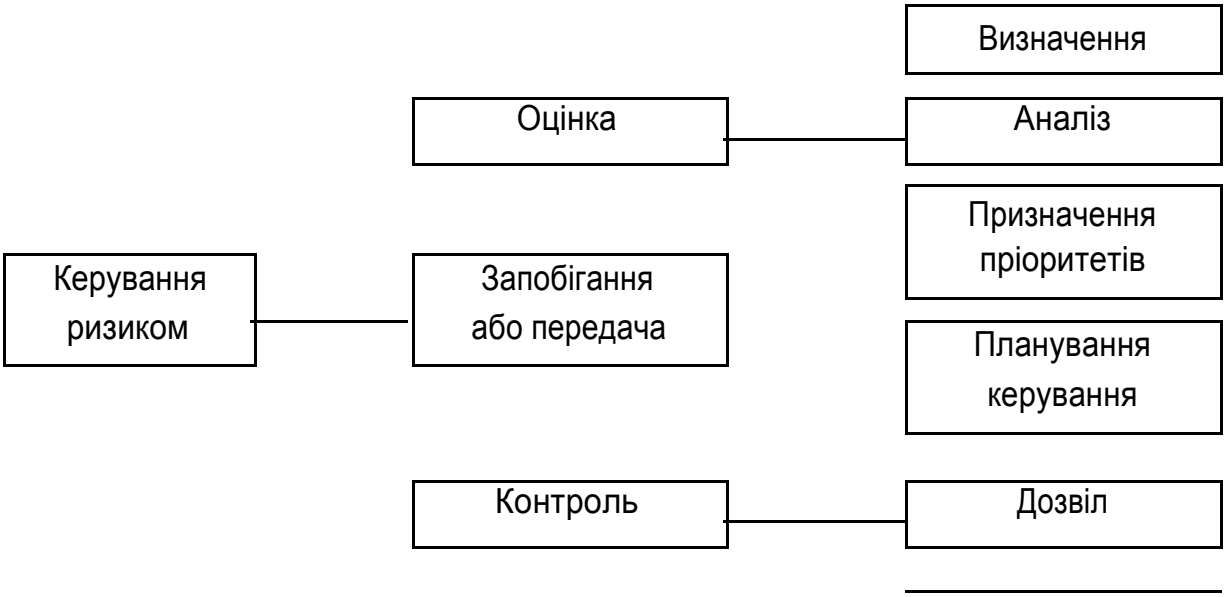
Ризик у реалізації програмних проектів – це потенційна проблема, що має істотну ймовірність негативно вплинути на успішність проекту, наприклад – на здачу його в строк, задоволення бюджетних обмежень, якість продукту, ефективність роботи команди.

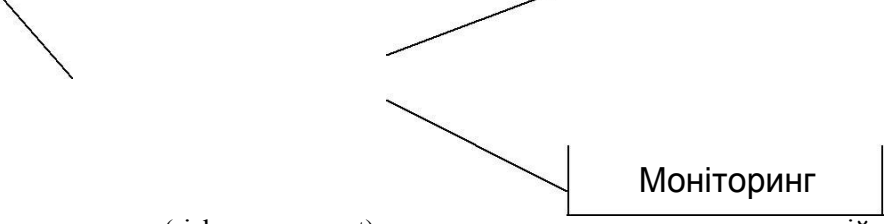
Керування ризиком – комплекс заходів щодо виявлення, оцінці, запобіганню й контролю ризиків проекту.

Як пише К. Вигерс , «Якщо що-небудь негарне вже відбулося з вашим проектом, те це – проблема, а не ризик…Керування ризиком означає роботу потенційною небезпекою до того, як вона перейде в кризову фазу». Менеджери проектів повинні виявляти ризик і управляти їм, починаючи з факторів, пов'язаних з вимогами, у співробітництві із представниками Замовника.

Стратегії й роботи з керування ризиком

Керування ризиком містить у собі дії, показані на мал.





Роботи з *оцінювання* ризику (risk assessment) починаються з *визначення* потенційних небезпек для проекту. Як методика виявлення може бути рекомендована методика мозкового штурму. Гарною підмогою для цього етапу робіт є наявна в Розроблювача класифікація ризиків.

Так, всі ризики прийнятий для ділити на прямі (ті, на які Розроблювач може так чи інакше впливати) і непрямі (незалежні від Розроблювача) [15].

М. Фаулер [7] запропонував розділити всі ризики на чотири категорії:

- ризики, пов'язані з вимогами,
- технологічні ризики,
- ризики, пов'язані із кваліфікацією персоналу,
- політичні ризики.

Розповсюджені фактори ризику, пов'язані з вимогами, включають невірне розуміння вимог, недостатнє залучення користувачів, неточності або зміни в масштабах і цілях проекту, постійно нестабільні вимоги. Докладний аналіз цих видів ризиків можна знайти в [8], глава 23.

Аналіз ризику зводиться до дослідження й опису потенційних наслідків конкретних факторів ризику для проекту, а також імовірності їхнього прояву.

Визначення пріоритетів складається з пошуку відповідей на два питання: наскільки ймовірний прояв ризику в проекті; наскільки руйнівні можуть бути наслідку його прояву.

Виявлені ризики містяться в спеціальний документ – *risk list.* Існують три основні стратегії поводження відносно ризиків:

Запобігання ризику, передача ризику, прийняття ризику.

Запобігання ризику (risk avoidance) – це процес реорганізації проекту таким чином, щоб ризик не міг на нього впливати. Наприклад – відмовитися від передових інструментів, що знову з'явилися, на користь випробуваних, не включати в план ті функції, які вимагають освоєння нових технологій.

Передача ризику – перерозподіл робіт проекту таким чином, щоб хтось іншої (Замовник, партнер і т.п.) відповідав за роботу з ним.

Прийняття ризику зобов'язує Розроблювача «піклуватися» про нього. Заходу щодо *контролю ризику* (risk control) включають планування, дозвіл і моніторинг.

Планування керування ризиком має на увазі створення плану дій для кожного окремого фактору, включаючи методи зм'якшення, плани на випадок непередбачених обставин, відповідальних осіб і строки виконання . Ціль дій по зм'якшенню впливу ризику — або не дозволити ризику стати проблемою, або зменшити його шкідливий вплив.

Деякі ризики можуть бути *дозволені* в процесі роботи над проектом, вони віддаляються зі списку ризиків, інших – навпроти, виявлені в ході виконання проекту й додані в цей документ.

Моніторинг ризиків покликаний здійснювати спостереження над ризиками зі списку, відслідковувати їхнє просування аж до дозволу, працювати з їхніми пріоритетами.

Сучасні тенденції в розвитку АИС і технологій їхнього створення

Індустрія виробництва АИС перетерпіла за 2 останні десятиліття якісні зміни . Це пов'язане з такими тенденціями комп'ютерного миру й світової економіки, як:

- розвитком і появою нової елементної бази,
- настанням епохи персональних комп'ютерів,
- появою й розвитком інтегрованих середовищ розробки,
- появою глобальних мереж передачі даних,
- глобалізацією бізнесу,
- ростом конкуренції,
- переходом до економіки, орієнтованої на споживача,
- появою й розвитком електронного бізнесу й ін.

АІС пройшли шлях від однокористувальницьких систем до систем масштабу робочої групи, малого підприємства, холдингу, транснаціональної корпорації.

Серед сучасних тенденцій у розвитку технологій створення АИС слід зазначити:

- швидкі методи прототипирования,
- орієнтацію на варіанти використання,
- перехід від розробці до настроювання.

Покупне або замовлене ПО - критерії вибору

У світі ІТ існує певна конкуренція між замовленими й покупними системами. Основний аргумент прихильників замовлених систем – «кожний серйозний бізнес унікальний і вимагає власної розробки; універсалізм – синонім бідності ». Аргументи їхніх супротивників – «кількість типових рішень звичайно й невелико. Ті самі організаційні рішення працюють у різних галузях промисловості».

Існує значна кількість аргументів на користь покупних систем, наприклад:

1. ERP-система X розробляється вендором Y уже Z років. За цей час він освоїв ринки найбільших країн Заходу, за тисячею впроваджень, що говорить про високу якість системи.
2. КИСНУВ базується на референтній (еталонної) моделі бізнесу. Дана модель апробована на підприємствах десятків держав і галузей промисловості й крім властиво системи ви (покупець) одержите докладні інструкції про те, як правильно вести бізнес і перемагати в конкурентній боротьбі.

На породження цих аргументів працюють відділи маркетингу таких гігантів цього сегмента ІТ-індустрії, як SAP, Oracle, SAGE, Microsoft, SSA Global і ін. И ці аргументи дійсно й серйозні й переконливі. Проте, незважаючи на потужний пресинг лідерів виробництва КИСНУВ, за даними оглядів, співвідношення замовлених і покупних систем автоматизації підприємств у світі оцінюється, приблизно, як 50:50.

По суті, вибір здійснюється навіть не з 2, а з 3 варіантів:

1) закупити рішення у вендора;
2) замовити ексклюзивне рішення у фірми-виробника прикладного програмного забезпечення.
3) виготовити рішення самостійно.

Основні критерії вибору:

- ціна;
- ступінь унікальності бізнесу компанії;
- рівень сервісного обслуговування. Цінові

питання зводяться до аналізу витрат на:

- закупівлю (або розробку),
- заходу щодо впровадження рішення,
- володіння (включаючи необхідні доробки).

Ступінь унікальності бізнесу компанії характеризується ступенем відбиття особливостей бізнесу компанії в рішеннях, представлених на ринку.

Рівень сервісного обслуговування характеризується наявністю підтримки покупного програмного забезпечення по місцю розміщення підприємства компанії, політикою вендора (розроблювача) в області підтримки, наявністю гарячої лінії й т.п.

На практиці найчастіше використається комбінація 1 і 3, або 2 і 3 варіантів: система закупається, або розробляється на стороні, впроваджується, потім її супровід і розвиток переходить до відділу автоматизації підприємства.

Стратегії вибору рішення

Перед фахівцем ІТ-відділу, або незалежним консультантом, якому доручений вибір рішення в області автоматизації підприємства, коштує нелегка задача. Адже на ринку представлені

сотні рішень в області автоматизації й щоб хоча б швидко ознайомитися з кожним з них може знадобитися кілька людино-літ.

На практиці, звичайно, навряд чи шлях докладного вивчення всіх відомих на ринку рішень можна розглядати всерйоз. У найпростішому випадку розглядаються аналітичні огляди, підготовлені незалежними експертами, оцінюються фінансові можливості підприємства впровадження й на третейський суд інвестора надаються 2-3 рішення.

Щоб зробити вибір обґрунтованим, використаються стратегії вибору рішення. Основні з них:

- аналіз вимог,
- аналіз невідповідності,
- підхід на основі «кращих практик».

Аналіз вимог

Раціональним прийомом, що дозволяє знизити витрати на підготовку варіантів рішення, а заодно знизити ризики, є формування документа вимог (не тільки до знову створюваного, але й до обираного продукту). Тим самим, частина роботи можна перекласти на представників маркетингових відділів вендоров – нехай вони пояснять і продемонструють, як на практиці реалізуються Ваші вимоги.

Практичне застосування методів аналізу вимог, розглянутих протягом лекційного курсу, стосовно до задачі вибору покупного рішення, вимагає відповіді на наступні питання:

- рамки проекту;
- широта аналізу вимог;
- глибина пророблення вимог;
- необхідні ресурси.

Рамки проекту. Які бізнеси-процеси, підрозділи, відділи підприємства необхідно автоматизувати? Практика впровадження ERP-систем показує, що навіть на самих передові в інформаційному змісті підприємствах ступінь охоплення бізнесу інформаційними технологіями рідко перевищує поріг в 90%. Можливо, великомасштабний проект автоматизації варто розбити на кілька етапів, почавши, допустимо , із планки в 30%. Інша стратегія – «всі й відразу» припускає спробу відразу вийти на максимально можливе охоплення функцій.

Широта аналізу вимог. Скільки вимог ви в стані сформулювати протягом розумного проміжку часу? Скільки часу піде у вендоров на аналіз документа вимог, що ви підготуєте? Д. О'Лири в [8] наступний порядок: документ опису вимог для підприємства з річним оборотом в $40 млн., містить близько 1000 позицій (вимог).

Глибина пророблення вимог. Яку вибрати ступінь деталізації вимог? Рівень документа «Концепція» навряд чи виявиться достатнім для серйозного розгляду. Необхідно відбивати як функціональні, та й нефункціональні вимоги. Можна зупинитися на коротких формулюваннях функцій (лаконічно, важко проверяемо). Можна перейти на мову сценаріїв (з'являється можливість доскональної перевірки, але це зажадає значних ресурсів) . Гарний варіант із комбінацією цих способів опису: критичний функціонал описати у формі сценаріїв, інше – у вигляді функцій.

Необхідні ресурси. Ключовий ресурс підприємства – людський ресурс. Чи знайдуть топ-менеджери підприємства час на скрупульозну оцінку вимог до системи? Наскільки добре зможуть сформувати вимоги керівники лінійних підрозділів? Наскільки вдасться задіяти ресурс постачальників рішень? Чи варто залучати зовнішніх консультантів?

ДОакова ціна обстеження? Як швидко вдасться сформувати вимоги? Який час варто виділити на одержання відповідей від вендоров, на аналіз цих відповідей перед остаточним вибором?

Аналіз невідповідності

Аналіз невідповідності при виборі КИСНУВ базується на класичних методах бізнесу-аналізу, предпологающих побудови моделей «як є», «як треба» і перехідної моделі, що показує шлях реформування підприємства.

Нюанс полягає в тім , що в цьому випадку аналіз «як є» застосовується не до бізнесу-системи в чистому виді, а до її комбінації з наявною автоматизованою системою. Він покликаний висвітити слабкі сторони й наявне рішення й пов'язані з ним проблеми бізнесу.

Аналіз «як треба» може вироблятися по одному з наступних сценаріїв: «реінжинірингу чистого аркуша» і «реінжинірингу, що запускає технологією».

У першому випадку використаються класичні підходи до реінжинірингу підприємств у комбінації з методами аналізу й формування вимог.

У другому випадку мова йде про реінжиніринг під конкретне Розв'язанн-вирішенн-ERP-рішення. У цьому випадку основою для перегляду діючий бізнес- процесів і рівня їхні автоматизації є «кращі практики», закладені у відповідній ERP-системі.

Як і в стратегії аналізу вимог, стратегія аналізу невідповідності залишає множину підводних каменів і питань. Допустимо, ми склали модель «треба» і розглядаємо специфікації конкуруючі ERP-системи. Як зіставити ці два документи? Як виміряти ступінь відповідності – по кількості функцій? Де гарантії того, що функції, що називаються однаково в розглянутих документах, *дійсно* однаково працюють?

Альтернативою розглянутим вище стратегіям, що дозволяє знизити ступінь невизначеності прийняття рішень, у певній мері є підхід на основі кращих практик .

Підхід на основі кращих практик

Підхід заснований на відмові від моделей «як є». Пропонується не зацикливаться на існуючим на підприємстві бізнесах-процесах, а запустити процес реінжинірингу, ґрунтуючись на «кращих практиках» впроваджуваного Розв'язанн-вирішенн-ERP-рішення. Основні доводи за застосування даного підходу.

1) Кожний пакет ERP відповідає потребам організації. Обрана ERP-система має приблизно той же набір кращих практик, що й будь-яка інша й всі вони приблизно еквівалентні.

2) Копіювати існуючі процеси не має змісту. Організація повинна здійснювати автоматизацію, ґрунтуючись на «реинжиниринговом потенціалі» ERP-системи.

3) Апріорний аналіз кращих практик не повинен використатися. Аналіз кращих практик повинен здійснюватися в контексті конкретної частини ПО обраної ERP-системи.

4) Треба не «загинати рішення під існуючий бізнес», а, навпроти, реорганізувати бізнес на основі кращих практик так, щоб зміни в ERP-системи були мінімальні. Це дозволить знизити ціну впровадження й ціну володіння ERP-системами.

Процес вибору рішення

Вибір рішення для побудови на підприємстві корпоративної АИС – дуже відповідальний процес, пов'язаний з питаннями захисту інвестицій і виживання на ринку. Значна кількість проектів по впровадженню ERP-систем закінчується провалом, особливо це стосується російської практики. Більше того, на Заході існують прецеденти, коли підприємства, що поставили «на карту ERP» свій добробут і плани, що зв'язали з ними, розвитку, попросту збанкрутували, не впоравшись із задачею «реінжинірингу під ERP».

Тому задачу вибору рішення варто розглядати в рамках проектного й процесного підходів, з усіма наслідками, що випливають звідси.

У формально пророблених процесах впровадження рішень недоліку немає. Як правило, вони розробляються й поставляються вендорами разом із самими рішеннями. Питання організації вибору рішень пророблені в літературі значно гірше.

Нижче розглянутий практичний процес, прийнятий компанією Chesapeake Display &

Packaging при виборі Розв'язанн-вирішенн-ERP-рішення – процес швидкого вибору для швидкого впровадження[30] на базі огляду.

Процес організований досить просто. Він передбачає виконання наступних етапів:

- сформировать команду «вибірників»;
- організувати демонстрацію КИЦЬ постачальниками;
- здійснити попередній відбір;
- здійснити вибір.

Сформувати команду . Команда повинна бути невеликий і ретельно підібраної. Критерії відбору – знання бізнесу й бізнесів-процесів; включення людей з різними поглядами, сполучення узкоспециализированных фахівців і фахівців широкого профілю. Обмеження на розміри команди – не більше 10 чоловік.

Організувати демонстрацію КИЦЬ постачальниками. Вибір обмеженої кількості виробників високоякісних КИЦЬ. Пропозиція обраним виробникам підготувати демонстрацію для команди «вибірників». Обмеження доступу представників виробників до членів команди до моменту демонстрації. Строк підготовки – 3 тижні. Тривалість демонстрації – 2 дні. Воля вибору для вендора встаткування й СУБД.

Здійснити попередній відбір. Постачальникам висловлюються наступні вимоги:

- показати, що КИСНУВ зможе працювати з вашим бізнесом;
- показати, що ви зможете впровадити його протягом необхідного часу;
- показати своє розуміння галузі.

Після проведення демонстрації на основі зазначених вимог здійснюється вибір трійки лідерів.

Здійснити вибір. Вибір здійснюється на основі голосування, більшістю голосів. Голосування виробляється на основі двох факторів:

- найбільш підходящі функціональні можливості;
- кращий персонал по впровадженню.

Кожна з опцій ПО оцінюється в трибальній шкалі (3 – найвищий бал). Оцінки здійснюються членами команди по групах компетенції.

Питання та завдання до змістового модулю № 2

1. Збір інформації про аналоги.
2. Структура таблиць звітності про аналоги.
3. Для чого створюється образ продукту?
4. Яка інформація повинна входити в опис образу продукту?
5. Застосування SWOT аналізу в процесі визначення образу продукту.
6. Управління масштабом складних проектів.
7. Визначення змісту проекту.
8. Які існують способи написання специфікацій вимог до ПЗ?
9. 2. Для чого використовуються діаграми прецедентів?
10. Створіть документ-концепцію «Віртуальної торгової площадки».

Перелік посилань

Основна література

1. Вигерс Карл Разработка требований к программному обеспечению/Пер, с англ. — М.: Издательско-торговый дом «Русская Редакция», 2004. —576с.: ил.
2. Леффингуелл Д., Уидриг Д. Принципы работы с требованиями к программному обеспечению. М.: ИД“Вильямс”, 2002.

Додаткова література

3. Введение в Rational Unified Process/ Ф. Кратчен – СПб.: Вильямс, 2002. – 240 с.
4. Алистер Коберн. Современные методы описания функциональных требований к системам.
5. Орлик С., Булуй Ю. Введение в программную инженерию и управление жизненным циклом ПО Программная инженерия. Программные требования. Copyright © Сергей Орлик, 2004-2005.
6. Унифицированный процесс разработки программного обеспечения/ А. Якобсон, Г. Буч, Дж. Рамбо– СПб.: Питер, 2002. – 496 с.

Інформаційні ресурси

7. http://www.sorlik.ru/swebok/3-1-software_engineering_requirements.pdf
8. http://www.intuit.ru/studies/courses/2188/174/info
9. Каталог електронних ресурсів кафедри (методичні вказівки по виконанню лабораторних робіт, інші навчальні матеріали та література по дисципліні).

Навчально-методична література

10. Методичні вказівки до виконання лабораторних робіт з дисципліни «Аналіз вимог до програмного забезпечення» для студентів спеціальності 6.050103 «Програмна інженерія» / Укладач Я.Є. Кадочнікова - Дніпродзержинськ: ДДТУ, 2015.

11. Методичні вказівки для самостійної роботи з дисципліни «Аналіз вимог до програмного забезпечення» для студентів спеціальності 6.050103 «Програмна інженерія» / Укладач Л.М. Божуха - Дніпродзержинськ: ДДТУ, 2015.
12. Конспект лекцій з дисципліни «Аналіз вимог до програмного забезпечення» для студентів спеціальності 6.050103 «Програмна інженерія» / Укладач Л.М. Божуха - Дніпродзержинськ: ДДТУ, 2015.

Конспект лекцій з дисципліни «Аналіз вимог до програмного забезпечення» для студентів напряму підготовки 6.050103 «Програмна інженерія».

Укладач: Л.М. Божуха, доцент, кандидат фіз.-мат. наук.

51918, м. Дніпродзержинськ, вул.. Дніпробудівська, 2

Підписано до друку ____. ____. 2015 р.

Формат _____ Обсяг _____ д. арк.

Тираж _____ екз.

Замовлення _____