Лекція 8

Діаграмна техніка у роботі із знаннями

Анотація: Випадки використання. Робота з вимогами. Випадки використання в управлінні розробкою. Ітеративний цикл автор/рецензент. Карти пам'яті.

Метод «Випадки використання»

Опис прикладу. Для прикладу розглянемо "Телефонну службу прийому заявок". Замовником даної системи є компанія, що володіє мережею продуктових магазинів. Ця компанія, окрім звичайної роздрібної торгівлі і оптових постачань продуктів окремим їдальням і ресторанам, хоче надавати ще і сервіс із обслуговування клієнтів за телефонними заявками. Клієнт реєструється в компанії, а потім за телефоном, в слушний для себе час, робить замовлення товарів, які привозять йому додому, а він розраховується. Для цього компанія хоче організувати у себе локальний телефонний центр, що складається з офісної багатоканальної АТС, штату операторів і відповідного програмного забезпечення. При цьому в компанії вже існує інформаційна система з обробки заявок від постійних дрібнооптових клієнтів, і система, яка замовляється, повинна бути об’єднана із нею.

Робота з вимогами. Випадки або **варіанти використання** (**use cases**) були запропоновані в кінці 90-х років Айвером Якобсоном, одним з головних авторів мови **UML**, як діаграмний підхід для витягування і первинної формалізації вимог до систем. Раніше вже зазначалось складність формування єдиної і зв'язної картини вимог до ПЗ. Необхідно витягнути вимоги зі всіх можливих джерел, формалізувати у деякому вигляді і обговорити із усіма зацікавленими персонами. Цей процес – витягування, формалізація, обговорення – є ітеративним, тобто все виконується не за один раз. Більш того, сам спосіб формалізації повинен бути зручний для обговорення, і в першу чергу з потенційними користувачами системи, які можуть бути абсолютно не компетентні в ІТ. Їх коментарі, схвалення і незгоди часто є основою ітеративного ***витягування вимог*** до системи. Крім того, цей спосіб роботи із інформацією повинен вести до створення моделей, які були б зручними для подальшої реалізації системи. Іншими словами, ясно формулювати початкові завдання для розробки ПЗ. Тобто, спосіб формалізації повинен бути простим, зрозумілим і володіти достатньою строгістю. Цим вимогам задовольняють ***діаграми випадків використання***, що є на сьогоднішній день складовою частиною стандарту UML.

Приклад *діаграми випадків* використання наведений на Рис. 8.1.

Отже, все починається з точної ***ідентифікації користувачів*** майбутньої системи. Це – основа гарних вимог і гарної системи, адже основне завдання системи – задовольняти потреби майбутніх

користувачів. Для цього слід їх знати в обличчя... У нашому випадку користувачами системи є *оператор*, *менеджер* і представники технічної підтримки та адміністрування. Система повинна також підтримувати зовнішній *інтерфейс* з системою обробки заявок. Це – четвертий користувач. Ще одним користувачем системи є *Петров А.Б. – директор* департаменту збуту товарів, який хоче періодично відслідковувати діяльність телефонної служби прийому заявок. Для нього має бути створено спеціальне призначене місце користувача *з екранними формами статистики*.

Різні користувачі ПЗ, які зображені на діаграмах випадків використання, називаються **акторами (actors)**. Актори можуть позначати:

- **типових користувачів** ("*Менеджер"*, "*Оператор"*, "Технічна підтримка") – працівників компанії, згрупованих по виконуваних обов'язках;
- **інші системи**, що взаємодіють з даною ("Система обробки заявок");
- **виділеного користувача** ("Петров А.Б.").

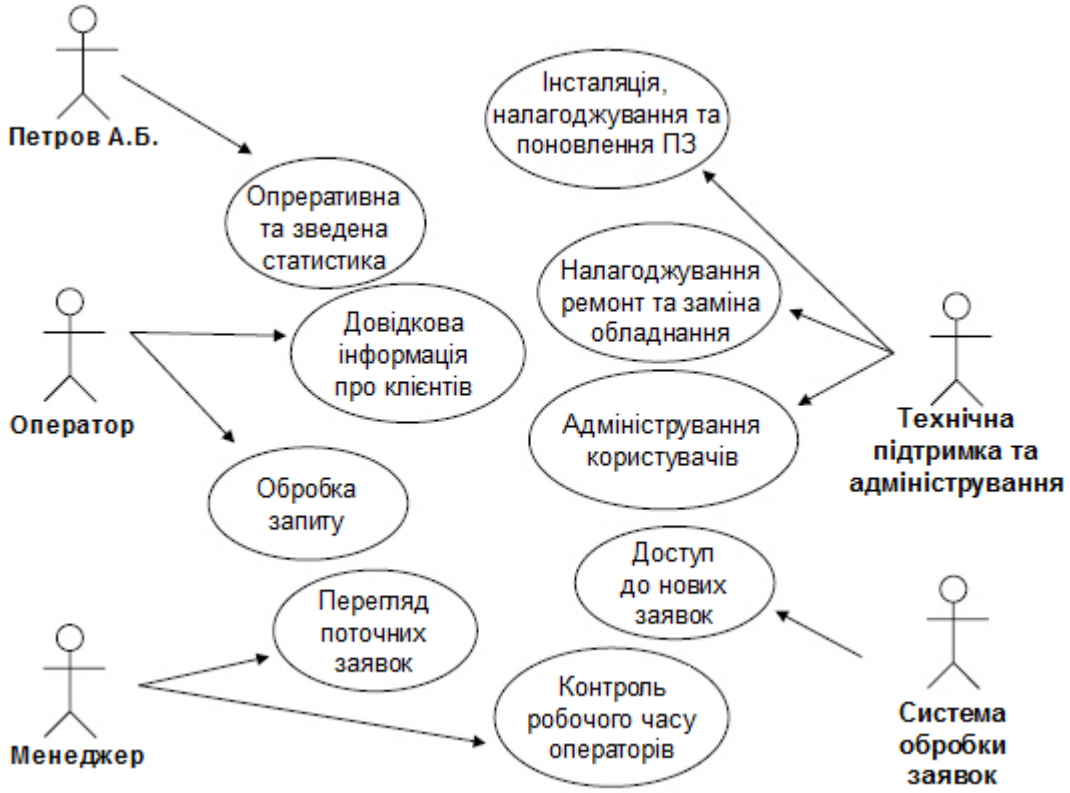

Рис. 8.1. Приклад діаграми випадків використання

Відзначимо, що ***виділений користувач*** істотно відрізняється від ***типового користувача***. Він, як правило, *Важлива Персона*, і узгодження функціональності для нього відбувається з ним особисто. Часто він впливає на фінансування проекту, від його думки про систему, багато в чому, залежить її успішна здача. Заради успіху проекту слід

вміти ідентифікувати **ТАКІ** персони, у межах всієї системи спеціально для них створювати деяку функціональність і дуже при цьому старатися!

Після ***ідентифікації користувачів*** відбувається ***визначення випадків використання*** системи цими користувачами. Перш за все, визначається та функціональність системи, яка безпосередньо допомагає користувачам виконувати їхню роботу, ***яка не пов'язана безпосередньо із експлуатацією системи***. У нашому випадку, для **оператора** важливим плюсом від використання системи виявляється **можливість отримувати швидкий доступ до довідкової інформації про клієнтів**, а також **оперативно обробляти запити**, що надійшли за телефоном на придбання (список товарів, ціни, оформлення замовлення і ін.). Для **менеджера** важливою є **можливість оперативного перегляду поточних запитів** (які були виконані, в роботі, відкладені, за певний період часу і ін.), а також **облік контроль робочого часу операторів** – хто і скільки часу витратив на роботи різного виду (телефонні розмови з клієнтами, оформлення заявки після закінчення розмови і так далі). При цьому важливо відзначити, що функція обліку робочого часу може зажадати певних дій з боку операторів – наприклад, натискати певну клавішу, у разі перерви на обід або на перекур. Проте, ми не позначили відповідний зв'язок з цим випадком використання з боку оператора, оскільки ця функціональність не допомагає йому в безпосередній роботі, а допомагає його начальникові. Крім того, ми не включили у випадки використання ряд сервісів, які пов'язані з експлуатацією системи, наприклад, функцію логіна в систему. Наявність чіткої точки зору під час складання діаграм – запорука їх корисності.

Отже, **випадок використання** (**use case**) – це незалежна частина функціональності системи, що має результуючу цінність для її користувачів.

"*Незалежність"* означає, що якщо випадок використання завжди виконується разом з деяким іншим випадком використання, то, мабуть, один з них слід включити в інший (який саме в який, як назвати випадок використання, що у результаті був створений, – залежить від обставин).

"*Результуюча цінність*" випадку використання для актора системи означає, що він, даний випадок використання, повинен приносити акторові деякий закінчений і цінний з погляду його бізнесу результат. Будучи реалізований системою, цей випадок використання дійсно робить бізнес актора більш ефективним, продуктивним. Тим самим ***розробка системи фокусується на бізнес-цілях***, а незначні випадки використання ігноруються, що є важливим для компактності моделі. Адже будується не абстрактна модель функцій системи, а ***набір найважливіших*** (для замовника і користувачів) ***сервісів***, щоб кожен з цих сервісів вірно зрозуміти і жоден не упустити. Надалі **контроль розробки системи** здійснюватиметься саме в термінах саме цього найважливішого – того, що конче потрібне замовникові і користувачам.

Випадки використання, що відповідають акторам "***Технічна підтримка і адміністрування***" і "***Служба обробки заявок***" дещо не відповідають наданому вище визначенню. Перш за все, самі ці актори не є користувачами ПЗ, які беруть участь в основному бізнес-процесі обробки телефонних заявок. "*Технічна підтримка і адміністрування*" зайнята підтримкою ПЗ і устаткування системи обслуговування телефонних заявок, а також її адмініструванням (додаванням нових користувачів, призначенням ним відповідних прав і ін.). "*Служба обробки заявок*" є інформаційною системою, яка вже існує в компанії, має базу даних і ряд сервісів щодо обробки заявок. ***Ідентифікація цих акторів і відповідних ним випадків використання важлива з погляду визначення вимог до системи.*** Для представників служби технічної підтримки необхідний спеціальний зручний інтерфейс з набором відповідних функцій. А всі заявки, що надійшли з телефону, повинні потрапити в єдину базу даних заявок і пройти єдиний цикл обробки. Втрата цих чинників може призвести до серйозних недоліків і проблем. Крім того, вони ні звідки безпосередньо не слідують і тому потребують особливих початкових вершин в дереві вимог – тобто саме ми вирішили, що їх доцільно помістити на головну діаграму випадків використання.

Відзначимо ще одну цікаву деталь. Клієнт магазина не є користувачем даного ПЗ. Він виявляється бізнес-користувачем всієї системи в цілому (включаючи відповідний бізнес-процес і устаткування). На Рис. 8.2 представлена бізнес-діаграма випадків використання.

Її можна малювати окремо (класики на цьому наполягають), але можна примальовувати клієнта і на загальну діаграму, якщо зв'язати стрілкою з оператором. Часто буває, що такий запис є не дуже концептуальним, але зручним і компактним.

Кожен випадок використання супроводжується невеликим текстовим описом, а надалі може містити цілі розділи в технічному завданні. **Діаграми випадків використання** можуть служити структурою для **технічного завдання** або його окремих частин.

Інші версії. На практиці *діаграми випадків* використання створюються не лише у такий спосіб, як вказано вище. Багато практик вважають за краще будувати дуже детальні моделі, промальовуючи на них всі невеликі випадки використання, а також численні зв'язки між ними (використання, розширення і так далі). Хтось рішуче протестує проти включення в актори системи, які взаємодіють з даною. Інші вважають неприйнятним суміщати звичайні діаграми і бізнес-діаграми випадків використання і так далі. Яку саме стратегію ви виберете в конкретному випадку, яку точку зору поставите основною задачею – вам вирішувати самим. Рецепту тут немає.

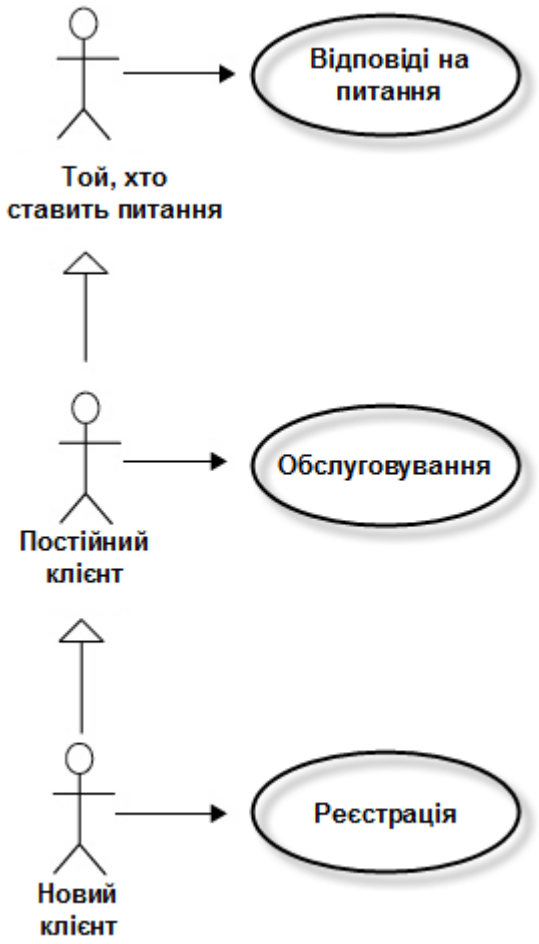

Рис. 8.2. Приклад діаграми бізнес-випадків використання

Важливо лише відзначити, що завжди потрібна правильна певна точка зору. Вона дозволяє чітко сфокусуватися, вирішувати конкретні, добре усвідомлювані завдання. А також таку точку зору подекуди можна усвідомлено, на догоду практичній корисності, порушувати.

Випадки використання в управлінні розробкою. Отже, вище ми показали, як діаграми випадків використання можуть бути корисними підчас виявлення первинної формалізації вимог. Але вони можуть виявитися корисними і після того, як цей процес завершений. Результуючі діаграми випадків використання можна використовувати під час управління розробкою. Менеджер проекту може відстежувати прогрес реалізації проекту за тим, скільки реалізовано функціональності, яка є необхідною користувачеві. Розробники можуть мати діаграми випадків використання десь перед очима, щоб не забувати про основну мету розробки. Ці самі діаграми можуть використовуватися в робочих зустрічах із проекту.

Здавалося б, що може бути простіше – реалізувати набір функцій, які необхідні користувачеві. Проте на ділі програмний проект може непомітно втратити цю мету. Замість цього можна, наприклад, дуже довго займатися розробкою складної і багатофункціональної архітектури, після реалізації якої розробники обіцяють, що всі призначені для

користувача функції будуть отримані майже відразу і дуже легко. Проте, як правило, виявляється, що це "відразу" було сильним перебільшенням і проект дуже вибивається з графіку, а багато замовлених користувачем функцій в цьому оточенні зробити важко або неможливо. Трапляється, що надмірна орієнтація на "внутрішню досконалість ПЗ" закінчується для проекту або великими проблемами, або повним крахом. Проте, бувають і інші випадки, коли лише така орієнтація згодом і рятує проект. Останнє трапляється, коли система довго розвивається і супроводжується, або коли вимоги до неї раптово і сильно змінюються, або коли на її основі створюються інші системи. Необхідний баланс між внутрішньою досконалістю програмного забезпечення і функціональністю, яка потрібна замовнику. При цьому система має бути доставленою йому вчасно. Розробка ПЗ в термінах випадків використання – гарний спосіб контролювати, чи рухається у потрібному напрямі процес створення системи.

Ітеративний цикл автор/рецензент

Опишемо одну цікаву і вкрай корисну техніку використання візуального моделювання під час видобування знань про яку-небудь предметну область через спілкування з експертами (фахівцями в цій предметній області). Ця техніка називається **цикл автор/рецензент** (Reader/Author Cycle review process) і може застосовуватися, наприклад, підчас роботи з діаграмами випадків використання у разі роботи як з UML, так і з будь-якою іншою мовою візуального моделювання. Ця техніка була визначена в межах методології SADT.

Активний співробітник – автор візуальних моделей (author), – вивчає не дуже знайому йому предметну область. При цьому авторові постійно потрібний зворотний зв'язок з експертами в цій предметній області для того, щоб він усвідомлював, наскільки вірно він зрозумів і адекватно формалізував той або інший аспект знань, що вивчався.

Як така галузь знань може виступати предметна область, для якої створюється інформаційна система. При розробці інформаційної системи її автори повинні добре розібратися в даній предметній області. Якщо майбутні користувачі або замовник системи не мали можливості детально ознайомитися з тим, як розробники зрозуміли і інтерпретували їх предметна область, то це неодмінно призведе до створення незатребуваної системи: дані будуть невірні або їх не вистачатиме, формати звітів будуть незручними і так далі.

Отже, для того, щоб створити адекватний опис системи, необхідно своєчасно отримувати оцінку створюваних моделей від тих людей, які врешті-решт будуть нею користуватися. Для цього вводяться такі ролі:

- **автор (author) моделі** – той, хто її створює;
- **експерт (commenter)** – це фахівець в тій наочній галузі, для якої будується дана модель; автор інтерв'ює експерта, отримує

необхідну для моделювання інформацію; експерт переглядає і коментує створені автором діаграми; важливо, що експерт **надає свої коментарі письмово** і розділяє з автором відповідальність за якість створюваних моделей; експерт може бути також архітектором системи, який активно бере участь в процесі розробки моделі аналізу, – але не як автор моделей (у нього вистачає інших турбот), а як активний критик (підчас розробки архітектури системи він активно використовуватиме цю модель);

· **читач (reader)** – у всьому схожий на експерта, але не зобов'язаний давати письмові коментарі до моделей і не несе відповідальності за якість моделювання.

Отримавши діаграми автора, експерт їх ретельно переглядає і пише свої коментарі (прямо на діаграмі, у вигляді приміток, червоною ручкою). Автор, після отримання своїх діаграм з коментарями, зобов'язаний відреагувати на кожне зауваження – помітити синьою ручкою на тій самій копії, чи приймає він зауваження чи ні. Прийняті зауваження він враховує в наступній версії діаграм, а відкинуті зауваження надсилає назад експертові з мотивуванням. У разі виникнення непорозуміння організовується зустріч автора і експерта, на якій вони залагоджують всі непорозуміння.

Окрім автора, експерта і читача в циклі "читач/автор" є також такі ролі:

· **бібліотекар (librarian)** – це головний координатор процесу моделювання; він стежить за тим, щоб всі учасники процесу вчасно отримували свіжі копії моделей, щоб ці копії не втрачалися і вчасно потрапляли в архів, а останній був би доступний; у його компетенцію входить також відстежувати, що всі зауваження експертів і читачів оброблені автором, не залишені без уваги; раніше, коли метод SADT лише з'явився, роль бібліотекаря була велика – моделі будувалися на папері; тепер же для цього використовують різні графічні пакети, а для зберігання різних версій моделі – програмні засоби управління версіями;

· **комітет технічного контролю (technical review committee)** – це група людей, яка стежить за тим, наскільки процес моделювання відповідає цілям проекту, чи буде можливість використовувати в подальшій роботі створювані діаграми; цей комітет стежить також за тим, коли моделювання потрібно завершити; адже час людей може коштувати істотних грошей, у проекту є терміни, а процес моделювання може продовжуватися дуже довго – наприклад, автор може захопитися генерацією ідей під час вивчення нової наочної галузі.

Слід відмітити, що цикл "читач/автор" може використовуватися в різних ситуаціях, коли необхідно ефективно витягувати інформацію з експертів деякої наочної області. Наприклад, така ситуація може скластися, коли технічний письменник створює документацію про

програмне забезпечення, або тестувальник вивчає систему для того, щоб ефективно її тестувати, або новий менеджер проекту вивчає систему, яка вже давно розробляється і створенням якої йому потрібно буде керувати, і так далі.

Крім того, цикл "автор/рецензент" може бути використаний і поза контекстом витягання знань, коли ми, під час створення візуальних моделей, хочемо отримувати регулярний і впорядкований зворотний зв'язок.

Різноманітність виробничих контекстів, де може застосовуватися дана техніка, а також особливості людських і організаційних відносин, призводять до того, що цикл "автор/рецензент" на практиці вимагає адаптації. Для його ефективного використання необхідне "тонке налагоджування" на особливості конкретної ситуації.

Зокрема, можуть варіюватися відповідальності різних ролей. Наприклад, експерт може відповідати за процес моделювання, або зовсім за нього не відповідати (вся відповідальність лежить на авторові). Само спілкування автора і експерта також може бути організоване по-різному. Наприклад, на відміну від приведених вище рекомендацій, експерт може висловлюватися лише усно, підчас особистих зустрічей з автором. На одній зустрічі експерт видає інформацію, на іншій перевіряє те, як вийшло у автора її формалізувати. Цей варіант представлений в прикладах нижче.

На **Рис. 8.3** показана початкова діаграма, яку намалював автор для першої зустрічі з експертом. На ній присутні лише інтерфейси з діаграм компонент UML і коментарі до них. На **Рис. 8.4** представлена завершальна діаграма, що вийшла у результаті численних ітерацій. На ній присутні численні компоненти системи, які згруповані за рівнями.

Карти пам'яті

Карти пам'яті (Mind Maps) – техніка роботи з різними знаннями, яка була запропонована і розвинена англійським психологом Тоні Бьюзеном в кінці 70-х років минулого століття. Вона дуже проста і використовується під час роботи з інформацією будь-якого вигляду для її структуризації, осмислення, кращого засвоєння і запам'ятовування. На листі паперу, в центрі, малюється ***об'єкт***, що позначає ту тему або предмет, який ми розглядаємо. Далі малюються вторинні об'єкти, які пояснюють і уточнюють даний і з'єднуються з ним *дугами*. І так далі. Приклад наведений на **Рис. 8.5.**

Рис. 8.3. Приклад початкової, першої діаграми.

Дизайн ідей. Карти пам'яті дозволяють виконати "**дизайн ідей**". Дуже часто ми, як слід не подумавши, починаємо щось робити – писати великий текст, з кимось зустрічатися, кимось керувати і ін. І виявляється, що розуміння підчас справи виникає важко і болісно. Більш того, ми вимушені переробляти те, що вже зробили без цього розуміння. Гарна ілюстрація – робота над текстом (диплома, курсової роботи, статті, книги і ін.). Кардинально переробляти текст дуже важко. А якщо при цьому співавторів декілька? Карти пам'яті тут дуже добре працюють, оскільки дозволяють в компактному вигляді виконувати спроби і усувати помилки, якщо бачити всю картину перед собою. Її легко також обговорювати у такому вигляді з іншими людьми. Але тут не потрібно впадати до фанатизму. Можна і написати текст, якщо він легко "виливається" з вас. І знову повернутися до схеми – багато що на ній може прояснитися.

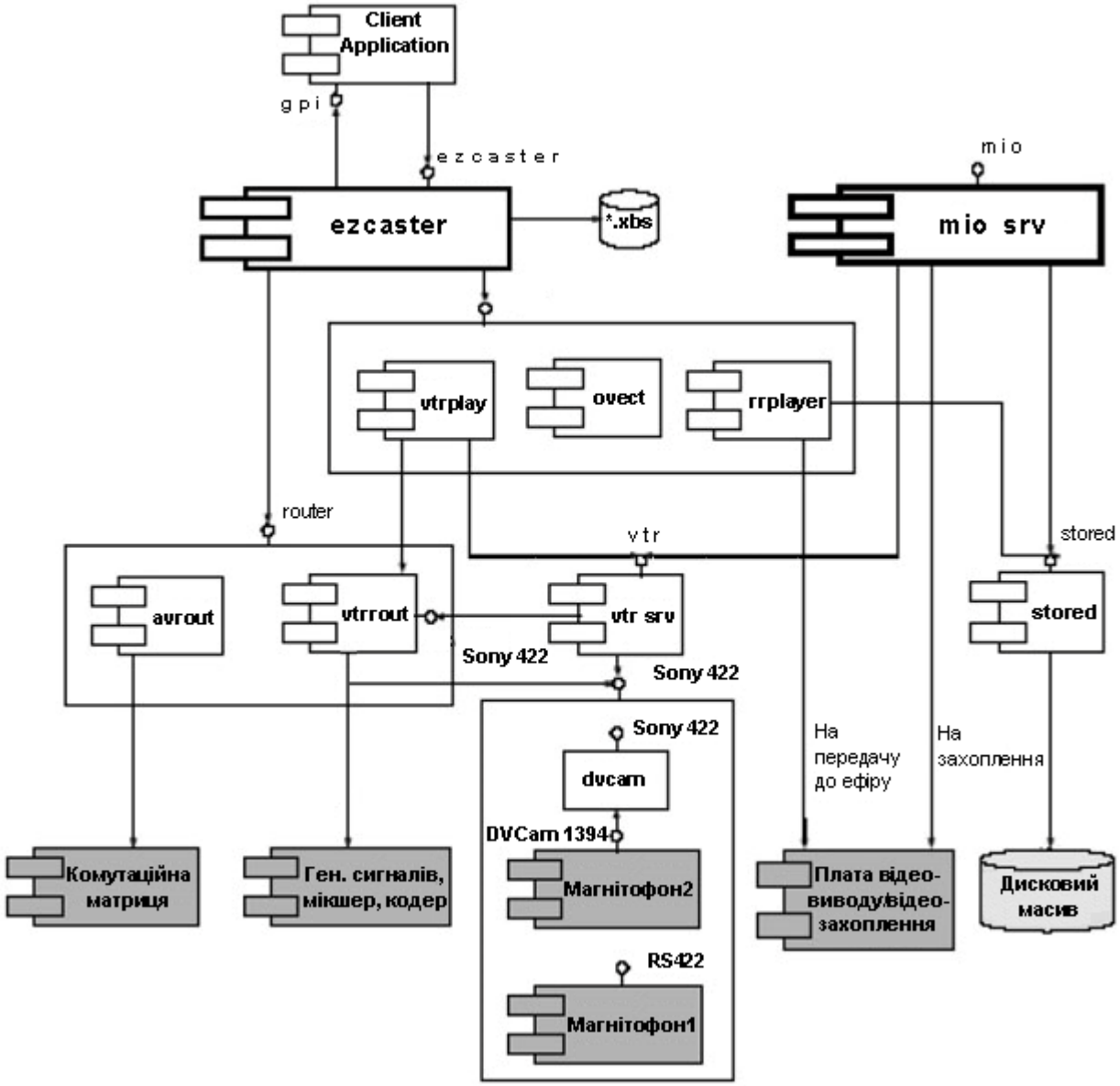

Рис. 8.4. Приклад підсумкової діаграми.

Планування детальної інформації. Метод дозволяє також виконати детальне планування великого об'єму інформації, що має величезну кількість важливих деталей. Наприклад, ми використовували карти пам'яті при проектуванні анкети – вона містила достатню кількість роз*галужень*, списки питань і ін. Все це в загальному вигляді, скорочено, було не представити, карти пам'яті, підтримані програмним інструментом Comapping1 нам тут дуже допомогли. Приклад наведений на **Рис. 8.6**.

Реструктуризація. Карти пам'яті корисні при реструктуризації знань. Наприклад, при реструктуризації статті. У нас був випадок, коли результати були отримані, матеріал зібраний і викладений, але достатньо хаотично. Ми виконали реструктуризацію статті за допомогою карт пам'яті (модель представлена на Рис. 8.7 і за цією моделлю швидко переписали текст. Виправлення безпосередньо за текстом затягнули б весь процес. Крім того, карти пам'яті дозволили розділити роботу між

співавторами – один створив новий план, а другий його реалізував в новій версії тексту.

Рис. 8.5.Приклад використання карт пам’яті

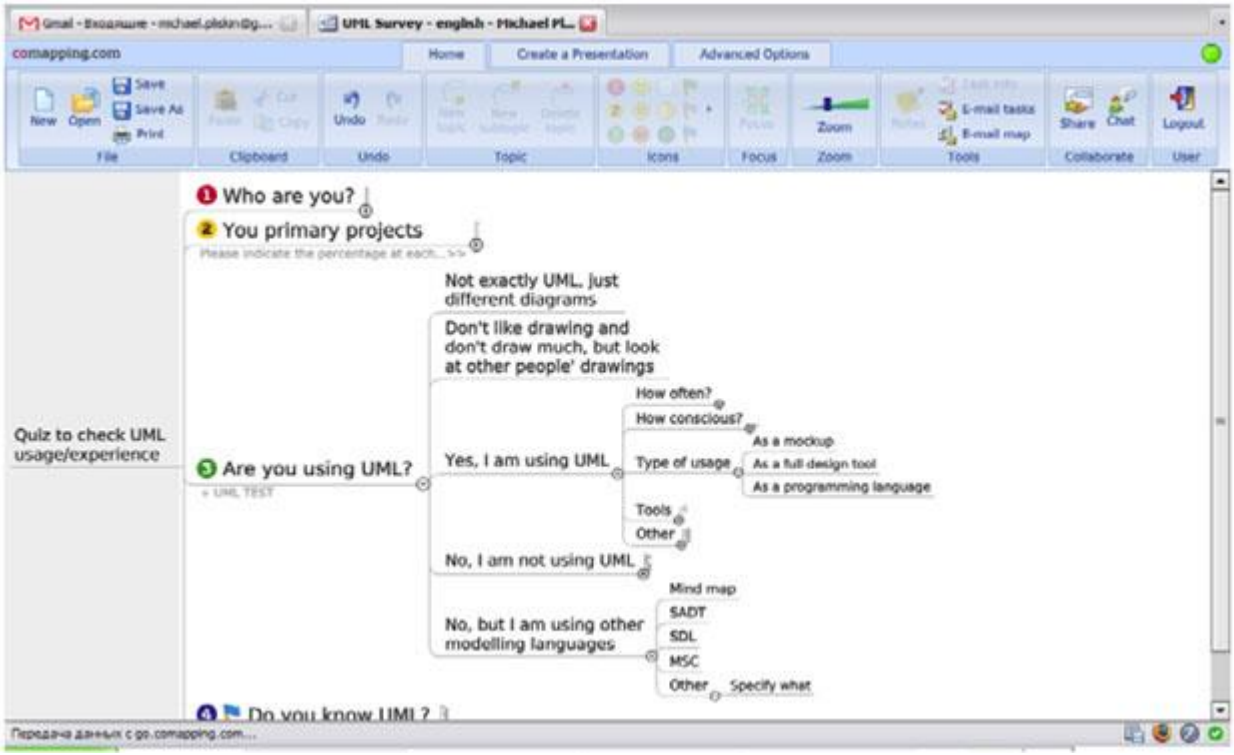

Рис. 8.6.

Метод реструктуризації широко використовується підчас роботи із знаннями, наприклад, у випадку виявлення та аналізу вимог. Побудувати подання тієї ж самої інформації, але з іншої точки зору – вірний спосіб знайти непомічені раніше суперечності, "темні кути", поглибити своє розуміння.

Робота з короткостроковою пам'яттю. Часто буває, що після прослуховування лекції, ми якийсь час пам'ятаємо її зміст (звичайно декілька днів), але *після* декількох місяців її зміст начисто випаровується з мозку. Так от, можна відразу, доки залізо ще є гарячим, зробити собі нарис основних аспектів із використанням карт пам'яті. Студенти говорять, що знайшовши потім такі конспекти-нагадування, вони швидко відновлюють учбовий матеріал в пам'яті. Це доцільно робити після лекції, коли цілісне враження від інформації ще є свіжим.

Колективна робота і продукт Comapping. Одна із головних переваг діаграм полягає в тому, що їх можна обговорювати з широким колом людей. Тест, наприклад, обговорювати важче – його потрібно спочатку проробити. А діаграму можна тут же дивитися і обговорювати. І виправляти. Більш того, за допомогою діаграм можна організовувати ефективні групові територіально розподілені процеси роботи із інформацією: планування, створення текстів, обмін результатами бесід, спілкування викладача і студента та ін.

Розповімо про один *програмний продукт*, який реалізовує карти пам'яті і підтримує широкі можливості групової роботи.

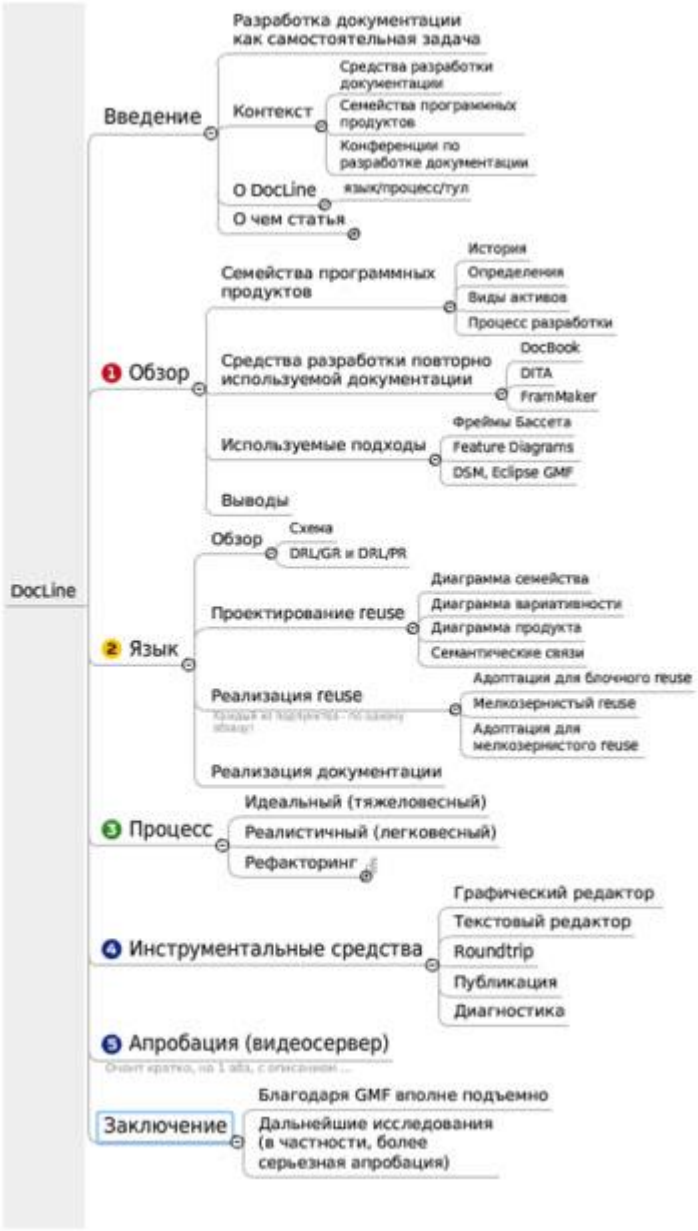
DocLine
Введение
Разработка документации как самостоятельная задача
Контекст
Средства разработки документации
Семейства программных продуктов
Конференции по разработке документации
О DocLine
язык/процесс/тул
О чем статья
1 Обзор
Семейства программных продуктов
История
Определения
Виды активов
Процесс разработки
Средства разработки повторно используемой документации
DocBook
DITA
FramMaker
Используемые подходы
Фреймы Бассета
Feature Diagrams
DSM, Eclipse GMF
Выводы
2 Язык
Обзор
Схема
DRL/GR и DRL/PR
Проектирование reuse
Диаграмма семейства
Диаграмма вариативности
Диаграмма продукта
Семантические связи
Реализация reuse
Адоптация для блочного reuse
Мелкозернистый reuse
Адоптация для мелкозернистого reuse
Реализация документации
3 Процесс
Идеальный (тяжеловесный)
Реалистичный (легковесный)
Рефакторинг
4 Инструментальные средства
Графический редактор
Текстовый редактор
Roundtrip
Публикация
Диагностика
5 Апробация (видеосервер)
Заключение
Благодаря GMF вполне подъемно
Дальнейшие исследования (в частности, более серьезная апробация)

Рис. 8.7.