ТЕМА 3. Виявлення вимог до програмної системи. Робота з замовником

Обстеження системи, спілкування з замовником, планування розробки, складання технічного завдання. Детальний аналіз предметної області, прийняття остаточного рішення про необхідність створення інформаційної системи, проектування загальної архітектури системи, вибір методу проектування.

Одним з найбільш важливих та зрозумілих методів одержання вимог є інтерв'ю з клієнтом; цей метод можна використовувати практично в будь-якій ситуації. Одна з основних задач інтерв'ювання – зробити все можливе, щоб упередження та переваги інтерв'юованих не впливали на вільний обмін інформацією. Це складна проблема. Соціологія показує, що неможливо сприймати оточуючий світ, не фільтруючи його відповідно до свого походження та накопичення досвіду.

Наступним методом визначення вимог є «мозковий штурм». Це метод проведення зібрання, при якому група людей намагається знайти рішення специфічної проблеми за допомогою накопичення всіх спонтанних ідей. Даний процес має ряд очевидних переваг:

1) підтримує участь всіх присутніх;
2) дозволяє учасникам розвивати ідеї один одного;
3) секретар веде запис всього ходу обговорення;
4) метод є застосовним за різних обставин;
5) як правило, в результаті використання цього методу можна одержати множину можливих рішень для будь-якої поставленої проблеми;
6) метод сприяє вільному мисленню, необмеженому звичайними рамками.

«Мозковий штурм» складається з двох фаз: генерація ідей та їх відбір. Основна мета на етапі генерації полягає в описі якомога більшої кількості ідей, не обов'язково глибоких. На етапі відбору головною задачею є аналіз всіх ідей. Відбір ідей включає відсікання, організацію, впорядкування, розвиток, групування, уточнення і т.і.

Наступним методом є сумісне розроблення додатків (JAD – Joint Application Design). Це зареєстрований товарний знак, який належить до компанії IBM. Він представляє собою груповий підхід до визначення вимог, при реалізації якого особлива увага приділяється вдосконаленною групового процесу та вірному підбору людей, залучених до роботи над проектом. Сеанси JAD аналогічні сеансам «мозкового штурму», однак не у всьому. Сеанси «мозкового штурму» тривають близько двох годин, а сеанси JAD – до трьох днів. На сеансах «мозкового штурму» відбувається швидке генерування ідей, а на сеансах JAD розробляються високо рівневі специфічні програмні моделі функцій, даних та ліній поведінки. Сеанс JAD має певну структуру, на ньому дотримуються певної дисципліни, він відбувається під керівництвом арбітра. В його основі лежить обмін інформацією з використанням документації, фіксованих вимог та правил роботи.

З моменту появи методики JAD на сеансах використовуються CASEінструменти та інші програмні засоби, призначені для побудови діаграм потоку даних DFD, діаграм взаємозв'язків між сутностями ERD, діаграм зміни станів та інших об'єктно-орієнтованих діаграм.

Наступний метод визначення вимог – розкадрування. Метою розкадрування є одержання ранньої реакції користувачів на запропоновані концепції додатку. За її допомогою можна на найбільш ранніх етапах ЖЦ спостерігати реакцію користувачів, до того як концепції будуть перетворені в код, а в багатьох випадках навіть до розроблення детальної специфікації. Розкадрування має наступні переваги: невисока вартість, дружність користувачу, інтерактивність, неформальність, забезпечення раннього аналізу інтерфейс ів користувача, легкість у створенні та модифікованість. Розкадрування можна використовувати для прискорення концептуального розроблення різних сторін додатку.

Вибір типу розкадрування залежить від складності системи і того, наскільки великий ризик, що колектив невірно розуміє її призначення. Для безпрецедентної нової системи, яка має

розпливчасте визначення, може знадобитись декілька розкадрувань, від пасивної до інтерактивної, по мірі вдосконалення колективом розуміння системи.

Ще одним методом визначення вимог є метод обігрування ролей, який дозволяє колективу розробників відчути весь світ користувача після перебування в його ролі. Концпеція, яка лежить в основі даного методу, досить проста: хоча, спостерігаючи і задаючи питання, ми підвищуємо рівень свого розуміння, наївно вважати, що за допомогою самого спостереження розробник/аналітик може одержати істинно глибоке розуміння вирішуваної проблеми або вимог до системи, яка покликана дану проблему вирішити.

Наступним методом визначення вимог є швидке прототипування. Швидке прототипування – це часткова реалізація системи ПЗ, створена з метою допомогти розробникам, користувачам та клієнтам краще зрозуміти вимоги до системи. Чим детальніше прототип, тим легше зрозуміти вимоги замонивка. З іншого боку, прототипи самі по собі є програмами, тому, чим детальнішим є прототип, тим він дорожчий.

Формулювання вимог до ПЗ є однією з найважливіших фаз розроблення ПЗ, оскільки визначає успіх всього проекту. Дана фаза включає етапи:

1) планування робіт, яке попереджає роботи над проектом.

Основні задачі:

- визначення цілей розробки; - попередня економічна оцінка проекту;
- побудова план-графіку виконання робіт;
- створення та навчання сумісної робочої групи;

2) проведення обстеження діяльності автоматизваного об’єкта (організації), в межах якого здійснюється:

- попереднє виявлення вимог до майбутньої системи;
- визначення структури організації;
- визначення переліку цільових функцій організації;
- аналіз розподілу функцій за підрозділами та співробітниками;
- виявлення функційних взаємодій між підрозділами, інформаційних потоків всередині підрозділу і між ними, зовнішніх по відношенню до організації об’єктів та зовнішніх інформаційних взаємодій;
- аналіз існуючих засобів автоматизації діяльності;

3) побудова моделей діяльності організацій, яка передбачає обробку матеріалів обстеження та побудову двох видів моделей:

- модель “AS-IS” («як є»), яка відбиває існуючий на момент обстеження стан справ в організації та дозволяє зрозуміти, яким чином функціонує дана організація, а також виявити «вузькі» місця і сформулювати пропозиції щодо покращення ситуації;
- модель “TO-BE” («як має бути»), яка відбиває уявлення про нові технології роботи організації.

Кожна модель включає в себе повну функцій ну та інформаційну модель діяльності організації, а також, у випадку необхідності, модель, яка описує динаміку поведінки організації. Перехід від моделі “AS-IS” до моделі “TO-BE” може виконуватись двома способами:

1) вдосконаленням існуючих технологій на основі оцінювання їх ефективності;

2)радикальною зміною технологій та перепроектуванням бізнеспроцесів (реінжиніринг бізнес-процесів).

Побудовані моделі мають самостійне практичне значення. Наприклад, модель “AS-IS” дозволяє виявити «вузькі» місця в існуючих технологіях та пропонувати рекомендації по вирішенню проблем незалежно від того, пропонується на даному етапі подальше розроблення ПЗ чи ні. Крім

того, модель полегшує навчання співробітників конкретним напрямкам діяльності організації за рахунок використання наочних діаграм («один малюнок вартий тисячі слів»).

Формулювання цілей проекту – під час постановки цілей програмного проекту варто розрізняти мету – створення ПЗ і мету, заради досягнення якої створюється ПЗ. Дуже рідко ПЗ створюється «заради самого себе». Як виключення можна вказати лише навчальні проекти, коли ПЗ розробляється заради демонстрації можливостей розробників, або ж академічні проекти, коли ПЗ розробляється заради доказу принципової можливості розроблення. Але в більшості випадків програмні продукти розробляються з метою наступного використання, тому дуже важливо не випустити з поля зору основну мету розроблення, яка формулюється замовниками і користувачами ПЗ, і не замінити цю основну мету власними цілями розробників ПЗ. Сучасне визначення якості ПЗ звучить наступним чином: «якість – це ступінь задоволеності користувачів, або ступінь відповідності ПЗ висунутим до нього вимогам користувачів». Якщо цілі проекту не відповідають потребам користувачів, то програмний продукт неможливо буде визнати якісним, навіть якщо при його розробленні були використані сучасні технології та були задіяні найкваліфікованіші розробники.

Цілі проекту повинні бути описані явно та задовольняти певним критеріям: 1) конкретність – мета проекту повинна бути конкретною, вираженою в термінах проекту, з вказанням умов та термінів.

Наприклад, мета «досягти максимального ступеня задоволеності користувачів продукту» не конкретна. В той час, як мета «одержати позитивний відгук на продукт від 90% користувачів за перші півроку продажів» досить конкретна для проекту по розробленню «коробкового» продукту; 2) реалістичність – мета повинна бути реалістичною, тобто досягенною з вархуванням наявних ресурсів. Наприклад, мета «розробити програму автоматичного художнього перекладу поезії з англійської мови на російську за рік» нереалістична для компанії середнього розміру. В той час, як мета «розробити програму підрядкового перекладу технічного тексту з англійської мови на російську за рік» може бути цілком реалістична, якщо є досвід та певні напрацювання;

3) вимірюваність – повинен бути вказаний ефективний критерій, який дозволяв би визначити, чи досягнуто мету і в якому ступені. Наприклад, мета «розробити програму, яка істотно підвищує ефективність процесу продажів» не вимірювана: неясно, як вимірюється ефективність і яке підвищення можна вважати істотною. В той час, як мета «розробити програму, яка прискорює процес оформлення заявки в середньому на 10%» вимірювана;

4) несуперечливість – цілі не повинні бути внутрішньо суперечливими і взаємно виключаючими одна одну. Наприклад, цілі «розробити продукт з максимальним набором функцій» та «витратити мінімальну кількість ресурсів на розроблення» суперечать одна одній (і до того ж, неконкретні).

Мінімальна кількість ресурсів (нуль) буде витрачено, якщо розроблення не проводити, але тоді й можливостей у продукта не буде. В той час, як мета «розробити продукт, який має не менше функцій, ніж конкурент X, і витратити на розроблення не більш Y людиногодин» не є суперечливою (але може виявитись нереалістичною).

Сформульовані цілі проекту необхідно підтвердити, тобто заручитись явною згодою з цілями всіх сторін, зацікавлених в успішності проекту. Сторонами, зацікавленими в успішності проекту, можуть бути:

1) розробники, які виконують проект;

2)інвестори, які його фінансують;

3) користувачі, які будуть застосовувати результати на практиці. Зацікавлені сторони проекту – всі особи і організації, які прямо або непрямо приймають участь в проекті та зацікавлені в його успішності. Ясно, що цілі, переслідувані сторонами, можуть бути досить різні. Наприклад, розробники можуть переслідувати мету засвоєння нової технології при виконанні проекту. Інвестори, швидше за все, зацікавлені у поверненні коштів та одержанні прибутку. Користувачі сподіваються, що розроблюване ПЗ зробить їх працю більш продуктивною. Підтвердження цілей проекту – це пошук компромісу, який задовольняв би всі зацікавлені сторони. В процесі виконання проекту його цілі можуть змінитись. Причини зміни цілей різноманітні і навряд чи можуть бути враховані наперед.

Наприклад, причинами змін цілей можуть бути:

1) зміни в бізнес-процесі, для автоматизації якого розробляєтьсяПЗ;

2) зміни кон'юнктури ринку, зокрема, поява нового конкуруючого продукту;

3) поява нової інформаційної технології,яка впливає на цільову предметну галузь.

В процесі виконання проекту постійно повинен проводитись моніторинг цілей для перевірки їх актуальності. У випадку необхідності цілі слід переглядати. Іноді перегляд цілей тягне за собою перегляд плану всього проекту.

Аналіз прикладної галузі. Щоб розробити програмну систему, яка приносить реальні вигоди певним користувачам, необхідно спочатку з'ясувати, які ж задачі вона повинна вирішувати для цих людей і які властивості мати.

Вимоги до ПЗ визначають, які властивості та характеристики воно повинно мати для задоволення потреб користувачів та інших зацікавлених осіб. Однак сформулювати вимоги до складної системи не так просто. В більшості випадків майбутні користувачі можуть перерахувати набір властивостей, який вони хотіли б бачити, але ніхто не дасть жодних гарантій, що це – вичерпний список. Крім того, часто саме формулювання цих властивостей буде незрозумілим більшості програмістів. Щоб ПЗ було дійсно корисним, важливо, щоб воно задовольняло реальні потреби людей та організацій, які часто відрізняються від безпосередньо виражених користувачами побажань. Для виявлення цих потреб, а також для виявлення змісту висказаних вимог доводиться проводити досить велику додаткову роботу, яка називається аналізом предметної галузі або бізнесмоделюванням, якщо мова йде про потреби організації.

В результаті цієї діяльності розробники повинні навчитись розуміти мову користувачів та замовників, з'ясувати цілі їх діяльності, визначити набір задач, вирішуваних ними.

На додаток варто з'ясувати, які взагалі задачі потрібно вміти вирішувати для досягнення цих цілей, з'ясувати властивості результатів, які хотілось би одержати, а також визначити набір сутностей, з якими доводиться мати справу при вирішенні цих задач. Крім того, аналіз предметної галузі дозволяє виявити місця можливих покращень і оцінити наслідки прийнятих рішень про реалізацію тих чи інших функцій. Після цього можна визначати галузь відповідальності майбутньої програмної системи і точніше формулювати вимоги.

Аналізом прикладної галузі займаються системні аналітики або бізнес-аналітики, які передають одержані ними знання іншим членам проектної команди, формулюючи їх більш зрозумілою розробникам мовою. Для передачі цих знань існує деякий набір моделей у вигляд графічних схем і тектосвих документів. Аналіз діяльності великої організації дає величезні обсяги інформації, з якої потрібно обирати суттєву та знаходити у ній «пробіли». Отже, одержану інформацію слід якимось чином систематизувати, для чого застосовується схема Захмана або архітектурна схема підприємства.

В основі схеми Захмана лежить наступна ідея: діяльність навіть дуже великої організації можна описати, використовуючи відповіді на прості запитання – навіщо, хто, як, де і коли – і різні рівні розгляду:

- цілі організації та базові правила, за якими вона працює;
- персонал, підрозділи та інші елементи організаційної структури, зв'язки між ними;
- сутності та дані, з якими має справу організація;
- виконувані організацією та різними її підрозділами функції та операції над даними;
- географічний розподіл елементів організації та зв'язки між географічно розділеними її частинами;
- часові характеристики та обмеження на діяльність організації, значущі для її діяльності події.

Також виділені декілька рівнів розгляду, з яких при бізнесмоделюванні особливо важливі три верхніх:

- найкрупніший – рівень організації в цілому, розглянутий в її розвитку спільно з оточенням, рівень загального планування її діяльності. Цей рівень містить тривалі цілі і задачі організації як цілісної системи, основні зв'язки організації з зовнішнім світом та основні види її діяльності;
- рівень бізнесу, на якому організація розглядається в усіх аспектах як окрема сутність, яка має певну структуру, відповідає її основним задачам;

- системний рівень, на якому визначаються концептуальні моделі всіх аспектів організації, без прив'язки до конкретних її втілень та реалізацій, наприклад, логічна модель даних у вигляді набору сутностей та зв'язків між нимим, логічна архітектура системи автоматизації у вигляді набору вузлів з прив'язаними до них функціями.

Найбільш зручною формою представлення інформації при аналізі предметної області є графічні діаграми різного роду. Вони дозволяють досить швидко зафіксувати одержані знання, швидко відновлювати їх у пам'яті і успішно спілкуватись із замовниками та іншими зацікавленими особами.

Часто для опису поведінки складних систем та діяльності крупних організацій використовуються діаграми потоків даних, які містять 4 види графічних елементів: процеси (трансформації даних в межах описуваної системи), сховища даних (зовнішні по відношенню до системи), сутності та потоки даних (між елементами трьох попередніх видів).

Використовуються декілька систем позначень для перерахованих елементів, найбільш відомі нотація ЙорданаДеМарко (Yourdon-DeMarco) і нотація Гейна-Сарсона (GaneSarson). В нотації Йордана-ДеМарко: процеси зображені колами, зовнішні сутності – прямокутниками, сховища даних – двома горизонтальними паралельними лініями. В нотації Гейна-Сарсона: процеси – прямокутники з заокругленими кутами, зовнішні сутності – прямокутники з тінню, сховища даних – витягнуті горизонтально прямокутники без правого ребра.

Фаза створення функційної специфікації. Функційна специфікація (Functional Specification) – це формальний опис, який пояснює, що і як буде робити програма. Вона достатньо детально показує будову всіх модулів та їх взаємодію з врахуванням проектних обмежень. Функційна специфікація – це документ, який описує потрібні характеристики програмної системи (функційність), а також необхідні для користувача системи вхідні та вихідні параметри.

Функційні специфікації допомагають усунути дублювання та невідповідності, дозволяють точно оцінити необхідні дії та ресурси, виступають в якості узгоджуючого та довідкового документів про внесені зміни, надають документацію про конфігурацію, дають можливість взаємодії осіб, які працюють з вісьмома основними функціями системного проектування. Вони дають точне уявлення про вирішення проблеми, підвищуючи ефективність розроблення системи та оцінюючи вартість альтернативних шляхів проектування. Вони служать вказівкою для випробувачів при верифікації (якісному оцінюванні) кожної технічної вимоги.

Функційна специфікація не визначає операцій, які відбуваються всередині даної системи і яким чином буде реалізовано її функцію. Замість цього, вона розглядає взаємодію із зовнішніми агентами (персонал, периферійні пристрої), які можуть взаємодіяти із системою.

Приклад з типової функційної специфікації: «Коли користувач натискає кнопку ОК, діалогове вікно закривається і в фокусі залишається головне вікно, яке було відкрите до появи діалогу». Така вимога описує взаємодію зовнішнього агента (користувач) і програмної системи. Специфікація може бути неформальною, тоді її можна сприймати як план або керівництво користувача з точки зору розробника, або формальною, тоді вона визначає математичні або програмні умови. З врахуванням призначення функційної специфікації та важких наслідків помилок у цьому документі, функційна специфікація повинна бути математично точною. Це означає, що вона повинна базуватись на поняттях, побудованих як математичні об'єкти, та твердженнях, однозначно зрозумілих для розробника ПЗ. Досить часто специфікація формулюється природньою мовою. Але використання математичних методів та формалізованих мов при розробленні функційної специфікації дуже бажане.

Функційна специфікація складається з 3-х частин:

1) опис зовнішнього інформаційного середовища, до якого повинні застосовуватись програми розроблюваної системи – на концептуальному рівні визначаються всі використовувані канали введення-виведення, всі інформаційні об'єкти, до яких застосовується розроблюване ПЗ, а також істотні зв'язки між цими об'єктами (наприклад, концептуальна схема бази даних або опис мережі приладів, якими керує розроблюване ПЗ);

2) визначення функцій програмної системи, визначених на множині станів цього інформаційного середовищі (зовнішні функції системи) – вводяться позначення всіх визначених

функцій, специфікуються всі вхідні дані та результати виконання кожної визначеної функції з їх типами, обмеженнями та співвідношеннями, а також визначається семантика кожної функції, що є найбільш важкою задачею;

3) опис небажаних (виключних) ситуацій, які можуть виникнути при виконання програм системи, а також реакцій на ці ситуації, які повинні забезпечити відповідні програми – перераховуються всі істотні з точки зору зовнішнього користувача випадки, коли ПЗ не зможе нормально виконувати ту чи іншу свою функцію, а також визначає реакцію ПЗ на кожен такий випадок.

Функційні специфікації можуть створюватись з різними цілями. Одна з основних цілей – привести групу розробників до єдиної думки про те, як в результаті повинна виглядати програма,перш ніж вдаватись до дій, які потребують значних часових витрат (написання сирцевого коду, тестування, відлагодження програм).

Специфікація неможлива без чіткого опису структур даних програми. Функційна специфікація розробляється під керівництвом та за безпосередньої участі архітектора (бізнес-аналітика) проекту. Функційна специфікація може сильно змінюватись від проекта до проекта. В великих комплексних проектах специфікації мають декілька рівнів деталізації. Першоджерелом для розроблення функцій них специфікацій є технічне завдання (Customer Requirements Specification), в якому описуються всі вимоги до ПЗ.

На верхньому рівні специфікування ПЗ розробляється зовнішня специфікація для замовника (Software Specification Document) – документ, який в бізнес-термінах, без перевантаження технічними подробицями, пояснює, що повинна робити система.

На другому рівні може бути розроблений концептуальний документ, який описує архітектуру системи (Software Architecture Document) – документ, який відображає функціювання всієї системи в цілому, не деталізуючи побудови окремих модулів, представляє структуру об’єктів та їх залежності. На заключному етапі розробляється технічна специфікація (Detail Design Document) – документ, який дозволяє повністю зосередитись на етапі кодування (реалізації) системи, описує низькорівневу організацію продукту, всі вимоги для кожного модуля (передавані параметри, глобальні структури та змінні, підпрограми, що викликаються і т.і.).

Задача розроблення специфікації вважається виконаною, якщо користувач чітко сформулював свої очікування до результуючого продукту, а програміст здатен однозначно реалізувати ці очікування в продукті без будь-яких інших знань про проект, керуючись лише специфікацією.

Фазу проектування ПЗ розглядають як діяльність, результат якої складається з двох складових частин:

1) архітектурне проектування - опис високорівневої структури та організації компонентів ПЗ (software architectural design, top-level design);

2) деталізація архітектури - деталізований опис кожного компонентами в потрібному для конструювання обсязі (software detailed design).

Отже, фаза проектування включає наступні 2 етапи:

1) розроблення системного проекту – на цьому етапі формується відповідь на запитання «Що повинна робити майбутня програмна система?», а саме: визначаються архітектура системи, її функції, зовнішні умови функціювання, інтерфейси та розподіл функцій між користувачами та системою, вимоги до програмних та інформаційних компонент, склад виконавців та терміни розроблення. Основу системного проекту складають моделі проектованої програмної системи, які будуються на основі моделі “TO-BE”. Документальним результатом є технічне завдання;
2) розроблення технічного проекту – на цьому етапі на основі системного проекту здійснюється власне проектування системи, яке включає проектування архітектури системи та детальне проектування. Таким чином, формується відповідь на запитання «Як побудувати систему, щоб вона задовольняла висунуті до неї вимоги?». Моделі проектованої програмної системи при цьому уточняються і деталізуються до необхідного рівня.

Архітектура ПЗ - це опис підсистем та компонентів ПЗ, а також зв'язків між ними. Архітектура намагається визначити внутрішню структуру розроблюваного ПЗ, задаючи спосіб, яким ПЗ організовується або конструюється.

Цілі етапу проектування:

1) підвищення продуктивності бізнес-процесів;

2) зменшення витрат;

3) покращення операційної бізнес-діяльності;

4) підвищення ефективності керування;

5) зменшення ризиків;

6) підвищення ефективності IT-організації;

7) підвищення продуктивності роботи користувачів;

8) підвищення інтероперабельності (можливості та прозорості взаємодії);

9) зменшення вартості підтримки життєвого циклу ПЗ;

10) покращення характеристик безпеки;

11) підвищення керованості.

Принципи проектування:

1) абстракція (Abstraction) - результатом є модель, яка спрощує поставлену проблему до рамок, значущих у даному контексті;

2) зв'язність та зчеплення (Coupling and Cohesion): зв'язність визначає силу зв'язку та взаємовпливу між модулями, тобто може розглядатись як ступінь самодостатності модуля; зчеплення визначає силу зв'язків всередині модуля (внутрішніх зв'язків), тобто функціональну залежність;
3) декомпозиція та розбиття на модулі (Decomposition and Modularization) - проводиться з метою одержання більш дрібних та відносно незалежних програмних компонентів, кожен з яких має різну функціональність;
4) інкапсуляція/приховування інформації (Encapsulation/information hiding) - групування та упаковування елементів та внутрішніх деталей абстракції по відношенню до реалізації з метою недоступності цих деталей користувачам компонентів;

5) розділення інтерфейсу та реалізації (Separation of interface and implementation) - визначення компоненту через вивільнення інтерфейсу від безпосередніх деталей реалізації;

6) достатність, повнота та простота (Sufficiency, completeness and primitiviness) моделі або проекту ПЗ.

Фундаментальними та ключовими проблемами проектування є:

1) паралелізм (Concurrency) - питання, піходи та методи організації процесів, задач і потоків для забезпечення ефективності, атомарності, синхронізації та розроділу в часі оброблення інформації;
2) контроль та оброблення подій (Control and Handling of Events) - методи оброблення подій, часто реалізованих як функції зворотнього виклику;

3) розподіл компонентів (Distribution of Components) - розподіл (і виконання) оброблення в термінах апаратного забезпечення, мережевої інфраструктури і т.і., використання проміжного зв'язуючого ПЗ (middleware);

4) оброблення помилок та виключних ситуацій, забезпечення відмовостійкості (Errors and Exception Handling and Fault Tolerance) - як попередити збої, або, якщо збій все ж відбувся, як забезпечити подальше функціонування ПЗ;

5) взаємодія та представлення (Interaction and Presentation) - представлення інформації користувачам та взаємодії користувачів з ПЗ з точки зору реакції ПЗ на дії користувачів;
6) збережуваність даних (Data Persistence) – вирішується питання, як повинні оброблятись "довгоживучі" дані.

Термінологічний розподіл різних видів проектування ПЗ:

1) D-дизайн (D-design, decomposition design) – декомпозиці структури ПЗ у вигляді набору фрагментів або компонентів;

2) FP-дизайн (FP-design, family pattern design) – сімейство архітектурних представлень, котрі базуються на шаблонах; 3) I-дизайн (I-design, invention) – створення високорівневої концепції представлення того, що собою являє програмна система.