**Лабораторна робота №04 Якість програмного забезпечення. Аналіз вимог замовника до програмного продукту**

**Мет а роботи**

1. Визначення існуючих проблем та обмежень в програмному продукті (ПП)
2. Розробка варіантів використання ПП
3. Формування вимог замовника на розробку (модернізацію) ПП

## **Порядок виконання роботи**

1. Опрацювати теоретичний матеріал до лабораторної роботи.
2. Для теми, що визначена на попередніх лабораторних (власний проект), згідно методичних вказівок інструкції повести попередній аналіз проблем і обмежень.
3. Розробити схеми для основних варіантів використання ПЗ.
4. Сформувати власний варіант документу «Вимоги до програмного засобу» (Software Requirements Specification, SRS).
5. Оформити і захистити лабораторну роботу.
6. Результати надсилати на електронну адресу викладача [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)

Файл повинен мати назву в такому форматі:

**QA <Номер групи><Номер лекції / практичної / лабораторної [літера позначення типу роботи L – лекція, R – лабораторна]<Прізвище англійською>**. Наприклад, **QA 4101R**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт-"близнюків" відповідь буде зараховуватися першому за часом надсилання.

**Строк виконання цієї роботи ІПЗ-41 10.11.2023**

## **Контрольні питання**

1. Назвіть стадії життєвого циклу програмного засобу.
2. З яких етапів складається аналіз вимог замовника?
3. Що таке варіанти використання програмного засобу?

**ТЕОРЕТИЧНА ЧАСТИНА**

## **Стадії життєвого циклу розробки програм**

ЖЦ розробки програм може сильно відрізнятися від проекту до проекту і від керівника проекту до керівника проекту. Проте, зазвичай він складається з наступних стадій:

1. Побудова життєвого циклу
2. Попередній аналіз
3. Аналіз побажань і вимог замовника
4. Уточнення функціональних характеристик
5. Створення технічного проекту (технічного завдання)
6. Реалізація технічного завдання
7. Системне тестування
8. Постpеалізаційна модифікація (доведення)
9. Супровід

## **Інженерія вимог**

**Інженерія вимог** – це розділ програмної інженерії, що займається проблемами отримання вимог до програмного забезпечення і їх документування, а також проблемами управління вимогами. Тобто, вона полягає у виявленні вимог, їх описі (створенні специфікації) і валідації. Цієї теми ми торкалися на лекціях з Програмної інженерії. Пригадаємо розглянуту схему

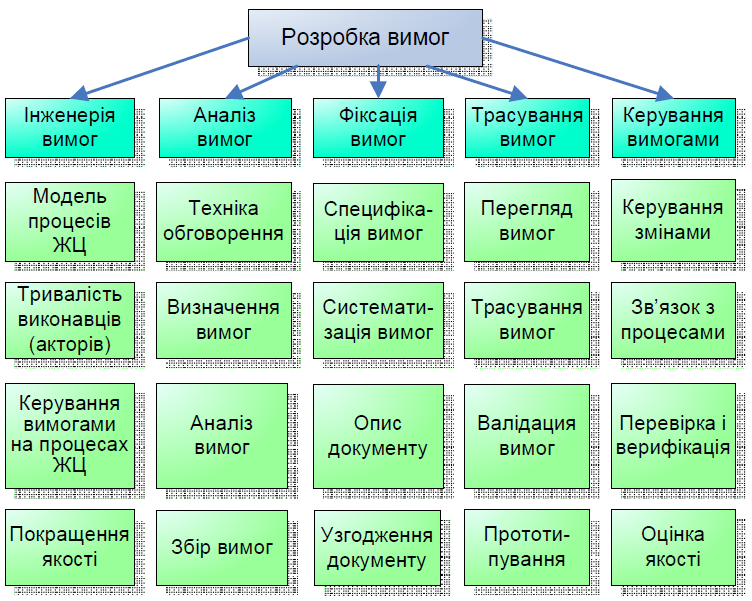


Рисунок 1 - Схема процесу розроблення вимог.

# Проблеми при виконанні проектів виникають через неформальний збір інформації, передбачуваної функціональності програмної системи (ПС), помилкових або неузгоджених нефункціональних вимог до системи, а також нерегламентованої процедури їх зміни, та інше.

В сучасних інформаційних технологіях фаза життєвого циклу, на якій фіксуються вимоги на розробку програмного забезпечення, є визначальною для його якості, термінів та вартості робіт. Саме на цій фазі мають бути зафіксовані реальні потреби користувачів, що стосуються функціональних, операційних та сервісних можливостей, які береться реалізувати розробник. Таким чином, на цій фазі домовляються замовник і виконавець, що визначає подальші дії виконавця.

Ціна помилок та нечітких неоднозначних формулювань на цій фазі дуже висока, бо час і засоби витрачаються на непотрібну замовникові програму, оскільки він мав на увазі зовсім інше, але не зумів сформулювати свої потреби. Внесення необхідних коректив при цьому може вимагати значних переробок, а інколи й повного перепроектування і, відповідно, перепрограмування. Тим часом статистика показує, що відсоток помилок у постановці завдань перевищує відсоток помилок кодування, і це є наслідком суб’єктивного характеру процесу формулювання вимог та майже повної відсутності засобів його формалізації. Так, згідно зі статистикою, в Америці витрачається щорічно до 82 млрд. доларів на проекти, визнані після реалізації такими, що не відповідають вимогам замовника, інакше кажучи, непотрібними. ***Помилки, допущені на стадії збору вимог, складають від 40 до 60% всіх дефектів проекту.***

**Вимоги** – це умови або можливості, якими повинна володіти система або системні компоненти з метою виконання контракту або задоволення відповідним формальним документам та стандартам; це сукупність тверджень відносно атрибутів, властивостей або якостей програмної системи, що підлягають реалізації.

# **Попередній аналіз**

Дуже важливим етапом є попередній аналіз. Розробники ПЗ повинні бути упевнені, що мають всю необхідну інформацію про клієнта, перш ніж розпочати реалізацію проекту.

## Що система повинна робити?

* Чи була чітко сформульована мета створення системи?
* Чи знає кінцевий користувач, що система дійсно повинна робити?

Дуже важливо знайти дійсну мету ПЗ, щоб мати можливість визначити межі проекту.

Це необхідно зробити настільки швидко, наскільки це можливо.

## Моделі даних і словники

Важливо, щоб дані, що обробляються в ПЗ, були виділені і визначені в поняттях, доступних як кінцевим користувачам, так і команді розробників. Часто трапляється, що заздалегідь не існує ніякої моделі даних, і проектувальник повинен створити словник і модель даних самостійно, а потім повернутися до користувача і обговорити з ним розроблену схему, щоб користувач зрозумів її і затвердив.

## Вихідні форми

При попередньому опитуванні користувача необхідно зробити ескізи всіх вихідних форм, оскільки може бути потрібно додаткове нарощування словника баз даних для забезпечення реалізації певних вимог користувача.

## Безпека і управління

Перш ніж почати розробку, кінцевий користувач повинен визначити необхідність забезпечення безпеки системи і даних. Обговорення системи забезпечення безпеки повинне розглядатися на якнайраніших стадіях проектування.

## Платформа і оточення

Hа якій платформі або платформах функціонуватиме створюване програмне забезпечення? Важливо оцінити оточення, в якому працюватиме система. Клієнти витрачають великі кошти на придбання апаратних засобів ще до того, як звертаються по розробку системи. Тому, потрібно з'ясувати всі деталі про:

1. Мережеві апаратні і програмні ресурси;
2. Типи наявних комп'ютерів;
3. Існуючі операційній системи;
4. Типи принтерів, моніторів, дисководів;
5. Інші периферійні пристрої.

## Варіанти використання

Залежно від ПЗ і його цілей, окремі деталі повинні бути обговорені детальніше. Ви повинні знати, що є пріоритетом для кінцевого користувача. Одні системи вимагають максимальної уваги до зовнішнього оформлення і особливостей експлуатації, інші - максимальної швидкості і зручності введення даних при максимально спрощеному зовнішньому вигляді системи. Швидкість часто знижується при використанні засобів обмеження доступу і захисту інформації. Упевніться, що користувачі розуміють значення:

1. Швидкості
2. Безпеки
3. Зовнішньої привабливості
4. Простоти використання
5. Розміру даних і способу їх організації

## Хто використовуватиме дану систему?

Часто поняття "хто" значно важливіше за поняття "що". Хороше розуміння категорій кінцевих користувачів може дати важливу стартову інформацію для початку створення проекту. Потрібно постійно вивчати, що хочуть кінцеві користувачі ПЗ. Різні типи користувачів можуть мати різні вимоги. Ці вимоги повинні бути описані і враховані при проектуванні програмного забезпечення.

## Програмний продукт для відкритого ринку

Якщо розробляється програмний продукт для відкритого ринку, розробники можуть створити тільки дуже грубе уявлення про кінцевого користувача. Якщо, наприклад, пишеться яка-небудь загальна програма обліку, то можна лише припустити, що потенційний клієнт буде мати загальне уявлення про комп'ютер і у нього буде необхідність щось облікувати.

## Програмний продукт для вертикального ринку

Якщо розробляється програмний продукт для підтримки офісу агентства подорожей і екскурсій, то відомо, що кінцеві користувачі використовуватимуть дану програму саме в цій області. Таким чином, можна оптимізувати систему обліку і розрахунків, зважаючи на специфіку конкретного бізнесу. Проте і тут немає ніякої інформації ні про досвід роботи кінцевих користувачів з обчислювальною технікою, ні про рівень їх професіоналізму в їх власному бізнесі.

## Програмний продукт для конкретного користувача

Якщо розробляється офісна система для офісу "Іванов і сини", то тут можна безпосередньо спілкуватися з кінцевими користувачами і з'ясовувати їх рівень пізнання обчислювальної техніки і досвід роботи у власному бізнесі. А це дає можливість заздалегідь передбачити більшість конфліктних ситуацій між програмним продуктом і кінцевими користувачами.

## Що очікують кінцеві користувачі?

Кожна група кінцевих користувачів має різні вимоги і очікування від вашої системи. Перед початком проектування системи необхідно з'ясувати, на що розраховує кінцевий користувач. Необхідно звернути увагу на наступні аспекти:

* + Початкове обстеження і складання технічного завдання
  + Інсталяція програмного продукту
  + Навчання кінцевих користувачів
  + Підтримка програмного продукту
  + Супровід та допомога в експлуатації

## Резюме

Як проектувальник системи, Ви повинні повернутися на рівень попереднього аналізу завдання і упевнитися, що вся необхідна інформація вами отримана. При недотриманні даної вимоги ви можете значно уповільнити реалізацію проекту унаслідок багатократного повторного звернення до користувача за уточненням невірно трактованих деталей і необговорених умов.

# **Аналіз побажань і вимог замовника**

Стандартна техніка розробки і програмування - зібрати весь код в невеликі, добре керовані модулі. Цикл розробки програм працює за аналогічною схемою для великих проектів. Величезна робота повинна бути виконана до того, як хоч би один рядок коди буде написаний. Існує величезна пропасти між ідеями користувачів і уявленням про можливі способи реалізації цих ідей конкретними розробниками. Мостом між цими двома поняттями повинні бути первинний цикл обстеження проекту і складання технічного завдання на даний проект. Це завдання ділиться на три стадії - вивчення вимог замовника, уточнення функціональної специфіки завдання і технічне завдання на проектування .

Аналіз вимог і побажань замовника починається з отримання замовлення на нову розробку (або на модифікацію тієї, що існує) і закінчується складанням документа, в деталях того, що описує дану розробку. Це може бути інтерактивний процес, в результаті якого з'являється документ, що повністю описує завдання і що задовольняє обидві сторони. Результуючий документ описує те, що кінцеві користувачі хотіли б отримати в результаті успішного завершення проекту. Цей документ повинен включати розгляд всіх проблем і вирішуваних завдань, безліч листів з вимогами і побажаннями замовника.

Найважливіша мета, яку необхідно досягти на цьому першому етапі, - це знайти і зрозуміти, що ж HАСПРАВДІ ХОЧЕ КОРИСТУВАЧ!

Часто це не так просто зробити, оскільки користувач не завжди уявляє, ЩО він дійсно хоче отримати. Банальним прикладом можуть служити користувачі, що замовляють, наприклад, одночасно декілька великих завдань типу "Облік заробітної плати", "Ведення складського обліку", "Складання табеля" і тому подібне, називаючи все це "Бухгалтерією". Якщо проігнорувати даний етап, то проект може врешті-решт бути засуджений на велику кількість доопрацювань, добудовування коду "на коліні" і неодмінне сидіння програмістів по вихідним, щоб зробити клієнтові дійсно те, що він хоче і що не було обумовлено заздалегідь. Можна порівняти цикл розробки програми з будівництвом будинку. Коли будинок спроектований, фундамент закладений і будівництво добігає кінця, немає можливості пересувати стіни або змінювати загальне планування будинку.

Очевидно, що будь-який проект починається з ідеї. Як тільки з'являється ідея, один або декілька чоловіків починають розвивати дану ідею. Ці люди - користувачі. Вони визначають початкові вимоги і ухвалюють рішення про створення того або іншого програмного продукту. Без них не було б і продукту. Таким чином, необхідно з'ясувати, що ж ці люди хочуть отримати від програмного продукту.

Перед початком обговорення майбутнього проекту дуже важливо впевнитися, що з обох боків столу переговорів сидять саме ті люди, які потрібні для сумісного обговорення проекту.

Три найбільш поширені помилки допускаються на даній стадії:

1. Замовники, які приступають до обговорення проекту, не є тими людьми, які ухвалюватимуть остаточне рішення про вимоги до програмної системи (тобто вони не є людьми, що мають повне уявлення про описуване ними завдання).
2. Учасники обговорення з боку розробників не є людьми, що мають відношення до технічної розробки проекту.
3. Технічні фахівці не розуміють користувачів (або розробники погано розбираються в діловодстві і бізнесі, або останню частину життя вони провели не відходячи від монітора, і можуть розмовляти тільки мовою бітів і байтів).

## Аналіз типу користувачів ПЗ

У одному випадку, користувачі, що беруть участь в обговоренні проекту, описують всі, із їхньої точки зору, деталі майбутнього завдання і залишаються задоволеними думкою, що вони точно виклали всі вимоги і побажання до завдання. На жаль, якщо користувачі, що брали участь в обговоренні не є кінцевими користувачами даної системи, то після складання кінцевого документа, який в деталях описує вирішувану задачу, у людей, що підписують даний документ, виникають питання і які-небудь нові пропозиції по вдосконаленню окремих деталей або їх зміні. Ця ситуація виникає в більшості подібних випадків. Як ви розумієте, така ситуація відкидає процес розробки ПЗ знову назад, на стадію аналізу майбутнього проекту. Маємо втрату часу і коштів.

Аналогічна проблема виникає при участі в розробці вимог таких людей, які ніколи не використовуватимуть створювану програму. Це загальна проблема проектування програмного забезпечення. Коли весь проект розроблений, реалізований, відтестований і представлений замовникові, кінцеві користувачі, ті хто дійсно використовуватиме створений програмний продукт, з'ясовують, що воно швидше перешкода, ніж допомога в їх роботі.

Третя проблема полягає в тому, що користувачі, які пред'явили мінімальні вимоги до системи на стадії системного проектування і залишили розробку проекту на розгляд виробника, починають обурюватися, що продукт не задовольняє тим або іншим вимогам, а тому працює некоректно і вимагає переробки.

Щоб уникнути описаних вище ситуацій, необхідно зробити наступне:

1. Переконатися, що люди, що беруть участь в обговоренні проекту, є людьми, що детально розбираються в тонкощах вирішуваного завдання.
2. Переконатися, що люди, що беруть участь в обговоренні проекту, зацікавлені в кінцевому результаті.
3. Дати користувачам можливість обговорити всі питання, які тільки можливо. Навіть якщо їх пояснення незв'язні і неорганізовані, спробуйте з'ясувати, що для користувачів є важливішим в створюваній програмі, а що менш важливим (що вони хотіли б отримати спочатку, а що потім).
4. Постаратися підключити до обговорення людей, які дійсно використовуватимуть створюваний продукт.

## Керівник проекту

Другою поширеною помилкою є вибір керівника проекту, що не володіє відповідними технічними знаннями для реалізації даного проекту. Ця проблема зазвичай зустрічається на великих проектах, де необхідна велика команда програмістів. Часто існує технічний лідер, який може управляти проектом так же добре, як і вирішувати технічні питання. Використання його як менеджера проекту буде значно ефективнішим, ніж використання простого адміністратора.

В процесі аналізу вимог замовника важливо, щоб в переговорах брав участь один з членів команди розробників, а в кращому разі, провідний технічний фахівець або технічний керівник проекту. На жаль, достатньо важко зібрати в одній кімнаті і в один час всіх людей, яким необхідно брати участь в обговоренні проекту.

Якщо в процесі обговорення бере участь тільки адміністративна особа, або керівник проекту, далекий від проблем безпосереднього кодування, то виникає безліч проблем і питань, пов'язаних з можливою оптимізацією окремих операцій, створенням словника баз даних, системними вимогами до створюваного програмного забезпечення і так далі. В даному випадку окремим членам доводиться повторно спілкуватися з кінцевими користувачами для з'ясування неврахованих або погано продуманих питань, що врешті-решт може зіпсувати відношення між командою розробників і кінцевими користувачами. З іншого боку, участь в обговоренні проекту технічних фахівців може привести до помітного спрощення проекту за рахунок приведення окремих вимог користувача до вже існуючих і раніше розроблених технологій задоволення даних вимог.

Коли необхідний технічний персонал просто не може бути присутнім на всіх засіданнях обговорення проекту, або технічний фахівець повинен тимчасово перемкнутися на інші дії в процесі обговорення, може допомогти так званий протокол засідань. Даний протокол містить нотатки про всі обговорювані на даному засіданні питання, а також імена, посади і телефони учасників обговорення як з однією, так і з іншого боку. Даний протокол також повинен містити інформацію про ухвалені рішення, обумовлені нюанси і будь-які деталі, обговорення яких вже проводилося. В кінці дані замітки винні перерости в кінцевий документ, що описує результат, отриманий в процесі обговорення всіх частин і деталей проекту – Специфікацію вимог замовника.

Якщо вказані рекомендації будуть дотримані, то технічна сторона розробки буде розглянута більш повно, що дасть можливість згодом уникнути багатьох помилок, пов'язаних з нерозумінням тією або іншою стороною технічних особливостей проекту.

## Програмісти, що працюють на рівні бітів і байтів

***Оберігайтеся*** програмістів, які можуть просидіти декілька діб над створенням функції, яка фактично не потрібна кінцевому користувачеві. Програмісти, які не уміють працювати з користувачами, не розуміють питань, пов'язаних із специфікою роботи кінцевого користувача, або що не уміють викласти більш менш складні технічні питання простою мовою не повинні брати участь в процесі обговорення проекту. Вони можуть створити додаткові труднощі технічного і часового характеру, починаючи детально з'ясовувати неістотні питання.

***На жаль***, багато програмістів не дуже добре розбираються в тому, що оточує їх у діловому світі. Їх спеціалізація - комп'ютери і програми, а не створення кінофільмів або управління лікарняним господарством. Виникає питання: "Чи дійсно необхідно команді розробників детально розбиратися в діловодстві і специфіці бізнесу кінцевих користувачів?" Недосвідчений програміст подумає, "Користувачі - професіонали в своїй області, а я - професіонал в своїй. Якщо ми почнемо навчати один одного нашим професіям, чи знадобимося ми один одному врешті-решт?"

***Це Hевіpно.*** Професійні знання в тій або іншій області не отримуються в процесі сумісного обговорення якого-небудь проекту. Hе отримуються вони і в процесі написання програми із заданої тематики. Професійні знання часто отримуються в процесі багатьох років навчання, наступних за не менш тривалим періодом проб і помилок.

***Коли користувачі намагаються описати***, що вони хочуть від окремих частин програми, не треба відразу переводити їх побажання в код. Необхідно розуміти, що хоче користувач, а потім постаратися зробити це найбільш оптимальним способом.

***Оптимальний варіант***, коли користувач має уявлення про технічну сторону обговорюваного завдання, а команда програмістів має досвід у сфері діяльності користувача. Коли поєднуються такі якості користувача і розробника, приблизно половина питань відразу знімається з обговорення за непотрібністю.

***У гіршому разі*** користувач є технофобом, а технічний фахівець нічого не знає про бізнес. В даному випадку хороша система все ж таки може бути розроблена, якщо обидві сторони підуть назустріч один одному і точно визначать всі вимоги і побажання. Проте окремі сторони проекту можуть бути неадекватно описані або взагалі упущені.

## Аналіз вимог замовника

Одне лише підключення до процесу розробки необхідних осіб з обох сторін не приведе до створення повноцінного документа, що описує завдання. Вся система не буде визначена до даного моменту. Головною метою є знаходження того, що хоче користувач. Іноді вартість контракту і час виконання завдання оцінюється після даного кроку, і Ви можете не підписати контракт зовсім або внести істотні поправки до первинних домовленостей на основі виконаного дослідження завдання. Важливо зберегти простоту процесу аналізу вимог і уникати обдумування, як буде реалізована та або інша функція або процедура. Необхідно пам'ятати, що аналіз вимог замовника може продовжитися від двох годин до декількох тижнів, залежно від складності поставленого завдання.

Може існувати велика кількість способів почати і проводити аналіз вимог, але всі вони повинні приводити до одного і тому ж результату - **складання документа, що описує всі вимоги і побажання користувача**.

Простий спосіб почати обстеження - зверху вниз. Що є головною метою системи? Які основні вимоги до системи? Визначення основних компонент системи може бути корисним для введення користувача в потрібне русло обговорення проблеми. Майже всі системи вимагають введення якоїсь інформації і виведення якихось звітних форм (у вигляді звітів і запитів), деякий вид конфігурації, можливості імпорту і експорту даних, архівації, і, можливо, сервісний розділ. Іноді процес імпорту - єдиний спосіб введення даних в систему. Якщо ж користувач вимагає багатотабличнy систему, тоді конфігурація може стати головною частиною системи. Також, який тип інтерфейсу потрібний? Випадні вікна? Графічний інтерфейс? Виходячи з цих даних, розробник може отримати інформацію про те, що повинно знаходитися в головному меню програми, і прикинути деякі деталі розробки ще до повного визначення проекту.

Hезалежно від прийнятого підходу до розгляду вимог користувача, результатом аналізу повинно бути ясне розуміння того, що вимагає користувач, і що він хоче. Тонка відмінність між цими двома поняттями є важливою! Вимоги користувача обмежуються уявленням користувача про пропоноване ним завдання. Ці вимоги користувач явно обговорює в процесі дискусії. Побажання ж користувача нерідко залишаються за кадром, не тому що користувач не обговорює їх спеціально, а тому, що він підсвідомо вважає деякі вимоги природними і такими, що не вимагають спеціального виділення.

## Також необхідно розставити пріоритети окремим ділянкам завдання.

Головна мета цієї стадії - упевнитися в тому, що ви розумієте потреби користувача і пріоритети напрямів розробки. Кінцевий документ складається для користувача, і він повинен його затвердити. Якщо дана фаза розробки проекту була ретельно проведена, то наступна фаза - Функціональна специфікація - не складе особливих труднощів.