**Практична робота №7-1. Застосування моделей якості і надійності програмних систем. Метрики якості**

**Мета:** навчитися обирати метрики якості та надійності для оцінювання програмного продукту.

**Завдання:**

1. Опрацювати лекцію №10 та теоретичні відомості, ознайомитися з матеріалами, наданими в списку додаткової літератури.
2. За цією інформацією визначте:

* метрики програмного продукту і їхні складові, а також які з цих метрик ви можете застосувати до вашого проекту;
* визначте стандарти з якості програмних систем (далі–ПС).

1. Робота повинна бути виконана згідно критеріїв оформлення документації та повинна містити

* Назва практичної роботи.
* Прізвище, група
* Назва проекту.
* Письмові відповіді на п.2.

По закінченню практичну роботу потрібно здати на перевірку викладачеві, надіславши електронною поштою на адресу [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com) . Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

Файл з роботою повинен мати назву в такому форматі:

**QA<Номер групи><Номер лекції / практичної / лабораторної>[-<Номер завдання>][літера позначення типу роботи L – лекція, P –практична, R – лабораторна]<Прізвище англійською>**.. Наприклад, **QA4101Р**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.

Тему в заголовку листа записати

**QA <Номер групи>-ЛР<Номер лабораторної>-<Прізвищеанглійською>**

**Строк виконання цієї роботи ІПЗ-41 – 19.02.2024**

**Теоретичні відомості**

**Метрики якості**

На сьогодні у програмній інженерії діють різні підходи до визначення набору метрик якості та методів вимірювання. Система вимірювання містить у собі метрики й моделі вимірювань, які використовуються для кількісної оцінки якості ПС. При визначенні вимог до ПС задаються відповідні ним зовнішні характеристики і їхні атрибути (характеристики), що визначають різні аспекти керування продуктом у певному середовищі. Для набору характеристик якості ПС, наведених у вимогах, визначаються відповідні метрики, моделі їхньої оцінки й діапазон значень мір для вимірювання окремих атрибутів якості.

Відповідно до стандарту *ISO/IEC* 9126. Infofmation Technology. – Software Quality Characteristics and metrics. 1997 метрики визначаються за моделями виміру атрибутів ПС на всіх процесах ЖЦ (проміжна, внутрішня метрика) і особливо на процесі тестування або функціонування (зовнішні метрики) продукту. Наведемо  класифікацію метрик ПС, правил для проведення метричного аналізу й процесу їхнього виміру.

**Типи метрик***.* Існує три типи метрик:

- метрики програмного продукту, які використовуються для вимірювання його характеристик – властивостей;

- метрики процесу, які використовуються для вимірювання властивості процесу ЖЦ створення продукту;

- метрики використання.

**Метрики програмного продукту** містять у собі:

- зовнішні метрики, що визначають властивості продукту, видимі користувачеві;

- внутрішні метрики, що визначають властивості, видимі тільки команді розробників.

*Зовнішні метрики*продукту – це метрики:

- надійності продукту, які використовують для визначення числа дефектів;

- функціональності, за допомогою яких визначають наявність і правильність реалізації функцій у продукті;

- супроводу, за допомогою яких вимірюють ресурси продукту (швидкість, пам'ять, середовище);

- застосування продукту, які сприяють визначенню ступеня доступності для вивчення й використання;

- вартості створеного продукту.

*Внутрішні метрики*продукту вміщують:

- метрики розміру, необхідні для вимірювання продукту за допомогою його внутрішніх характеристик;

- метрики складності, необхідні для визначення складності продукту;

- метрики стилю, які використовуються для визначення підходів і технологій створення окремих компонентів продукту і його документів.

Внутрішні метрики дозволяють визначити продуктивність продукту і є релевантними відносно зовнішніх метрик.

Зовнішні й внутрішні метрики задають на процесі формування вимог до ПС і є предметом планування й керування досягненням якості кінцевого програмного продукту.

Метрики продукту часто описуються комплексом моделей для встановлення різних властивостей, значень моделі якості або прогнозування. Вимірювання виконують, як правило, після калібрування метрик на ранніх процесах проекту. Загальна міра – ступінь трасування, що визначають числом трас, які простежуються за моделями сценаріїв типу UML й оцінкою кількості:

– вимог;

– сценаріїв і дійових осіб;

– об'єктів, вміщених у сценарій, і локалізація вимог до кожного сценарію;

– параметрів й операцій об'єкта й ін.

Стандарт ISO/IEC 9126–2 визначає такі типи мір:

– міра розміру ПС в різних одиницях вимірювання (число функцій, рядків у програмі, розмір дискової пам'яті й ін.);

– міра часу (функціонування системи, виконання компонента й ін.);

– міра зусиль (продуктивність праці, трудомісткість й ін.);

– міра обліку (кількість помилок, число відмов, відповідей системи та ін.).

Спеціальною мірою може бути рівень використання повторних компонентів, яку вимірюють як відношення розміру продукту, виготовленого з готових компонентів, до розміру системи в цілому. Така міра використовується також при визначенні вартості і якості ПС. Приклади метрик:

- загальне число об'єктів і число повторно використовуваних;

- загальне число операцій, повторно використовуваних і нових операцій;

- число класів, що успадковують специфічні операції;

- число класів, від яких залежить певний клас;

- число користувачів класу або операцій та ін.

При оцінці загальної кількості певних величин часто використовують середньостатистичні метрики (середнє число операцій у класі,  класу нащадків або операцій класу й ін.).

Як правило, міри є суб'єктивними й залежать від знань експертів, що виконують кількісні оцінки атрибутів компонентів програмного продукту.

Прикладом широко використовуваних зовнішніх метрик програм є метрики Холстеда – це характеристики програм, виявлених на основі статичної структури програми конкретною мовою програмування: число входжень операндів й операторів, що найчастіше зустрічаються; довжина опису програми як сума числа входжень всіх операндів й операторів та ін.

На основі цих атрибутів можна обчислити час програмування, рівень програми (структурованість та якість) і мови програмування (абстракції засобів мови й орієнтація на проблему) та ін.

Як метрику процесу можна використовувати час розробки, число помилок, знайдених на процесі тестування та ін. Частіше застосовують такі метрики процесу:

- загальний час розробки й час окремо для кожної стадії;

- час модифікації моделей;

- час виконання робіт на процесі;

- число знайдених помилок при інспектуванні;

- вартість перевірки якості;

- вартість процесу розробки.

**Метрики використання**призначено для вимірювання ступеня задоволення потреб користувача для розв’язання задач. Вони допомагають оцінити не властивості самої програми, а результати її експлуатації – експлуатаційну якість. Як приклад – точність і повнота реалізації завдань користувача, а також витрачені ресурси (трудовитрати, продуктивність та ін.) на ефективне розв’язання задач користувача. Оцінка вимог користувача виконується за допомогою зовнішніх метрик.

**Стандартна оцінка показників якості**

Оцінка якості ПС згідно з чотирьохрівневою моделлю якості починається з нижнього рівня ієрархії, тобто з найелементарнішої властивості оцінюваного атрибута показника якості згідно з встановленими  мірами. При  проектуванні системи  визначають значення оцінних елементів з кожного атрибута показника аналізованого ПС, що відображений у  вимогах.

За визначенням стандарту ISO/IES 9126-2 метрика якості ПС являє собою «модель вимірювання атрибута, що зв'язує його з показником якості». При вимірюванні показників якості даний стандарт дає змогу визначати такі типи мір:

-  міри розміру в різних одиницях вимірювання (кількість функцій, розмір програми, обсяг ресурсів й ін.);

-  міри часу – періоди реального, процесорного або календарного часу (час функціонування системи, час виконання компонента, час використання й ін.);

-  міри зусиль – продуктивний час, витрачений на реалізацію проекту (продуктивність праці окремих учасників проекту, колективна трудомісткість та ін.);

-  міри інтервалів між подіями, наприклад, час між послідовними відмовами;

-  міри лічби – лічильники для визначення кількості виявлених помилок, структурної складності програми, числа несумісних елементів, числа змін (наприклад, число виявлених відмов та ін.).

Метрики якості, що використовують для оцінки ступеня простоти тестування за допомогою даних (безвідмовна робота, виконуваність функцій, зручність застосування інтерфейсів користувачів, БД і т.п.) після проведення випробувань ПС на множині тестів.

Напрацювання на відмову як атрибут надійності – це середній час між появою загроз, що порушують безпеку, складова вимірюваної оцінки збитку, завданого відповідними загрозами.

Дуже часто оцінка програми здійснюється за кількістю рядків. При зіставленні двох програм, що реалізують одне застосування, перевага надається короткій програмі, оскільки її створює більш кваліфікований персонал, в ній менше прихованих помилок, її легше модифікувати. За вартістю вона дорожче, хоча часу на налагодження й модифікацію витрачають більше. Отже, довжину програми можна використати як допоміжну властивість для порівняння програм з урахуванням однакової кваліфікації розробників, єдиного стилю розробки й спільного середовища.

Якщо у вимогах до ПС було зазначено отримання кількох показників, то обчислений після збирання даних показник множиться на відповідний ваговий коефіцієнт, а потім підсумовуються всі показники для отримання комплексної оцінки рівня якості ПС.

На основі вимірювання кількісних характеристик і проведення експертизи якісних показників із застосуванням вагових коефіцієнтів, що нівелюють різні показники, визначається підсумкова оцінка якості продукту шляхом підсумовування результатів за окремими показниками й порівняння їх з еталонними показниками ПС (вартість, час, ресурси й ін.).

Іншими словами, при проведенні оцінки окремого показника за допомогою оцінних елементів визначають ваговий коефіцієнт *k-метрики*, *j-показника*, *i-атрибута*. Наприклад, як *j-показник* візьмемо переносність. Цей показник буде обчислюватися за п'ятьма атрибутами (*i*= 1, ..., 5), причому кожний з них буде множитися на відповідний коефіцієнт *ki*.

Всі метрики *j-атрибута* підсумовують й утворюють *i-показник* якості. Коли всі атрибути оцінено за кожним з показників якості, виробляється сумарна оцінка окремого показника, а потім й інтегральна оцінка якості з урахуванням вагових коефіцієнтів всіх показників ПС.

В остаточному підсумку результат оцінки якості є критерієм ефективності й доцільності застосування методів проектування, інструментальних засобів і методик оцінювання результатів створення програмного продукту на процесах ЖЦ.

Для викладу оцінки значень показників якості використовують стандарт *ДСТУ* 3230–1995. Управление качеством и обеспечение качества. Термины и определения., у якому представлено такі методи: вимірювальний, реєстраційний, обчислювальний та експертний (а також комбінації цих методів).

*Вимірювальний метод*базується на використанні вимірювальних і спеціальних програмних засобів для отримання інформації про характеристики ПС, наприклад, визначення обсягу, числа рядків коду, операторів, кількості гілок у програмі, число точок входу (виходу), реактивність та ін.

*Реєстраційний метод* використовують для підрахунку часу, числа збоїв або відмов, початку й кінця роботи  системи у процесі його виконання.

*Розрахунковий метод* базується на статистичних даних, зібраних при проведенні випробувань, експлуатації й супроводі ПС. Розрахунковими методами оцінюються показники надійності, точності, стійкості, реактивності й ін.

*Експертний метод* здійснюється групою експертів – фахівців, компетентних у розв’язку  певної задачі або типу ПС. Їхня оцінка базується на досвіді й інтуїції, а не на безпосередніх результатах розрахунків або експериментів. Цей метод здійснюється шляхом перегляду програм, кодів, супровідних документів і сприяє якісному оцінюванню створеного продукту. Для цього встановлюють контрольовані ознаки, які корелюють з одним або декількома показниками якості й вміщені в карти опитування експертів. Метод використовується для оцінювання таких показників, як простота аналізу, документованість, структурованість ПС та ін.

Для оцінювання значень показників якості залежно від особливостей використаних ними властивостей, призначення, способів їхнього визначення застосовуються:

- шкала метрична (1.1 – абсолютна, 1.2 – відносна, 1.3 – інтегральна);

- шкала порядкова (рангова), що дозволяє визначати ранг характеристики шляхом порівняння з опорними;

- класифікаційна шкала, що характеризує наявність або відсутність розглянутої властивості в оцінюваному програмному забезпеченні.

Показники, які обчислюються за допомогою метричних шкал, називають кількісними, а визначені за допомогою порядкових і класифікаційних шкал – якісними.

Атрибути програмної системи, що характеризують її якість, вимірюють з використанням метрик якості. Метрика визначає міру атрибута, тобто змінну, якій привласнюється значення, отримане внаслідок вимірювання.

Для правильного використання результатів вимірювань кожна міра ідентифікується відповідною шкалою.

Стандарт ISO/IES 9126–2 рекомендує застосовувати п’ять  видів шкал вимірювання значень, які впорядковано від менш строгої до більш строгої:

- номінальна шкала відображає категорії властивостей оцінюваного об'єкта без їхнього впорядкування;

- порядкова шкала впорядковує характеристики за зростанням або спадання шляхом порівняння їх з базовими значеннями;

- інтервальна шкала задає істотні властивості об'єкта (наприклад, календарна дата);

- відносна шкала задає певне значення за вибраною одиницею;

- абсолютна шкала вказує на фактичне значення величини (наприклад, число помилок у програмі дорівнює 10).

Приклад використання метрик якості.

На погляд багатьох фахівців оцінка за кількістю рядків у коді тягне за собою спокусу написати більше рядків, щоб взяти побільше грошей. Зрозуміло, про оптимізацію в такому продукті ніхто вже думати не стане.

Кількість рядків вихідного коду (Lines of Code - LOC, Source Lines of Code - SLOC) є найбільш простим і розповсюдженим способом оцінки обсягу робіт за проектом.

Спочатку даний показник виник як спосіб оцінки обсягу роботи за проектом, в якому застосовувалися мови програмування, що володіють досить простою структурою: «один рядок коду = одна команда мови». Також давно відомо, що одну й ту ж функціональність можна написати різною кількістю рядків, а якщо візьмемо мову високого рівня (С++, Java), то можливо і в одному рядку написати функціонал 5-6 рядків — це не проблема. І це було б півбіди: сучасні засоби програмування самі генерують тисячі рядків коду на дріб'язкову операцію. Тому метод LOC є тільки оціночним методом (який треба брати до відома, але не спиратися в оцінках) і ніяк не обов'язковим. Залежно від того, яким чином враховується подібний код, виділяють два основні показники SLOC:

* кількість «фізичних» рядків коду — SLOC (використовуються абревіатури LOC, SLOC, KLOC, KSLOC, DSLOC) — визначається як загальне число рядків вихідного коду, включаючи коментарі і порожні рядки (при вимірюванні показника на кількість порожніх рядків, як правило, вводиться обмеження —— при підрахунку враховується кількість порожніх рядків, що не перевищує 25% загального числа рядків у вимірювальному блоці коду);
* Кількість «логічних» рядків коду — SLOC (використовуються абревіатури LSI, DSI, KDSI, де «SI» — source instructions) — визначається як кількість команд і залежить від мови програмування. У тому випадку, якщо мова не допускає розміщення кількох команд в одному рядку, то кількість «логічних» SLOC буде відповідати числу «фізичних», за винятком числа порожніх рядків і рядків коментарів. У тому випадку, якщо мова програмування підтримує розміщення кількох команд в одному рядку, то один фізичний рядок повинен бути врахований як кілька логічних, якщо вона містить більше однієї команди мови.

Для метрики SLOC існує велике число похідних, покликаних отримати окремі показники проекту, основними серед яких є:

* + - кількість порожніх рядків;
    - число рядків, що містять коментарі;
    - відсоток коментарів (відношення рядків коду до рядків коментаря, похідна метрика стилістики);
    - середнє число рядків для функцій (класів, файлів);
    - середнє число рядків, що містять вихідний код для функцій (класів, файлів);
    - середнє число рядків для модулів.

***Недоліки SLOC***

Потенційні недоліки SLOC, на які орієнтована критика:

* Некрасиво і неправильно зводити оцінку роботи людини до декількох числових параметрів і по них судити про продуктивність. Менеджер може призначити найбільш талановитих програмістів на складну ділянку роботи; це означає, що розробка цієї ділянки займе найбільше часу і породить найбільшу кількість помилок, через складність завдання. Не знаючи про ці труднощі, інший менеджер з отриманими показниками може вирішити, що програміст зробив свою роботу погано.
* Метрика не враховує досвід співробітників та їх інші якості.
* Спотворення: процес вимірювання може бути спотворений за рахунок того, що співробітники знають про вимірювання показників і прагнуть оптимізувати ці показники, а не свою роботу. Наприклад, якщо кількість рядків вихідного коду є важливим показником, то програмісти будуть прагнути писати якомога більше рядків і не будуть використовувати способи спрощення коду, що скорочують кількість рядків.
* Неточність: немає метрик, які були б одночасно і значущі і досить точні. Кількість рядків коду — це просто кількість рядків, цей показник не дає уявлення про складність вирішуваної проблеми. Аналіз функціональних точок був розроблений з метою кращого вимірювання складності коду і специфікації, але він використовує особисті оцінки вимірювального, тому різні люди отримають різні результати.

І головне пам'ятати: метрика SLOC не відображає трудомісткості за створення програми.

***Метрики стилістики й зрозумілості програм***

Іноді важливо не просто порахувати кількість рядків коментарів в коді і просто співвіднести з логічними рядками коду, а дізнатися щільність коментарів. Тобто код спочатку був документований добре, потім — погано.

***Метрики складності***

Крім показників оцінки обсягу робіт за проектом дуже важливими для отримання об'єктивних оцінок є показники оцінки його складності. Як правило, дані показники не можуть бути обчислені на самих ранніх стадіях роботи над проектом, оскільки вимагають, як мінімум, детального проектування. Однак ці показники дуже важливі для отримання прогнозних оцінок тривалості і вартості проекту, оскільки безпосередньо визначають його трудомісткість.

***Метрики Опис***

*Зважена насиченість класу 1* (Weighted Methods Per Class (WMC). Відображає відносну міру складності класу на основі циклічної складності кожного його методу. Клас з більш складними методами і великою кількістю методів вважається більш складним. Згідно обчислень метрики батьківські класи не враховуються.

*Зважена насиченість класу 2* (Weighted Methods Per Class (WMC2)). Міра складності класу, заснована на тому, що клас з великою кількістю методів, є більш складним, і що метод з великою кількістю параметрів також є більш складним. Згідно обчислень метрики батьківські класи не враховуються.

*Глибина дерева успадкування* (Depth of inheritance tree). Довжина найдовшого шляху спадкоємства, що закінчується на даному модулі. Чим глибше дерево успадкування модуля, тим може опинитися складніше передбачити його поведінку. З іншого боку, збільшення глибини дає більший потенціал повторного використання даним модулем поведінки, визначеного для класів-предків.

*Кількість нащадків* (Number of children). Число модулів, безпосередньо успадковують даний модуль. Більші значення цієї метрики вказують на широкі можливості повторного використання при цьому занадто велике значення може свідчити про погано вибраної абстракції.

*Зв'язність об'єктів* (Coupling between objects). Кількість модулів, пов'язаних з даним модулем в ролі клієнта або постачальника. Надмірна зв'язність говорить про слабкість модульної інкапсуляції і може перешкоджати повторному використанню коду.

*Відгук на клас* (Response For Class). Кількість методів, які можуть викликатися екземплярами класу, обчислюється як сума кількості локальних методів, так і кількості вилучених методів.

***Метрики Холстеда***

Метрика Холстед відноситься до обчислювальної метрики, на підставі аналізу числа рядків і синтаксичних елементів початкового коду програми.

Основу метрики Холстед складають чотири вимірювані характеристики програми:

* NUOprtr (Number of Unique Operators) — число унікальних операторів програми, включаючи символи—роздільники, імена процедур і знаки операцій (словник операторів);
* NUOprnd (Number of Unique Operands) — число унікальних операндів програми (словник операндів);
* Noprtr (Number of Operators) — загальна кількість операторів в програмі;
* Noprnd (Number of Operands) — загальна кількість операндів в програмі.

***Метрики циклічної складності за Мак-Кейбом***

Показник циклічної складності є одним з найпоширеніших показників оцінки складності програмних проектів. Даний показник був розроблений вченим Мак—Кейбом в 1976 р., належить до групи показників оцінки складності потоку управління програмою і обчислюється на основі графу керуючої логіки програми (control flow graph). Даний граф будується у вигляді орієнтованого графу, в якому обчислювальні оператори або вирази позначаються у вигляді вузлів, а передача управління між вузлами — у вигляді дуг.

Показник циклічної складності дозволяє не тільки зробити оцінку трудомісткості реалізації окремих елементів програмного проекту і скорегувати загальні показники оцінки тривалості і вартості проекту, а й оцінити пов'язані ризики і прийняти необхідні управлінські рішення.

Спрощена формула обчислення циклічної складності відображається наступним чином:

C = e — n + 2,

де e — число ребер, а n — число вузлів на графі керуючої логіки.

Як правило, згідно обчислень циклічної складності логічні оператори не враховуються.

У процесі автоматизованого обчислення показника циклічної складності, як правило, застосовується спрощений підхід, відповідно до якого побудова графа не здійснюється, а обчислення показника проводиться на підставі підрахунку кількості операторів керуючої логіки (if, switch і т.д.) і можливої кількості шляхів виконання програми.

Цикломатичне число Мак-Кейба показує необхідну кількість проходів для покриття всіх контурів сильнозв’язаного графу або кількості тестових прогонів програми, необхідних для вичерпного тестування за принципом «працює кожна гілка». Показник циклічної складності може бути розрахований для модуля, методу та інших структурних одиниць програми.

*Існує значна кількість модифікацій показника циклічної складності.*

* «Модифікована» цикломатична складність — розглядає не кожне розгалуження оператора множинного вибору (switch), а весь оператор як єдине ціле.
* «Строга» цикломатична складність — включає логічні оператори.
* «Спрощена» обчисленість циклічної складності — передбачає обчислення не на основі графа, а на основі підрахунку керуючих операторів.

***Метрики Чепіна***

Існує кілька її модифікацій. Розглянемо більш простий, а з точки зору практичного використання — досить ефективний варіант цієї метрики.

Суть методу полягає в оцінці інформаційної міцності окремо взятого програмного модуля за допомогою аналізу характеру використання змінних зі списку вводу-виводу.

Всі безліч змінних, що складають список вводу-виводу, розбивається на чотири функціональні групи.

1.   Безліч «Р» — що вводяться змінні для розрахунків та для забезпечення виведення. Прикладом може служити змінна, що використовується в програмах лексичного аналізатора, яке містить рядок вихідного тексту програми, тобто сама змінна не модифікується, а лише містить вихідну інформацію.

2.   Безліч «М» — модифікуються або створювані усередині програми змінні.

3.   Безліч «С» — змінні, що беруть участь в управлінні роботою програмного модуля (керуючі змінні).

4.   Безліч «Т» — не використовуються в програмі ( "паразитні") змінні. Оскільки кожна змінна може виконувати одночасно декілька функцій, необхідно враховувати її в кожній відповідній функціональній групі.

Далі вводиться значення метрики Чепіна:

0 = а1Р + а2М + аЗС + а4Т,

де а1, а2, аЗ, а4 — вагові коефіцієнти.

Вагові коефіцієнти використані для відображення різного впливу на складність програми кожної функціональної групи. На думку автора метрики найбільшу вагу, що дорівнює трьом, має функціональна група С, так як вона впливає на потік управління програми. Вагові коефіцієнти інших груп розподіляються таким чином: а1 = 1; а2 = 2; а4 = 0.5. Ваговий коефіцієнт групи Т не дорівнює нулю, оскільки "паразитні" змінні не збільшують складності потоку даних програми, але іноді ускладнюють її розуміння. З урахуванням вагових коефіцієнтів вираз набуде вигляду:

0     = Р + 2М + ЗС + 0.5Т.

***Попередня оцінка якості***

Попередня оцінка якості на основі статистичних методів в залежності від етапу розробки програми.

При використанні інтегрованих інструментальних засобів у компаній, що розробляють типові рішення (під цю категорію потрапляють так звані «інхаузери» — компанії, що займаються обслуговуванням основного бізнесу) з'являється можливість будувати прогнози складності програм, грунтуючись на зібраній статистиці. Статистичний метод добре підходить для вирішення подібних типових завдань і практично не підходить для прогнозу унікальних проектів. У випадку унікальних проектів застосовуються інші підходи, обговорення яких знаходиться за рамками даного матеріалу.

Типові завдання падають на відділи розробки з бізнесу, бо попередня оцінка складності могла б сильно спростити завдання планування та управління, тим більше, що є накопичена база по проектах, в якій збережено не лише остаточні результати, а й всі початкові та проміжні.

*Виділимо типові етапи в розробці програм:*

* розробка специфікації вимог до програми;
* визначення архітектури;
* опрацювання модульної структури програми, розробка інтерфейсів між модулями;
* опрацювання алгоритмів;
* розробка коду і тестування.

Тепер спробуємо розглянути ряд метрик, що часто використовуються для попередньої оцінки на перших двох етапах.

Попередня оцінка складності програми на етапі розробки специфікацій вимог до програми. Для оцінки за результатами роботи даного етапу може бути використана метрика прогнозованого числа операторів Nпрогн програми:

Nпрогн = NF \* Код

де: NF — кількість функцій чи вимог у специфікації програми, що розробляється;

Код — одиничне значення кількості операторів (середня кількість операторів, що припадають на одну середню функцію або вимогу).

Значення Код — статистичне.

**Запитання:**

1.    Навести рівні цілісності програмного забезпечення.

2.    Як проаналізувати процес підвищення якості (Quality Improvement).

3.    Надати визначення поняттю метрики як основи вимірювання.

4.    Оцінка якості коду в програмі.

5.    Описати призначення Loc-оцінок якості, визначити їх роль при забезпеченні контролю якості програмних продуктів.

6.    Метрика стилістики і зрозумілості програм.

7.    Проаналізувати метрики Холстеда, визначити їх роль при забезпеченні контролю за якістю програмного забезпечення.

8.    Метрики циклічної складності за Мак-Кейбом.

9.    Метрики Чепіна.

10. Попередня оцінка якості на основі статистичних методів в залежності від етапу розробки програми.

11. Попередня оцінка складності програми на етапі розробки специфікацій вимог до програми.