МІНІСТЕРСТВО КУЛЬТУРИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ   
КУЛЬТУРИ І МИСТЕЦТВ

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

**РЕФЕРАТ НА ТЕМУ:**

ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Студента 3-го курсу

групи КН-22

Педини Ігоря Васильовича

Перевірив: Булига Костянтин Борисович

КИЇВ 2015

**ЗМІСТ**

[ВСТУП 3](#_Toc418956241)

[РОЗДІЛ 1. ОСНОВИ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (SOFTWARE QUALITY FUNDAMENTALS) 4](#_Toc418956242)

[1.1 КУЛЬТУРА І ЕТИКА ПРОГРАМНОЇ ІНЖЕНЕРІЇ (SOFTWARE ENGINEERING CULTURE AND ETHICS) 4](#_Toc418956243)

[1.2 ЗНАЧЕННЯ І ВАРТІСТЬ ЯКОСТІ (VALUE AND COSTS OF QUALITY) 4](#_Toc418956244)

[1.3 МОДЕЛІ І ХАРАКТЕРИСТИКИ ЯКОСТІ (MODELS AND QUALITY CHARACTERISTICS) 5](#_Toc418956245)

[1.4 ПІДВИЩЕННЯ ЯКОСТІ (QUALITY IMPROVEMENT) 8](#_Toc418956246)

[РОЗДІЛ 2. ВИЗНАЧЕННЯ ЯКОСТІ ПЗ 11](#_Toc418956247)

[2.1 ТРИ АСПЕКТИ ВИЗНАЧЕННЯ КОСТІ 11](#_Toc418956248)

[2.1 ПІДТВЕРДЖЕННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (SOFTWARE QUALITY ASSURANCE, SQA) 14](#_Toc418956249)

[ВИСНОВОК 17](#_Toc418956250)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 18](#_Toc418956251)

# ВСТУП

Протягом багатьох років окремі автори і цілі організації визначали термін «якість» по-різному. Філ Кросбі (Phil Crosby) в 1979 році дав визначення якості як «відповідність користувальницьким вимогам». Уотс Хемпфри (Watts Hamphrey, оригінальний автор концепції моделі оцінки зрілості CMM, а також PSP і TSP – People Software Process і Team Software Process) описує якість як «досягнення відмінного рівня придатності до використання». Компанія IBM, в свою чергу, ввела в обіг фразу «якість, як кероване ринковими потребами» («market-driven quality»). Критерій Белдріджа (Baldrige) для організаційної якості використовує схожу фразу – «якість, яке встановлюється споживачем» («customer-quality driven»), розглядаючи задоволення споживача в якості головного міркування щодо якості. Частіше, поняття якості використовується відповідно до визначення системи менеджменту якості ISO 9001 як «ступінь відповідності властивих характеристик вимогам» (саме так це сформульовано в офіційному перекладі ІСО 9000-2000 «Системи менеджменту якості. Основні положення та словник»). Дані погляди перегукуються із введеним автором «прийнятною якістю», визначуваним не тільки рівнем запитів кінцевих споживачів щодо параметрів створюваного продукту, але і заданим контекстом (обмеженнями проекту). Це не означає, що «прийнятна якість» протиставляється «якості, яка визначена замовником». Звичайно, не варто і проводити паралель «прийнятної якості» з «продуктом другої свіжості». Введення категорії «прийнятності» щодо якості є лише прагматичним поглядом на бажану ступінь досконалості створюваного продукту (послуги), здатну задовольнити користувачів і досяжну в рамках заданих проектних обмежень

Таким чином, прийнятна якість може розглядатися як кількісно виражений компроміс між замовником і виконавцем щодо характеристик продукту, створюваного виконавцем в інтересах вирішення завдань замовника з урахуванням інших обмежень проекту (зокрема, вартістю, що часто іменується як «cost of quality» .

# РОЗДІЛ 1. ОСНОВИ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (SOFTWARE QUALITY FUNDAMENTALS)

Згода, досягнута щодо вимог до якості (в оригіналі – quality requirements), нарівні з чітким доведенням до інженерів того, що становить якість одержуваного продукту, вимагають обговорення і формального визначення багатьох аспектів якості. Інженери повинні розуміти сенс, що вкладається в концепцію якості, характеристики і значення якості щодо розроблюваного або супроводжуваного програмного забезпечення. Важливою ідеєю є те, що програмні вимоги визначають необхідні характеристики якості програмного забезпечення, а також впливають на методи кількісної оцінки і сформульовані для оцінки цих характеристик відповідні критерії приймання.

## 1.1 КУЛЬТУРА І ЕТИКА ПРОГРАМНОЇ ІНЖЕНЕРІЇ (SOFTWARE ENGINEERING CULTURE AND ETHICS)

Очікується, що інженери з програмного забезпечення сприймають питання якості програмного забезпечення як частину своєї професійної культури. SWEBOK дає посилання на джерела, що описують здорову культуру програмної інженерії. Етичні аспекти можуть відігравати значну роль у забезпеченні якості програмного забезпечення, культури і щодо інженерів до своєї роботи. IEEE Computer Society і ACM розробили кодекс етики («моральний кодекс» – code of ethics) і професійної практики, що базується на восьми принципах, які допомагають інженерам зміцнити їх відношення до якості і незалежність у вирішенні питань забезпечення гідної якості створюваних програмних продуктів в їх повсякденній роботі.

## 1.2 ЗНАЧЕННЯ І ВАРТІСТЬ ЯКОСТІ (VALUE AND COSTS OF QUALITY)

Поняття «якість», насправді, не настільки очевидне і просте, як це може здатися на перший погляд. Для будь-якого інженерного продукту існує безліч інтерпретацій якості, в залежності від конкретної системи. Характеристики якості можуть знадобитися в тій чи іншій мірі, вони можуть бути відсутні або можуть задавати певні вимоги, все це може бути результатом певного компромісу (що цілком перегукується з розумінням «прийнятної якості», як менш жорсткої точки зору на забезпечення якості, як досягнення досконалості).

Вартість якості (cost of quality) може бути диференційована на вартість попередження дефектів (prevention cost), вартість оцінки (appraisal cost), вартість внутрішніх (internal failure cost), а також зовнішніх збоїв (external failure cost).  
Рушійною силою програмних проектів є бажання створити програмне забезпечення, що володіє певною цінністю (значуще для вирішення певних завдань або досягнення цілей). Цінність програмного забезпечення може виражатися у формі вартості, а може і ні. Замовник, як правило, має своє уявлення про максимальні вартісних вкладення, повернення яких очікується в разі досягнення основних цілей створення програмного забезпечення. Замовник може також мати певні очікування щодо якості. Іноді, замовники не замислюються про питання якості і пов'язаної з ними вартістю. Є характеристики якості чисто декоративними (світоглядної), чи все ж це невід'ємна частина програмного забезпечення? Відповідь, ймовірно, знаходиться десь посередині, як майже завжди буває в таких випадках, і є предметом обговорення ступеня залучення замовника до процесу прийняття рішень і повного розуміння замовником вартості і вигоди, пов'язаної з досягненням того чи іншого рівня якості. В ідеальному випадку, більшість такого роду рішень повинно прийматися процесі роботи з вимогами, проте ці питання можуть підніматися на протязі всього життєвого циклу програмного забезпечення. Не існує якихось стандартних правил того, як саме необхідно приймати такі рішення. Однак, інженери повинні бути здатні представити різні альтернативи (у досягненні різного рівня якості і їх вартість. Тут SWEBOK наводить деякі джерела, в яких більш детально обговорюються питання значущості якості і відповідних характеристик вартості.

## 1.3 МОДЕЛІ І ХАРАКТЕРИСТИКИ ЯКОСТІ (MODELS AND QUALITY CHARACTERISTICS)

В різних джерелах і моделях термінологія характеристик якості програмного забезпечення відрізняється. Кожна модель включає різне число рівнів ієрархії і загальне число розпізнаних характеристик якості. Різні автори створили різні моделі якості зі своїм набором характеристик і атрибутів. Ці моделі можуть бути корисні для обговорення, планування, адаптації і оцінки якості програмних продуктів. ISO/IEC визначає три пов'язаних моделі якості програмного забезпечення (ISO 9126-01 Software Engineering – Product Quality, Part 1: Quality Model) – внутрішня якість, зовнішнє якість і якість в процесі експлуатації, а також набір відповідних робіт з оцінки якості програмного забезпечення (ISO14598-98 Software Product Evaluation).

Управління якістю (software quality management) і якість процесів програмної інженерії (software engineering process quality) мають безпосереднє відношення до якості створюваного програмного продукту. Моделі і критерії оцінки можливостей організацій, що займаються розробкою програмного забезпечення, насамперед стосуються розгляду організації проектних робіт і аспектів управління. Відповідно, вони розглядаються в областях знань SWEBOK «Управління програмною інженерією» і «Процес програмної інженерії».  
Звичайно, неможливо повністю відокремити якість процесу від якості продукту.  
Якість процесу, обговорюване в галузі знань «Процес програмної інженерії», впливають на характеристики якості продукту, які, в свою чергу, відображаються в сприйнятті якості продукту у процесі експлуатації з боку замовника.  
Існує два найважливіших стандарти в області якості програмного забезпечення. TickIT – стосується розгляду загальної системи менеджменту якості ISO 9001-00 в додатку до програмних проектів (і, зокрема, поєднання такого погляду з положеннями стандарту життєвого циклу ISO 12207) і представлений також у вигляді спеціальних рекомендацій ISO 90003-04 «Software and Systems Engineering – Guidelines for the Application of ISO9001:2000 to Computer Software». Інший важливий стандарт – CMMI, обговорюваний в галузі знань «Процес програмної інженерії», надає рекомендації щодо вдосконалення процесу. (тут не можна не згадати і ISO 15504 «Information Technology – Software Process Assessment», відомий як SPICE – Software Process Improvement and Capability dеtermination, який також розглядається у згаданій галузі знань). Безпосередньо з управлінням якістю пов'язані процесні області (області компетенції) CMMI: забезпечення якості процесу та продукту (process and product quality assurance, категорія процесів CMMI «Support»), перевірка (verification, категорія «Engineering») і атестація (validation, категорія «Engineering»). При цьому, CMMI класифікує огляд (review) аудит (audit) в якості методів верифікації, але не як самостійні процеси, на відміну, наприклад, від стандарту 12207.

Дебати щодо того, який саме стандарт варто використовувати інженерам для забезпечення якості програмного забезпечення – CMMI або ISO 9001, тривають з самого створення цих стандартів. Сьогодні можна сказати про те, що дані стандарти все ж розглядають як взаємодоповнюючі і, що сертифікація по ISO 9001 допомагає в досягненні старших рівнів зрілості за CMMI.

Насамперед, інженери повинні визначити цілі створення програмного забезпечення. У цьому контексті особливо важливо пам'ятати, що вимоги замовника – первинні і містять вимоги у відношенні якості, а не тільки функціональності (функціональні вимоги). Таким чином, інженери відповідальні за витяг вимог до якості, які не завжди представлені явно, а також обговорення їх важливості і ступеня складності їх досягнення. Всі процеси, пов'язані з якістю (наприклад, складання, перевірка та підвищення якості), повинні проектуватися з урахуванням цих вимог і несуть на собі тягар додаткових витрат (як важливу складову частину вартості програмного забезпечення). Стандарт ISO 9126-01 (Software Engineering – Product Quality, Part 1: Quality Model) визначає для двох з трьох описаних у ньому моделей, пов'язані характеристики і суб-характеристики якості, а також метрики, корисні для оцінки якості програмних продуктів.

Розуміння терміну «продукт» розширено включенням всіх артефактів, створених на виході всіх процесів, використовуваних для створення кінцевого програмного продукту. Прикладами продукту є (але не обмежуються цим): повна специфікація системних вимог (System requirements specification), специфікація програмних вимог для програмних компонент системи (software requirements specification, SRS), моделі, код, тестова документація, звіти, що створюються в результаті робіт з аналізу якості. Хоча, найчастіше термін якість використовується у відношенні кінцевого продукту і поведінки системи в процесі експлуатації, хорошою інженерної практикою є вимога до того, щоб відповідність заданим характеристикам якості оцінювалося і для проміжних результатів продуктів життєвого циклу в рамках всіх процесів програмної інженерії.

## 1.4 ПІДВИЩЕННЯ ЯКОСТІ (QUALITY IMPROVEMENT)

До програмної інженерії застосовні теорії і концепції, що лежать в основі вдосконалення якості. Наприклад, запобігання та рання діагностика помилок, постійне вдосконалення (continuous improvement) та увагу до вимогам замовника (customer focus), складові принцип «building in quality». Ці концепції ґрунтуються на роботах експертів з якості, які прийшли до думки, що якість продукту безпосередньо пов'язана з якістю використовуваних для його створення процесів.  
Такі підходи, як TQM (Total Quality Management – загальне управління якістю) PDCA (Plan, Do, Check, Act – Планування, Дія, Перевірка, Реакція/Коригування), що є інструментами досягнення завдань, пов'язаних з якістю. Підтримка менеджменту допомагає у виконанні процесів, оцінки продуктів і отримання всіх необхідних даних. Крім цього, розробляється програма вдосконалення (improvement program, зазвичай є цільовою і охоплює роботу підрозділу чи організації в цілому) детально ідентифікує всі дії і проекти по поліпшенню окремих аспектів діяльності в рамках певного періоду часу, за який такі проекти можна здійснити з успішним вирішенням відповідних завдань. При цьому, підтримка менеджменту означає, що всі проекти по поліпшенню володіють достатніми ресурсами для досягненням поставлених цілей. Підтримка менеджменту тісно пов'язана з реалізацією активної взаємодії в колективі, і повинна попереджати виникнення потенційних проблем (і пасивного або навіть активної протидії реалізації програми вдосконалення або окремих її проектів). Формування робочих груп, підтримка менеджерів середньої ланки та виділені ресурси на рівні проекту – ці питання обговорюються в галузі знань «Процес програмної інженерії».

Засоби для поліпшення функціональної якості включають ручні інструменти тестування, які дозволяють досліджувати тестер програмного забезпечення через інтерфейс користувача, а також інструменти для автоматизованого тестування, таких як фрейм ворки для модульного тестування. Інструменти для навантажувального тестування та тестування продуктивності також може допомогти виміряти і поліпшити ті компоненти функціонального якості.

Інструменти, які допомагають покращити структурні якості програмного забезпечення надають такі послуги, як рефакторинг, який дозволить розробнику поліпшити код організований без зміни чого-небудь. Структурна якість може також надати інструменти статичного аналізу коду, код для вивчення проблем безпеки (таких як потенціал для атак шляхом впровадження коду SQL) і інші проблеми, поряд з динамічного аналізу коду, які можуть включати в себе профілювання продуктивності, заходи тестового покриття, і багато іншого. Ці інструменти можуть також забезпечити різні метрики коду, такі як вимірювання циклометричної складності.

Інструменти для поліпшення якості процесу допомагають відстежувати і керувати процесом розвитку. Вони включають підтримку для відстеження статусу процесу, можливо шляхом зіставлення вимог щодо прогресу вимірювань для розробника, відповідальних за кожний з них. Процес якісного інструменту може також дати подання обробленого коду, тобто число рядків, доданих або змінених щотижня, прогресу в пошуку і виправлення помилок, хід виконання плану тестування та інших заходів проекту здоров'я.

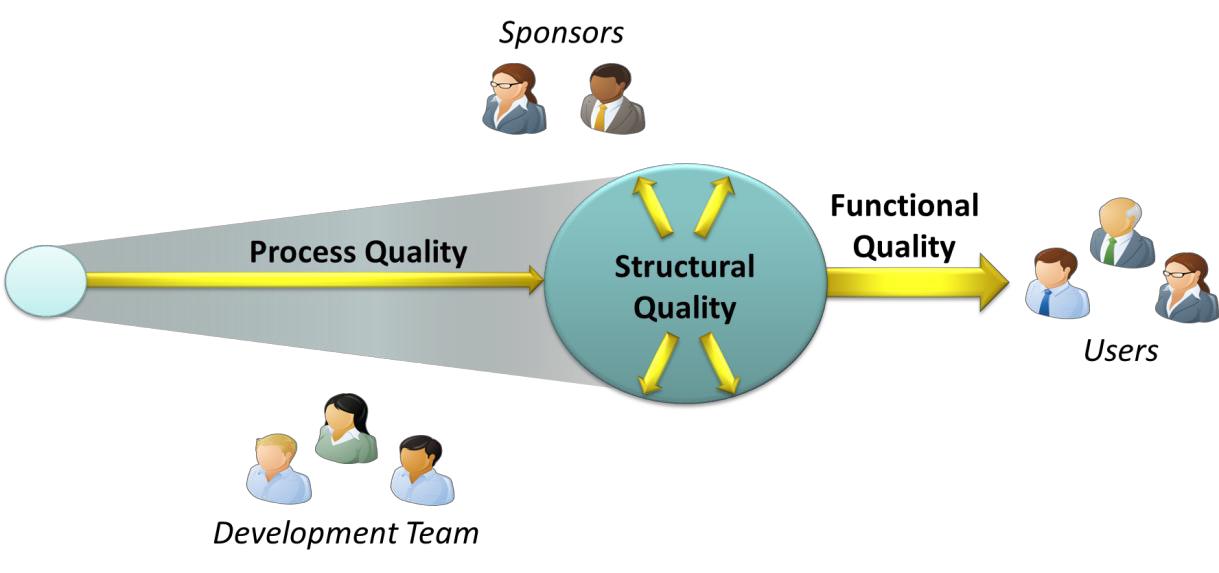
Що б вони не робили, важливо розуміти, що на відміну від інструментів для функціонального якості та структурної якості, які зазвичай використовуються виключно командою розробників, інструменти для якості процесу використовуються також в рамках проекту спонсорів (і, можливо, навіть з допомогою програмного забезпечення користувачів). Це означає, що ці інструменти повинні бути доступні через менш технічно орієнтованих інтерфейсів, таких як електронні таблиці і програмне забезпечення для спільної роботи. Роблячи їх доступними тільки через інструменти розробника не достатньо.

Інструменти не вся історія, звичайно. Такі заходи, як група перевірки коду та ефективного управління може також мати великий вплив на різні аспекти якості програмного забезпечення – люди матерії. Ще як і з кожним іншим аспектом розробки програмного забезпечення, використовуючи хороші інструменти, безумовно, допомагає.

# РОЗДІЛ 2. ВИЗНАЧЕННЯ ЯКОСТІ ПЗ

## 2.1 ТРИ АСПЕКТИ ВИЗНАЧЕННЯ КОСТІ

Немає єдиного правильного шляху, щоб задуматися про якість програмного забезпечення – це складна область. Однак, згрупувати різні компоненти можна за трьома розділами.



Мал 1. Якість програмного забезпечення можна розглядати у вигляді трьох аспектів: функціональний, структурний, якість і процес

Трьома аспектами якості програмного забезпечення є функціональна якість, структурна якість, якість і процес. На кожному з них варто зупинитися більш докладно.

**Функціональна якість** означає, що програма коректно виконує завдання, які перед нею ставляться, і які треба зробити для своїх користувачів. Серед атрибутів функціональні якості є:

**Засідання зазначеної вимоги**. Чи вони надходять від спонсорів проекту або програми, призначені користувачам, конференційні вимоги є неодмінною умовою функціональні якості. В деяких випадках, це може навіть включати дотримання чинних законів і положень. І оскільки вимоги часто міняли протягом всього процесу розробки, досягнення цієї мети вимагає розвитку команди, щоб зрозуміти і впровадити відповідну вимогам, які були спочатку визначені для проекту. Створюючи програмне забезпечення, яке має декілька дефектів. Серед них помилки, які знижують надійність програмного забезпечення, які можуть порушити його безпеку або обмежити його функціональність. Досягнути того, щоб програма не мала дефектів дуже складно, користувачі рідко бувають задоволені таким програмним забезпеченням, вони сприймають як баги.

**Досить гарна продуктивність**. Легкість навчання і простота використання. Для користувачів, програмне забезпечення, інтерфейс додатка, і тому ці атрибути функціональної якості найбільш часто забезпечують ефективний інтерфейс та добре продуманий користувальницький робочий процес. Естетика інтерфейсу – наскільки це красиво – теж може бути важливою, особливо в споживчих додатках. Тестування програмного забезпечення зазвичай орієнтований на функціональні якості. Всі, перелічені характеристики можуть бути випробувані, принаймні до певної міри, і тому значна частина забезпечення функціонального якості зводиться до тестування.

Другий аспект якості програмного забезпечення, **структурна якість**, означає, що сам код добре структурований. На відміну від функціональних, якісних, структурну якість складно перевірити (хоча є інструменти, щоб допомогти виміряти її, як описано далі).

Атрибути цього типу якості включають:

* Тестування коду. Код організований таким чином, що робить тестування легким
* Код ремонтопридатність. Наскільки легко додати новий код або змінити існуючий код без появи помилок
* Зрозумілість коду. Код читається. Він є більш складним, ніж він повинен бути. Вони мають великий вплив на те, як швидко нові розробники можуть почати роботу з існуючою базою коду.
* Ефективність коду. Особливо в умовах обмежених ресурсів ситуаціях, написання ефективного програмного коду може бути критично важливою.
* Код безпеки. Надає програма можливість поширених атак, таких як переповнення буфера і SQL-ін'єкції? Вона невпевнено іншими способами?

Як функціональні, якісні і структурні якості є важливими, і вони зазвичай отримують левову частку уваги у дискусіях якості програмного забезпечення.

**Третій аспект, якість процесу**, також є критично важливим. Якість процесу розробки істотно впливає на значення одержуваних користувачами, розвиток команди, і спонсори, і так усі три групи мають частку в поліпшенні даного аспекту якості програмного забезпечення. Найбільш очевидні ознаки якості процесу включають дані:

Дотримання терміну постачання. Щоб програмне забезпечення було доставлено вчасно. Програмного забезпечення, що надається за очікувану суму грошей. Повторюваний процес розробки, що надійно забезпечує якість програмного забезпечення. Якщо процес має перші дві характеристики – програмне забезпечення доставлений вчасно і в рамках бюджету – але так підкреслює розвиток команди, її кращих членів кинути, це не якість процесу. Правда якість процесу означає бути несуперечливим від одного проекту до іншого.

Існує безліч зв'язків між цими трьома аспектами якості програмного забезпечення. Наприклад, процес поліпшення якості з гнучкої розробки методів підвищує шанси проекту вимогам права, які також покращують функціональні якості. Є компроміси, де поліпшенню якості в одній області, можуть знизити якість в іншому. Організації можуть прискорити процес розробки проекту, щоб укластися в термін – поліпшення якості процесу – тільки, щоб знайти, що кількість помилок у програмному забезпеченні збільшилися. Аналогічно, ріжучі характеристики можуть знизити функціональні якості, оскільки користувачі отримують менше того, що вони шукають, але підвищення якості процесу за рахунок збільшення коефіцієнтів зустрічі дату і релізу. Загалом, кожен проект розвитку впливає на інтереси всіх трьох груп – і всі три аспекти якості – один проти одного. Різні проекти роблять різні компроміси. Не дивно, що кожен, хто бере участь у програмному проекті піклується найбільше про аспекти якості, які безпосередньо впливають на них. Користувачі піклуватися в першу чергу про функціональну якість, так це те, що вони бачать. Вони також, ймовірно, можуть піклуватися про деякі аспекти процесу якості, такі як дата поставки кінцевого програмного забезпечення. Користувачі зазвичай не піклуються про структурну якість, хоча її відсутність цілком може впливати на них через програмне забезпечення життя.

Команда розробників, звичайно ж, турбується про структурну якість, однак, оскільки вони саме ті люди, які будуть порушені проблеми, викликані низькою якістю. Вони також дбають про функціональну якість, хоча, можливо, трохи менше, ніж користувачі. Розвиток команд також хвилює якість процесу, в тому, що він надає безліч показників, з допомогою яких вони виміряються.  
Третя група, спонсори, піклується про все: функціональну якість, структурну якість, якість і процес. Якщо вони розумні, люди платять за проект, знають, що в будь-яку область потрібно вкладати кошти. Зрештою, спонсори прагнуть створювати цінність для бізнесу, і кращий спосіб зробити це, беручи широке тлумачення якості програмного забезпечення. Вони повинні також зрозуміти зв'язок між якістю та ризиками. Ризик, пов'язаний з прийомом низької, якості програмного забезпечення, скажімо, громадський інтернет-сайт, містить набагато менше ризику, що дозволяє знизити якість польоту літака системи управління. Роблячи вибір відповідно часто вимагає компромісів між конкуруючими цілями.

## 2.1 ПІДТВЕРДЖЕННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (SOFTWARE QUALITY ASSURANCE, SQA)

Процеси SQA забезпечують підтвердження того, що програмні продукти та процеси життєвого циклу проекту відповідають заданим вимогам. Таке підтвердження здійснюється на основі планування (planning), постановки робіт (enacting) і виконання (performing) набору дій, спрямованих на те, щоб якість стало невід'ємною частиною програмного забезпечення. Такий погляд означає ясне і точне формулювання проблеми, а також те, що визначено і чітко виражені вимоги до відповідного програмного рішенням. SQA домагається забезпечення якості в процесі розробки і супроводу за рахунок виконання різних дій на всіх етапах життєвого циклу, що дозволяє ідентифікувати проблеми ще на ранніх стадіях, які практично неминучі в будь-якій складній діяльності.

Така ідентифікація можлива у багатьох випадках (якщо навіть не в більшості ситуацій), коли проблема ще є ризиком. Це – завдання управління ризиками, яке цілком можна було б винести в якості самостійної галузі знань SWEBOK, в силу вже досить великого сукупного досвіду не тільки індустрії ІТ або дисципліни управління проектами. Так чи інакше, можна сказати, що вже було підкреслено при обговоренні SQM, управління ризиками (Risk Management) є серйозним додатковим інструментом для забезпечення якості програмного забезпечення. Однак, обмежуватися згадкою управління ризиками тільки в контексті SQM було б неправильно, так як сьогоднішнє розуміння Risk Management включає в себе не тільки питання попередження ризиків, але і управління процесом вирішення проблем. SQA, як це сформульовано SWEBOK, концентрується на процесах. Роль SQA полягає в тому, щоб забезпечити відповідне планування процесів, подальше виконання процесів на основі заданого плану і проведення необхідних вимірювань процесів з передачею результатів вимірювань зацікавленим сторонам (організаційними структурам і особам). SQA-план визначає засоби, які будуть використовуватися для забезпечення відповідності розроблюваного продукту заданим призначеним для користувача вимогам з максимальним рівнем якості, можливим при заданих обмеженнях проекту (тобто, в термінології автора – прийнятним рівнем якості, прим. автора ). Для того, щоб цього досягти, в першу чергу необхідно, щоб цілі були чітко визначені і зрозумілими (а також, однозначно інтерпретуючими, що є обов'язковою умовою будь-яких цілей і відповідних вимог, прим. автора). Це, в обов'язковому порядку, повинно бути відображено у відповідних планах управління проектом, розробки і супроводу. Подробиці можна знайти в стандарті IEEE 730-02 «IEEE Standard for Software Quality Assurance Plans». Конкретні роботи та завдання по забезпеченню якості структуруються з деталізацією вимог щодо їх вартості та асоційованим ресурсів, цілям з точки зору управління та відповідним розкладом в контексті цілей, визначених планами управління, розробки і супроводу. SQA-план повинен узгоджуватися з планом конфігураційного управління (див. область знань «Software Configuration Management»). План SQA ідентифікує документи, стандарти, практики і угоди, що застосовуються при контролі проекту, а також те, як ці аспекти будуть перевірятися і відслідковуватися для забезпечення достатності і відповідності заданим планом. Також, SQA-план ідентифікує метрики, статистичні техніки, процедури формування повідомлень про проблеми і проведення коригувальних дій, такі кошти (в оригіналі SWEBOK використовується термін resources, прим. автора), як інструменти, техніки та методології, питання безпеки фізичних носіїв (це, швидше, питання базової інфраструктури проектів, а не SQA-плану, прим. автора), тренінги, а також формування звітності та документації, що відносяться до питань SQA. Крім того, SQA-план стосується і питань робіт по забезпеченню якості, що відносяться до інших видів діяльності, перелічених у <різних> плани по створенню програмного забезпечення, до яких також відносяться постачання, установка, обслуговування (підтримка та супровід, прим. автора) замовних та/або тиражованих/готових програмних рішень (Commercial off-the-shelf, COTS), необхідних для даного проекту програмного забезпечення. Нарешті, SQA-план може містити необхідні для забезпечення якості критерії приймання програмного забезпечення і дії по формуванню звітності та управління і контролю над роботами.

# ВИСНОВОК

Не існує єдиного кращого способу підвищити якість програмного забезпечення – різні точки зору підкреслюють різні речі. Просте розділення описано тут мається на увазі надати один корисний спосіб думати про цю тему.  
Шляхом забезпечення рівних аспектів на функціональну якість, структурну якість, і якість процесу, цей підхід допомагає розширити наш погляд, допомагає включати речі, які важливі для всіх трьох зацікавлених сторін: користувачів, розвитку команди, і спонсорів. Включаючи всі три зони також допомагає нам думати про те, які інструменти потрібні нам, щоб поліпшити якість програмного забезпечення.

Таким чином, можна сказати, що якість програмного забезпечення – це характеристика [програмного забезпечення](http://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), ступінь відповідності ПЗ до [вимог](http://uk.wikipedia.org/wiki/%D0%92%D0%B8%D0%BC%D0%BE%D0%B3%D0%B8_%D0%B4%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F). При цьому вимоги можуть трактуватись по-різному, що породжує декілька незалежних визначень терміну. Якість ПЗ – набір властивостей продукту (сервісу або програм), що характеризують його здатність задовольнити встановлені або передбачувані потреби замовника. Поняття якості має різні інтерпретації залежно від конкретної програмної системи і вимог до неї.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Рекомендации по преподаванию программной инженерии и информатики в университетах.–Computing Curricula-2001: Computer Science.–Пер. с англ. – Интернет– Ун. информац. технологий, М.: 2007.–462с.
2. Бабенко Л.П., Лавріщева К.М. Основи програмної інженерії.– Навч. посібник.–К.: Знання, 2001. –269 с.
3. Лаврищева Е.М., Грищенко В.Н. Области знаний программной инженерии – SWEBOK и подход к обучению этой дисциплине// Управляющие системы и машины.–2005. – №1.– С.38–54.
4. Pfleeger S.L. Software Engineering. Theory and practice. – Printice Hall: Upper Saddenle River, New Jersey, 1998. – 576 p.
5. Jacobson I. Object-Oriented Software Engineering. A use Case Driven Approach, Revised Printing. – New York: Addison-Wesley Publ. Co., 1994. – 529 p.