**Додаткові вимоги**

## СПЕЦИФІКАЦІЯ ДОДАТКОВИХ ВИМОГ

Специфікація додаткових вимог(SS - Supplementary Specification) містить загальні нефункціональні вимоги до системи (зручність використання, надійність, продуктивність, здатність до супроводу, тощо). Нагадаємо, що специфікація сценаріїв використання також містить нефункціональні вимоги, якщо вони застосовуються тільки до одного сценарію використання.

Деякі функціональні вимоги теж можуть бути представлені в документі SS:

* вимоги, які поширюються за межі системи, і внаслідок цього, складні для відображення в сценаріях використання (запитати підказку через соціальну мережу);
* вимоги, які є простими і їх подання у вигляді сценаріїв варіантів використання є недоречним (вмикання чи вимикання звуку);
* вимоги, які не пов'язані з певним сценарієм використання (друк, он-лайн довідник, звіти).

Нефункціональні вимоги та функціональні вимоги до системи, які містяться в документі SS є додатковими вимогами (SUPL).

## ЗБІР ДОДАТКОВИХ ВИМОГ

* Збір додаткових вимог проводиться на етапі Збору вимог (інтерв'ю, анкетування, семінар);
* Додаткові вимоги можуть бути задокументовані окремо-новий тип вимоги - Додаткові запити зацікавлених осіб (Supplementary Stakeholder's Request - SSTRQ);
* Потребують додаткових пояснень від Розробників Замовнику про суть вимог, їх вартості та наслідків.

Як для визначення функціональних, так і для визначення нефункціональних вимог використовуються робочі групи, члени яких визначають, перевіряють і затверджують вимоги. Для виявлення нефункціональних вимог особливо важливо залучити до цієї роботи не тільки аналітиків і користувачів, але й архітекторів і ключових розробників продукту або системи, а також групу тестування. Архітектор сприймає нефункціональні вимоги як вхідні дані для вибору і проектування архітектури системи, а група тестування планує ті сценарії навантажувального тестування, які будуть використовуватися для перевірки виконання нефункціональних вимог.

Ролі, які при цьому відіграють учасники робочої групи з визначення нефункціональних вимог такі:

* *Користувачі* - дають оцінки значень параметрів, які використовуються для визначення нефункціональних вимог. Параметри, як правило, прив'язані до сценаріїв використання, в яких користувачем повинні виконуватися певні дії з певними обмеженнями за певний час.
* *Системний аналітик* - збирає, аналізує і документує і систематизує нефункціональні вимоги.
* *Системний архітектор, ключові розробники* - беруть участь у визначенні та аналізі нефункціональних вимог і перевіряють їх на можливість реалізації.
* *Група тестування* - бере участь у визначенні та аналізі нефункціональних вимог і розробляє сценарії тестування для перевірки нефункціональних вимог.

## КЛАСИФІКАЦІЯ ДОДАТКОВИХ ВИМОГ FURPS+

Додаткові вимоги можна класифікувати за різними методиками. В RUP застосовується класифікація FURPS+, яку розробив Роберт Грейді (Robert Grady) із Hewlett-Packard 1992 р.

Нижче наводиться класифікація вимог за FURPS+.

**Функціональність**

**F** - Functionality - - функціональні вимоги, які висунуті до системи і описують її базові властивості.

Цей пункт містить функціональні вимоги, що не були відображені в жодному сценарії використання. Він звичайно включає деякі загальні функції, доступні в багатьох частинах системи. Наприклад: друк, он-лайн довідник, звіти. Тим не менш, якщо функціональність більш складна і не може бути виражена в парі пропозицій, може знадобитися створення додаткового варіанту використання для її опису.

В Додатку наведена таблиця 1 із розбиттям функціональних вимог на підкатегорії.

Приклад: Он-лайн довідник повинен бути доступний з меню на кожній сторінці.

Приклад: В системі повинен бути довідник ролей користувачів (Касир, Адміністратор складу, Системний адміністратор). В системі повинен бути довідник користувачів. У системного адміністратора повинна бути можливість реєстрації користувача і призначення користувачеві ролі.

**Зручність користування**

**U** – Usability **–** визначають скільки зусиль потрібно прикласти користувачеві для введення даних, експлуатації та виведення кінцевої інформації (описують інтерфейс користувача, діалог із користувачем, процедури введення даних).

Категорії вимог: естетика і логічність користувальницького інтерфейсу, захист від людського фактору, експлуатаційна документація, її склад (керівництва користувачів, адміністраторів та ін), галузеві та державні стандарти оформлення, кваліфікація користувачів та їх навчання, довідкова інформація в системі.

Приклад: Поля введення на одній сторінці повинні бути вирівняні вертикально.

В Додатку наведена таблиця 1 із розбиттям вимог до зручності користування на підкатегорії та відповідними прикладами.

**Надійність**

**R** – Reliability- визначають припустиму періодичність, тривалість і серйозність збоїв, можливість відновлення системи після них, точність обчислень, режим роботи, ступінь передбачуваності поведінки системи.

В Додатку наведена 1 таблиця із розбиттям вимог до надійності на підкатегорії та відповідними прикладами.

Приклад: Після випуску релізу система не повинна мати критичних і важливих помилок, допускається не більше ніж 20 незначних помилок.

Приклад: Грошові розрахунки повинні виконуватися і зберігатися з точністю до двох десятих.

**Продуктивність**

**P** - Performance **–** визначають потужність та ефективність системи.

Приклад вимог: швидкість обробки інформації, час відповіді; наскільки потужною є система: припустима кількість користувачів, об’єм інформації, що пересилається за 1 раз, час відновлення, час завантаження/виходу, ємність, використання ресурсів (можуть бути також описані в Implementation Requirements).

В Додатку наведена таблиця 1 із розбиттям вимог до продуктивності на підкатегорії та відповідними прикладами.

Приклад: Система повинна обслуговувати 5000 користувачів одночасно.

Приклад: Система повинна зберігати в базі даних не більше ніж один мільйон транзакцій. Якщо база даних перевищила ліміт, старі транзакції повинні бути відправлені в резервне сховище і потім видалені з операційної бази даних. Ці вимоги можуть бути також описані в Implementation Requirements (Вимоги Реалізації).

**Здатність до Супроводу**

S – Supportability – характеристики системи, щодо можливостей її масштабування та розширення, сумісності з іншими системами, переносимості, характеристики з обслуговування та супроводу, можливість багаторазового застосування, локалізація і т.п.

В Додатку наведена таблиця 1 із розбиттям здатності до супроводу на підкатегорії та відповідними прикладами.

Приклад: Після виходу системи наступні версії системи повинні бути сумісні зверху вниз. Всі введені в попередній версії транзакції повинні бути доступні в новій версії.

Приклад: Додаток повинен підтримувати Англійська, французька та іспанська мови.

**Обмеження**

+ - Contrains - вимоги, які обмежують розробників при виборі засобів.

Приклад вимог: реалізація коду, вимоги інтерфейсу, формати вхідних та вихідних даних, вимоги до апаратного забезпечення, вимоги документації, вимоги ліцензій та юридичних норм.

В Додатку наведена таблиця 1 із розбиттям цього типу вимог на підкатегорії та відповідними прикладами.

Приклад: Керівництво Адміністратора має бути доступне у форматі PDF.

Приклад: Система повинна бути заснована на архітектурі J2EE.

Жодна з перерахованих категорій не є обов'язковою. Багато з них можуть бути пропущені, якщо вони не підходять до конкретної системи. Тим не менш, вимога повинна виключатися тільки в результаті запланованого рішення замовника та системного аналітика, а не тому що його важливість не була проаналізована.

## КРИТЕРІЇ ЯКІСНИХ НЕФУНКЦІОНАЛЬНИХ ВИМОГ

Як до функціональних, так і до нефункціональних вимог застосовуються критерії якості вимог - тобто опис тих характеристик, яким повинні задовольняти якісні вимоги.

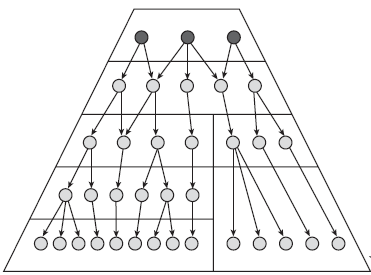
Нижче наведені основні характеристики якісних вимог.

* **Повнота** (окремої вимоги і системи вимог) - вимога повинна містити всю необхідну інформацію для її реалізації. У неї включається вся інформація про описуваний параметр, яка відома на момент опису. Система вимог також не повинна містити невиявлених і не визначних вимог.
* **Однозначність** - вимога повинна бути внутрішньо несуперечливо і всі, хто з нею працює, повинні розуміти її однаково. Вимоги слід висловлювати просто, стисло і точно, використовуючи відомі терміни. Зазвичай базові знання читачів специфікації вимог до ПЗ різняться. Тому в її склад потрібно включити розділ з визначенням понять прикладної області, які використовувалися при визначенні вимог, або подавати їх у Словнику.
* **Коректність** окремої вимоги і узгодженість (несуперечність) системи вимог - вимога не повинна містити в собі невірної, неточної інформації, а окремі вимоги в системі вимог не повинні суперечити одна одній.
* **Необхідність** - вимога повинна відображати можливість або властивості ПЗ, дійсно необхідну користувачам.
* **Здійсненність** – вимога, що включається в специфікацію, має бути здійсненною при заданих обмеженнях операційного середовища. Здійсненність вимог перевіряється в процесі аналізу здійсненності розробником. Зокрема, для нефункціональних вимог перевіряється можливість досягнення зазначених чисельних значень при існуючих обмеженнях.
* **Тестованість** - означає, що існує кінцевий і розумний за вартістю процес ручної або машинної перевірки того, що ПЗ задовольняє цій вимозі. Кожна вимога (особливо нефункціональна) повинно містити достатньо інформації для однозначної перевірки її реалізації. Це може бути наявність чисельних значень характеристик якості продукту або системи.
* **Трасованість** – зв'язок вимоги із вимогами інших рівнів деталізації.

Якість нефункціональних вимог безпосередньо визначає якість розроблюваного продукту або системи і досягається за рахунок ітеративного процесу визначення та аналізу нефункціональних вимог при злагодженій роботі всієї групи, яка бере участь у їх розробці.

## ТРАСУВАННЯ ВИМОГ

Існує три рівні вимог, які найчастіше зустрічаються в проектах (рис. 1). Зведений опис вимог, які відповідають цим рівням, подано у таблиці 1.



ПОТРЕБИ

ФУНКЦІОНАЛЬНІ

ОСОБЛИВОСТІ

USE CASES (СЦЕНАРІЇ ВИКОРИСТАННЯ)

СЦЕНАРІЇ

(АЛГОРИТМЫ)

ТЕСТОВІ СЦЕНАРІЇ

ДОДАТКОВІ ВИМОГИ

ТЕСТОВІ СЦЕНАРІЇ

Рисунок 1.Пираміда вимог

Таблиця 1. Зведений опис вимог

|  |  |  |
| --- | --- | --- |
| **Елемент Трасування**  **(Тип Вимоги)** | **Артефакт (Тип Документа)** | **Опис** |
| Stakeholder Requests (STRQ - Запити Зацікавлених Осіб) | STR - Ключові запити зацікавлених осіб | Вимоги високого рівня |
| Feature (FEAT - Функціональна Особливість) | Vision – Концепція (Бачення) | Функціональні можливості та властивості системи |
| Use Case (UC - Варіант Використання - ВВ) | Use Case (UC) – Короткий опис Варіантів використання | Варіанти використання, що містять функціональні вимоги |
| Supplementary Requirement (SUPL - Додаткова Вимога) | Supplementary Specification (SS) - Додаткова Специфікація | Нефункціональні вимоги, які не відображені в моделі ВВ та функціональні вимоги, які не відображені у ВВ |

**Трасування** - це спосіб представлення відношень між вимогами різного рівня в системі. Воно допомагає визначити джерело будь-якої вимоги.

Кожна потреба (STRQ) відповідає кільком функціональним особливостям (FEAT). Зазвичай це відношення «багато-до-багатьох», оскільки одна потреба може трасуватися до багатьох функціональних особливостей, але ж і з кількох потреб може бути отримана одна функціональна особливість. Одна потреба (STRQ) що відповідає одній функціональній особливості (FEAT) - це також загальний випадок.

Функціональні особливості (FEAT) відповідають сценаріями використання (UC) у відношенні «багато-до-багатьох». Функціональні особливості (FEAT) також відповідають додатковим вимогам (SUPL) у відношенні «багато-до-багатьох».

Рисунок 2 показує структуру трасування, яку можна використати в проекті.

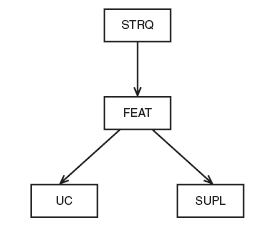
****

Рисунок 2. Діаграма трасування

* STRQ (Потреби Зацікавлених Осіб) будуть трасуватися в FEAT (Функціональні Особливості), визначені в документі Vision (Баченні).
* FEAT (Функціональні Особливості) будуть трасуватися в UC (Сценарії) або в SUPL (Додаткові Вимоги). Кожна підтверджена функціональна особливість може бути трасована, принаймні, в один UC або SUPL. Також можуть бути відносини багато-до-багатьох між FEAT, UC і SUPL.

Головне **призначення** трасування полягає в наступному:

* Підтвердити, що реалізація відповідає вимогам (вся необхідна функціональність була реалізована і протестована).
* Підтвердити, що система робить тільки те, для чого вона призначена (вся реалізована функціональність була запитана замовниками).
* Допомогти в управлінні змінами (аналізувати наслідки зміни вимог – як вплине зміна вимоги на пов’язані з нею вимого інших рівнів).

В таблицях 2, 3, 4 наведено приклади **матриць трасування** вимог, де K, N, L, M – кількість вимог типу STRQ, FEAT, UC, SUPL відповідно.

Таблиця 2. Матриця трасування вимог STRQ в FEAT

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | STRQ 1:  Зміст вимоги | STRQ 2  Зміст вимоги | STRQ 3 | STRQ … | STRQ K |
| FEAT 1:  Зміст вимоги | х |  | - | - | - |
| FEAT 3  Зміст вимоги |  | х |  |  |  |
| FEAT …  Зміст вимоги |  |  |  |  |  |
| FEAT N  Зміст вимоги |  |  |  |  |  |

Таблиця 3. Матриця трасування вимог FEAT в UC

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FEAT 1  Зміст вимоги | FEAT 2  Зміст вимоги | FEAT 3  Зміст вимоги | FEAT …  Зміст вимоги | FEAT N  Зміст вимоги |
| UC 1  Назва прецеденту | х | х |  |  |  |
| UC …  Назва прецеденту |  |  |  |  |  |
| UC L  Назва прецеденту |  | х |  |  |  |

Таблиця 4. Матриця трасування вимог FEAT в SUPL

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FEAT 1  Зміст вимоги | FEAT 2  Зміст вимоги | FEAT 3  Зміст вимоги | FEAT …  Зміст вимоги | FEAT N  Зміст вимоги |
| SUPL 1  Зміст вимоги |  |  |  |  |  |
| SUPL 2  Зміст вимоги |  |  |  |  | х |
| SUPL …  Зміст вимоги |  |  |  |  |  |
| SUPL M  Зміст вимоги |  |  |  |  |  |

Та ж інформація може бути відображена у формі **дерева** трасування (таблиця 5).

Таблиця 5. Приклад дерева трасування FEAT в UC

|  |  |
| --- | --- |
| FEAT 2: | Зміст вимоги |
| UC 1: Назва прецеденту (Зміст вимоги)  UC L: Назва прецеденту (Зміст вимоги) |

Подання трасування (зв'язків) допомагають швидко знайти певну проблему:

* Сценарій Використання (UC) або Додаткова Вимога (SUPL) не трасуються ні з яких вимог (це означає, що ви реалізували щось, що не було запитано замовниками).
* Функціональна Особливість (FEAT) не трасується ні в якій UC або SUPL (це означає, що Ви не реалізували щось із замовником необхідної функціональності).

Додаток

Таблиця 1.

**Таблиця із поясненнями та прикладами додаткових вимог за класифікацією FURPS+**

|  |  |  |
| --- | --- | --- |
| Категорія | Підкатегорія | Пояснення, приклади |
| Functionality | журналювання, аuditing | інструменти відстеження дій користувачів і системи шляхом запису в журнал безпеки конкретних типів подій. |
| ліцензування, licensing | засоби для відстеження, придбання, встановлення та контролю над використанням ліцензій. |
| локалізація, localization | засоби підтримки різних природних мов. |
| пошта, mail | служби відправки та отримання повідомлень. |
| допомога, online help | можливість надавати підтримку користувачів в реальному часі  (наприклад, "Необхідна система online допомоги"). |
| друк, printing | засоби для друку документів. |
| звітність, reporting | інструменти створення і отримання звітів. |
| безпека, security | засоби захисту доступу до певних ресурсів інформації. |
| управління системою, system management | інструменти, що дозволяють керувати програмами в розподіленому середовищі. |
| технологічний процес, workflow | підтримка документообігу, включаючи процеси перевірки, візування та затвердження. |

|  |  |  |
| --- | --- | --- |
| USABILITY | доступність | Легкість доступу до системи та використання особливої функціональності.  Приклад: Функціональність бронювання квитка на літак повинна бути доступна з домашньої сторінки.  Приклад: Функціональність бронювання автомобіля повинна бути доступна після не більше ніж одного клацання з домашньої сторінки. |
| естетичність | Естетичність користувача інтерфейсу.  Приклад: Поля введення на одній сторінці повинні бути вирівняні вертикально. |
| відповідність інтерфейсу користувача | Приклад: Інтерфейс повинен відповідати стандарту IBM CUA. [CUA91a] [CUA91b]  Щоб уникнути неясності при згадці стандартів, краще надати посилання на джерело з описом стандарту. |
| ергономічність | Ергономічні аспекти користувальницького інтерфейсу (уникнення непотрібних дій, незручних переміщень мишкою і т.д.).  Приклад: При відкритті діалогового вікна акцент повинен бути на першому полі введення. |
| легкість у використанні | Легкість у навчанні та використанні системи.  Приклад: Для використання системи користувачі не повинні володіти спеціальними технічними навичками (крім уміння користуватися браузером).  Приклад: Представник служби послуг повинен мати можливість вивчати систему он-лайн.  Приклад: Середній час процедури бронювання номера в готелі повинно бути не більше десяти хвилин. |

|  |  |  |
| --- | --- | --- |
| RELIABILITY | працездатність | Час доступності системи у відсотках (середній час між помилковими станами).  Приклад: Середній час між відмовами має бути, принаймні, 30 днів.  Приклад: Система повинна бути доступна 99.93% часу. |
| міцність | Здатність системи протистояти зовнішнім порушенням, наприклад невірному введені даних або нестачі ресурсів.  Приклад: Для кожного невірного введення користувачем система повинна відображати відповідне повідомлення про помилку з поясненням, якого формату повинні бути дані, що вводяться. |
| точність | Точність, з якою система розраховує значення.  Приклад: Грошові розрахунки повинні виконуватися і зберігатися з точністю до двох десятих. |
| відновлюваність | Наскільки майстерно система відновлюється зі стану непрацездатності. У цьому пункті ми дбаємо про мистецтво відновлення і відсутності наслідків, в той час як час відновлення описується в пункті Продуктивність. Тим не менш, можна скомбінувати ці два аспекти в одному місці. |
| коректність | Наскільки правильно система повинна працювати (без помилок).  Приклад: При виведенні списку рейсів система не повинна пропускати прямі рейси або рейси з однією зупинкою.  Приклад: Після випуску релізу система не повинна мати критичних і важливих помилок, допускається не більше ніж 20 незначних помилок.  В ідеалі система не повинна мати помилок зовсім. Однак ця мета практично завжди нереальна для досягнення в рамках встановлюваних часових обмежень. |
| відмовостійкість | Чутливість модулів системи до відмов у роботі. |
| безпека (доступу до даних, з’єднання ) | Будь небезпеку по відношенню до користувачів, даних, системним компонентам або системам, представлені в процесі використання системи. |
| захист | Рівень захисту щодо доступу до певних модулів системи. |

|  |  |  |
| --- | --- | --- |
| PERFORMANCE | швидкість обробки інформації | Час, за який система виконує завдання. Вона може бути виражена, наприклад, в кількості транзакцій на хвилину.  Приклад: Система повинна обробляти 1000 процедур бронювання квитків на хвилину. |
| час відповіді | Наскільки швидко система відповідає на запити.  Приклад: Середній час відповіді системи має бути менше двох секунд. |
| час відновлення | Наскільки швидко система відновлюється зі стану непрацездатності. Важливо розрізняти час, коли система стає робочої з точки зору користувача (зазвичай тому що резервна система дозволяє виконання різних операцій), і час, коли проблема буде фактично виправлена. Приклад: У разі відмови системи резервна система повинна дозволяти виконання операцій протягом 30-ти секунд.  Приклад: Середній час відновлення повинно бути менше години. |
| час завантаження або вимкнення | Тривалість часу завантаження системи і її вимкнення.  Приклад: Система повинна бути працездатною протягом однієї хвилини після завантаження. |
| ємність | Кількість користувачів, яке система може обслуговувати.  Приклад: Система повинна обслуговувати 5000 користувачів одночасно. |
| використання ресурсів | Використання пам'яті, дискового простору, бази даних і т.д.  Приклад: Система повинна зберігати в базі даних не більше ніж один мільйон транзакцій. Якщо база даних перевищила ліміт, старі транзакції повинні бути відправлені в резервне сховище і потім видалені з операційної бази даних.  Ці вимоги можуть бути також описані в Implementation Requirements (Вимоги Реалізації). |

## SUPPORTABILITY (ЗДАТНІСТЬ ДО СУПРОВОДУ)

|  |  |  |  |
| --- | --- | --- | --- |
| Тестованість | Наскільки легко тестувати систему. Чи потрібна інтеграція з яким-небудь інструментом для тестування?  Приклад: Інтерфейс не повинен містити компонентів, що перешкоджають автоматичному тестуванню з використанням IBM Rational Functional Tester. | | |
| Пристосовність | Наскільки легко система адаптується до нової програмної середовищі.  Приклад: Час розгортання на новій версії WebSphere Application Server має займати не більше ніж один день. | | |
| Здатність до налаштування | Наскільки легко знаходити і виправляти помилки.  Приклад: Лог помилок, що містить інформацію про всі критичні помилки, повинен бути доступний системному адміністраторові через Інтернет, щоб він міг перевірити його віддалено в будь-який час. | | |
| Сумісність | Ступінь сумісності системи з попередніми версіями системи, з замінної системою і з інтерфейсами.  Приклад: Після виходу системи наступні версії системи повинні бути сумісні зверху вниз. Всі введені в попередній версії транзакції повинні бути доступні в новій версії. | | |
| Здатність до оновлення | Наскільки легко розширити систему новими функціональними особливостями.  Приклад: Чи не повинна турбуватися установка на клієнтському робочому місці. Всі системні оновлення і нові релізи повинні здійснюватися на сервері. | | |
| Здатність до установки | Легка установка системи.  Приклад: Установка нової версії системи не повинно вимагати ніякої установки на робочому місці користувача. | | |
| Розширюваність | Наскільки легко система розширює обсяги даних або користувачів.  Яка кількість користувачів система повинна підтримувати з часом  Приклад: Після шести місяців роботи, система повинна мати можливість обслуговувати додатково 5000 користувачів. | | |
| Портативність | Наскільки легко переміщати систему на інше програмне забезпечення або платформу.  Приклад: Зміна бази даних системи в майбутньому не повинно вимагати перезапису логіки додатка. | | |
| Можливість багаторазового застосування | Наскільки легко використовувати частини системи в інших системах.  Приклад: Головна функціональність системи (бронювання рейсу, покупка квитка на літак) повинна бути відображена в компонентах, які можуть бути використані в додатку клієнт / сервер (не через Інтернет). | | |
| Взаємодія | Наскільки легко взаємодіяти з іншими системами.  Взаємодія - це можливість програм, систем або бізнес-процесів працювати разом для виконання загального завдання.  Приклад: Система повинна автоматично бронювати квиток з Airline Reservation System без необхідності втручання людини. | | |
| Відповідність технічним умовам | Наскільки добре система задовольняє всім стандартам і правилам.  Приклад: Процес збору персональної інформації у користувача, який купує квиток на літак, повинен відповідати PatriotAct. | | |
| Здатність до локалізації | Які мови підтримує система. Наскільки легко розширити систему новими мовами.  Приклад: Додаток повинен підтримувати Англійську, французьку та іспанську мови. | | |
| Змінність | Наскільки легко змінювати функціональність системи. | | |
| Здатність до аналізу | Наскільки легко аналізувати систему. | | |
| Здатність до аудиту | Наскільки легко операції системи піддаються аудиту. | | |
| Здатність до налаштування конфігурації | Під здатністю до налаштування конфігурації і оновленню іноді розуміють гнучкість. | | |
| + Contrains | на дизайн  design | Вимоги, що відносяться до системного дизайну та архітектури.  Приклад: Система повинна бути заснована на архітектурі J2EE. |
| вимоги реалізації  implementation | Приклади вимог реалізації:  • Мова програмування, що використовується для розробки системи.  • Управління системами і їх версіями.  • Використання баз даних.  • Сторонні компоненти.  • Обмеження на ресурси: пам'ять, дисковий простір.  • Стандарти кодування. |
| вимоги інтерфейсу  interface | Цей пункт описує різні інтерфейси:  • Інтерфейс користувача.  • Інтерфейс апаратного забезпечення.  • Інтерфейс програмного забезпечення.  • Інтерфейс комунікацій. |
| вимоги апаратного забезпечення  physical | зазвичай ставляться тільки до обладнання, на якому розгортається система. Тут можна вказати, наприклад, форму, розмір або вагу пристрою. Це не відноситься до веб-додатків. |
| вимоги документації | Вимоги до документації можуть містити:  • Роздрукована документація.  • Документація, доступна на СD.  • Документація, доступна он-лайн.  • Он-лайн довідник.  Приклад: Керівництво Адміністратора має бути доступне у форматі PDF.  Вимоги до он-лайн документації іноді описуються в пункті Functionality (Функціональність). |
| вимоги ліцензій та юридичних норм | При розробці системи можуть накладатися й інші обмеження, зокрема юридичні:  • міжнародні угоди: одиниці виміру, мови,  • авторське право,  • угоди про ліцензування,  • законодавство,  • галузеві стандарти.  Приклад: На сторінках, де наводяться дані користувача, має бути посилання на сторінку з політикою конфіденційності. |