ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 004.415.5.

***Національний авіаційний університет**
****Тернопільський національний технічний університет**
***О.Г. Харченко, *І.О. Галай,**
****І.О. Боднарчук, **В.В. Яцишин**

Проектування архітектури WEB–застосування на основі моделі якості

Розглянуті питання забезпечення вимог якості при проектуванні архітектури програмної системи. Для цього використано підхід, базований на концепції моделей якості стандарту ISO 25010. Вимоги користувачів до якості програмних систем, сформульовані в термінах моделі якості у використанні відображені на вимоги зовнішньої якості, з яких виділяються вимоги якості до архітектури програмних систем. Запропонований підхід застосовано до проектування архітектури WEB – застосувань.

Рассмотрены вопросы обеспечения требований по качеству при проектировании архитектуры программной системы. Для этого использовано подход, что базируется на концепции моделей качества стандарта ISO 25010. Требования пользователей к качеству программных систем, сформированы в определениях модели качества в использовании, отображены на требования внешнего качества, с которых выделяются требования качества к архитектуре программных. Предложенный подход использован при проектировании архитектуры WEB – применений.

Were considered the issues of ensuring quality requirements while designing architecture of software systems. For this purpose was used approach based on ISO 25010 models quality concept. Users' PS quality requirements formulated in the definitions of quality models in use communicats on external quality demand from which marked out quality requirements on PS architecture. The proposed approach is used while designing Web-applications architecture.

Ключові слова: архітектура програмного забезпечення, вимоги якості, комунікація вимог, зразки проектування, багатокритерійна оптимізація.

Вступ

Сучасний рівень розвитку технологій проектування програмних систем (ПС), дозволяє за короткий час створювати складні багатофункціональні програмні комплекси.

Головною метою цих технологій є забезпечення функціональних вимог до ПС, а виконання вимог якості контролюється на останній стадії життєвого циклу (ЖЦ), до готового програмного продукту.

При невиконанні вимог якості, вносяться зміни в проміжні продукти на всіх попередніх стадіях ЖЦ, що призводить до перевитрат бюджету проекту, подовження термінів його виконання, а можливо і до відхилення проекту замовником [1].

В стандарті з процесів життєвого циклу ПС ISO 12207 передбачено два процеси, які мають відношення до якості, це процес «забезпечення гарантії якості» та процес «управління якістю».

Цілями першого процесу є забезпечення гарантії якості ПС шляхом впровадження в технології розробки ПС стандартів якості та відповідних процедур.

Процес «управління якістю» призначений для реалізації моніторингу забезпечення вимог замовника ПС до якості.

Процес «забезпечення гарантії якості» інтегрується з процесом «управління якістю», таким чином, що результати моніторингу якості використовуються для планування дій по задоволенню вимог якості. Причому, ці процеси повинні реалізуватися на всіх стадіях ЖЦ на основі користувацьких вимог, тобто виконати процедури комунікації вимог [2]. При цьому виникає ряд проблем, пов’язаних із складом та формою представлення вимог до проміжних продуктів, та визначенням зв’язків між вимогами сусідніх стадій ЖЦ.

Для розробки вимог якості до архітектури можна використовувати моделі вимог якості у використанні, зовнішньої та внутрішньої якості стандарту ISO 25010 [3].

Можна сформулювати вимоги до ПС в цілому, вимоги до архітектури та до модулів в єдиній уніфікованій формі.

Для визначення зв’язків між вимогами можна використати методи статистики, або експертні методи [4].

Дана стаття містить результати досліджень по розробці вимог якості до ПС та процедур комунікацій вимог.

Сучасний стан технології проектування архітектури ПС.
Проектування архітектури на основі патернів.

Наступним етапом після розробки вимог є проектування архітектури ПС. Цей етап є дуже важливим, оскільки на ньому вирішується задача задоволення як функціональних вимог, так і вимог якості.

Найбільш широко, в даний час, при проектуванні архітектури використовується концепція типових рішень, яка базується на виборі з репозиторію архітектур ПС, отриманих з практики, найбільш підходящої для реалізації поставлених функціональних вимог до ПС.

М. Шоу і Д. Гарлан в [5] класифікували архітектури ПС, виділивши п’ять груп по реалізації ними певного набору стандартних загальних функцій. Е. Гамма в [6] розробив архітектурні зразки проектування (патерни), які представляють комбінацію компонентів, зазвичай класів або об’єктів, що вирішують певні проектувальні задачі. Зразки проектування в [5] розділені на структурну, креаційну і поведінкову категорії. Структурні зразки моделюють способи представлення об’єктів, такі як дерева або зв’язні списки. Вони дозволяють користуватися множиною об’єктів як одним цілим, тому надають певні зручності.

Креаційні зв’язки пов’язані із методом створення складних об’єктів, таких як, лабіринти і дерева.

Поведінкові зразки проектування дозволяють відстежувати поведінку об’єктів, наприклад видавати звіт про колекцію об’єктів в певному порядку.

В загальному випадку патерн проектування представляє собою взаємодію об’єктів і класів, пристосованих для розв’язання загальної задачі проектування, яка вирішується в конкретних умовах. Слід зазначити що і в інших відомих технологіях проектування архітектури ПС, таких як, MVC (Model/View/Controller) [7] або модель шарів (layers) [8] теж достатньо плідно використовується концепція патернів.

Вибір архітектури ПС з врахуванням показників якості.

Оскільки при використанні концепції патернів вибирається декілька альтернативних архітектур, які задовольняють функціональним вимогам, то вибір конкретної архітектури проводиться шляхом оптимізації, з врахуванням вимог якості.

Однім з найбільш відомих підходів до вибору архітектурних рішень з врахуванням вимог якості є технологія ADD - ATAM (Attribute - Driven Design, Architecture Trade of Analis Method) [9]. Перша частина технології ADD для вибору архітектурних зразків проектування шляхом розробки тактик, кожна з яких призначена для реалізації певного атрибута якості, причому в кожному зразку реалізується декілька тактик. Тобто, тактики представляють собою вид проектного рішення, яке впливає на керування реакцією за даним атрибутом якості. ADD - є рекурсивним процесом декомпозиції, на кожному з етапів якого відбувається відбір тактик і архітектурних зразків, які задовольняють певним сценаріям якості.

Реалізація відібраних тактик і архітектурних зразків, для вибору найкращої архітектури виконується методом ATAM. Метод ATAM реалізується за чотири етапи, перші два з яких представляють оцінні операції і є аналітичними. На цих етапах виконується виявлення та аналіз архітектурних методик і патернів, а також їх відбір на основі так званого «дерева корисності».

Ця технологія має ряд недоліків методологічного, організаційного та технічного характеру.

По-перше, в ній використовуються нестандартизовані та неуніфіковані показники якості, і метрики, що може породжувати неоднозначність їх трактувань. Також в технології відсутня процедура комунікації вимог якості, сформульованих до ПС, на вимоги до архітектури, що може приводити до суттєвого впливу суб’єктивних факторів.

Являючись досить об’ємною і вартісною технологією ADD - ATAM може використовуватись лише великими колективами розробників для розробки великих програмних проектів.

Використання моделей якості для розробки вимог.

Питання розробки вимог якості до ПС на основі моделей якості, розглядались в ряді публікацій [12, 13]. Тут розглядаються моделі трьох категорій якості, а саме якості у використанні стандарту ISO 25010, зовнішньої та внутрішньої якості. На основі моделі якості у використанні розробляються загальні вимоги якості користувача, або замовника. Модель вимог

якості у використанні ПС можна представити у вигляді:

$V_{use} = \{H^u_i, A^u_{ik}, C^u_{ik}, M^u_{ik}\}$, i e N_u, K =1,S_i (1)

тут H^u_i - характеристика якості у використанні;

A^u_{ik} - атрибут характеристики якості;

C^u_{ik} - обмеження на значення атрибута;

M^u_{ik} - метрика атрибута.

Характеристики і метрики підбираються із стандарту [13], а атрибути і обмеження из вимог користувача та аналізу предметної області.

Вимоги зовнішньої якості представляються у вигляді структури моделі зовнішньої якості і інтерпретуються як вимоги до ПС в цілому, в тому числі і до архітектури. Ці вимоги записуються у вигляді

$V_{ex} = \{H^e_i, П^e_{ik}, A^e_{ik}, C^e_{ik}, M^e_{ik}\}$, i e N_e, K =1,R_i (2)

тут H^e_i - характеристики;

$П^e_{ik}$ - підхарактеристики зовнішньої якості;

A^e_{ik}, C^e_{ik}, M^e_{ik} - атрибути, обмеження та метрики відповідно.

Комунікація (трасування) вимог якості (1) на структуру вимог (2) виконується з використанням методу SQFD [4]. На основі отриманих вимог зовнішньої якості формулюються вимоги якості до архітектури.

Розробка вимог якості до WEB–застосувань. Питання оцінювання якості WEB – застосувань досліджувались в ряді публікацій [10], [14]. Однак в них досліджувались лише певні аспекти якості, а не весь спектр характеристик, причому були використані неуніфіковані і нестандартизовані характеристики якості, що ускладнює порівняння отриманих результатів, та використання їх на практиці. В роботі [10] була проведена систематизація характеристик якості WEB – застосувань шляхом віднесення їх до наступних груп:

- технічні критерії надійності і безпеки;
- технічні критерії швидкодії і оптимізації трафіка;
- системні критерії;
- психологічні критерії;
- естетичні та художні.

Проектування якісних WEB – застосувань на основі технічних критеріїв не викликає труднощів і для кожного класу WEB – систем визначено типові проектні рішення. Але врахування додаткових критеріїв викликає значні ускладнення, повязані з їх конфліктністю, повною схожістю критеріїв з різних груп та інше.

Хоча WEB – системи відрізняються великим різноманіттям, для них можна розробити систему критеріїв якості, загальну для всіх систем. Одним з шляхів вирішення цієї задачі є використання моделі якості стандарту ISO 25010, яка містить повний набір уніфікованих та стандартизованих характеристик і метрик якості. Наявність в стандарті трьох типів моделей якості дає можливість віднести їх до відповідних стадій життєвого циклу WEB і реалізувати процедури комунікації вимог.

Перейдемо до побудови моделі якості, яка відображає вимоги якості у використанні для WEB-застосувань. Виходячи із загальних потреб користувачів і замовників, вимоги якості до web-ресурсів можна сформулювати наступною сукупністю критеріїв:

– Доступність.

– Наявність системи навігації.

– Наявність системи пошуку інформації.

– Зрозумілість структури сайту.

– Швидкість надання інформації.

– Точність надання інформації.

– Естетичне оформлення сайту.

– Скорочення часових ресурсів на виконання поставлених завдань.

– Скорочення фінансових затрат на виконання поставлених задач.

– Наявність системи ведення відвідуваності сайту.

– Наявність системи для ведення статистики отриманих послуг.

– Відповідність сайту галузевим чи міжнародним стандартам.

– Безперебійна робота протягом визначеного періоду часу.

– Безпечність зберігання та контролю над даними.

– Безпека користувачів.

– Надійність сайту (захист від атак ззовні).

– Наявність засобів наповнення інформаційного контенту і т.д.

Визначивши потреби та обмеження відобразимо атрибути на характеристики і метрики моделі якості у використанні

$$\{H^u_i, M^u_{ij}\} i = \overline{1,4},\ j = \overline{1,B_i}.$$

Для відображення атрибутів на характеристики моделі і вибору відповідних метрик проведемо їх класифікацію. У результаті отримаємо стандартизоване представлення вимог користувачів бізнес-системи у вигляді моделі якості

у використанні. Для зручності використання в технології проектування вимоги можна представити у вигляді шаблону. Наведемо приклад опису для атрибута «Наявність системи навігації». Провівши класифікацію цього атрибута, заповнення полів шаблону матиме вигляд як показано у таблиці 1.

Таблиця 1.Опис атрибута

Елемент шаблону	**Екземпляр елементу**	
Характеристика	**Задоволеність**	
Підхарактеристика	-	
Назва атрибуту	**Наявність системи навігації**	
Визначення атрибуту	Даний атрибут визначає необхідність наявності системи навігації для руху по сторінках web-ресурсу	
Мета/Мотивація	Призначення атрибута полягає у реалізації можливості перегляду сторінок сайту користувачем з координацією переміщень по сайту	
Шкала вимірювань	Тип шкали порядковий (присутність, відсутність)	
Процедура визначення, протокол, Х	Процедура визначення атрибута – аналіз предметної області, потреба користувача.	
	Примітки	
Тип збору даних та підрахунку	Тип збору атрибута ручний	
Метрика	Абсолютна продуктивність	
Пріоритетність	0,7	

Нижче наведемо специфікацію вимог якості у використанні до WEB – сайтів, структуровані відповідно моделі якості:

– Ефективність
– Доступність.
– Точність надання інформації.
– Зрозумілість структури сайту.
– Продуктивність
– Швидкість надання інформації
– Економія часових ресурсів.
– Економія фінансових затрат.
– Безпечність
– Безпечність зберігання та контролю над даними.
– Безперебійна робота протягом визначеного періоду часу.
– Безпека користувачів.
– Надійність сайту.
– Задоволеність
– Наявність системи навігації.
– Зрозумілість структури сайту.
– Естетичне оформлення сайту.
– Наявність системи ведення відвідуваності сайту.
– Наявність системи для ведення статистики отриманих послуг.
– Відповідність сайту галузевим чи міжнародним стандартам.
– Наявність засобів наповнення інформаційного контенту

Отже, у наведеній специфікації вимог до web-сайтів, які сформовані на основі моделі якості в уніфікованій формі подано потреби замовника та користувачів ресурсів web-простору. Застосування запропонованої моделі дозволяє систематизувати вимоги до програмного забезпечення, на відміну від технологій, які наведені у [10]. При використанні відсутня чітка структуризація вимог, тобто фактично неможливо встановити чітку приналежність вимоги до потреб замовника і до певного фізичного модуля архітектури, де вона буде реалізована. Крім того, такий підхід до розробки вимог не є уніфікованим і в ньому суттєвий вплив має субєктивний фактор.

Побудова моделі зовнішньої якості та виділення вимог до архітектури.

Для розробки специфікації вимог до архітектури у термінах моделі зовнішньої якості web-сайтів використано каскадну процедуру проектування [12]. Для цього проведемо аналіз характеристик

$$H_i^u, \quad i \in N_u^k, \; K = \overline{1, S_i},$$

атрибутів $A^u_{ik}, \quad i \in N^k_u, \; K = \overline{1, S_i}$
та метрик $M^u_{ik}, \quad i \in N^k_u, \; K = \overline{1, S_i}$
моделі якості у використанні.

Для прикладу наведемо відображення атрибута «Наявність системи навігації» на атрибути моделі зовнішньої якості для web-застосувань. Аналізуючи системи навігації web-сайтів, їх можна розділити на два типи: головна навігація та контекстна навігація. Виходячи з цього, атрибут «Наявність системи навігації» моделі якості у використанні відображається у два атрибути зовнішньої моделі якості: «наявність головного навігаційного меню» і «наявність контекстного навігаційного меню». Опишемо один з них у вигляді шаблону (табл. 2.).

Таблиця 2.Опис атрибуту наявність головного навігаційного меню

Елемент шаблону	**Екземпляр елементу**
Характеристика	**Функціональність**
Підхарактеристика	**Навігація та перегляд інформації**
Назва атрибуту	**Головне навігаційне меню**
Визначення атрибуту	Навігаційне меню – це кнопки, іконки, посилання, що об'єднані в блок меню і служать для переміщення по сайту. Головне навігаційне меню – це меню, яке присутнє і незмінне на всіх сторінках сайту, доступних відвідувачу.
Шкала вимірювань	Порядкова (присутнє, відсутнє)
Процедура визначення, протокол, Х	Значення атрибуту можна визначити лише шляхом перегляду і визначення наявності чи відсутності головного меню. Цей атрибут не вказує на якість структури і оформлення меню, а лише на його наявність чи відсутність.
	Примітки
Тип збору даних та підрахунку	Ручний
Інтерпретація значення оцінки	Чим вище – тим краще
Метрика	Завершеність функціональної реалізації
Пріоритетність	0,8

Враховуючи особливості WEB-застосувань, зокрема за функціональним призначенням та взаємодією з користувачем, підхарактеристику «Функціональна повнота» характеристики «Функціональність» для зручності використання та чіткості формулювання вимог пропонується розділити на декілька підхарактеристик (рис. 1).

Рис. 1 Модифікована модель зовнішньої якості для області WWW

Прикладами атрибутів і метрик для такої характеристики якості як надійність, можуть бути середній час між відмовами, число усунутих дефектів при тестуванні, інтенсивність від-

мов та ін. Однак внутрішні, зовнішні й атрибути якості у використанні взаємозалежні. Досягнення якості у використанні залежить від задоволення атрибутів зовнішньої якості, які, у свою чергу, залежать від задоволення відповідних атрибутів внутрішньої якості.

Виходячи з вищесказаного, для кінцевого використання побудованої моделі якості WEB-застосувань та іншого програмного забезпечення для WEB, модель якості (рис. 1.) довизначається атрибутами та метриками якості.

Слід зауважити, що побудована модель якості без значних модифікацій може бути застосована не лише при розробці вимог до web-застосувань, але й для оцінки більшості сайтів або іншого програмного забезпечення для WWW. Однак, атрибути, що відносяться до під характеристики «Функціональна повнота» характеристики «Функціональність», потрібно обов'язково переглянути відносно конкретної області застосування і вони, як правило, різні для різних web-застосувань. Так, наприклад, атрибути вищезгаданої підхарактеристики будуть абсолютно різні для web-системи дистанційного навчання та будь-якого електронного магазину, що пояснюється зовсім різним призначенням цих систем.

Також для різних систем при оцінюванні їх якості потрібно переглянути пріоритетність визначену для атрибутів якості, оскільки вони вказують на рівень важливості даного атрибуту для конкретної предметної області. Для визначення пріоритету та рівня залежності атрибутів зовнішньої моделі якості використаємо метод SQFD.

Попередньо на основі простого алгоритму вибору визначимо пріоритети атрибутів моделі якості у використанні. Для цього було проведено опитування (анкетування) експертів відносно переваги одного атрибута моделі якості у використанні над іншим по кожній з характеристик H_i^u, $i \in N_u^k$, $K = \overline{1, S_i}$. При цьому вагові множники вибирались за транзитивною шкалою та базою 2.

Наведемо приклад для визначення пріоритету деяких атрибутів характеристики «Задоволеність», моделі якості у використанні, на основі простого алгоритму вибору. Для цього введемо наступні позначення:

H_1^U - характеристика «Задоволеність» моделі якості у використанні;

$A_{ij}^U, i = 1, j = 1, N_i$ – атрибути характеристики «Задоволеність», зокрема

A_{11}^U – «Наявність системи навігації»;

A_{12}^U – «Зрозумілість загальної структури сайту»;

A_{13}^U – «Естетичне оформлення сайту»;

A_{14}^U – «Наявність системи ведення відвідуваності сайту»;

A_{15}^U – «Наявність системи для ведення статистики отриманих послуг»;

A_{16}^U – «Наявність засобів наповнення інформаційного контенту».

Провівши статистичну обробку результатів анкетування, отримано наступні значення переваг одного атрибута над іншим: $\alpha_{12} = 1, \alpha_{23} = 1/2, \alpha_{34} = 8, \alpha_{45} = 1, \alpha_{56} = 1/4$. Виходячи з рівності $\alpha_{ij} = \alpha_i / \alpha_j$, отримаємо значення вектора пріоритетів

$\alpha = (0{,}18;\ 0{,}18;\ 0{,}36;\ 0{,}045;\ 0{,}045;\ 0{,}18)$

Отримані значення пріоритету вимог якості у використанні за характеристикою «Задоволеність» записують справа у «будинку якості». Виходячи із представлення вектора $\alpha = (0{,}18;\ 0{,}18;\ 0{,}36;\ 0{,}045;\ 0{,}045;\ 0{,}18)$, можна зробити висновок про те, що найбільш важливими для користувачів є атрибути, які відображають естетичні аспекти оформлення сайту, менш важливими – наявність системи навігації та інформаційного наповнення сайту, а також зрозумілість структури сайту. Практично не цікавлять користувачів властивості сайту, які відображають загальну його відвідуваність та ведення статистики отриманих послуг.

Частковий випадок представлення «будинку якості», який показує кореляцію та пріоритети між вимогами якості у використанні та вимогами зовнішньої якості для області WWW, наведено у табл. 3. При цьому компоненти моделі зовнішньої якості позначено наступним чином:

H_1^x – характеристика «Функціональність» моделі зовнішньої якості.

$\ddot{I}_{11}^x$ – підхарактеристика «Навігація і перегляд інформації» характеристики «Функціональність», до якої входять наступні атрибути:

A_{111}^x – «Наявність контекстного меню»;

A_{112}^x – «Наявність головного навігаційного меню»;

$\ddot{I}^{x}_{12}$ – підхарактеристика «Керування інформаційним наповненням сайту» характеристики «Функціональність» до якої входять атрибути:

A^{x}_{121} – «Редагування стилів»

A^{x}_{122} – «Редагування сторінок та розділів сайту»

H^{x}_{2} – характеристика «Зручність використання» моделі зовнішньої якості;

$\ddot{I}^{x}_{21}$ – підхарактеристика «Зрозумілість» характеристики «Зручність використання», до складу якої входять наступні атрибути:

A^{x}_{211} – «Зрозумілість загальної структури сайту»;

A^{x}_{212} – «Карта сайту»;

$\ddot{I}^{x}_{22}$ – підхарактеристика «Зручність навчання» характеристики «Зручність використання», до складу якої входить атрибут A^{x}_{221} – «Питання, що часто задають».

$\ddot{I}^{x}_{23}$ – підхарактеристика «Зручність роботи» характеристики «Зручність використання», до складу якої входить атрибут A^{x}_{231} – «Зручність і наочність навігації»;

$\ddot{I}^{x}_{24}$ – підхарактеристика «Привабливість» характеристики «Зручність використання», до складу якої входять наступні атрибути:

A^{x}_{241} – «Однорідність стильового оформлення»;

A^{x}_{242} – «Групування головних елементів управління»;

A^{x}_{243} – «Однорідність кольорового оформлення посилань»;

A^{x}_{244} – «Сталість елементів керування»;

A^{x}_{245} – «Підтримка різних мов»;

H^{x}_{3} – характеристика «Зручність супроводу» моделі зовнішньої якості;

$\ddot{I}^{x}_{31}$ – підхарактеристика «Аналізованість» характеристики «Зручність супроводу»;

A^{x}_{311} – атрибут «Ведення лог-файлів» підхарактеристики «Аналізованість».

Таблиця 3. Матриця залежностей та пріоритетів атрибутів якості у використанні та зовнішньої якості

		H^{x}_{1}				H^{x}_{2}									H^{x}_{3}	Пріоритет вимог	Пріоритет атрибутів моделі якості у використанні	
		$\ddot{I}^{x}_{11}$		$\ddot{I}^{x}_{12}$		$\ddot{I}^{x}_{21}$		$\ddot{I}^{x}_{22}$	$\ddot{I}^{x}_{23}$	$\ddot{I}^{x}_{24}$					$\ddot{I}^{x}_{31}$			
		A^{x}_{111}	A^{x}_{112}	A^{x}_{121}	A^{x}_{122}	A^{x}_{211}	A^{x}_{212}	A^{x}_{221}	A^{x}_{231}	A^{x}_{241}	A^{x}_{242}	A^{x}_{243}	A^{x}_{244}	A^{x}_{245}	A^{x}_{311}			
H^{U}_{1}	A^{U}_{11}	6	9	0	0	3	3	0	9	0	6	0	0	0	0	0,18	45	16%
	A^{U}_{12}	0	6	0	0	9	9	6	9	3	3	0	6	6	0	0,18	72	23%
	A^{U}_{13}	0	0	0	0	0	0	0	0	9	6	9	9		0	0,45	39	4%
	A^{U}_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0,45	9	1%
	A^{U}_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0,37	69	44%
	A^{U}_{16}	0	0	9	9	0	0	0	0	0	0	0	0	0		0,18	33	12%
		Пріоритет атрибутів моделі зовнішньої якості																
		6	21	9	15	36	21	15	33	12	15	9	15	15	18			
		2%	9%	3%	5%	13%	10%	8%	14%	7%	6%	1%	8%	7%	7%			

Задавши граничне значення пріоритету для атрибутів зовнішньої якості Pg = 15, отримаємо набір атрибутів, які включаються у вимоги зовнішньої якості.

$\{A^{x}_{122}, A^{x}_{221}, A^{x}_{242}, A^{x}_{245}, A^{x}_{244}, A^{x}_{311}\}$ (3)

Вибір архітектури WEB - застосування

На основі аналізу функціональності WEB - застосування та вимог користувача та зовнішньої

якості виберемо альтернативні архітектури із каталогу зразків проектування [6].

Архітектура незалежних елементів складається з джерела даних і деякої кількості клієнтів, дані клієнтів обновляються, коли відбувається зміна в джерелі даних. Для цього можна використати зразок проектування Observer. В якості альтернативи розглянемо також систему яка керується подіями, відповідну архітектуру незалежних компонентів і зразок проектування State. В даній архітектурі застосування представляють такими, що складаються з компонентів, кожний з яких знаходиться в стані чекання, поки не відбудеться певна подія.

Третій зразок проектування візьмемо з архітектури структурних патернів, а саме зразок Decorator.

Це задача багатокритерійної оптимізації.

Для її рішення можна використати метод аналізу ієрархій Сааті. Але його застосування повязане з обчисленням власних чисел матриці попарних порівнянь а також може привести до некоректних результатів. Тому було використано простий метод розв'язання задачі [16].

Представимо схему методу на конкретному прикладі.

Розглянемо проблему визначення оптимальної архітектури з трьох альтернативних за сукупністю критеріїв якості (3).

Будемо розглядати наступні частинні критерії оптимальності, що характеризують в сукупності вихідний показник:

f1 – A^x_{122}
f2 – A^x_{221}
f3 – A^x_{242}
f4 – A^x_{245}
f5 – A^x_{244}
f6 – A^x_{311}

Передбачається, що всі введені частинні показники необхідно максимізувати, тобто більшому значенню кожного показника буде відповідати більш бажаний стан для замовника. Розглянемо ситуацію вибору, коли маємо три альтирнативні архітектури, позначені буквами State - A, Observer - B, Decorator - C.

Будемо слідувати запропонованому простому алгоритму вибору з транзитивною шкалою і базою **a** = 2. Побудуємо вектор вагових множників для сформульованих частинних критеріїв. Для цього експертами визначається значення коефіцієнтів переваги одного показника над іншим. Будемо вважати, що по результатам експертної оцінки були отримані наступні дані:

$\alpha_{12} = 2; \alpha_{23} = 4; \alpha_{34} = 1/4; \alpha_{45} = 1; \alpha_{56} = 4.$

Тут рівність $\alpha_{12} = 2$, наприклад, значить, що частинний критерій f1 в два рази більш «важливий» ніж критерій f2 і т. д.

Скориставшись відношенням $\alpha_{ij} = \alpha_i / \alpha_j$ і за умови нормованості вектора α^1, безпосередньо отримуємо:

$\alpha_1 = 0{,}364; \alpha_2 = 0{,}182; \alpha_3 = 0{,}045; \alpha_4 = 0{,}182; \alpha_5 = 0{,}182; \alpha_6 = 0{,}045.$

Далі переходимо до процедури обчислювання значень частинних критеріїв оптимальності, що відповідають трьом варіантам A, B, C. Спочатку, з допомогою того ж підходу, ранжируємо варіанти A, B, C по критерію f1. Нехай експерт вказав наступні значення коефіцієнтів досконалості: $\alpha^1_{12} = 1; \quad \alpha^1_{23} = 1/2.$

Відповідний вектор вагових множників α^1 має компоненти 0,250; 0,250; 0,500. які інтерпретуються як значення функції f1 для трьох варіантів A, B, C:

$$f_1(A) = 0{,}250; \; f_1(B) = 0{,}250; \; f_1(C) = 0{,}500.$$

Аналогічно визначаємо значення наступних частинних критеріїв для варіантів A, B, C:

$$\alpha^2_{12} = 2; \quad \alpha^2_{23} = 2;$$
$$f_2(A) = 0{,}571; \; f_2(B) = 0{,}286; \; f_2(C) = 0{,}143$$
$$\alpha^3_{12} = 1; \; \alpha^3_{23} = 1;$$
$$f_3(A) = 0{,}333; \; f_3(B) = 0{,}333; \; f_3(C) = 0{,}333$$
$$\alpha^4_{12} = 1/2; \; \alpha^4_{23} = 1/4;$$
$$f_4(A) = 0{,}091; \; f_4(B) = 0{,}182; \; f_4(C) = 0{,}727$$
$$\alpha^5_{12} = 4; \; \alpha^5_{23} = 2;$$
$$f_5(A) = 0{,}727; \; f_5(B) = 0{,}182; \; f_5(C) = 0{,}091.$$
$$\alpha^6_{12} = 1/2; \; \alpha^6_{23} = 2;$$
$$f_6(A) = 0{,}250; \; f_6(B) = 0{,}500; \; f_6(C) = 0{,}250.$$

Скориставшись методом лінійної згортки,

$$J(x_i) = \sum_{k=1}^{6} \alpha_k f_k(x_i),$$

Отримаємо значення узагальненого критерія оптимальності для трьох варіантів $x_1 = A, x_2 = B, x_3 = C$:

Відповідно, до цього методу, найбільш оптимальною визначилася архітектура **C.** Однак бачимо, що вибір **A** теж є прийнятним.

Можна також замість лінійної згортки в якості узагальненого критерія використати згортку Джоффриона, засновану на комбінації лінійної і максимінної згортки.

Легко підрахувати, що в запропонованому методі загальне число порівнянь обєктів по важливості, що виконується користувачем, дорівнює: $N_1 = m - 1 + m(n-1) = mn - 1$, де m – число частинних критеріїв, n – кількість альтернатив.

В стандартних процедурах Сааті і Коггера и Ю число порівнянь дорівнює

$$N_2 = \frac{m^2 - m}{2} + m\frac{n^2 - n}{2},$$ що може бути суттєво більше N1. Так для m=6, n=10 маємо:

$N_1 = 59, N_2 = 285$

Крім того, як уже зазначалося, при використанні запропонованого підходу нема необхідності вирішувати лінійні системи і шукати власні числа матриць.

Висновок

В даній роботі для забезпечення якості WEB- застосування пропонується застосовувати моделі якості ISO 25010.

Використання комплексу моделей якості, пов'язаних з відповідними етапами проектування ПС, дозволяє проводити комунікацію вимог якості до готового програмного продукту, сформульованих замовником, на вимоги до проміжних продуктів, отриманих на відповідних етапах проектування. А це дозволяє реалізувати процеси управління та забезпечення якості ПС, визначені стандартом ISO 12297 при проектуванні архітектури та програмних модулів.

Запропоновані процедури були використані для проектування архітектури WEB-сайту з врахуванням критеріїв якості.

Використана література:

1. *Андон Ф. И.* Основы инженерии качества программных систем / Ф.И.Андон, Г.И.Коваль, Т.М.Коротун, Е.М.Лаврищева, В.Ю.Суслов. - К. Академпериодика, 2007.-672с.
2. ISO/IEC 12207. Information technology - Software life cycle processes. 2002.
3. ISO/IEC 25010 Software engineering. Software product Quality, Reguirement and Evaluation (SquaRe), Quality Model, 2008.
4. *Харченко О.Г.* Інструментальний засіб розробки та комунікації вимог якості до програмних систем / О.Г. Харченко, В.В. Яцишин, І.Е.Райчев. - Інженерія програмного забезпечення № 2, 2010. - 29-34 с.
5. *Garlan D., Show M.* Software Architecture. Perspectives on an Emerging Discipline, Englewood Clitts, NJ: Prentice Hall, 1996.
6. *Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д.* Приёмы обьектно-ориентированного проектирования.Патерны проектирования. СПб: Питер, 2010.- 366 с.
7. *Glenn E.,* Krasner and Stephen T.Pope. F cookbook for using the model - view controller user interface paradigm in Smalltalk - 80. Jornal of Object - Orient Programming 1 (3):26-49, August 1998.
8. *Фаулер М.* Архитектура корпоративных программных приложений: Пер. с англ.-М.: Издательский дом «Вильямс», 2004 – 544 с.
9. *Басс Л., Клементс П., Кауман Р.* Архитектура программного обеспечения на практике. 2-е издание. - СПб.:Питер, 2006. – 575 с.
10. *Пелещишин А.М.* Методи побудови ефективних WWW – систем /А.М. Пелещишин // Вісник національного університету «Львівська політехніка»: Інформаційні системи та мережі. -№464.-2002.- С.240-255.
11. *Коваль Г.І.* Моделювання вимог до якості програмних систем оброблення даних / Г.І. Коваль, Г.Б. Мороз // Проблеми програмування, 2006.-№2, №3-С.273-244.
12. *Харченко О.Г.* Розроблення та керування вимогами до програмного забезпечення на основі моделі якості / О.Г.Харченко, В.В.Яцишин // Вісник ТДТУ.-Том 14.№1.-2009.- С.201-207.
13. ISO/IEC 9126. (1-4) Software Engineering-Product Quality/2001-2004.
14. *Яцишин В.В.* Технологія оцінювання якості WEB-застосувань / В.В. Яцишин // Вісник ТДТУ.-Том 14.-№4.-2009.-С.132-140.
15. *Черноруцкий И.Г.* Методы принятия решений.-СПб.БХВ-Петербург.-2005.-416с..

Відомості про авторів:

Харченко Олександр Григорович – кандидат технічних наук, професор кафедри комп'ютерних інформаційних технологій факультету комп'ютерних наук Національного авіаційного університету. Наукові інтереси – технології розробки програмного забезпечення, технології проектування інформаційних систем.

Галай Ірина Олександрівна – аспірант кафедри інформаційних управляючих систем і технологій факультету комп'ютерних наук Національного авіаційного університету. Наукові інтереси – технології оцінювання якості програмних систем, проектування програмного забезпечення.
E-mail: irinagalay@ukr.net

Боднарчук Ігор Орестович – асистент кафедри комп'ютерних наук Тернопільського національного технічногно університету ім. Івана Пулюя. Наукові інтереси – оцінка якості програмного забезпечення, якість архітектури програмних систем.
E-mail: bodnarchuk.io@gmail.com

Яцишин Василь Володимирович – асистент кафедри комп'ютерної інженерії Тернопільського національного технічного університету ім. Івана Пулюя. Напрямки наукових досліджень – оцінювання якості програмних систем, технології проектування web-застосувань.

Стаття надійшла до редакції 26.11.2010 р.

ТЕХНОЛОГІЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 681.3

Институт программных систем НАН Украины
А.Е. Дорошенко, К.А. Жереб

Техника и инструментарий переписывающих правил для инженерии программного обеспечения графических ускорителей

Графические ускорители позволяют достичь высокой производительности за счет большого количества вычислительных ядер, однако их программирование является сложным и трудоемким процессом. В работе предложен подход к созданию эффективных программ для графических ускорителей с использованием техники переписывающих правил. Предложенный подход и разработанный инструментарий позволяют достичь высокой производительности параллельных программ, а также повысить производительность разработчиков.

Графічні прискорювачі дозволяють досягти високої продуктивності за рахунок великої кількості обчислювальних ядер, проте їх програмування є складним і трудомістким процесом. У роботі запропонований підхід до створення ефективних програм для графічних прискорювачів з використанням техніки переписуючих правил. Запропонований підхід і розроблений інструментарій дозволяють досягти високої продуктивності паралельних програм, а також підвищити продуктивність розробників.

The graphic accelerating allow to attain a high yield due to plenty of calculable kernels, however their programming is a difficult and labour intensive process. In-process offered approach near creation of the effective programs for the graphic accelerating with the use of technique of rewritting rules. Offered approach and the worked out tool allow to attain the high yield of the parallel programs, and also promote the productivity of developers.

Ключевые слова: graphical processing units, rewriting rules technique, automated program transformations, formal design methods, development tools, Microsoft .NET framework.

Введение

В последнее время активно развивается новая аппаратная и программная платформа для высокоэффективных параллельных вычислений – графические ускорители (graphical processing units, GPUs [1]). Эти специализированные устройства, изначально предназначенные только для обработки графики, оказались весьма эффективными для решения вычислительных задач, не связанных с графикой (general purpose computations on GPUs, GPGPU [2]). Их преимуществом является очень высокая степень возможного параллелизма, за счет большого количества вычислительных ядер и специального планировщика, позволяющего эффективно управлять исполнением миллионов потоков. Кроме того, графические ускорители поддерживают иерархию памяти, от быстрой, но ограниченной по объему, до большой по объему и медленной. Это позволяет использовать разные виды памяти для разных задач, что также повышает эффективность параллельных программ.

Однако для графических ускорителей характерен и весьма существенный недостаток, а именно сложность разработки приложений для этих платформ. Первоначально для программирования вычислений на графических ускорителях использовались средства разработки графических приложений – шейдеры (shaders). Вычисления представлялись в виде действий с графическими объектами, такими как текстуры [3]. В последнее время интерес к использованию графических ускорителей в качестве средств высокопроизводительных вычислений поддерживается усилиями ведущих разработчиков аппаратуры. Так, компания NVidia представляет платформу для вычислений на графическом ускорителе CUDA [4]. Аналогично, компания AMD выступила с инициативой