**УДК 004.415.5**

Харченко О.Г.1, Боднарчук І.О.2, Галай І.О.,Райчев І.Е.

1 Національний авіаційний університет

Кафедра комп’ютерних інформаційних технологій

2 Тернопільський національний технічний університет

Кафедра комп’ютерних наук

**ІНСТРУМЕНТАЛЬНИЙ ЗАСІБ ПОРІВНЯЛЬНОГО ОЦІНЮВАННЯ**

**І БАГАТОКРИТЕРІАЛЬНОГО ВИБОРУ АРХІТЕКТУРИ**

**ПРОГРАМНИХ СИСТЕМ**

*В представленому інформаційно-програмному комплексі реалізуються функції класифікації та комунікації вимог якості до програмної архітектури, а також порівняльне оцінювання альтернативних архітектур і вибір кращої з них по множині критеріїв якості. Оцінювання якості архітектури виконується методом аналізу ієрархій (МАІ) з використанням оптимізаційного алгоритму визначення ваг альтернатив. Вибір архітектури по множині критеріїв проводиться з врахуванням компромісів між критеріями якості та чутливості рішення до зміни вимог якості.Комплекс функціонально є системою підтримки прийняття рішень (СППР) архітектора програмних систем.Приведено опис програмного комплексу,який реалізує дану систему.*

*В представленном информационно-программном комплексе реализуются функции классификации и коммуникации требований к качеству программной архитектуры, а также сравнительное оценивание альтернативных архитектур и выбор наилучшей из них по множеству критериев качества. Оценивание качества архитектуры выполняется методом анализа иерархий (МАИ) с использованием оптимизационного алгоритма определения весов альтернатив. Выбор архитектуры по множеству критериев проводится с учетом компромиссов между критериями качества и чувствительности решения к изменению требований к качеству. Функционально, комплекс является системой поддержки принятия решений (СППР) архитектора программных систем. Приведеноописаниепрограммногокомплекса, которыйреализуетданнуюсистему.*

*In the submitted information program complex functions of classification and the communications of requirements to quality of program architecture, and also comparative estimationalternative architecture and a choice of best of them on set of criteria of quality are realized. Estimation ofqualities of architecture it is carried out by a method of the analysis of hierarchies (MAH) with use optimizationalgorithm of definition of weights of alternatives. The choice of architecture on set of criteria is carried out in view of compromises between criteria of quality and sensitivity of the decision change of requirements quality. Functionally, the complex is a system of support of decision-making of the architect of program systems. The description of a program complex which realizes this system is resulted.*

# Вступ

Створення сучасних програмних продуктів вимагає від проектувальників враховувати великі об’єми даних, знань, факторів для прийняття ними ефективних рішень. Програмні системи (ПС) є високоінтелектуальним продуктом, що ускладнює формалізацію процесів його проектування, а це, в свою чергу, затрудняє розробку та використання засобів автоматизації цих процесів. Тому актуальним є застосування формальних методів, таких як математичне моделювання, оптимізація, теорія прийняття рішень для розробки моделей процесів проектування та побудови на їх основі засобів автоматизації підтримки прийняття рішень проектувальниками.

Особливо важливо це на етапах специфікації вимог і проектування архітектури, оскільки ці етапи є першими і виправлення результатів неефективних рішень, прийнятих на них, приводить до великих втрат.

При проектуванні програмних систем широко застосовується компонентна технологія яка базується на вживанні компонентів повторного використання (КПВ) ,які взяті з раніше виконуваних проектів. Архітектура в цій технології проектується вибором ,на основі вимог до ПС ,каркасу і заповненням його необхідними компонентами ,взятими з репозиторію ,або інтернету. *Каркас* являє собою високорівневу абстракцію проекту ПС , і поєднує множину взаємодіючих між собою об'єктів у деяке інтегроване середовище [1].Розширенням поняття компонента є *шаблон* (паттерн) – абстракція, що містить у собі опис взаємодії сукупності об'єктів узагальній кооперативній діяльності, для якої визначені ролі учасників і їхнявідповідальність [2]. Розроблено велику кількість компонентів, які класифіковані по типам задач ,і видам застосувань,а також технологія їх використання [3].

Оскільки в репозиторії патернів,як правило ,є декілька компонентів,які реалізують одну і ту ж функцію ,то отримаємо певну множину альтернативних архітектур ПС. Для вибору найбільш прийнятного варіанта архітектури необхідно знайти оцінки альтернатив відносно критеріїв якості ,при заданих обмеженнях.

На практиці використовується декілька методів оцінювання програмної архітектури. Найбільш відомими з них є методи ,які базуються на розробці сценаріїв використання та перевірки ,чи задовольняє даний варіант архітектури вимозі по певному критерію якості. Наприклад, в методі ATAM (ArchitectureTrade-OffAnalysisMethod) оцінюються ризики того, що архітектура не задовольняє концептуальним вимогам, які описуються сценарієм [4]. Недоліками цих методів є те що в сценарії перевіряється лише один атрибут, і одна альтернатива, неможливість отримати порівняльні оцінки альтернатив, а також недостатня формалізація і висока трудомісткість.

Більш ефективним в даній задачі є метод аналізу ієрархій Сааті , який дозволяє отримати порівняльні оцінки множини альтернатив по задоволенню критеріїв якості [5]. Cуттєвим недоліком застосування МАІ є обмежена кількість альтернатив ,які можна оцінювати одночасно (n<=7±2), що викликано неузгодженістю елементів матриць парних порівнянь Для вирішення цієї проблеми, в [6] запропонована модифікація МАІ, в якій вагові множники альтернатив визначаються з умови мінімізації неузгодженості матриці парних порівнянь, що приводить вихідну задачу до задачі математичного програмування. В [7] розглянуті питання застосування модифікованого МАІ (ММАІ) до задачі вибору оптимальної архітектури програмних систем. Отримані відносні оцінки альтернатив в ММАІ можуть використовуватись для аналізу конфліктів між атрибутами якості і пошуку компромісного рішення при виборі архітектури по множині критеріїв якості. Для остаточного вибору варіанта архітектури з врахуванням сукупності критеріїв обчислюють значення інтегрального критерію у вигляді скалярної згортки, для чого необхідно задати коефіцієнти пріоритетів критеріїв якості. Але пріоритети критеріїв різних груп фахівців суттєво різняться,тому для отримання компромісного результату необхідно проводити додаткові дослідження [5],[8]. Також важливо при виборі архітектури враховувати чутливість отриманого ранжування альтернатив до зміни пріоритетів критеріїв якості ,викликаною зміною вимог предметної області [8].

Для ефективного вирішення архітектором перерахованих вище задач необхідно розробити інформаційно-програмний комплекс який би автоматизував рішення всього комплексу задач.

**1**. **ФУНКЦІОНАЛЬНІСТЬІНСТРУМЕНТАЛЬНОГО ЗАСОБУ.**

Загальний вигляд системи зображено на рисунку 1.

Інтерфейсна підсистема забезпечує взаємодію користувачів з системою. База знань використовується для формування альтернативних архітектур зі стандартних патернів. Тут зберігаються правила побудови архітектури у відповідності з вимогами.

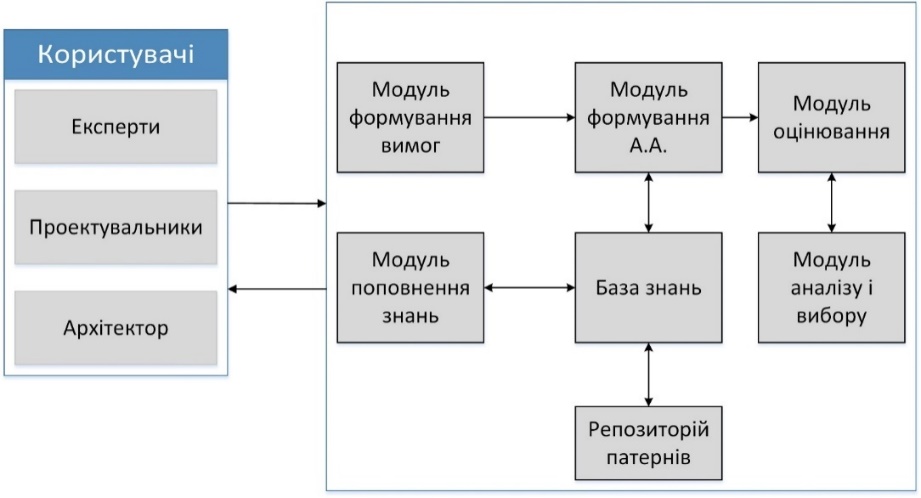


Рис. 1. Структура комплексу (А.А. – альтернативні архітектури).

* 1. **Модуль визначення вимог якості архітектури.**

Множина вимог якості архітектури  повинна визначатися таким чином, щоб побудована на основі даної архітектури ПС задовольняла вимогам якості до ПС . Для визначення  по заданих  в [9] розроблена процедура і програмний засіб, базовані на технології QFD, яка полягає в прямому визначенні експертами кореляцій між вказаними множинами. Для зменшення впливу суб’єктивних факторів на результат в даному модулі використано алгоритм методу базових протоколів і експертний метод парних порівнянь

Специфікації вимог – це сукупність властивостей, яким повинна задовольняти поведінка ПС. Ці властивості можуть бути представлені за допомогою формалізму базових протоколів.

Базовий протокол представляє собою вираз виду , де *x* – список типізованих параметрів, α і β – формули базової мови,*u* – процес протоколу. Формула α називається передумовою, а формула β – пост умовою базового протоколу. Після умова виражає стан системи після виконання базового протоколу. Базовий протокол розглядається, як формула темпоральної логіки, яка виражає той факт, що коли (для певних значень параметрів) стан системи має розмітку, яка задовольняє умові α, то процес *u*переводить систему в стан, в котрому розмітка задовольняє вимоги умови β. Базові протоколи формалізують вимоги до ПС. Для запису α і β використовується мова, близька до мови специфікації. Для означення процесів використовується мова алгебри процесів, така як CCS, CSP або такі мови, як UML, MCS чи SQL. В роботі [10] цей формалізм був успішно застосований для верифікації функціональних вимог до телекомунікаційних систем.

В нашому випадку передумовою α є вимоги якості до ПС, специфіковані в термінах стандарту ISO/IEC 25010, а пост умовою β – вимоги якості до архітектури ПС. Процес *u* є процесом комунікації (перетворення) α в β, який реалізується експертною технологією парних порівнянь. Суть полягає в побудові матриці , де  виражає, на скільки вплив *i-*го критерію якості архітектури переважає вплив *j*-го критерію на реалізацію *s*-го критерію якості ПС. З побудованої матриці обчислюються вагові множники  критеріїв якості архітектури. Призначивши пріоритети критеріям якості ПС , обчислюємо інтегральні вагові множники

 (1)

і, ввівши обмеження , визначимо критерії якості архітектури, які включаються в β.

Використання формалізму базових протоколів дозволив уніфікувати представлення α і β, що дало змогу зменшити кількість помилок в специфікаціях, а також формалізувати процедуру комунікації вимог.

* 1. **Модуль формування альтернативних архітектур.**

При створенні даного модуля для подання архітектури була прийнята концепція шарів, розвинута М.Фаулером[11], в якій функціональність розділена на шари. Корпорація Microsoft розробила технологію, яка базується на даній концепції [3]. В цій технології для кожного шару розроблено набори компонентів (патернів), які реалізують функціональність даного шару. Компоненти згруповані у категорії і призначені для вирішення певних стандартних задач.

Таким чином, кожне програмне застосування, яке буде проектуватись, можна розділити на логічні частини, які відповідають шарам. Визначивши категорії задач, які будуть розв’язуватись певним шаром, можна вибрати деякий компонент з існуючого набору і, таким чином, створити каркас архітектури. А оскільки для кожної категорії розроблено, як правило, декілька компонентів, то можна створити множину альтернативних архітектур.

Для автоматизації цього процесу пропонується використовувати експертну технологію, засновану на базі знань. Знання у системі організовані у вигляді фрейму, зображеного на рисунку 2.

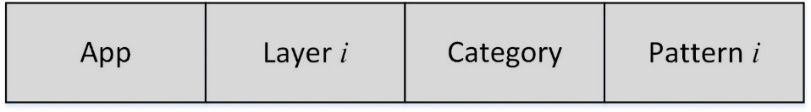


Рис. 2. Структура фрейму бази знань експертної системи.  
App – ім’я програми; Layer – ім’я шару; Category – ім’я категорії задач; Pattern – ім’я шаблону.

Експерти (архітектор) ділять застосування на шари і визначають задачі, які розв’язуються на певних шарах. Потім заповнюється фрейм-шаблон, в якому незаповненим є останній слот. На основі пошуку у репозиторії шаблонів знаходиться відповідний компонент і поміщується в каркас, створюючи таким чином архітектуру.

* 1. **Модуль порівняльного оцінювання архітектур.**

Із скомпонованих альтернативних архітектур необхідно визначити найкращу. Для цього проводиться порівняльне оцінювання альтернатив, як по кожному показнику якості, так і по їх сукупності. Структурна схема процесу оцінювання зображена на рисунку 3.

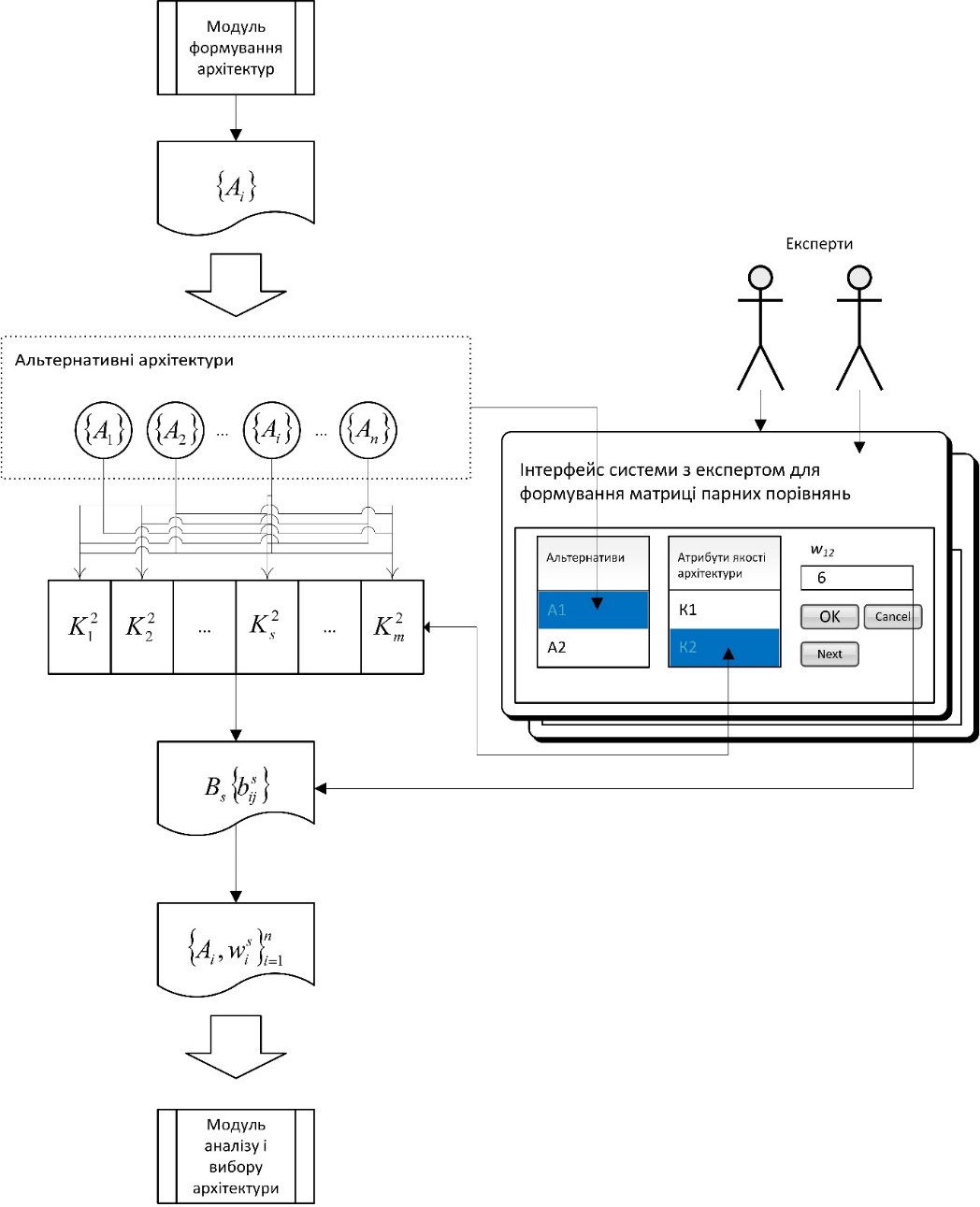
 – критерії якості ПС у відповідності зі стандартом ISO/IEC 25010; 

Рис. 3. Ієрархічне представлення задачі оцінювання архітектури.

– критерії якості архітектури;

–альтернативні архітектурні рішення.

Архітектурне рішення вибирається з умови оптимізації сукупності критеріїв ,. Це задача багатокритеріальної ієрархічної оптимізації і для її розв’язання використовується модифікований метод аналізу ієрархій Сааті.

При використанні МАІ ваги альтернатив (критеріїв)на кожному рівні знаходяться з використанням матриць парних порівнянь, які заповнюють експерти (тут визначає перевагу *i*-тої альтернативи над *j*-ю). В роботі [6] для рішення задачі оцінювання при великій кількості альтернатив (*n*>9) використано алгоритм обчислення ваг  з умови мінімізації неузгодженості матриці. Ця задача зводиться до задачі лінійного програмування, математична модель якої визначається вибраною формою неузгодженості матриці.Тут приведено рішення цієї задачі для наступної форми міри неузгодженості:

, (2)

де – вибране порогове значення.

Задача мінімізації (2) зводиться до задачі лінійного програмування:

 (3)

 (4)

Для отримання порівняльних оцінок якості архітектур по кожному з критеріїв експерти формують матрицю , де  показує, наскільки *i*-та альтернатива переважає *j*-ту по реалізації *s*-го критерію. Для отриманих матриць визначаються набори вагових множниківяк рішення задачі (4),(5).Для ранжування альтернатив по множині критеріїв необхідно знайти їх пріоритети. Для визначення пріоритетів критеріїв якості архітектури  по їх впливу на реалізацію критеріїв якості ПС  експертами заповнюється матриця парних порівнянь , де величинавизначає, на скільки вплив критерію  переважає вплив критерію  на реалізацію критерію якості ПС.Розв’язавши задачу (3), (4) для матриць , отримаємо набори пріоритетів критеріїв якості архітектури , ,.Тоді вага альтернативної архітектуривідносно реалізації критерію якості ПСвизначатиметься за формулою:

, (5)

де –вагові множники, визначені на попередньому етапі.

Тепер ранжуються альтернативиза величиноюдля кожного .Тобто, отримаємо множини альтернатив і відповідні їм значення ваг.

Застосування оптимізаційного алгоритму до задачі оцінювання архітектур достатньо повно розкрито у роботах [7],[8].В даній системі, після оцінювання альтернатив, може рішатись задача їх оперативного корегування. Така задача виникає тоді, коли експерти і архітектор при виборі надають перевагу певній альтернативі , хоча вона за деякими критеріями має не найкращі оцінки. Ставиться задача збільшити оцінки за цими критеріями за рахунок зменшення за іншими, але так, щоби оцінки альтернативи  за всіма критеріями були не гірші за інші.

Така ситуація є типовою при проектуванні розподілених програмних застосувань, коли бажана архітектура відома, а корегування критеріїв може здійснюватись шляхом підбору стандартних функціональних компонентів

До розв’язування цієї задачі застосовується аксіоматичний підхід В. Подіновського [11], який полягає в попарному заміщенні критеріїв. Критерії  і  є порівняними за заміщенням, якщо для деякої альтернативи  можлива компенсація за перевагою будь-якої зміни критерію  зміною критерію .

Тобто, якщо  – це альтернатива, яка заміщує  шляхом корекції  і компенсації , то їх скореговані значення будуть

, , , (6)

тут  – вектор значень критеріїв.

Запишемо співвідношення для компенсації при заміщенні для множини компонент вектору альтернативи , яку ми хочемо зробити кращою за :

, (7)

де  – можливе зменшення компоненти  з метою збільшення ;

 – множина індексів *r*, для яких , ;

 – задана для  множина індексів, така, що компоненти  можуть брати участь у заміщенні компонентів ;

 – задані коефіцієнти пропорційності .

Компоненти вектору  після заміщення визначаються наступними співвідношеннями:

 (8)

Оптимізація заміщення (8), приводить вихідну задачу до задач лінійного програмування, математичні моделі яких залежать від стратегії прийняття рішень [12].

Для Парето-оптимальної стратегії модель оптимізації буде такою:

 (9)

при обмеженнях:

;

;

 (10)

.

змінними тут є .

Після деяких перетворень для розв’язування задачі застосовується стандартний симплекс-метод.

Модуль аналізу і прийняття рішень

Для прийняття остаточного рішення по вибору архітектури проводиться аналіз результатів, отриманих в модулі оцінювання.

Оскільки в розробці ПС беруть участь декілька груп фахівців,які мають різні пріоритети для кожного з атрибутів якості, то визначаються пріоритети кожної з груп шляхом формування ними матриць парних порівнянь, до яких застосовується експертний метод парних порівнянь і знаходяться пріоритети критеріїв – номер групи експертів. Компромісне рішення приймається як середнє геометричне, або як усереднене,з врахуванням показника компетентності груп експертів, ( – показники компетентності).

Якщо потрібно визначити ранжування альтернатив відносно глобальної якості ПС, то необхідно визначити пріоритети критеріїв якості ПС, , застосувавши модифіковану процедуру МАІ.

Визначити ваги альтернатив відносно реалізації глобального критерію якості ПС можна за значенням показника:

|  |  |
| --- | --- |
| . | (11) |

Тоді показником важливості альтернативи по множині критеріїв буде

|  |  |
| --- | --- |
| , | (12) |

і ранжування проводиться за значеннями .

Отримані проранжовані як по окремих показниках якості,так і по їх сукупності,альтернативні архітектури використовуються для прийняття остаточного рішення . Однак, як відмічалось раніше, при застосуванні МАІ використовуються експертні оцінки при парних порівняннях, а також при призначенні пріоритетів критеріям якості. Це може привести до помилок при прийнятті рішень, особливо коли оцінки альтернатив мало відрізняються. Також використання інтегрального показника приховує взаємозв’язки між критеріями якості і міру прийнятого компромісу між ними при призначенні ваг критеріям.

**Аналіз компромісів атрибутів якості.**

Для аналізу прийнятих компромісів між критеріями якості при виборі архітектури за значеннями інтегрального показника якості використовується графічне зображення їх взаємозалежностей. На рис.4 зображена діаграма компромісів між вартістю та модифікованістю для за результатами оцінювання альтернатив,отриманими в [8]. Таке представлення дає змогу візуалізувати можливі компроміси, їхні відносні розміри та відношення між альтернативами розробки з точки зору компромісів.

Рис.4. Компроміси між критеріями якості

Коли конфліктуючі атрибути якості мають однаковий пріоритет, то графічно компроміс між ними буде представлений точкою на діагоналі першого квадранту. Коли архітектурне рішення має бути обране з врахуванням пріоритету одного з атрибутів якості, то обирають Парето-оптимальне рішення, яке знаходиться найвище відносно діагоналі першого квадранту, коли пріоритетний атрибут відкладається по осі ординат. У випадку, коли пріоритетний атрибут відкладається по осі абсцис, то Парето-оптимальним рішенням буде те, що знаходиться нижче від діагоналі у крайньому правому положенні.

**Аналіз чутливості.**

Забезпечена також можливість дослідження чутливості рішень, отриманих з використанням МАІ, до зміни пріоритетів критеріїв якості для прийняття обґрунтованого рішення по вибору архітектури.

Нехай ми маємо проранжовані архітектурні альтернативи, отримані методом аналізу ієрархій. Приведемо формулу для визначення мінімальної зміни абсолютної величини ваги атрибуту якості , таку, що порядок слідування  та  поміняється на протилежний:

. (13)

Тут – мінімальна зміна величини пріоритету *Ps*критерію якості *Ks* ,яка змінює порядок слідування сусідніх альтернатив *Ai* та *Aj* у проранжованому списку на зворотній. Найменше значенняпоказує, що пріоритет *Ps* атрибуту *Ks*є критичним до змін оцінок в парних порівняннях. Це рівняння можна також використати у випадку зміни вимог до ПС у процесі проектування, яке може привести до зміни пріоритетів відносно критеріїв якості.

Для кожного атрибуту якості можлива наявність декількох значень , які можуть спричинити зміну порядку слідування сусідніх альтернатив.

**2. Програмна система комплексу**

**2.1. Об'єктна модель комплексу**

Для проектування і подальшої побудови системи відповідно до об’єктно-орієнтованого підходу,застосовано графічне моделювання з використанням діаграм UML. Для розробки статичної моделі об’єктів системи використані діаграми класів, а для розробки функціональних вимог–Usecase діаграми (діаграми сценаріїв).

В системі “Архітектор ПС” наявні три типи(ролі) користувачів:

*адміністратор* – додає у базу нові архітектури додатків, шари, модулі йпатерни та обслуговує репозиторій патерніві систему в цілому.

*архітектор* – комбінує патерни, для вирішення класів специфікованих задач шляхом формування наборів альтернативних архітектур.

*експерт* – оцінює сформовані архітектором множини альтернативних архітектур програмних додатків за певними визначеними критеріями якості.

“Архітектор програмних систем” складається з трьох підсистем:

1. Підсистема управління репозиторієм патернів (*адміністратор*).
2. Підсистема створення альтернативних архітектур (*Архітектор*).
3. Підсистема оцінювання архітектур (*Експерт*).

***Діаграми сценаріїв системи***

функціональні можливості *адміністратора*забезпечуються використанням підсистеми управління репозиторіємпатернів, діаграма сценаріїв якоїпоказана на рис5.



Рис. 5. Сценарії роботи *Адміністратора* з управління репозіторієм патернів архітектур

Актор на діаграмі –це*Адміністратор репозиторію патернів*, який керує роботою з базою даних архітектур.Варіанти використання (сценарії):

* Редактор архітектур – базовафункція редагування репозиторію патернів.
* Створення БД – створення нової БД архітектур патернів.
* Створення архітектури – створення нової архітектури в БД.
* Редагування архітектури – редагувати існуючу архітектуру в БД.
* Видалення архітектури – видалення існуючої архітектури, яка вибирається із списку архітектур репозиторію.
* Збереження архітектури – збереження внесених змін у архітектуру додатку.
* Створення шару – створення шару в архітектурі,обраній для редагування.
* Редагування шару – редагування існуючого шару архітектури додатку.
* Видалення шару – видалення шару з архітектури, відкритої для редагування.
* Створення модулю – створення модулю (категорії)у відкритій архітектурі.
* Редагування модулю – редагування модулю у відкритій архітектурі.
* Видалення модулю – видалення модулю з відкритої архітектури.
* Менеджер патернів – менеджер патернів для зручного управління патернами в модулях (категоріях) архітектур програмних додатків.
* Візуалізація архітектури – відображення вигляду поточної архітектури,що редагується.
* Редактор патернів – редактор патернів викликається при редагуванні набору патернів поточного модулю (категорії):
  + Створення патерну – створення нового патерну.
  + Редагування патерну – редагування вже існуючого патерну.
  + Видалення патерну – видалення патерну вибраного зі списку.
  + Візуалізація патерну – візуалізувати текстовий описпатерну у графічну структуру патерну.
  + Збереження патерну – зберегти внесені зміни до патерну в БД.

функціональні можливості *Архітектора*з абезпечуються використанням підсистеми створення альтернативних архітектур, діаграма сценаріїв якої показана на рис 6.

Тут актор є *Архітектором*, який запускає процес створення альтернативних архітектур, обирає батьківську архітектуру та патерни для підстановки у модулі (категорії) шарів програмного додатку, вводить описову частину завдання та приймає конструктивні рішення щодо побудови альтернативних архітектур програмних додатків.



Рис. 6. Сценарії роботи *Архітектора* з конструювання нових варіантів архітектур

Варіанти використання (сценарії) підсистеми:

* Створення альтернативних архітектур–базовий Usecase, який представляє головну задачу підсистеми і викликає інші сценарії.
* Під'єднання до бази даних – вибір бази даних (репозиторію патернів) для подальшої роботи з нею.
* Вибір архітектури – обрання батьківської архітектури для побудови альтернативних архітектур додатку.
* Вибір патернів за модулями – вибір патернів для кожного модуля відповідного шару архітектури та формування альтернатив.
* Огляд створених архітектур – огляд архітектором створених альтернативних архітектур програмного додатку перед збереженням.
* Збереження альтернативних архітектур – виконання збереження сформованих альтернатив у вигляді завдання для подальшої роботи *Експертів*.
* Введення описової частини завдання.
* Збереження завдання з альтернативними архітектурами додатків.

функціональні можливості *Експерта* забезпечуються використанням підсистеми оцінювання архітектур,діаграма варіантів використання якої показана на рис. 7. Тут актор – *Експерт*, який запускає процес оцінювання, оцінює альтернативні архітектури програмних додатків попарно та виставляє оцінки.



Рис. 7. Сценарій роботи *Експерта* з оцінювання існуючих варіантів архітектур

Підсистема має такі варіанти використання (сценарії) :

* Оцінка альтернативних архітектур – базовий Usecase, який забезпечує головну задачу підсистеми і визиває інші сценарії.
* Вибір завдання для оцінки – виконання вибору завдання для подальшого оцінювання.
* Під'єднання до бази даних – вибір бази даних (*репозиторію патернів архітектур*) для подальшої роботи з нею.
* Порівняння архітектур – порівняння альтернативних архітектур між собою попарно та проведення оцінювання.
* Порівняння пари архітектур у текстовому (табличному) вигляді.
* Порівняння пари архітектур у графічному вигляді (у виді діаграм).
* Виставлення оцінок,як результат попарного порівняння.
* Збереження результатів оцінювання.

2.2. Програмна реалізація системи

Діаграми класів “Архітектора програмних систем” відображають логічну структуру й функції системивідповідно до трьох складових підсистем. діаграма класів підсистеми управління репозиторіємпатернів показана на рис. 8.



Рис.8. Діаграма класів підсистеми управління репозиторієм патернів

Класи на діаграмі:

* Architecture – клас архітектури зі змінними і функціями для роботи з архітектурами програмних додатків.
* Layer – клас шар представляє собою набір даних і функцій для роботи з шарами програмного додатку.
* Module – клас модуль представляє собою комплект змінних і функцій для роботи з категоріями (модулями) шарів архітектури.
* Pattern – клас патерн представляє собою комплект даних і функцій для роботи з шаблонами (патернами) різних категорій.
* Task – клас задача, що використовується для збереженння множини побудованих альтернативних архітектур для поточного типу додатка.
* result\_info – клас, який повертає значення щодо виконання операції (якщо відбулася помилка – повертається стек помилок).
* Derby DB Manager – клас роботи з БД архітектур.
* patern\_e\_C – клас-контролер редактора паттернів, який містить в собі функції кнопок і пунктів меню.
* main\_C – клас-контролер головного вікна підсистеми управління репозиторієм патернів.

діаграма класів підсистеми створення альтернативних архітектур показана на рис.9.



Рис. 9. Діаграма класів підсистеми створення альтернативних архітектур

Класи на діаграмі:

* Create\_arch – клас, який являє собою головне виконавче тіло підсистеми створення альтернативних архітектур.
* Arch\_work – допоміжний клас, який містить у собі функції роботи з архітектурами програмних додатків.

Операції класів (функції):

* Вибір бази даних (репозиторію)() – вібір бази даних (репозиторію), де зберігаються архітектури типів програмних додатків і патерни, над якими будуть виконуватись операції.
* Вибір архітектури додатку() – вибір батьківської архітектури для побудови альтернатив.
* Вибір патернів по модулях() – вибір патернів для кожного модуля з метою формування альтернатив.
* Візуалізація готових альтернативних архітектур().
* Створення завдання() – ввід описової частини завдання на оцінювання.
* Збереження альтернативних архітектур у БД().
* Формування структури архітектури() – формування архітектури, з врахованням структури архітектури додатку та насиченості її патернами.

діаграма класів підсистеми оцінювання архітектур показана на рис. 10.



Рис. 10. Діаграма класів підсистеми оцінювання архітектур

Класи на діаграмі:

* Rating\_arch – клас, що являє собою головне виконавче тіло підсистеми, в якому проводиться оцінювання альтернативних архітектур.

Операції класів (функції ):

* Вибір бази даних() – вібір бази даних (репозиторію), в якому зберігаються архітектури програмних додатків і патерни, над якими будуть виконуватись операції.
* Вибір завдання() – вибір завдання на оцінювання.
* Формування текстового зображення архітектури().
* Виставлення оцінки() – при виставленні оцінки матриця оцінок доповнюється та візуалізується нова пара архітектур для подальшого оцінювання.
* Візуалізація матриці оцінок() та збереження оцінок().

# Висновки

Застосування запропонованої системи при проектуванні програмного забезпечення дозволить за рахунок використання формалізованих моделей, репозиторію готових архітектурних патернів та методів оптимізації підвищити якість програмних продуктів, скоротити терміни розробки і зменшити витрати.

Використання методу формалізації вимог якості дозволяє зменшити кількість помилок в специфікаціях приблизно на 30% [10]. Метод комунікації вимог якості дає змогу обґрунтовано вибирати критерії якості архітектури, а алгоритм,розроблений на його основі,дозволяє автоматизувати дану процедуру.

Застосування методу МАІ дозволяє вибирати найкращу архітектуру з множини альтернативних по сукупності критеріїв якості, на відміну від відомих методів ATAM, CBAM та інших,в яких архітектура оцінюється по одному критерію.

Застосування запропонованого алгоритму визначення ваг альтернатив в МАІ з умови мінімуму неузгодженості матриці парних порівнянь розширює межі застосування методу на більшу кількість альтернатив (*n*>9) , тобто вибір рішення здійснюється на всій множині альтернатив. А при використанні стандартного МАІ для цього випадку множина альтернатив ділиться на кластери з*n*<9, і задача розв’язується для кожного кластера окремо,а потім порівнюються кращі варіанти з кожного кластера. А це значно збільшує час на пошук рішення.Так,для прикладу, розглянутого в [11],при *n*=15множину альтернатив треба поділити на два кластери, і тоді час на пошук найкращої альтернативи при використанні стандартного алгоритму буде більш, ніж в два рази перевищувати час для модифікованого алгоритму. Використання співвідношення (12) при зміні вимог дозволить уникнути повторного ранжування альтернатив,якщо величини змін вкладаються у визначені інтервали. Застосування методу компоновки альтернативнихархітектуртарепозиторіюпатернівдаєзмогуефективновикористовуватиготовіархітектурніпатерни при виборі оптимального рішення для широкого спектру категорій програмних застосувань (додатків).

**ЛІТЕРАТУРА**

1.Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования [Текст] / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. – СПб. : Питер, 2010. – 366 с. – ISBN 978-5-469-01136-1.

2. Лавріщева К.М. Програмна інженерія. [Текст] / Лавріщева К.М.– К.: Академперіодика, 2008.– 319с.

3. Руководство Microsoft по проектированию архитектуры приложений. 2-е издание. Microsoft, 2009. – 529 с

4. Kazman, R. ATAM: Method for Architecture Evaluation [Teкст] / RickKazman, MarkKlein, Paul Clements.–Pittsburgh, PA:Software Engineering Institute, Carnegie Mellon University, August 2000. – CMU/SEI-2000-TR-004, ADA377385. – 83p.

5. T. Al-Naeem, I. Gorton, M.A. Babar, F. Rabhi, and B. Benatallah, “A quality driven systematic approach for architecting distributed software applications”,Proceedings of the 27th International Conference on Software Engineering(ICSE), St. Louis, USA, 2005.

6. Павлов А.А. Математические модели оптимизации для нахождения весов объектов в методе парных сравнений. Павлов А.А, Лищук Е.И., Кут В.И. // Системні дослідження та інформаційні технології. – К.: ІПСА, – 2007. №2, с. 13 – 21.

7. HarchenkoAlexandr, BodnarchukIhor, HalayIryna, YatcyshynVasyl. Software Architecture Design on the Base of Method of Hierarchic Optimization // Proceeding of VIIIth International Conference on Perspective Technologies and Methods in MEMS Design. pp. 39–40, Polyana, 2012.

8. Харченко О.Г. Метод багатокритеріальної оптимізації програмної архітектури на основі аналізу компромісів [Текст] / Харченко О.Г., Боднарчук І.О., Галай І.О. // Інженерія програмного забезпечення. – 2012. – № 3–4 (11–12). – с. 5–11

9. Харченко О.Г. Інструментальний засіб розробки та комунікації вимог якості до програмних систем [Текст] / О.Г. Харченко, В.В. Яцишин, І.Е. Райчев // Інженерія програмногозабезпечення. – 2010. – № 2. – с. 29–34.

10. Letichevsky A. Basic Protocols, Message Sequence Charts, and the Verification of Requerements Specifications[Text] / A. Letichevsky, J. Kapitonova, A. Letichevsky Jr., V. Volkov, S. Baranov, V. Kotlyarov//ISSRE 2004, WITUL, Rennes, 4 November 2005.– pp. 112–142.

11.Harchenko, A. Decision support system of software architect. // A. Harchenko, I. Bodnarchuk, I. Halay / Proceedings of IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2013. Berlin. 2013. Vol.1, – pp. 265 – 269.

11.Fowler, Martin. Patterns of Enterprise Application Architecture [Text] / Martin Fowler, David Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, Randy Stafford.: Addison-Wesley, 2002. – 533 p.

11.Подиновский В. В. Введение в теорию важности критериев в многокритериальных задачах принятия решений. [Текст] / Подиновский В. В. – М.: Физматлит, 2007. – 64 с.

12. Павлов О.А.Оперативные алгоритмы принятия решений в иерархической системе Саати, основанные на замещении критериев [Текст]/ Павлов О.А., Ліщук К.І. // Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка. К.: “ВЕК+”, 2008.– №48. – с. 78 – 81