Серія: Механіко-технологічні системи та комплекси. – Харків : НТУ «ХПІ», 2017. – No 33(1255). – С. 39–42.– Бібліогр.: 19 назв. – ISSN 2079-5459.

Анализ полупроводниковых сенсоров современных газоанализаторов/ Дейнеко Н. В. // Вісник НТУ «ХПІ». Серія: Механіко-технологічні системи та комплекси. – Харків : НТУ «ХПІ», 2017. – No 33(1255). – С. 39-42.– Бібліогр.: 19 назв. – ISSN 2079-5459.

Analysis of semiconductor sensors of modern gas analyzers/ Deyneko N. //Bulletin of NTU "KhPI". Series: Mechanical-technological systems and complexes. – Kharkov: NTU "KhPI", 2017. – № 33 (1255).– P. 39–42.– Bibliogr.:19. – ISSN 2079-5459

Відомості про авторів / Сведения об авторах / About the Authors

Дейнеко Наталя Вікторівна **–** кандидат технічних наук, старший науковий співробітник наукового відділу з проблем цивільного захисту та техногенно-екологічної безпеки науково-дослідного центру, Національний університет цивільного захисту України, вул. Чернишевська, 94, м. Харків, Україна, 61023; e-mail: natalyadeyneko@gmail.com.

Дейнеко Наталья Викторовна – кандидат технических наук, старший научный сотрудник научного отдела по проблемам гражданской защиты и техногенно-экологической безопасности научно-исследовательского центра, Национальный университет гражданской защиты Украины, ул. Чернышевская, 94, г. Харьков, Украина, 61023; e-mail: natalyadeyneko@gmail.com.

Deyneko Natalia – PhD, Senior Researcher of the Scientific Division for Civil Defense Issues and Technogenic and Environmental Safety Research Center, National University of Civil Protection of Ukraine, Chernyshevskaya, 94, Kharkiv, Ukraine, 61023; e-mail: natalyadeyneko@gmail.com.

УДК 621.391

Я. Ю. ДОРОГИЙ

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ПРЕДСТАВЛЕННЯ ТА ОБГРУНТУВАННЯ АРХІТЕКТУРИ КРИТИЧНОЇ ІТ-ІНФРАСТРУКТУРИ

Розглядаються підходи до обґрунтування проектних рішень щодо архітектури критичної ІТ-інфраструктури. Проведено детальний порівняльний аналіз основних підходів, що використовуються для проектування архітектури на базі запропонованих критеріїв порівняння. Визначені їх основні недоліки та переваги, перевірена можливість використання даних підходів при проектуванні критичної ІТ-інфраструктури. Зроблено висновок про необхідність створення нового підходу до обґрунтування проектних рішень щодо архітектури критичної ІТ-інфраструктури через неможливість застосування розглянутих методів та підходів.

Ключові слова: архітектура підприємства, критична ІТ-інфраструктура, компаративний аналіз, ADDT, V&B, AREL, QOC, IBIS.

Рассматриваются подходы к обоснованию проектных решений по архитектуре критической ИТ-инфраструктуры. Проведен детальный сравнительный анализ основных подходов, используемых для проектирования архитектуры на базе предложенных критериев сравнения. Определены их основные недостатки и преимущества, проверена возможность использования данных подходов при проектировании критической ИТ-инфраструктуры. Сделан вывод о необходимости создания нового подхода к обоснованию проектных решений по архитектуре критической ИТ-инфраструктуры из-за невозможности применения рассмотренных методов и подходов.

Ключевые слова: архитектура предприятия, критическая ИТ-инфраструктура, компаративный анализ, ADDT, V & B, AREL, QOC, IBIS.

The article deals with issues related to the problem of justification of design decisions regarding the architecture of critical IT infrastructure.

The result of the study is: a comparative analysis of the approaches to justify and present the architecture of the enterprise based on imposed criteria, identification of their disadvantages and advantages, possibility of its using to justify the design of critical IT infrastructure. It is concluded that there is a need to create a new approach to the justification of design solutions for the architecture of critical IT infrastructure due to the impossibility of applying the methods and approaches considered.

As a scientific novelty, for the first time, a detailed comparative analysis of the methods of presentation and justification of design solutions of the enterprise architecture from the point of view of their application for the design of the critical IT infrastructure was considered.

The practical significance of the proposed analysis lies in the fact that it can be used to create rationionale model for automated decision support systems for designing and justifying design decisions about the architecture of a critical IT infrastructure, while developing additional tools for architects of critical infrastructure architectures and public critical infrastructure objects.

Keywords: enterprise architecture, critical IT infrastructure, comparative analysis, ADDT, V & B, AREL, QOC, IBIS.

Вступ. Мови представлення архітектури підприємств (АП) розглядаються як інструмент цілісного описування підприємства [1], що пов'язують всі напрямки діяльності організації, які зазвичай розглядаються окремо. Маючи такий опис, можна розглянути загальносистемний вплив змін [2] на всіх рівнях від бізнес-процесів до ІТ [3]. Наприклад, для новоствореного ІТ-застосування мови моделювання АП можуть бути використані для розгляду впливу на бізнес-процеси, людські ресурси, організаційні цілі тощо.

Хоча мови моделювання АП можуть бути використані для вираження цілісного дизайну організації, вони не надають проектних рішень за отриманими моделями. Ми повинні бути обережними у використанні аналогії, так як досвід у галузі архітектури про

грамного забезпечення показує, що залишення неявним обґрунтування дизайну веде до «випаровування архітектурних знань» [4]. Це означає, що без конструктивного обґрунтування залишаються неявними критерії проектування, причини проектування та альтернативні рішення.

Аналіз літературних даних та постановка проблеми. У результаті недостатньої раціоналізації архітектори не можуть підтвердити якість своїх проектів. Крім того, нові проекти не беруть до уваги обмеження, що випливають з попередніх проектних рішень [5]. Ці обмеження можуть бути бізнесового або технічного характеру, наприклад вибір мови програмування.

Крім того, дослідження серед практиків проектування АП [6] дає вказівки на корисність раціоналізації проектування з метою мотивації проектних рішень та архітектурного забезпечення. У той же час, однак, дослідження показує, що професіонали часто відмовляються від використання структурованого шаблону/підходу при раціоналізації архітектури, покладаючись, натомість, на фіксацію спеціальної інформації за допомогою таких простих інструментів, як Microsoft Office.

Проте раціоналізаторські підходи з фокусуванням на програмну архітектуру зосереджені на проблемах програмного забезпечення (наприклад, кодова документація). Проте ці проблеми відрізняються від тих, які виникають в корпоративній архітектурі [7] і, особливо, в архітектурі критичних інфраструктур. Архітектори підприємств займаються повним стеком проблем бізнес-ІТ [3]. Наприклад, вони аналізують вплив змін в ІТ-інфраструктурі на бізнес-процеси та навпаки. Таким чином, архітектори підприємств вирішують різні, перехресні організаційні питання, в той час, як програмні архітектори займаються головним чином проблемами програмного забезпечення. Тому завдання полягає в визначенні такого підходу, який би використав переваги цих методів, зневілював їх негативні сторони та надав би можливість дослідникові використовувати його при проектуванні критичної ІТ-інфраструктури.

Ціль та мета дослідження. Метою дослідження є визначення основних особливостей підходів до раціоналізації архітектури та можливості їх використання для проектування та обґрунтування проектних рішень щодо критичної ІТ-інфраструктури.

Задачею дослідження є визначення недоліків основних підходів до раціоналізації архітектури, їх порівняння з точки зору використання як підходу до раціоналізації архітектури критичної ІТ-інфраструктури.

Для досягнення поставленої мети були поставлені наступні завдання:

1. Дослідження існуючих підходів та визначення їх недоліків та переваг.

2. Дослідження можливості їх застосування для обґрунтування проектних рішень щодо критичної ІТ-інфраструктури.

Методи обґрунтування проектних рішень архітектури. Прийняття рішення щодо вибору дизайну системи включає наступні компоненти:

– обґрунтування проектних рішень;

– стратегії прийняття цих рішень;

– чинники, що впливають на їх прийняття.

Обґрунтування рішення. Просте пояснення обґрунтування проектного рішення – *явне* / *чітке* уявлення про процес міркування над проектом. Існують і інші аспекти проектування. Принцип проектування може бути наміром мотивувати створення артефакту проектування, бути обмеженням або припущенням, яке впливає на артефакт проектування, компромісом між вимогами, судженням для вибору з ряду варіантів проектування та аргументацією «за» чи «проти» проектної пропозиції.

Є різні підходи до процесу проектного міркування. Один з них – аргументація. Основним аргументаційним представленням є використання вузлів та посилань для позначення знань та відносин. Він бере свій початок від аргументованого представлення Тулміна, яке будується за допомогою даних, вимог, гарантій, підстав та спростувань [8]. У схемі Тулміна прикладом вузла є дані або претензія, які являють собою спостереження та висновок відповідно. *Тип*/*вид* зв'язку типу «так» може бути використаний для з'єднання вузлів для представлення індуктивних відносин. Цей підхід став базою для численних аналогічних аргументальних підходів, таких як інформаційна система на основі проблем (IBIS) [1] та мова обґрунтування проектування (DRL) [9]. Вони в основі демонструють проблему, аргумент та вирішення аргументації проектування.

Інший підхід до представлення концепції проектування полягає у використанні методологій на основі шаблонів. Ці методи використовують стандартні шаблони, які входять в процес розробки, щоб полегшити схему обґрунтування проектування. На відміну від методів, що базуються на аргументації, практики, які використовують методи на основі шаблонів, не створюють аргументаційні діаграми для обговорення, а натомість фіксують результати міркування. Цей підхід орієнтований на практичне застосування обґрунтування проектування в галузі. Прикладами цього підходу є шаблон опису рішення архітектури (ADDT) [10] та шаблон Views and Beyond (V&B) [11].

Використання підходів обґрунтування проектів заснованих на аргументації іноді недостатньо, щоб визначити правильне рішення серед альтернатив. На проектне рішення може впливати багато факторів та вимог. Деякі з них мають більш високе значення або пріоритет, ніж інші. Тому, додатково у процесі прийняття рішень можуть використовуватися кількісні методи оцінки пріоритетів, витрат та переваг. Прикладом такого підходу є метод аналізу витрат (CBAM) [12].

Розглянемо більш детально основні методи обґрунтування проектних рішень.

Інформаційна система на основі проблем (IBIS) та її варіанти. Інформаційна система, що базується на проблемах (IBIS), є методом структурування та документування обґрунтування проекту [1], що базується на обговоренні проблеми. Подібний підхід був використаний і в інших областях проектування, таких як будівництво архітектурного проекту та містобудування.

Ключовим аспектом дизайну IBIS є артикуляція проблем. За кожною проблемою слідують позиції, що

відповідають на питання. Позиція може бути підтримана або заперечена аргументами. Позиція може походити від або бути наслідком артефакту. На рисунку 1 представлена діаграма, яка показує такі відносини. Різні проблеми обговорення пов'язані такими відносинами, як «аналогічно до», «замінює», «тимчасовий наступник» тощо.

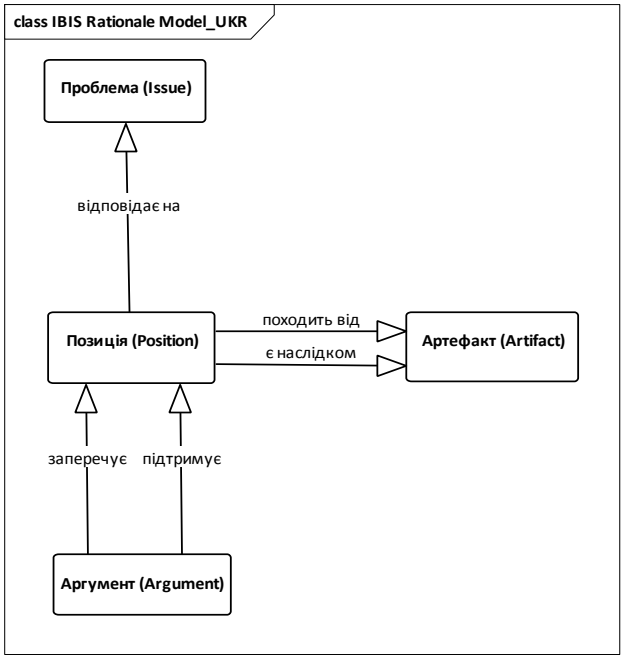

Рис. 1 – Модель обгрунтування дизайну IBIS

З 1970 по 1980 рр. було зроблено багато спроб використання IBIS, але жодна з цих систем не пройшла етап пілотного проекту [13]. Це пов'язано з тим, що в IBIS не використовуються наступних два важливих моменти:

– пов'язані між собою питання, такі як субпитання, не можуть бути представлені в IBIS в якості залежності, і тому важко моделювати деякі обговорення;

– плюси і мінуси альтернативних позицій не обговорюються.

Мета-модель методу прийняття рішення представлена на рис. 2.

IBIS підходить для обґрунтування простого дизайну, але може не спрацювати, коли проекти стають складнішими та захараченими.

Щоб подолати ці обмеження, МакКолл розробив процедурну ієрархію проблем (PHI) для документування обгрунтування проекту [14]. PHI використовує більш широке визначення концептуальних позицій і використовує новий принцип для об'єднання проблем. У IBIS позиції визначають проектні питання, які обговорюються. В PHI враховується кожне проектне питання незалежно від того, обговорюється воно чи ні. PHI спрощує взаємовідносини в IBIS за допомогою спеціального типу відносин. Проблема A слугує проблемі B, якщо і тільки якщо вирішення A впливає на вирішення B. Таким чином, домінуючий тип відносин в PHI є спеціальний тип відносин «підпроблема».

PHI є простою квазіієрархічною структурою, яка з'єднує проблеми лише за допомогою спеціального типу відносин (рисунок 3). Подібно до IBIS, PHI забезпечує взаємозв'язок залежностей між проблемами і фіксує плюси і мінуси альтернативних позицій. PHI має лише обмежений успіх у прийнятті проектних рішень для промислового використання [15].

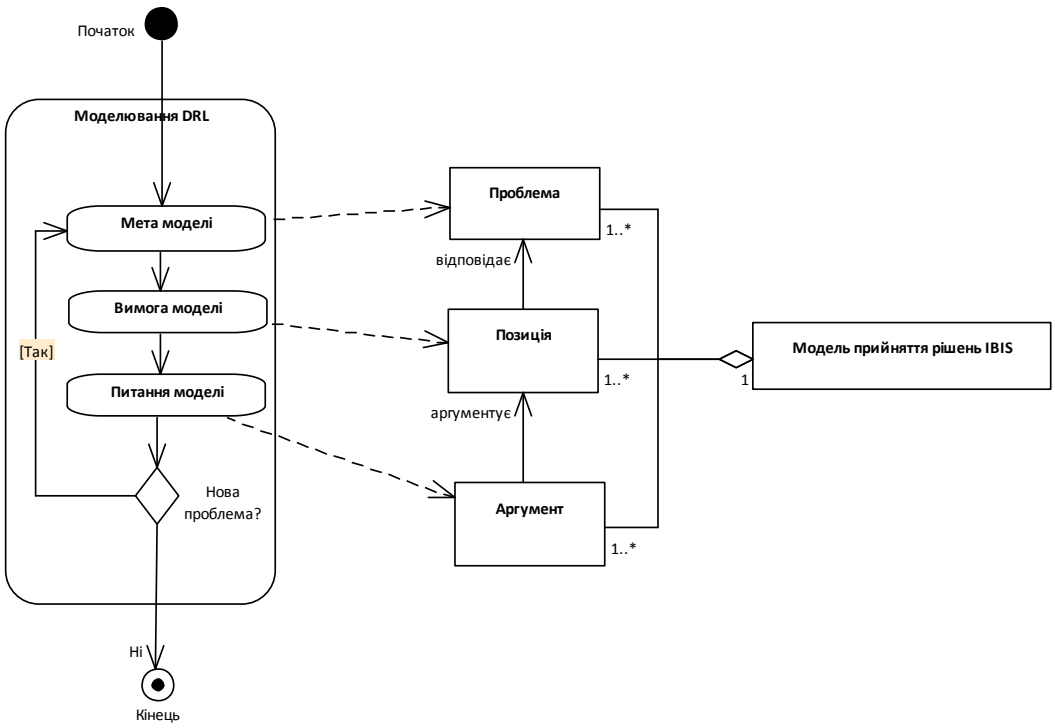

Рис. 2 – Модель рішення IBIS

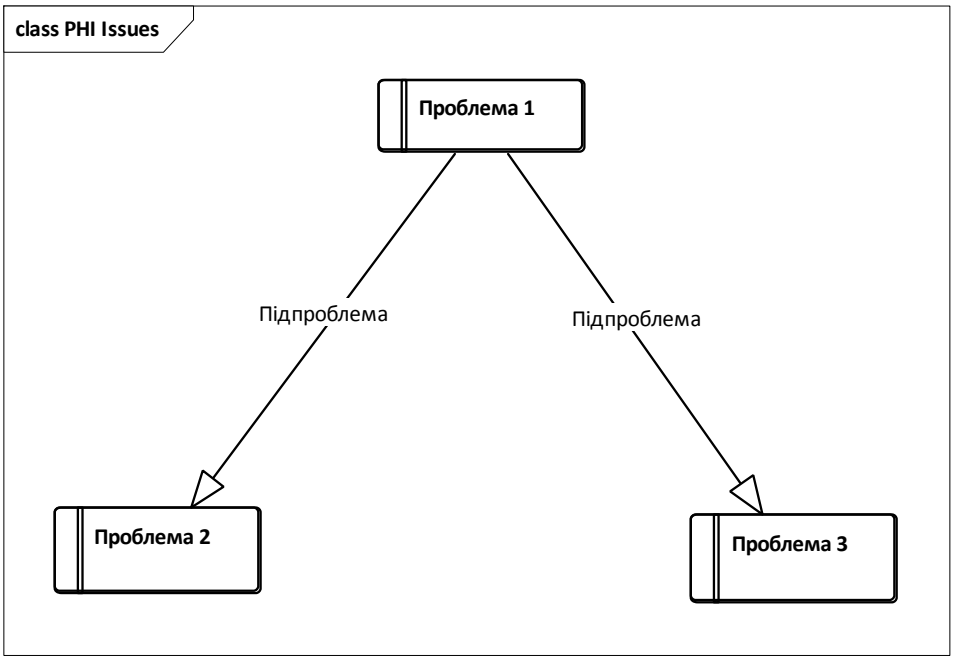

Рис. 3 – Проблеми в PHI

gIBIS покращує IBIS, дозволяючи використовувати «інший» тип вузла для включення зовнішніх матеріалів [16]. Він підтримує агрегацію дерева issue-position-argument (IPA) у вузол з'єднання IPA. Графічне представлення gIBIS реалізовано в MCC. Графічний макет gIBIS складається з:

– глобального представлення даних мережі;
– представлення локальної мережі;
– представлення індексу, що відображає ієрархію проблем і аргументи;
– представлення вузла, що показує вміст та атрибути вибраного вузла;
– представлення панелі керування.

Система представлення та обслуговування процесу знань (REMAP) також базується на методі IBIS [17]. Вона розширює основні конструкції IBIS, таких як проблема, позиція та аргумент з вимогами, обмеженнями та об'єктами проектування. Це розширення забезпечує зв'язок між вимогами та об'єктами дизайну для його обґрунтування (рис. 4)

Дана модифікація дозволяє зафіксувати історію проектних рішень в циклі розробки в якості знання процесу. Проблеми піднімаються та вирішуються на етапі проробки вимог при створенні об'єктів проектування.

Мова обґрунтування проектування (DRL). DRL визначає обґрунтування проекту, описуючи, як артефакт служить або задовольняє очікувані функції. DRL є мовою, яка представляє якісні елементи в просторі міркувань навколо рішень [18]. У DRL можливі варіанти проектування містяться в альтернативному просторі, а аргументи в підтримку або заперечення проекту містяться в аргументному просторі. Кожна конструктивна можливість оцінюється і результати розміщуються в просторі оцінки. Оцінка виконується відповідно до певних критеріїв, що містяться в просторі критеріїв. Проблеми, які є явними та містять альтернативи, оцінки та критерії, містяться в просторі проблеми. Основні типи об'єктів в DRL – мета, проблеми, вимоги та альтернативи. Структура графа прийняття рішення за допомогою цих елементів показана на рис. 5.

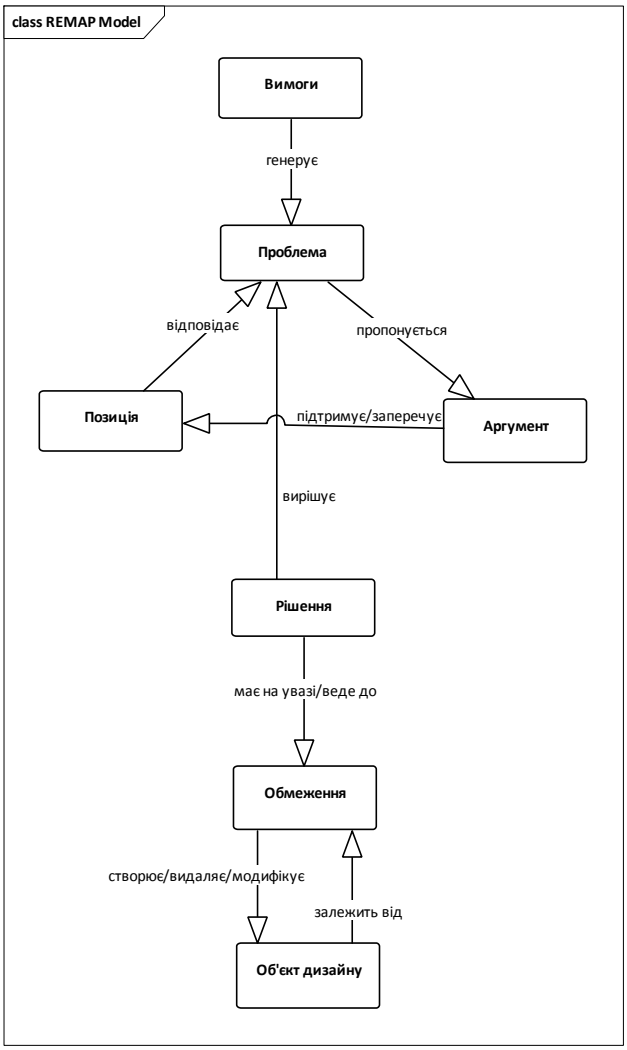

Рис. 4 – Модель REMAP

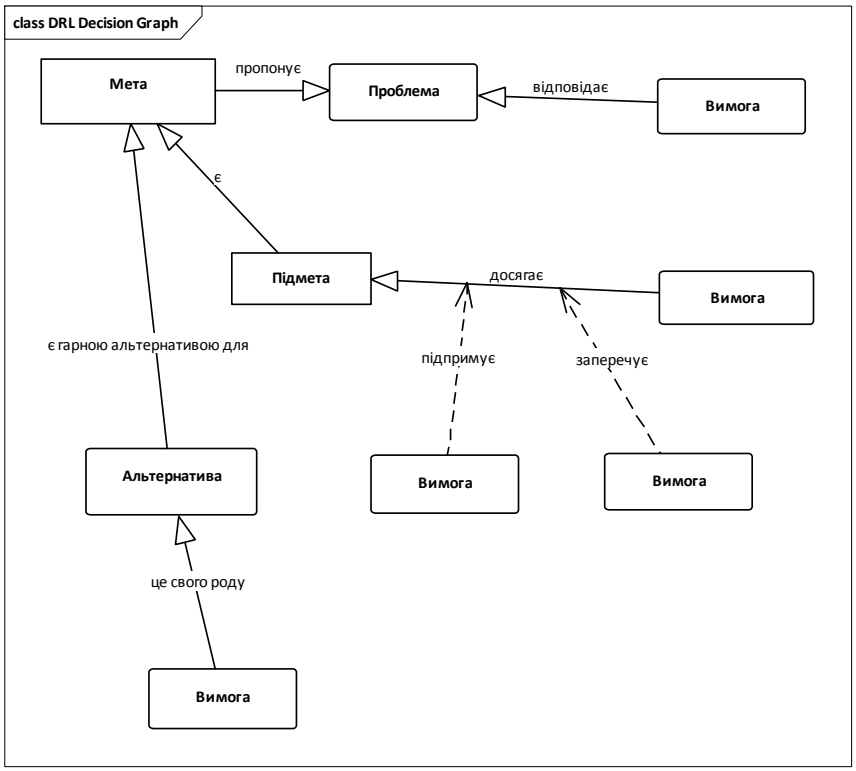

Рис. 5 – Структура графу прийняття рішень DRL

Мета – це критерії, які необхідно задовольнити. Альтернатива являє собою варіант, який розглядається. Його зв'язок з метою полягає в визначенні, чи це хороша альтернатива. Проблема є питанням, яке виникає з мети, на яку потрібно відповісти. Вимога є відповіддю на запитання і дозволяє досягти або не досягти мети. Мета-модель методу представлена на рис. 6.

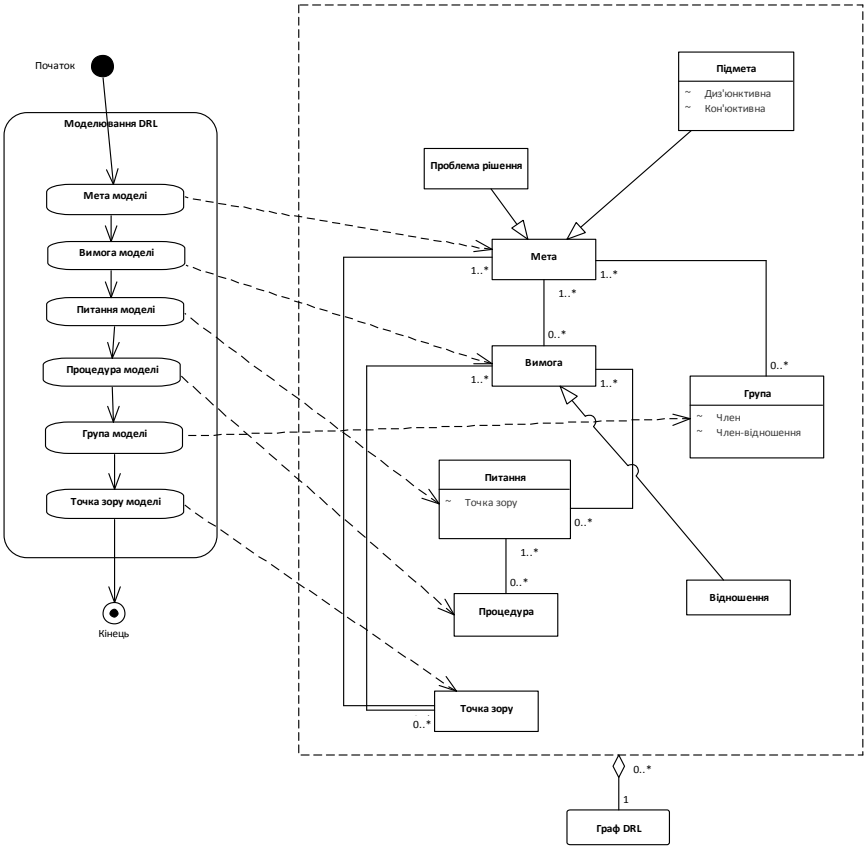

Рис 6 – Модель прийняття рішення DRL

Аргументація проектування в DRL побудована з наведених вище елементів. Вони в свою чергу згруповані в різні конструктивні та аргументаційні простори. DRL реалізується системою під назвою SIBYL [19].

Інженерія програмного забезпечення на базі обґрунтування проектного рішення (SeuRAT). Бург розробив систему SeuRAT для підтримки використання обґрунтування проектування під час обслуговування програмного забезпечення, асоціювання обґрунтування з кодом та здійснення ряду висновків за обґрунтуванням з метою забезпечення консистенції та повноти принципів проектування [20].

Система представлення обґрунтування проектування RATSpeak – це модифікація DRL. На рисунку 7 показані елементи, які представлені в RATSpeak. Ключовою відмінністю між RATSpeak і DRL є те, що RATSpeak представляє концепцію вимог, а не цілей. Вимоги складаються як з функціональних, так і з нефункціональних вимог. Проблема прийняття рішення пов'язана з вимогами для підтримки аргументу щодо альтернативних рішень. Аргументна онтологія – це ієрархія типових типів аргументів, які обслуговують типи вимог. Вони забезпечують можливість підтримки в SeuRAT синтаксичного та семантичного виведення. Синтаксичне виведення пов'язане з правильністю структури графу обґрунтування рішення. Наприклад, якщо існує альтернатива (тобто потенційне рішення) для прийняття рішення (тобто проблема), то альтернатива повинна мати аргумент, що її підтримує (тобто підтвердження). Семантичне виведення вимагає від дослідника вивчення змісту обґрунтування. Наприклад, дозволяє ідентифікувати вибрану альтернативу, яка не так добре підтримує іншу альтернативу, порівнюючи їх обґрунтування.

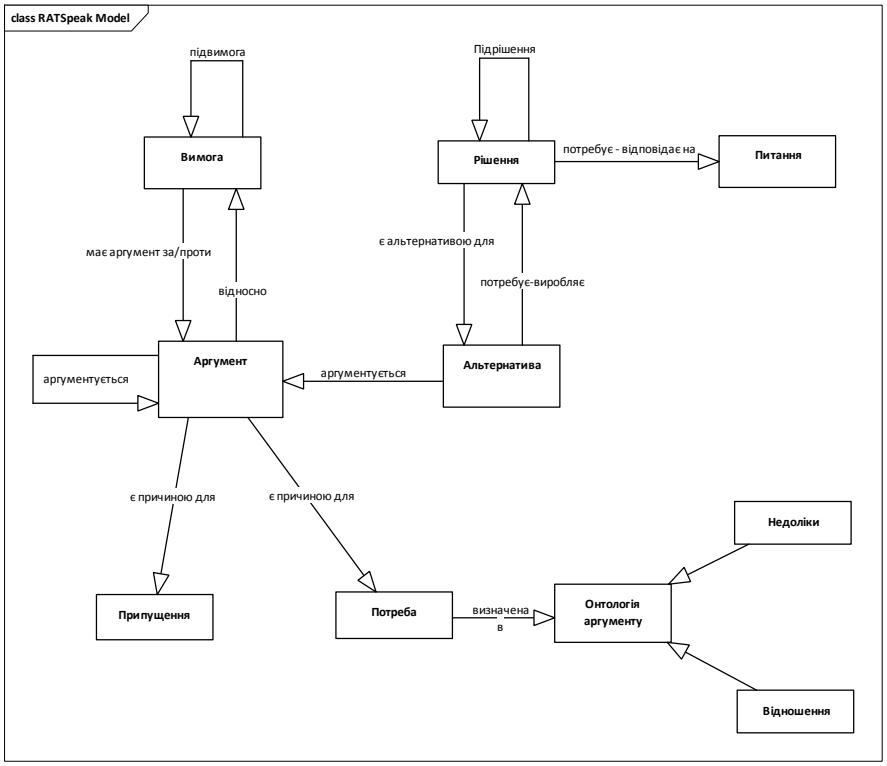

Рис. 7 – Модель обґрунтування рішення RATSpeak

Запитання, варіанти та критерії (QOC). QOC – напівформальна нотація, яке представляє аналіз простору проекту артефакту. Вона пояснює, як і чому артефакт проекту вибирається з простору можливостей. Основними елементами QOC є питання, які визначають основні проблеми проектування, варіанти, які надають можливі відповіді на поставлені питання, а також критерії оцінки та порівняння варіантів проектування.

Представлення QOC супроводжується аналізом простору проектування. Маклін та співавтори стверджують, що дослідники здатні аналізувати та аргументувати з QOC при розробці артефактів у просторі проекту в природному стилі [20]. Таким чином, аналіз обґрунтування проекту є просто додатковим продуктом проектування. На рисунку 8 зображене представлення QOC, яке показує приклад проектування об'єкту екрана. Питання в тому, чи повинен об'єкт екрану бути широким або вузьким. Якщо об'єкт є широким, то він використовує екранну нерухливість, але на ньому легко натиснути мишею. Якщо ж він вузький, то зберігає екранну нерухливість, але на ньому важко натиснути мишею. Оскільки обґрунтування є аргументом, а не доказом, Маклін та співавтори стверджують, що елементи QOC повинні використовуватися для обґрунтування проектування, навіть якщо вони можуть бути предметом додаткових аргументів. Таким чином, QOC може бути розширений на будь-який довільний рівень розробки. Дизайнери архітектур, що використовують QOC, повинні застосовувати ці аргументи тільки там, де вони будуть корисними, але не поширювати їх на всі можливі деталі проекту, оскільки це не є корисним.

QOC підтримує розвиток простору альтернатив. Це дає змогу розробникам враховувати критерії, що можуть

визначати життєздатність варіантів. На відміну від IBIS, PHI та REMAP, які відображають історію розробки проекту, QOC зосереджується на варіантах проектування. Мета-модель методу представлена на рис. 9.

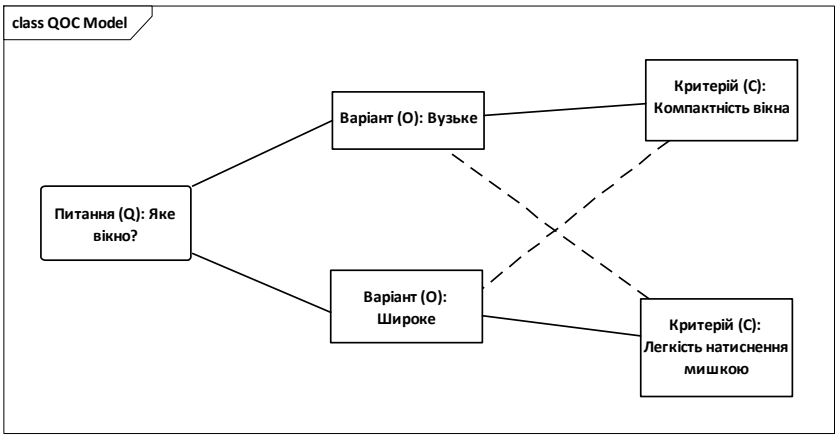

Рис. 8 – Приклад рішення QOC: суцільна лінія – позитивна оцінка, пунктирна лінія – негативна оцінка

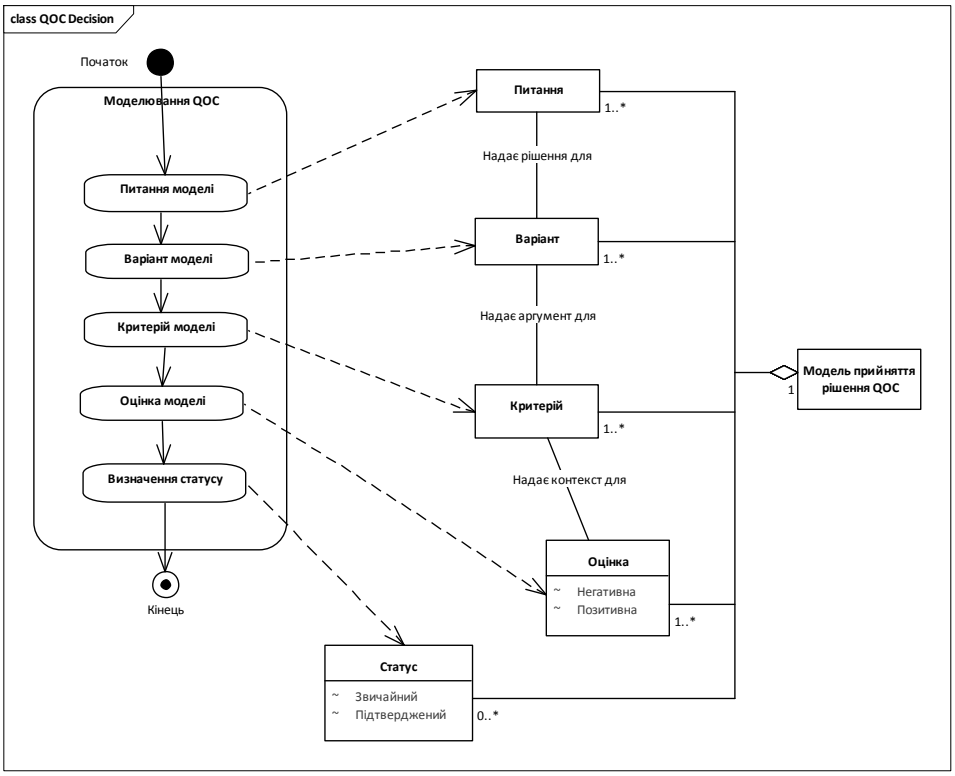

Рис. 9 – Модель прийняття рішення QOC

Дютоїт і Піч запропонували метод обґрунтування використання конкретного дизайну на основі обґрунтування, який поєднує в собі специфікацію використання з методом аргументації QOC [21]. Аргументація випадків використання, які складаються з функціональних та нефункціональних вимог, потім фіксуються в розширеній моделі QOC. Цей метод забезпечує кращу інтеграцію між специфікацією вимог, специфікацією проекту та його обґрунтуванням.

Views and Beyond (V&B). V&B – це сукупність методів, запропонованих Клеменсом та ін. для документування архітектури програмного забезпечення [11]. Він пропонує декілька типів представлень для опису архітектури програмного забезпечення:

– *модульний;*
– *компонентно-конекторний;*
– *виділення.*

Кожен тип представлення дає інший погляд на структуру системи. Наприклад, модульний тип представлення має декілька стилів подання, таких як розбиття коду та структура модулів за шарами.

Крім того, документуючи архітектуру, автори стверджують про те, що важливо документувати і те, чому саме така архітектура. Багато рішень приймається в проекті, але не всі рішення повинні бути виправданими. Запропоновано ряд принципів керівництва документацією щодо прийняття рішення:

– проектна команда витратила значний час, оцінюючи варіанти прийняття рішення;
– рішення має вирішальне значення для досягнення вимоги/мети;
– рішення вимагає розгляду нетривіальної під-

готовчої інформації;

– проблеми в нетривіальному рішенні;

– рішення має широке поширення на решту проекту архітектури і його буде важко скасувати;

– документування рішення є більш прийнятним зараз, а не пізніше.

Подібно до підходу ADDT, використовується прагматичний підхід до документування обґрунтування проекту. Замість документування обговорюваних відносин, фіксується наступна важлива інформація:

– *рішення* – резюме рішення;

– *обмеження* – основні обмеження, що виключають можливості;

– *альтернативи* – розглянуті варіанти та причини їх усунення;

– *ефекти* – наслідки та результати рішення;

– *докази* – будь-яке підтвердження того, що рішення є гарним.

Мета-модель методу представлена на рис. 10.

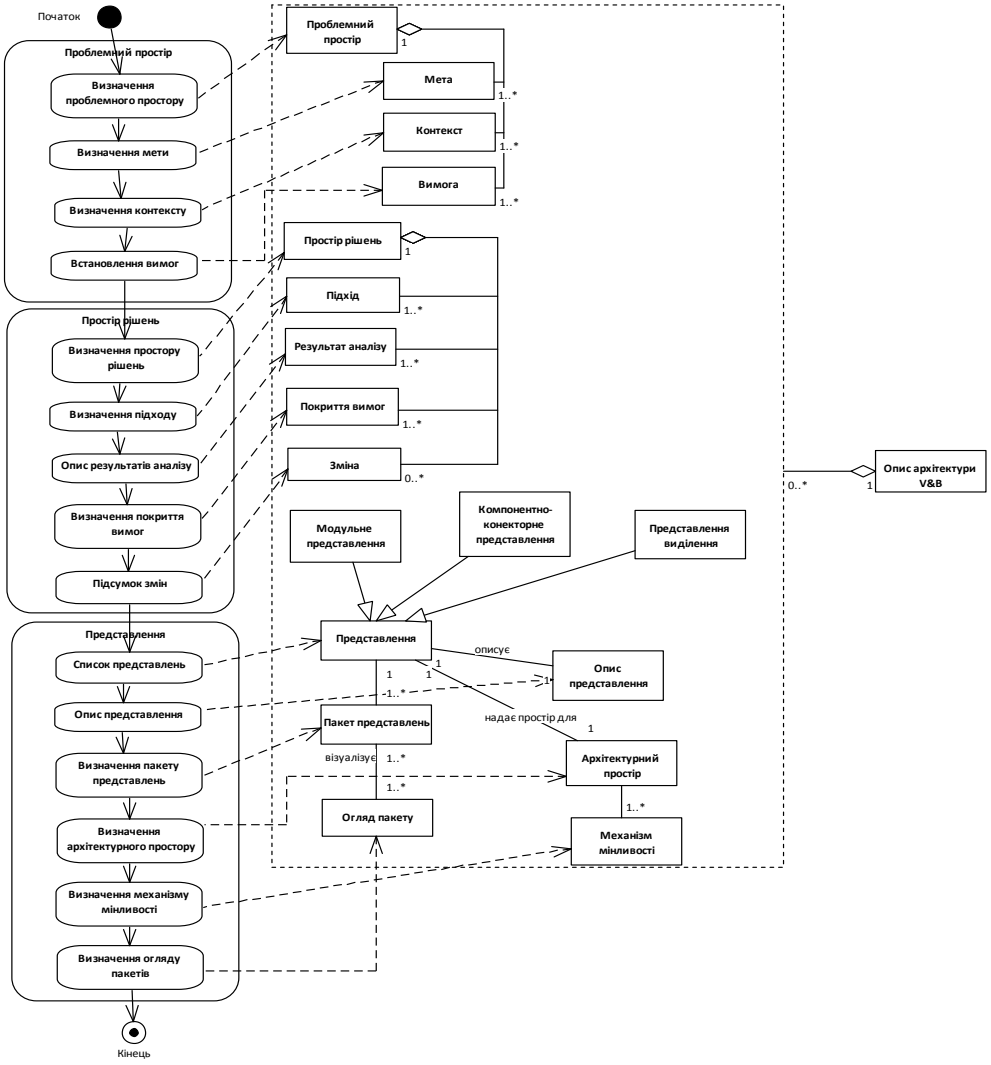

Рис. 10 – Модель прийняття рішення V&B

Вищенаведений список аргументів дизайну можна легко віднести до шаблону ADDT. ADDT надає більш детальну класифікацію, щоб допомогти дизайнерам зафіксувати всі обґрунтування проектних рішень. Шаблон V&B є більш загальним і простішим у використанні.

Подібно до ADDT, підхід V&B зосереджується на наданні дизайнерам повного шаблону та керівництва про те, як архітектура програмного забезпечення повинна бути задокументована. Є повний список, який описує, які елементи містять обґрунтування проекту, які його деталі слід задокументувати та як ці елементи пов'язані між собою.

Шаблон опису рішення щодо архітектури (ADDT). Тирі та Акерман запропонували прагматичний підхід до представлення обґрунтування проекту [10]. Замість того, щоб зосередити увагу на обґрунтуванні дизайну та моделі представлення, вони надають шаблон, який дозволяє відобразити обґрунтування рішення.

Шаблон опису рішення містить декілька ключових елементів:

– *проблема* – описує проблему архітектури;

– *рішення* – вказує остаточне рішення щодо архітектури (наприклад, вибрана позиція);

– *статус* – статус рішення: очікують на розгляд, вирішується або затверджене;

– *групування* – групування рішень в галузі архітектури за такими типами, як інтеграція, презентація та дані. Ця онтологія використовується для організації множини рішень;

– *припущення* – основні припущення, які впливають на рішення;

– *обмеження* – обмеження, які рішення можуть спричинити для середовища;

– *позиції* – життєздатні варіанти вирішення проблеми;

– *аргумент* – причини для підтримки позиції;

– *наслідки* – наслідки прийняття рішення, які можуть призвести до необхідності прийняття інших рішень, запровадження нових вимог, нових обмежень тощо;

– *пов'язані рішення* – рішення, які є взаємозалежними;

– *супутні вимоги* – цілі чи вимоги, пов'язані з рішенням;

– *пов'язані артефакти* – пов'язані з архітектурою, проектуванням і документами сфери застосування;

– *супутні принципи* – узгоджений набір принципів проектування, з якими рішення узгоджено;

– *примітки* – ідеї, котрі обговорювалися командою і зафіксовані як додаткова інформація.

Інформація, зібрана в цьому шаблоні, є повною. Вона забезпечує підтримку знань під час та після процесу проектування архітектури. Тирі та Акерман стверджують, що ця інформація – це той набір, який є потрібним для спілкування з іншою організацією [10].

Мета-модель методу представлена на рис. 11.

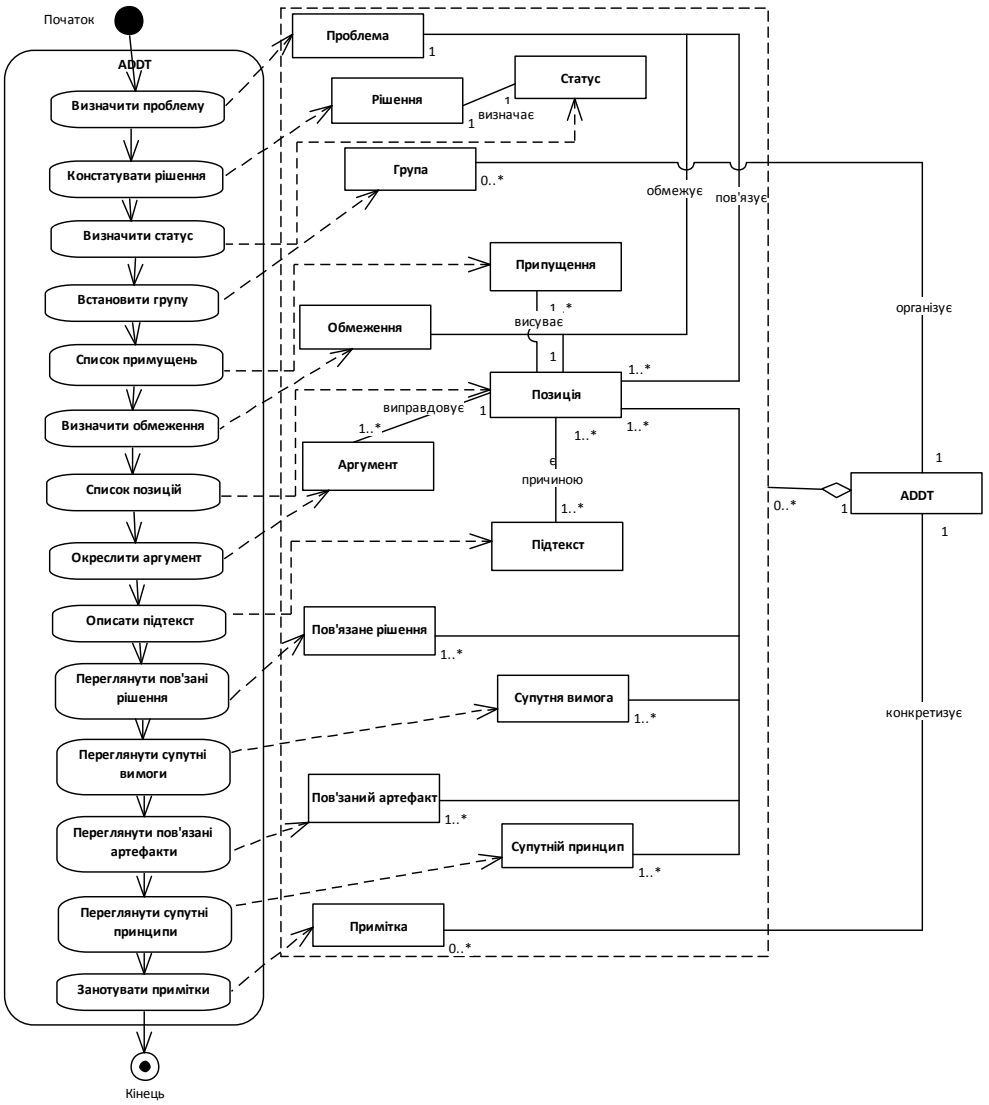

Рис. 11 – Модель прийняття рішень ADDT

У ADDT немає конкретних методів обґрунтування та обговорення для управління процесом обґрунтування проекту.

ADDT не надає сам процес міркування над проектом, а також не надає повного керівництва про те, як слід аргументувати ту чи іншу позицію. На відміну від IBIS та QOC, ADDT надає повний список щодо збору знань при обґрунтуванні проекту. Ця інформа-

ція часто залишається недоступною або частково доступною при використанні інших підходів. Використовуючи ADDT, архітектори можуть чітко відслідковувати, які рішення були прийняті, чому вони були зроблені, і будь-яку додаткову інформацію, яка має відношення до рішення. Головною перевагою ADDT є всебічність і повнота проектування, яка фіксується за допомогою шаблону. Однак ADDT не пропонує методів обґрунтування чи аргументації, що супроводжує шаблон. Іншим недоліком можна вважати ресурсомістке завдання заповнювання всього шаблону, оскільки не всі рішення повинні бути повністю задокументовані через їх простоту.

Архітектура обґрунтування та зв'язування елементів (AREL). AREL має на меті допомогти архітекторам створювати та документувати архітектурний дизайн з акцентом на архітектурні рішення та обґрунтування проекту [5]. AREL охоплює три типи знань архітектури: проектні питання, проектні рішення та результати проектування. Ці об'єкти знань представлені стандартними об'єктами уніфікованого моделювання (UML). Проблема проектування – це матеріали, які впливають на прийняття дизайнерських/проектних рішень. Ця сутність інкапсулює такі поняття, як функціональні вимоги (наприклад, сценарії та діаграма співпраці), нефункціональні вимоги (наприклад, всі атрибути якості) та контексти проекту. Він також фіксує інформацію про проектні рішення та обґрунтування проекту. Результати проектування включають в себе отримані рішення. Прикладами є класи, компоненти, інтерфейс та спосіб використання. AREL було реалізовано як плагін для інструменту моделювання під назвою Enterprise Architect (EA). Плагін AREL використовує стандартні позначення UML для представлення об'єктів проектування та рішень. Будь-який індивідуальний тип об'єкту архітектурного знання фіксується шляхом застосування заздалегідь визначеного тегового шаблону стереотипу. Інструменти імпорту доступні для отримання інформації із специфікації вимог на базі документу Word, а вбудовані в плагін інструменти дозволяють виконувати графічне відстеження та аналіз інформації. Концептуальна модель підходу представлена рис. 12.

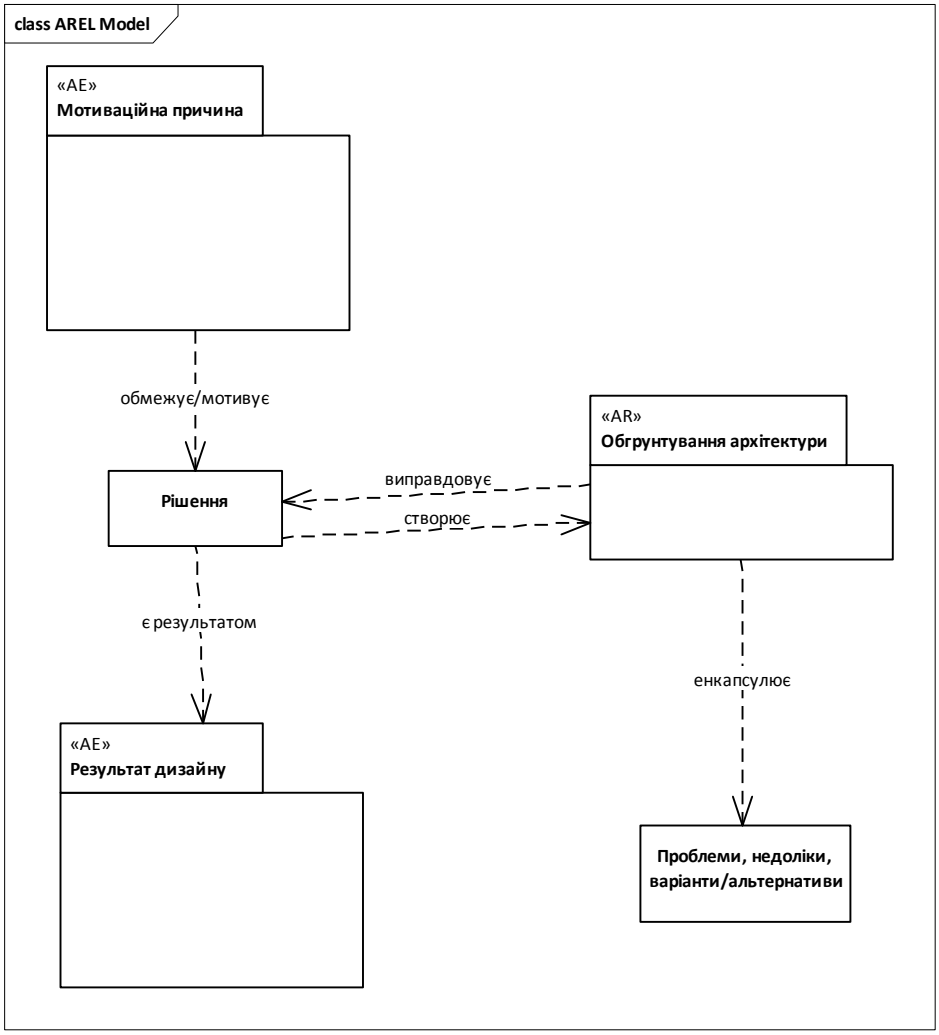

Рис. 12 – Концептуальна модель AREL [5]

Цей інструмент авторами перевірено на специфікації промислової електронної платіжної системи. Основною відмінністю між AREL та іншими системами є те, що AREL, на відміну від IBIS, QOC та DRL, – це як якісний, так і кількісний метод представлення обґрунтування проекту з вбудованим пошуко-

вим механізмом, який надає конструктору мову, що допомагає процесу обговорення та дозволяє зафіксувати текстові обґрунтування.

Кількісні методи обґрунтування. Проектування архітектури передбачає вибір оптимального рішення шляхом усунення альтернативних варіантів проектування нижчого рівня. Один із способів полегшити такий процес відбору полягає в тому, щоб певним чином кількісно оцінити варіанти проектування. Існує ряд методів, які використовують кількісний аналіз як засіб у процесі відбору. Метод аналізу недоліків архітектури (ATAM) – це спосіб використання компромісів для обґрунтування прийняття архітектурних рішень [22]. Метод ATAM дозволяє виконати наступні кроки:

– визначити атрибути якості, як цілі в рішенні;
– визначити відносну пріоритетність цілей;
– ідентифікувати підходи до архітектури (варіанти);
– створити дерево використання для переліку атрибутів якості, уточнення їх атрибутів та сценаріїв;
– провести аналіз альтернативних підходів шляхом вивчення точок чутливості (тобто, які якісні атрибути впливають), точки компромісу (тобто компроміс, необхідний для задоволення атрибутів якості), ризиків та не ризиків.

Метод аналізу витрат (CBAM) розглядає витрати та переваги, пов’язані з рішеннями. Метод розраховує очікувану віддачу стратегії щодо архітектури шляхом зважування переваг та витрат. Вигоди та витрати обчислюються шляхом розрахунку зважених вигод та зважених витрат усіх пов’язаних сценаріїв. Цей зважений метод забезпечує кількісне обґрунтування прийняття рішень [12].

Результати компаративного аналізу методів обґрунтування проектних рішень. Незважаючи на те, що методи раціонального проектування заклали основу для конструкторських міркувань, більшість з них все ще мають проблеми з точки зору зручності використання. Для їх порівняння використані наведені нижче критерії успішного впровадження концепції розробки:

– *Ефективне представлення обґрунтування дизайну* – на основі аргументації моделей фіксуються як міркування, так і структура аргументації обґрунтування дизайну. Структура аргументації є часовитратною і її важко простежити. Тому її слід спростити, не втрачаючи основну інформацію про обґрунтування дизайну.

– *Ефективна комунікація обґрунтування дизайну* – дизайнери хочуть знати проблеми, обґрунтування, потенційні альтернативи та елементи дизайну, на які впливає рішення. Отже, необхідність повторювати процес обговорення, як це пропонується аргументованими моделями, таким чином зменшена. Отже, проблеми в моделях, що базуються на аргументації, розташовуються вище над представленням їх обговорення та нище, ніж їх представлення взаємозв'язків з дизайнерськими артефактами.

– *Фокус на артефактах дизайну* – вимоги та специфікації дизайну об'єктів використовуються для підтримки системи оцінки та технічного обслуговування. Тому для обґрунтування дизайну необхідно пояснити артефакти дизайну, що містяться в цих специфікаціях.

– *Простежуваність та аналіз впливу* – основне використання концепції дизайну полягає в тому, щоб допомогти дизайнерам зрозуміти обґрунтування та проблеми дизайну з метою виконання подальшого технічного обслуговування. Одним з таких заходів є аналіз впливу змін. Методи обґрунтування дизайну повинні підтримувати аналіз ефектів пульсації (хвильовий вплив), використовуючи можливість простежувати обґрунтування конкретних проектних рішень з метою пояснення конструктивної залежності.

– *Обґрунтування комплексного дизайну* – обґрунтування дизайну складається з багатьох типів інформації. Міркування може ґрунтуватися на аргументації або на кількісному аналізі. Тому методи обґрунтування дизайну повинні бути всеосяжними та гнучкими, щоб відобразити ці різні типи міркувань.

– *Загальна підтримка інструментів* – розробники архітектури завжди стикаються з дуже обмеженим графіком проектування. Використання іншого інструменту для представлення конструкторського обґрунтування збільшує накладні витрати на документацію та відокремлює знання від дизайну. Обґрунтування дизайну слід зафіксувати як побічний продукт, за допомогою того самого інструмента, в якому розробляється сам дизайн.

– *Корисність підходу для розробки критичних IT-інфраструктур* – можливість, корисні засоби та елементи у підходах, які можна використати для проектування та обґрунтування дизайну критичних IT-інфраструктур.

Проаналізуємо корисність існуючих методів обґрунтування дизайну за критеріями, викладеними вище. Кожен метод оцінюється відповідно до того, чи відповідає він критеріям повністю (2 бали), частково (1 бал) чи не відповідає (0 балів) (табл. 1).

Таблиця 1 – Аналіз корисності існуючих методів обґрунтування дизайну

Метод / Критерій	gIBIS	PHI	REMAP	DRL	SeuTAT	QOC	V&B	ADDT	ARCHEL
Ефективне представлення обґрунтування дизайну	0	0	0	0	0	0	2	2	2
Ефективна комунікація обґрунтування дизайну	0	0	0	0	0	0	2	2	2
Фокус на артефактах дизайну	1	1	2	1	2	0	2	1	2
Простежуваність та аналіз впливу	0	0	1	0	1	0	0	1	2
Обґрунтування комплексного дизайну	0	0	0	0	0	0	1	1	2
Загальна підтримка інструментів	0	0	0	0	0	0	2	2	2
Корисність підходу для розробки критичних IT-інфраструктур	0	0	0	0	0	0	0	1	1
Всього	**1**	**1**	**3**	**1**	**3**	**0**	**9**	**10**	**13**

Підхід QOC підходить для простого, аргументованого проектування, де рішення мають чіткі варіанти, а критерії та елементи можуть бути узагальнені. Він розроблений для надання архітекторам простого методу, що стимулює міркування. Однак, це не настільки однозначно у випадку складного дизайну. Проектування архітектури є досить складним завдання і вимагає зазвичай залучення багатьох варіантів з власними критеріями та оцінками. Прості конструктивні рішення можуть легко стати заплутаними і незрозумілими.

Як і IBIS, QOC, DRL призначена для того, щоб забезпечити архітекторам модель і словниковий запас, що стимулює обговорення проектування. Велика різниця методу, на відміну від IBIS і QOC, полягає в тому, що DRL призначений для асинхронного використання. Іншими словами, DRL слід використовувати після того, як сам процес проектування вже здійснений. Він використовується для побудови обґрунтування за допомогою аналізу історичних записів дизайнерських сеансів. Очевидний і його недолік при аналізі моделі, її складність швидко зростає при ускладненні дизайну.

Використовуючи задокументоване обґрунтування рішення, SeuRAT дозволяє супроводжуючим користувачам отримувати та перевіряти обґрунтування для проведення технічного обслуговування. Бург виявив, що SeuRAT забезпечує кращі результати серед протестованих людей, які не є експертами в Java, ніж результати їх відповідної контрольної групи для виявлення проблем та виконання завдань [23].

Мета ADDT і V&B полягає в наданні методів обґрунтування дизайну для практиків. Тому вони мають більш прагматичний підхід, ніж інші методи. З точки зору представлення та комунікації дизайну обґрунтування, їх структура менш формальна, ніж інші методи. Через це вони можуть бути реалізовані за допомогою різних типів програмних інструментів і легко можуть бути прийняті організацією. Проте обмежене графічне представлення в цих двох методах обмежує їх здатність надавати обґрунтоване обчислення і можливість відстеження. Інший підхід AREL не має вад двох вище наведених методів обґрунтування дизайну, але також не може бути використаний у вихідному стані для проектування та обґрунтування проектних рішень критичної IT-інфраструктури.

Обговорення результатів дослідження. У даній роботі представлений детальний компаративний аналіз підходів до обґрунтування проектних рішень, визначені їх недоліки. Як результат. Таким чином, зроблено перший крок на шляху до створення інструментарію документування та обґрунтування процесу прийняття проектних рішень щодо архітектури критичної IT-інфраструктури.

Для майбутніх досліджень, насамперед, планується розробка мета моделі нового підходу, що можна буде застосувати для проектування архітектури критичної IT-інфраструктури та програмного забезпечення для створення архітектури критичної IT-інфраструктури, яке б підтримувало запропоновану мета-модель підходу.

Крім того, буде розглянуто можливість застосування підходу для модернізації вже існуючих критичних IT-інфраструктур.

Нарешті, одним з головних завдань є вивчення *способів* зменшення зусиль при проектуванні з використанням запропонованого підходу.

Висновки. В результаті проведених досліджень:

1. Проаналізовано основні існуючі підходи до обґрунтування проектних рішень щодо архітектури підприємства, визначені їх недоліки та переваги.

2. Досліджено можливість їх подальшого застосування при проектуванні архітектури критичної IT-інфраструктури.

Список літератури:

1. *Kunz, W.* Issues as Elements of Information Systems [Text] / *W. Kunz, H. Rittel*. – Berkeley, 1970. doi: 10.1515/9783038210665.147
2. *Op't Land, M.* Enterprise architecture: creating value by informed governance [Text] / *M. Op't Land, E. Proper, M. Waage, J. Cloo, C. Steghuis*. – Springer, 2009. – 145 p. doi: 10.1007/978-3-540-85232-2
3. *Aier, S.* Virtual Decoupling for IT/Business Alignment – Conceptual Foundations, Architecture Design and Implementation Example [Text] / *S. Aier, R. Winter* // Business & Information Systems Engineering. – 2008. – Vol. 1, Issue 2. – P. 150–163. doi: 10.1007/s12599-008-0010-7
4. *Jansen, A.* Software Architecture as a Set of Architectural Design Decisions [Text] / *A. Jansen, J. Bosch* // 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05). – 2005. doi: 10.1109/wicsa.2005.61
5. *Tang, A.* A rationale-based architecture model for design traceability and reasoning [Text] / *A. Tang, Y. Jin, J. Han* // Journal of Systems and Software. – 2007. – Vol. 80, Issue 6. – P. 918–934. doi: 10.1016/j.jss.2006.08.040
6. *Plataniotis, G.* Capturing Decision Making Strategies in Enterprise Architecture – A Viewpoint [Text] / *G. Plataniotis, S. de Kinderen, H. A. Proper* // Lecture Notes in Business Information Processing. – 2013. – P. 339–353. doi: 10.1007/978-3-642-38484-4_24
7. *Coggins, C.* The methodology for business transformation v1.5: A practical approach to segment architecture [Text] / *C. Coggins, J. Speigel* // Journal of Enterprise Architecture. – 2007.
8. *Toulmin, S.* The Uses of Argument [Text] / S. Toulmin. – 2nd ed. – Cambridge University Press, 2003. – 247 p. 10.1017/cbo9780511840005
9. *Lee, J.* Extending the Potts and Bruns model for recording design rationale [Text] / *J. Lee* // 13th International Conference on Software Engineering. – 1991. doi: 10.1109/icse.1991.130629
10. *Tyree, J.* Architecture Decisions: Demystifying Architecture [Text] / *J. Tyree, A. Akerman* // IEEE Software. – 2005. – Vol. 22, Issue 2. – P. 19–27. doi: 10.1109/ms.2005.27
11. *Clements, P.* Documenting Software Architectures: Views and Beyond [Text] / *P. Clements* et. al. – 1st ed. – Addison Wesley, 2002.
12. *Asundi, J.* Using Economic Considerations to Choose Amongst Architecture Design Alternatives [Text] / *J. Asundi, R. Kazman, M. Klein* // Defense technical information center. – 2001. doi: 10.21236/ada399151
13. *Fischer, G.* Making Argumentation Serve Design [Text] / *G. Fischer, A. Lemke, R. McCall* // Design Rationale: Concepts, Techniques and Use. – Lawrence Erlbaum Associates, 1996. – P. 267–294.
14. *McCall, R. J.* PHI: a conceptual foundation for design hypermedia [Text] / *R. J. McCall* // Design Studies. – 1991. – Vol. 12, Issue 1. – P. 30–41. doi: 10.1016/0142-694x(91)90006-i
15. *Riddle, W.* Software Technology Maturation [Text] / *W. Riddle* // Proceedings of the 8th International Conference on Software Engeneering (ICSE). – 1985. – P. 189–200.
16. *Conklin, J.* gIBIS: a hypertext tool for exploratory policy discussion [Text] / *J. Conklin, M. L. Begeman* // Proceedings of the 1988 ACM conference on Computer-supported cooperative work – CSCW '88. – 1988. doi: 10.1145/62266.62278

17. *Ramesh, B.* Supporting systems development by capturing deliberations during requirements engineering [Text] / *B. Ramesh, V. Dhar* // IEEE Transactions on Software Engineering. – 1992. – Vol. 18, Issue 6. – P. 498–510. doi: 10.1109/32.142872

18. *Lee, J.* What's in Design Rationale [Text] / *J. Lee, K. Lai* // Design Rationale: Concepts,Techniques and Use. – 1996. – Vol. 2. – P. 21–52.

19. *Lee, J.* SIBYL: a tool for managing group design rationale [Text] / *J. Lee* // Proceedings of the 1990 ACM conference on Computer-supported cooperative work – CSCW '90. – 1990. doi: 10.1145/99332.99344

20. *Maclean, A.* Questions, Options and Criteria [Text] / *A. Maclean, R. Young, V. Bellotti, T. Moran* // Design Rationale: Concepts, Techniques and Use. – Lawrence Erlbaum Associates, 1996. – P. 53–106.

21. *Dutoit, A. H.* Rationale-Based Use Case Specification [Text] / *A. H. Dutoit, B. Paech* // Requirements Engineering. – 2002. – Vol. 7, Issue 1. – P. 3–19. doi: 10.1007/s007660200001

22. *Bass, L.* Software Architecture in Practice [Text] / *L. Bass, P. Clements, R. Kazman*. – Boston: Addison Wesley, 2003.

23. *Burge, J.* Software Engineering Using design RATionale [Text]: PhD dissertation / *J. Burge*. – Worcester Politechnic Institute, 2005.

Bibliography (transliterated):

1. Kunz, W., Rittel, H. (1970). Issues as Elements of Information Systems. Berkeley. doi: 10.1515/9783038210665.147

2. Op't Land, M., Proper, E., Waage, M., Cloo, J., Steghuis, C. (2009). Enterprise architecture: creating value by informed governance. Springer, 145. doi: 10.1007/978-3-540-85232-2

3. Aier, S., Winter, R. (2008). Virtual Decoupling for IT/Business Alignment – Conceptual Foundations, Architecture Design and Implementation Example. Business & Information Systems Engineering, 1 (2), 150–163. doi: 10.1007/s12599-008-0010-7

4. Jansen, A., Bosch, J. (2005). Software Architecture as a Set of Architectural Design Decisions. 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05). doi: 10.1109/wicsa.2005.61

5. Tang, A., Jin, Y., Han, J. (2007). A rationale-based architecture model for design traceability and reasoning. Journal of Systems and Software, 80 (6), 918–934. doi: 10.1016/j.jss.2006.08.040

6. Plataniotis, G., de Kinderen, S., Proper, H. A. (2013). Capturing Decision Making Strategies in Enterprise Architecture – A Viewpoint. Lecture Notes in Business Information Processing, 339–353. doi: 10.1007/978-3-642-38484-4_24

7. Coggins, C., Speigel, J. (2007). The methodology for business transformation v1.5: A practical approach to segment architecture. Journal of Enterprise Architecture.

8. Toulmin, S. (2003). The Uses of Argument. Cambridge University Press, 247. 10.1017/cbo9780511840005

9. Lee, J. (1991). Extending the Potts and Bruns model for recording design rationale. 13th International Conference on Software Engineering. doi: 10.1109/icse.1991.130629

10. Tyree, J., Akerman, A. (2005). Architecture Decisions: Demystifying Architecture. IEEE Software, 22 (2), 19–27. doi: 10.1109/ms.2005.27

11. Clements, P. et. al. (2002). Documenting Software Architectures: Views and Beyond. Addison Wesley.

12. Asundi, J., Kazman, R., Klein, M. (2010). Using Economic Considerations to Choose Amongst Architecture Design Alternatives. Defense technical information center. doi: 10.21236/ada399151

13. Fischer, G., Lemke, A., McCall, R. (1996). Making Argumentation Serve Design. Design Rationale: Concepts, Techniques and Use. Lawrence Erlbaum Associates, 267–294.

14. McCall, R. J. (1991). PHI: a conceptual foundation for design hypermedia. Design Studies, 12 (1), 30–41. doi: 10.1016/0142-694x(91)90006-i

15. Riddle, W. (1985). Software Technology Maturation. Proceedings of the 8th International Conference on Software Engeneering (ICSE), 189–200.

16. Conklin, J., Begeman, M. L. (1988). gIBIS: a hypertext tool for exploratory policy discussion. Proceedings of the 1988 ACM Conference on Computer-Supported Cooperative Work – CSCW '88. doi: 10.1145/62266.62278

17. Ramesh, B., Dhar, V. (1992). Supporting systems development by capturing deliberations during requirements engineering. IEEE Transactions on Software Engineering, 18 (6), 498–510. doi: 10.1109/32.142872

18. Lee, J., Lai, K. (1996). What's in Design Rationale. Design Rationale: Concepts,Techniques and Use, 2, 21–52.

19. Lee, J. (1990). SIBYL: a tool for managing group design rationale. Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work – CSCW '90. doi: 10.1145/99332.99344

20. Maclean, A., Young, R., Bellotti, V., Moran, T. (1996). Questions, Options and Criteria. Design Rationale: Concepts, Techniques and Use. Lawrence Erlbaum Associates, 53–106.

21. Dutoit, A. H., Paech, B. (2002). Rationale-Based Use Case Specification. Requirements Engineering, 7 (1), 3–19. doi: 10.1007/s007660200001

22. Bass, L., Clements, P., Kazman, R. (2003). Software Architecture in Practice. Boston: Addison Wesley.

23. Burge, J. (2005). Software Engineering Using design RATionale. Worcester Politechnic Institute.

Надійшла (received) 01.11.2017

Бібліографічні описи / Библиографические описания / Bibliographic descriptions

Порівняльний аналіз методів представлення та обґрунтування архітектури критичної IT-інфраструктури/ Дорогий Я. Ю. // Вісник НТУ «ХПІ». Серія: Механіко-технологічні системи та комплекси. – Харків : НТУ «ХПІ», 2017. – No 33(1255). – С.42–54. – Бібліогр.:23 назв. – ISSN 2079-5459.

Сравнительный анализ методов представления и обоснования архитектуры критической ИТ-инфраструктуры/ Дорогий Я. Ю. // Вісник НТУ «ХПІ». Серія: Механіко-технологічні системи та комплекси. – Харків : НТУ «ХПІ», 2017. – No 33(1255). – С. 42–54.– Бібліогр.: 23 назв. – ISSN 2079-5459.

A comparative analysis of the methods for presenting and substantiating the architecture of a critical IT infrastructure/ Dorogyy Ya. //Bulletin of NTU "KhPI". Series: Mechanical-technological systems and complexes. – Kharkov: NTU "KhPI", 2017. – № 33 (1255).– P.42–54. – Bibliogr.:23. – ISSN 2079-5459

Відомості про авторів / Сведения об авторах / About the Authors

Дорогий Ярослав Юрійович – кандидат технічних наук, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», доцент кафедри "Автоматика і управління в технічних системах"; Київ, пр. Перемоги, 37; e-mail: cisco.rna@gmail.com.

Дорогий Ярослав Юрьевич – кандидат технических наук, Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», доцент кафедры "Автоматика и управление в технических системах"; Киев, пр. Победы, 37; e-mail: cisco.rna@gmail.com.

Dorogyy Yaroslaw – candidate of technical sciences, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"; associate professor of the department «Automation and Control in Technical Systems»; av. Victory, 37, Kyiv