ECONOMICS COLLEGE IN STALOWA WOLA

ENGINEERING SCIENCES: DEVELOPMENT PROSPECTS IN COUNTRIES OF EUROPE AT THE BEGINNING OF THE THIRD MILLENNIUM

Collective monograph

Volume 1

Stalowa Wola, Poland
2018

Recommended for publication
by the Academic Council of Economics College in Stalowa Wola

Responsible for release: dr Małgorzata Korecka, rector
(Economics College in Stalowa Wola)

Engineering sciences: development prospects in countries of Europe at the beginning of the third millennium: Collective monograph. Volume 1. Riga : Izdevnieciba "Baltija Publishing", 2018. 464 p.

ISBN 978-9934-571-63-3

CONTENTS

МЕТОДИ ПРЕДСТАВЛЕННЯ ТА ОБГРУНТУВАННЯ АРХІТЕКТУРИ КРИТИЧНОЇ ІТ-ІНФРАСТРУКТУРИ

Дорогий Я. Ю., Цуркан В. В.

ВСТУП

Мови представлення архітектури підприємств (АП) розглядаються як інструмент цілісного описування підприємства[1], що пов'язують усі 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, оскільки досвід у галузі архітектури програмного 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

[1] Kunz W. Issues as Elements of Information Systems. Berkeley, 1970.

[2] Enterprise architecture: creating value by informed governance / M. Op't Land and etc. Springer, 2008.

[3] Aier S., Winter R. Virtual decoupling for it/business alignment-conceptual foundations, architecture design and implementation example. *Business & Information Systems Engineering*. 2009. Vol. 1. P. 150–163.

[4] Jansen A., Bosch J. Software architecture as a set of architectural design decisions. *Software Architecture*. 2005. P. 109–120.

уваги обмеження, що випливають із 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 під час 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). Дані проблеми відрізняються від тих, які виникають у корпоративній архітектурі[7] і, особливо, в архітектурі критичних інфраструктур. Архітектори підприємств займаються повним стеком проблем бізнес-ІТ. 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 у визначенні такого підходу, який би використав переваги цих методів, зневілював їх негативні сторони та надав би можливість досліднику використовувати його під час проектування критичної ІТ-інфраструктури.

---

[5] Tang A., Jin Y., Han J. A rationale-based architecture model for design traceability and reasoning. *Journal of Systems and Software*. 2007. Vol. 80. P. 918–934.

[6] Plataniotis G., de Kinderen S., Proper H.A. Capturing decision making strategies in enterprise architecture – a viewpoint. *Enterprise, Business-Process and Information Systems Modeling*. 2013. Vol. 147. P. 339–353.

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

Мета дослідження

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

Задачею дослідження є визначення недоліків основних підходів до раціоналізації архітектури, їх порівняння з точки зору використання як підходу до раціоналізації архітектури критичної IT-інфраструктури, а також створення нового підходу до побудови критичної IT-інфраструктури, який би врахував виявлені в ході дослідження недоліки існуючих підходів.

Для досягнення поставленої мети були поставлені такі завдання:

1. Дослідження існуючих підходів та визначення їх недоліків та переваг.
2. Дослідження можливості їх застосування для обґрунтування проектних рішень щодо критичної IT-інфраструктури.
3. Аналіз можливих стратегій прийняття рішень та чинників, які на них впливають, із метою подальшого їх використання в пропонованій моделі.
4. Розроблення мета-моделі обґрунтування проектних рішень під час проектування критичних IT-інфраструктур, яка, на відміну від існуючих, дозволяє враховувати специфіку, притаманну саме критичним інфраструктурам.

Методи обґрунтування проектних рішень архітектури

Прийняття рішення щодо вибору дизайну системи включає такі 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 із 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) 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 у 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].

---

[8] Toulmin S. The Uses of Argument, 2nd ed. Cambridge University Press, 2003.

[9] Extending the Potts and Bruns Model for Recording Design Rationale. *13th International Conference on Software Engineering*. 1991. P. 114–125.

[10] Tyree J., Akerman A. Architecture decisions: Demystifying architecture. *Software, IEEE*. 2005. Vol. 22. P. 19–27.

[11] Documenting Software Architectures: Views and Beyond, 1st ed. / *P. Clements and etc.* Addison Wesley, 2002.

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 в 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.

[12] Asundi J. Using Economic Considerations to Choose Amongst Architecture Design Alternatives. 2001.

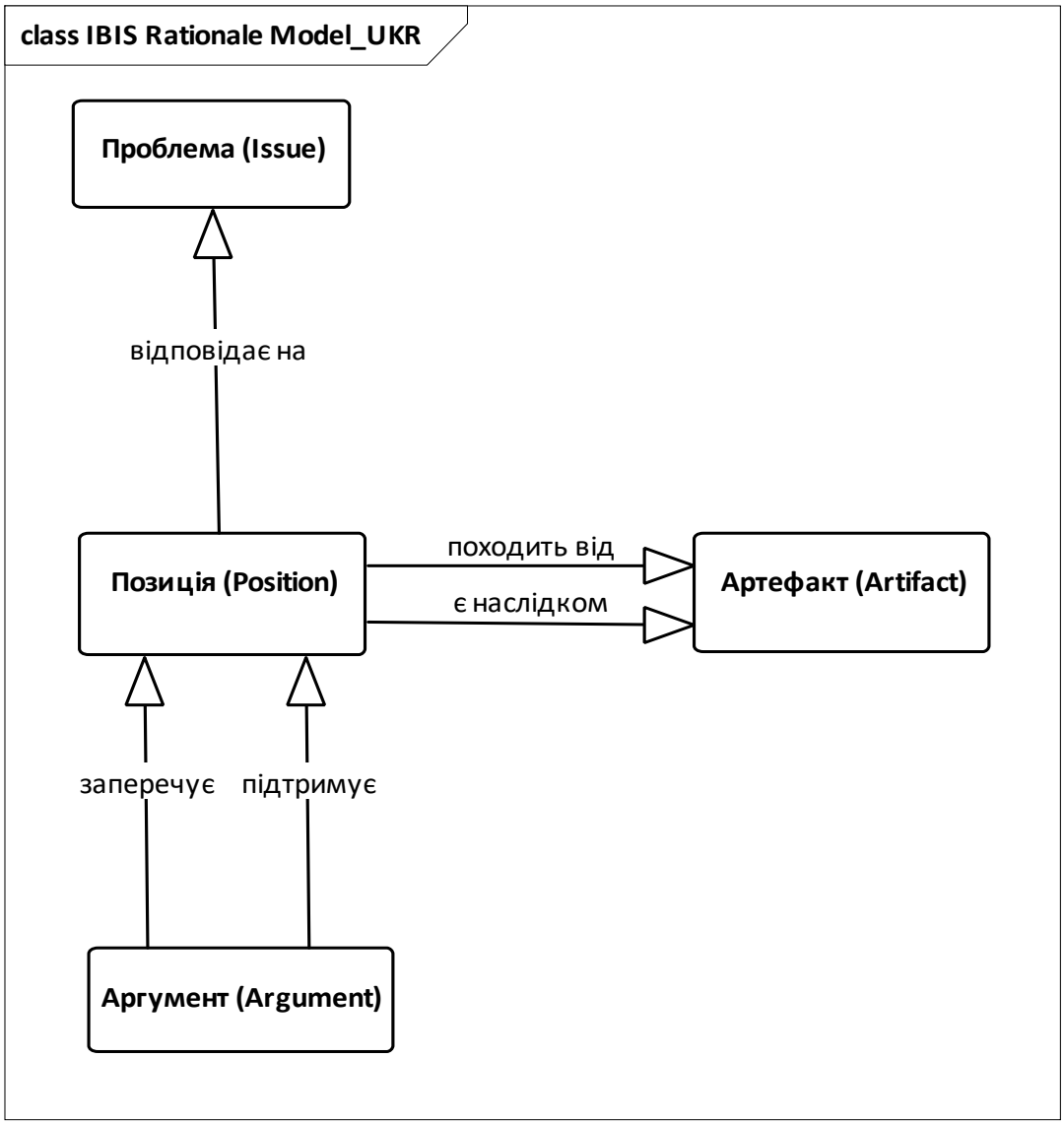

Рис. 1. Модель обґрунтування дизайну IBIS

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

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

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

– плюси і мінуси альтернативних позицій не обговорюються.

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

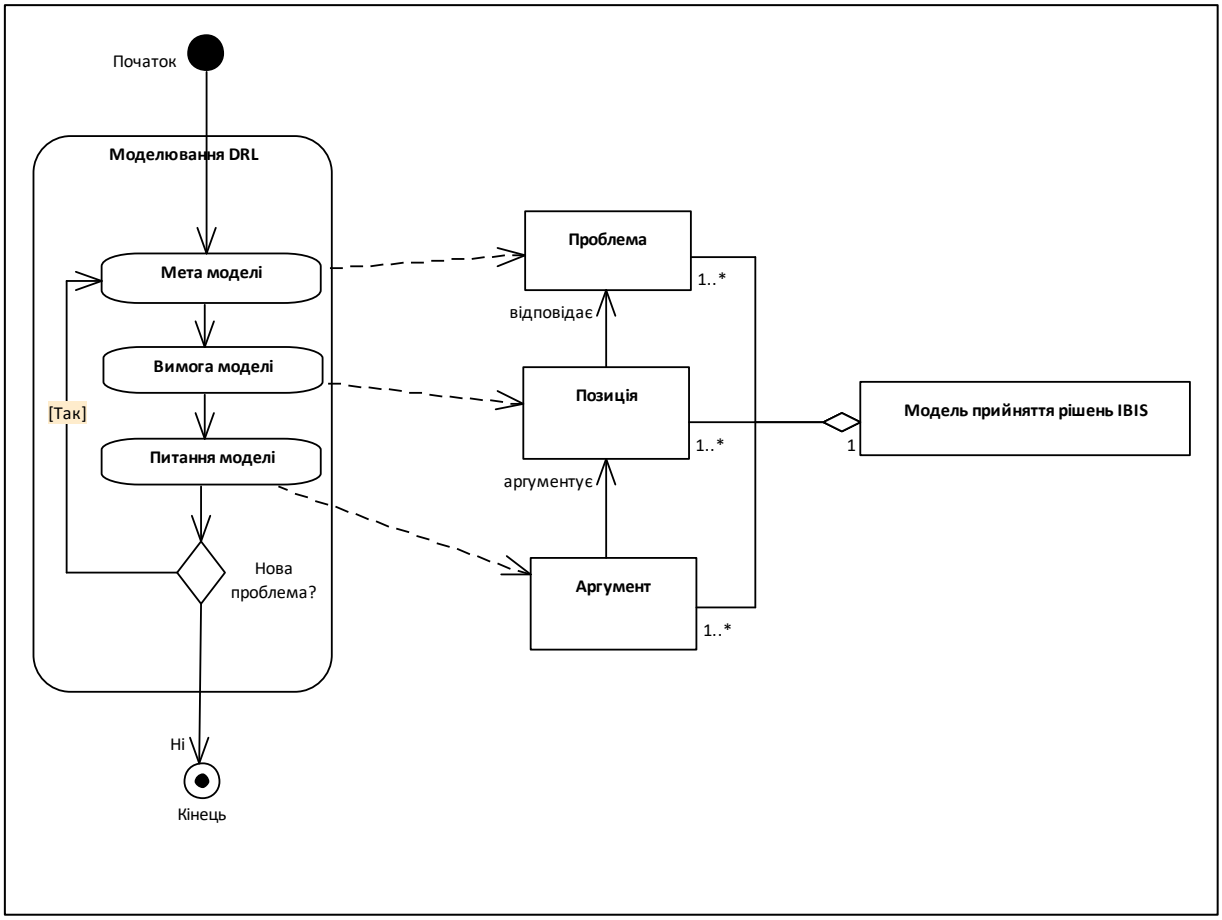

Рис. 2. Модель рішення IBIS

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 обґрунтування 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. У PHI враховується кожне проектне питання незалежно від того, обговорюється воно чи ні.

[14] PHI: A conceptual foundation for design hypermedia. *Design Studies*. 1991. Vol. 12. № 1. P. 30–41.

PHI спрощує взаємовідносини в IBIS за допомогою спеціального типу відносин. Проблема А слугує проблемі В, якщо і тільки якщо вирішення А впливає на вирішення В. Таким чином, домінуючим типом відносин у 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].

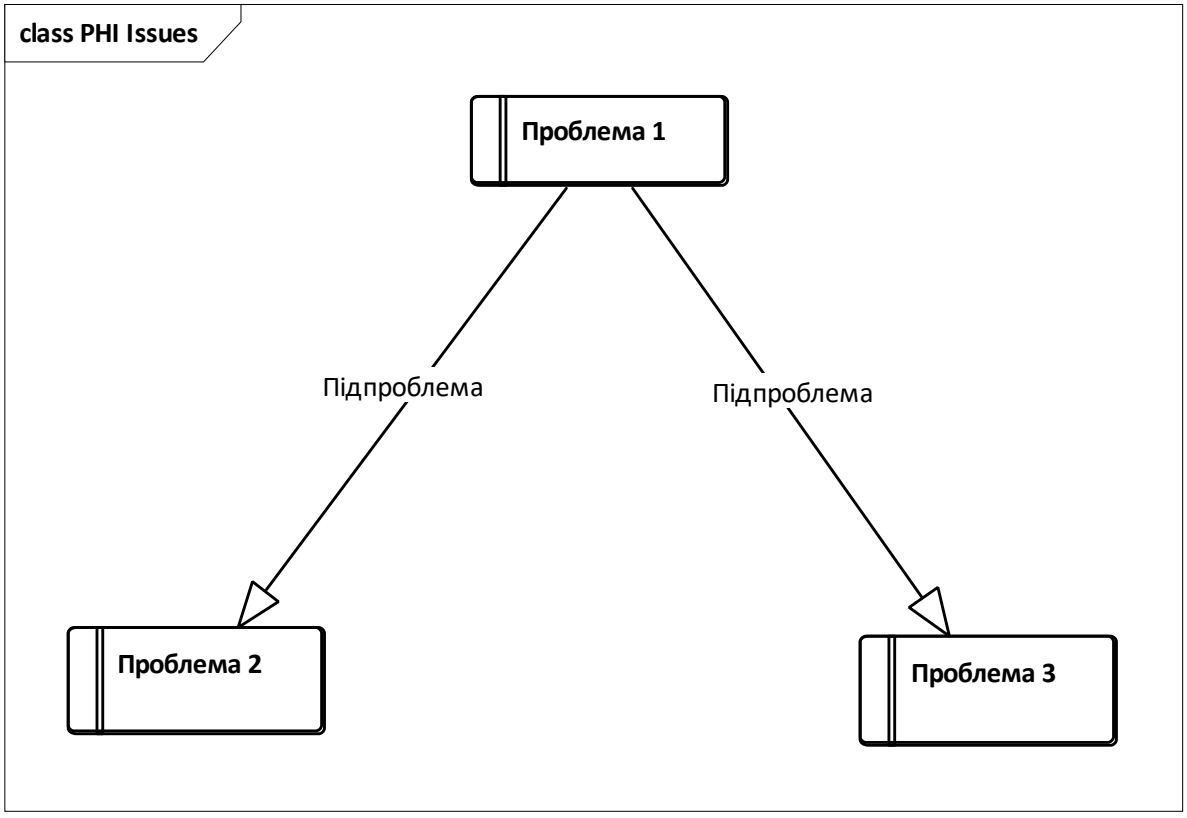

Рис. 3. Проблеми в PHI

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

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, такі як проблема, позиція та аргумент із вимогами, обмеженнями та об'єктами проектування. Це розширення забезпечує зв'язок між вимогами та об'єктами дизайну для його обґрунтування (рис. 4).

[16] Conklin J., Begeman M. IBIS: A hypertext tool for exploratory policy discussion. *Proceedings ACM Conference on Computer-Supported Cooperative Work*. 1988.

[17] Ramesh B., Dhar V. Supporting Systems Development by Capturing Deliberations During Requirements Engineering. *IEEE Transactions on Software Engineering*. 1992. Vol. 18. № 6. P. 498–510.

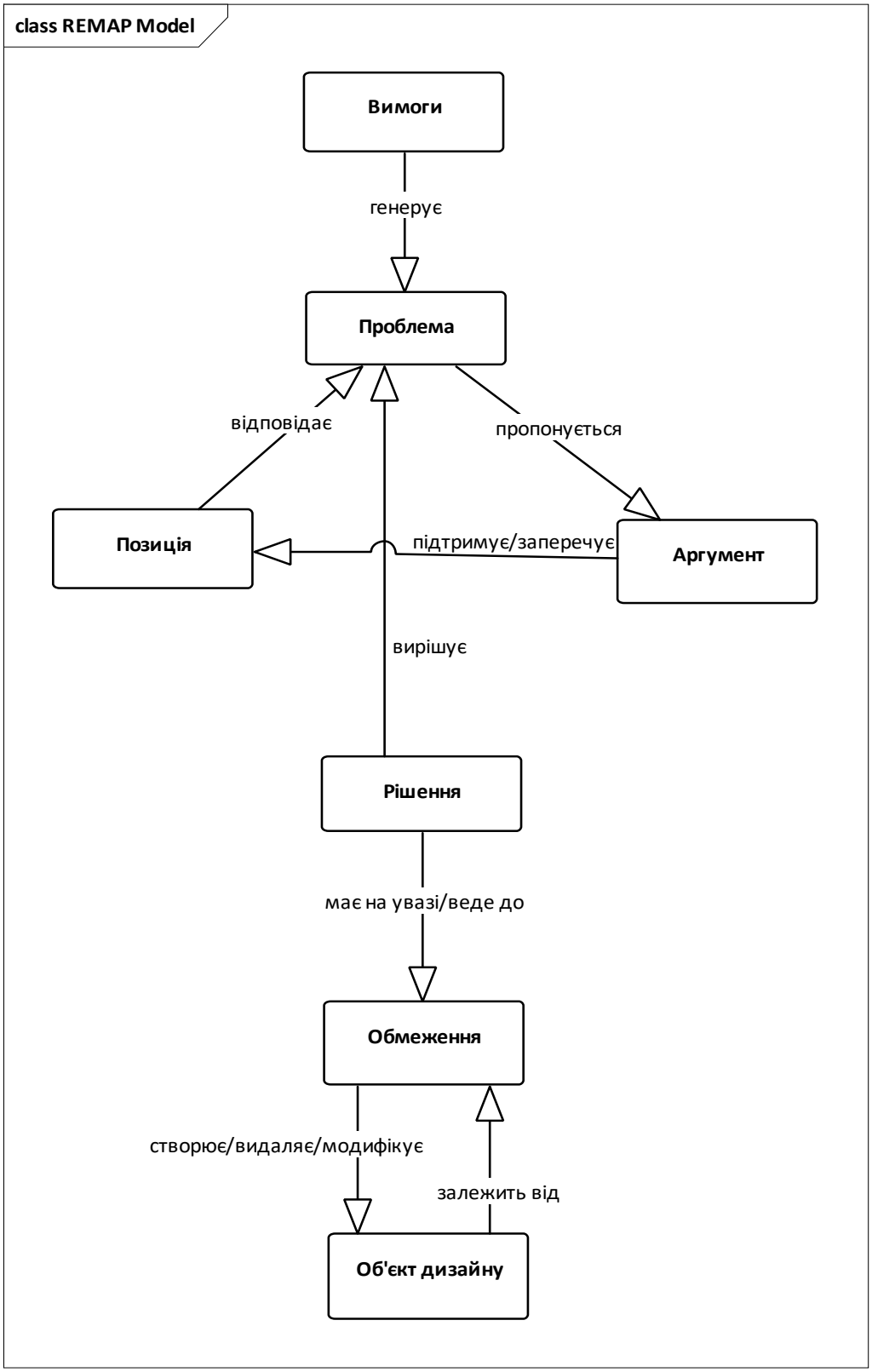

Рис. 4. Модель REMAP

Дана модифікація дозволяє зафіксувати історію проектних рішень у циклі розробки в якості знання процесу. Проблеми піднімаються та вирішуються на етапі проробки вимог під час створення 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 на 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 рисунку 5.

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

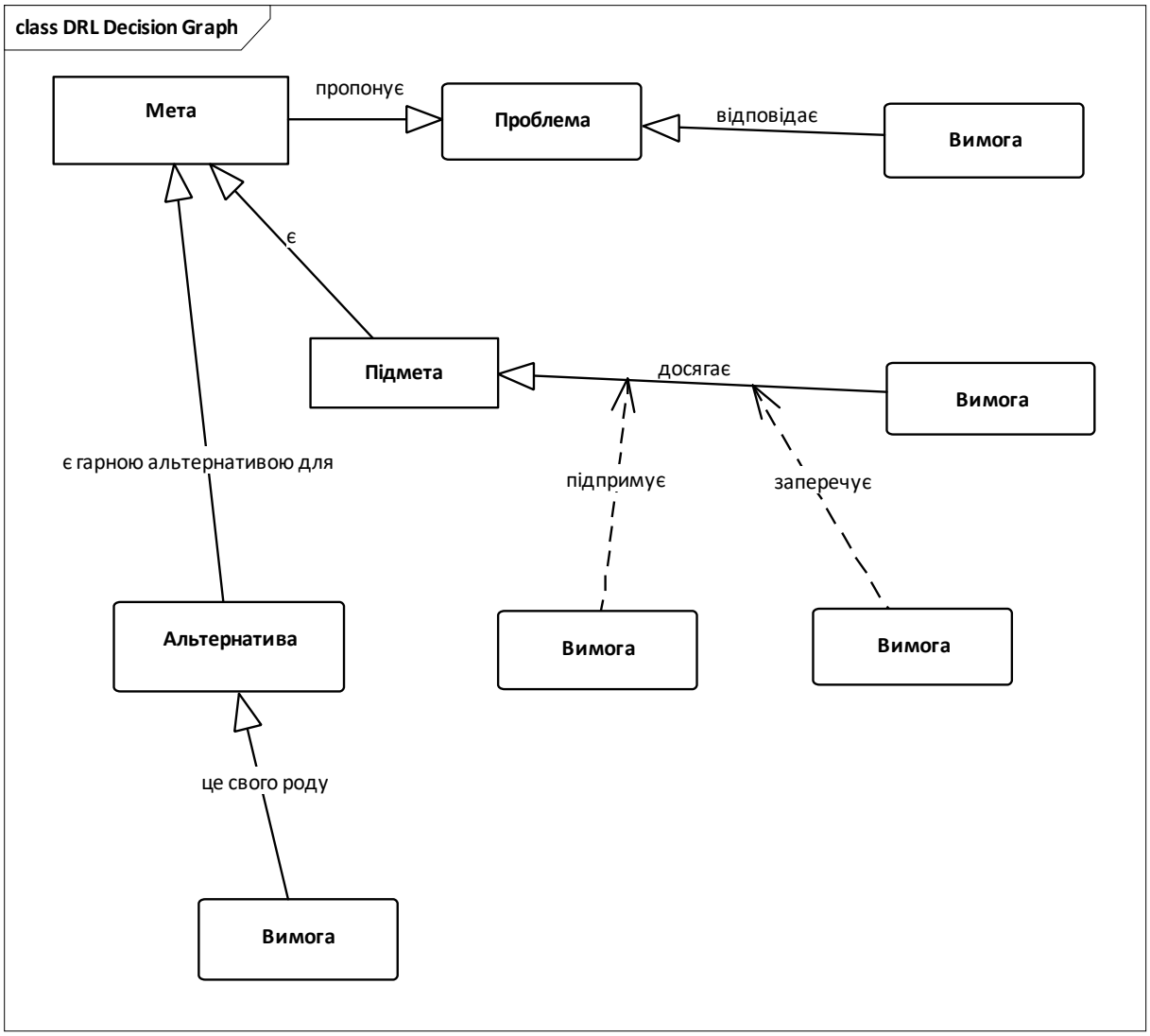

Рис. 5. Структура графу прийняття рішень DRL

Мета – це критерії, які необхідно задовольнити. Альтернатива являє собою варіант, який розглядається. Його зв'язок із метою полягає у визначенні, чи це добра 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.

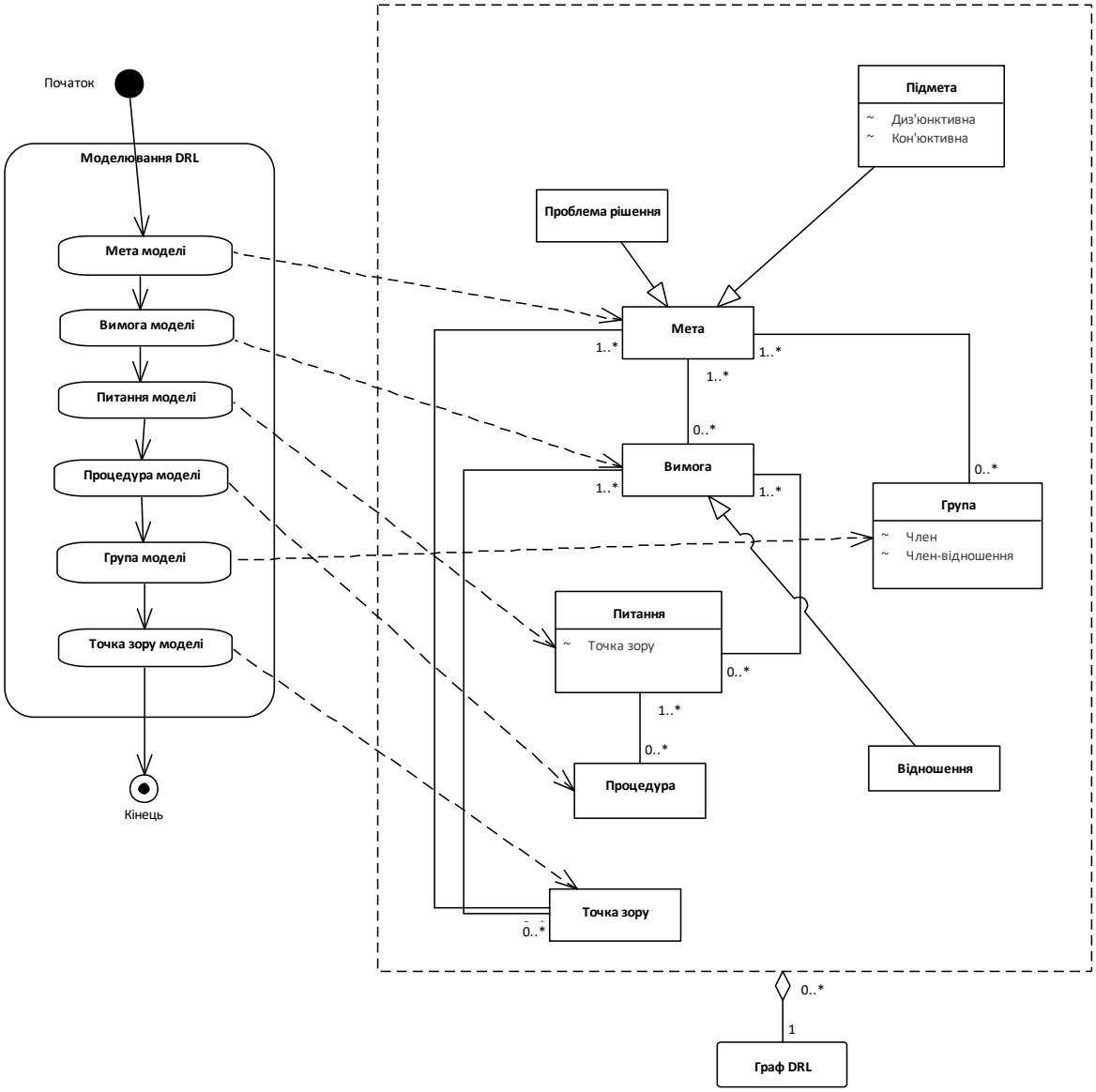

Рис 6. Модель прийняття рішення DRL

Аргументація проектування в DRL побудована з наведених вище елементів, які 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

[19] Lee J. SIBYL: A Tool for Managing Group Decision Rationale. *Proceedings of the Conference on on Computer-Supported Coopoerative Work*. 1990. P. 77–92.

підтримки використання обґрунтування проектування під час обслуговування програмного забезпечення, асоціювання обґрунтування з кодом та здійснення ряду висновків за обґрунтуванням із 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 із 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.

[20] Maclean A., Young R., Bellotti V., Moran T. Questions, Options and Criteria *Design Rationale: Concepts, Techniques and Use*. Ch. 3. Lawrence Erlbaum Associates, 1996. P. 53–106.

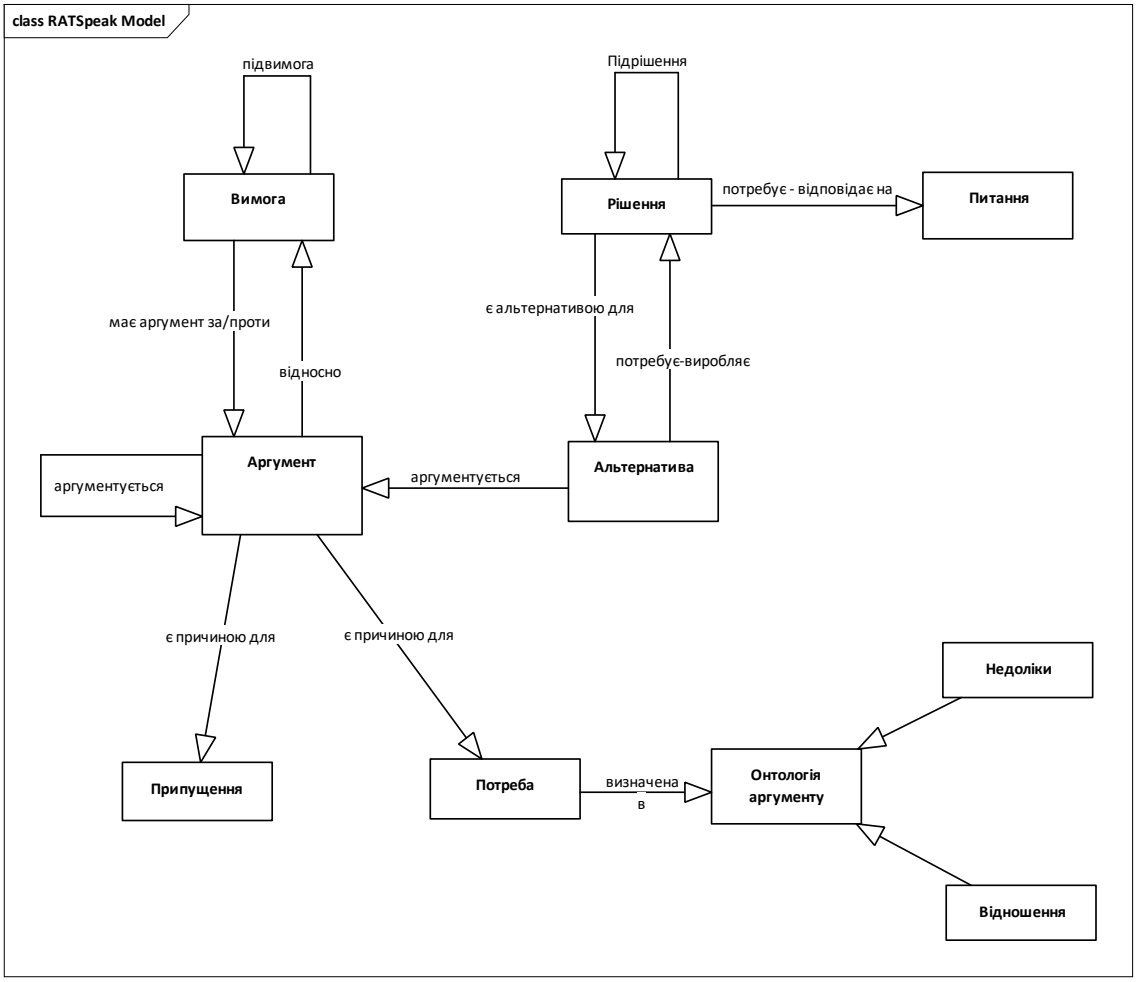

Рис. 7. Модель обґрунтування рішення RATSpeak

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 під час розробки 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.

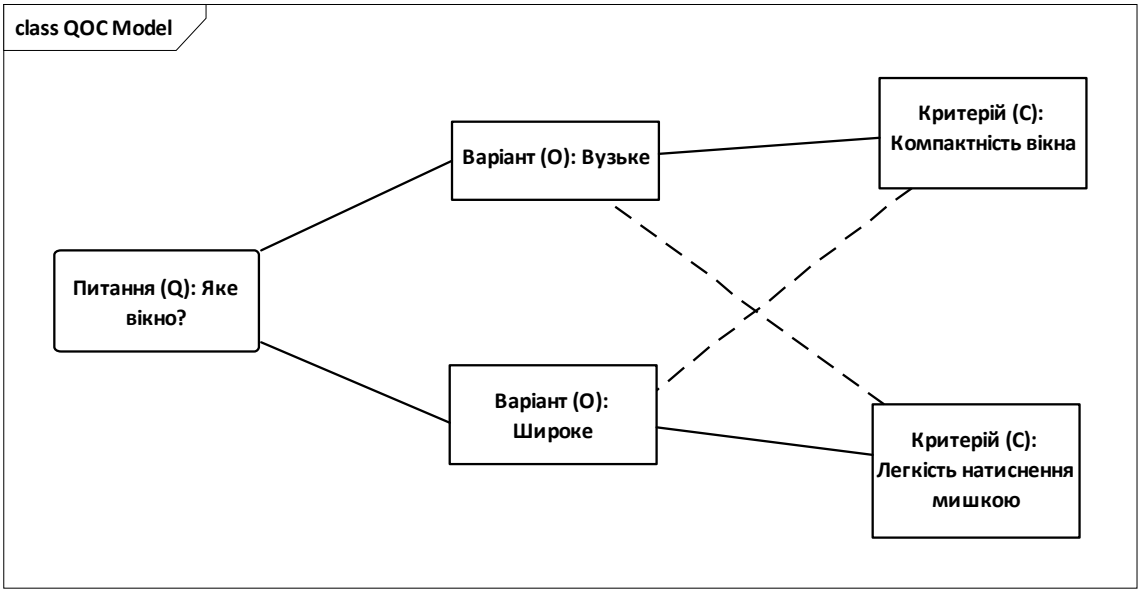

Рис. 8. Приклад рішення QOC: суцільна лінія – позитивна оцінка, пунктирна лінія – негативна оцінка

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.

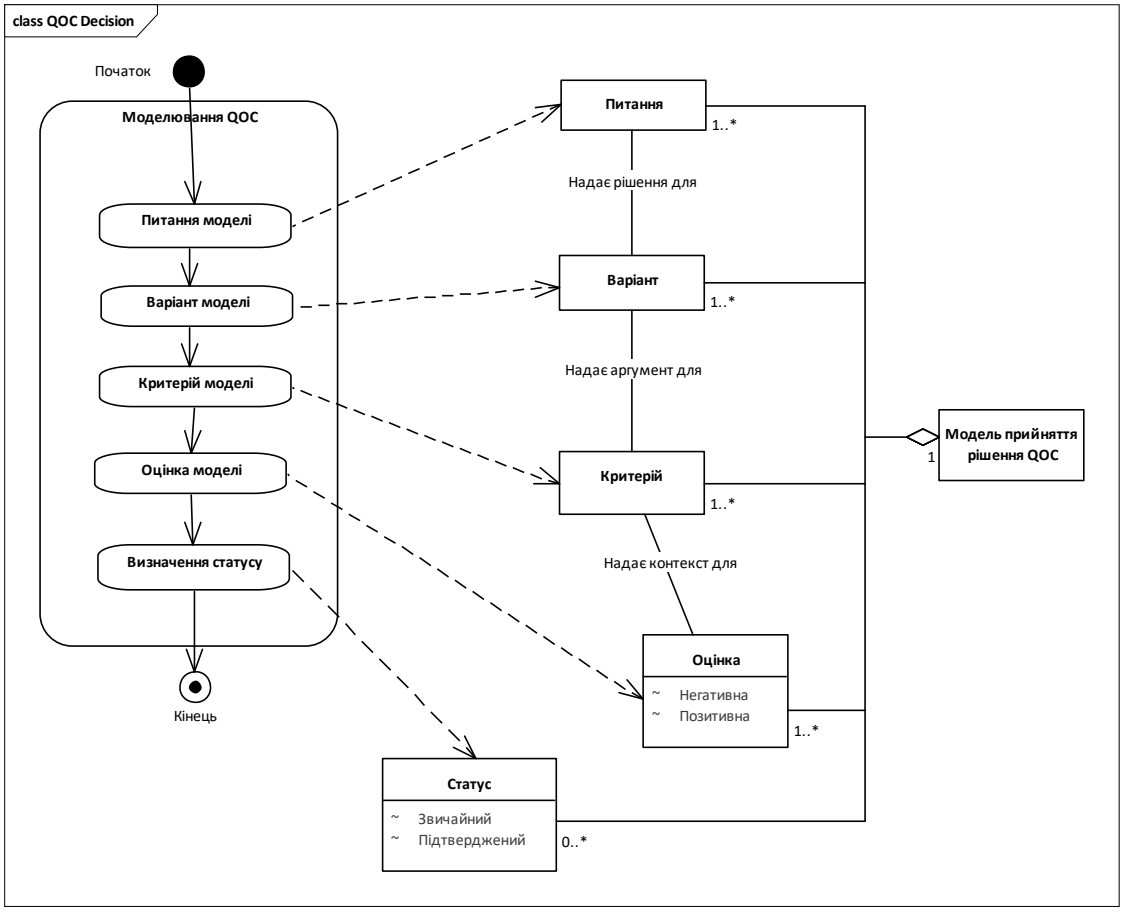

Рис. 9. Модель прийняття рішення QOC

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.

[21] Dutoit A.H., Paech B. Rationale-Based Use Case Specification. *Requirements Engineering*. 2002. Vol. 7. No. 1. P. 3–19.

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 підготовчої 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 така 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.

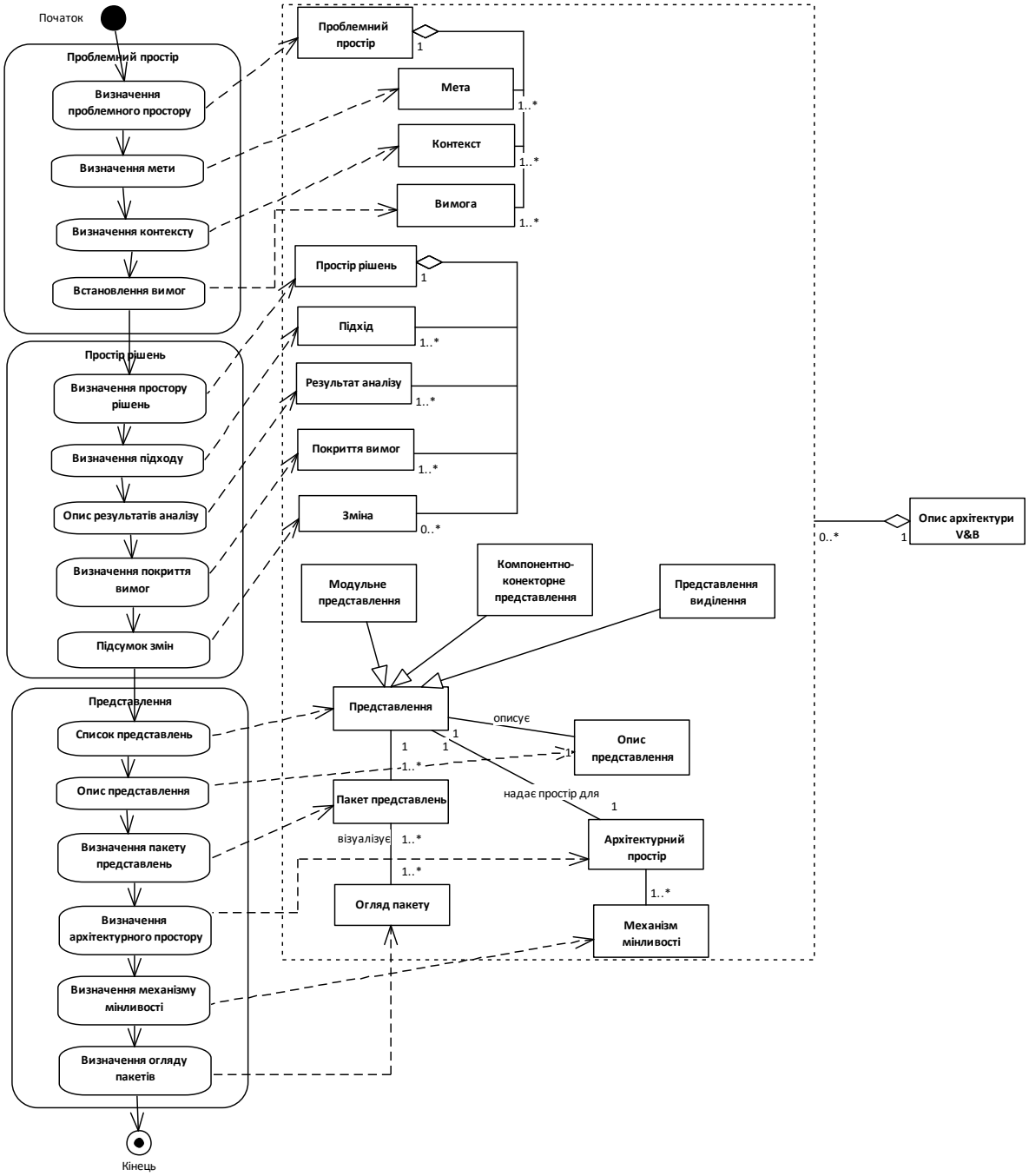

Рис. 10. Модель прийняття рішення V&B

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 у 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, які 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.

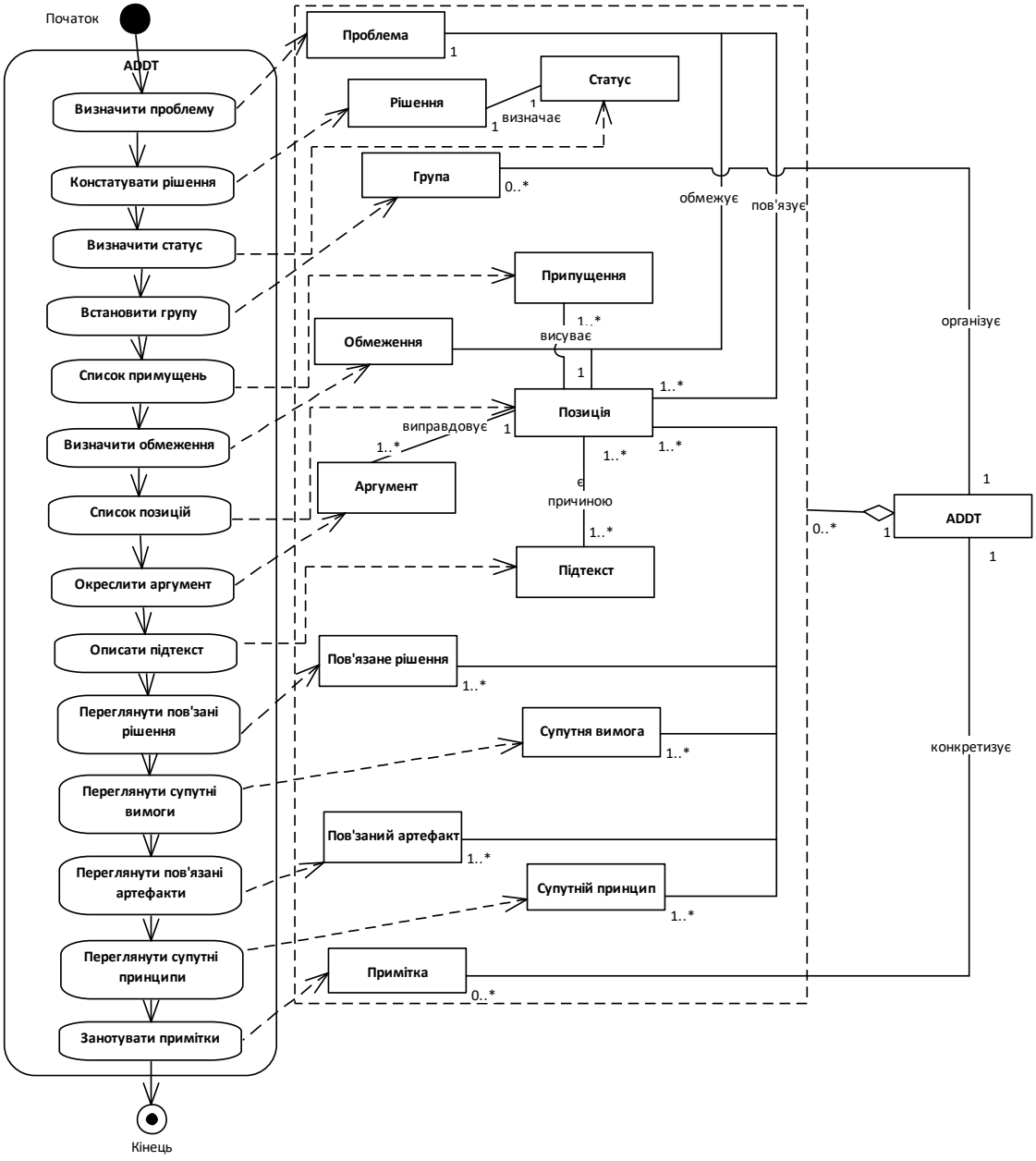

Рис. 11. Модель прийняття рішень ADDT

В 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 під час обґрунтування проекту. Ця інформація часто залишається недоступною або частково доступною у 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 у 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.

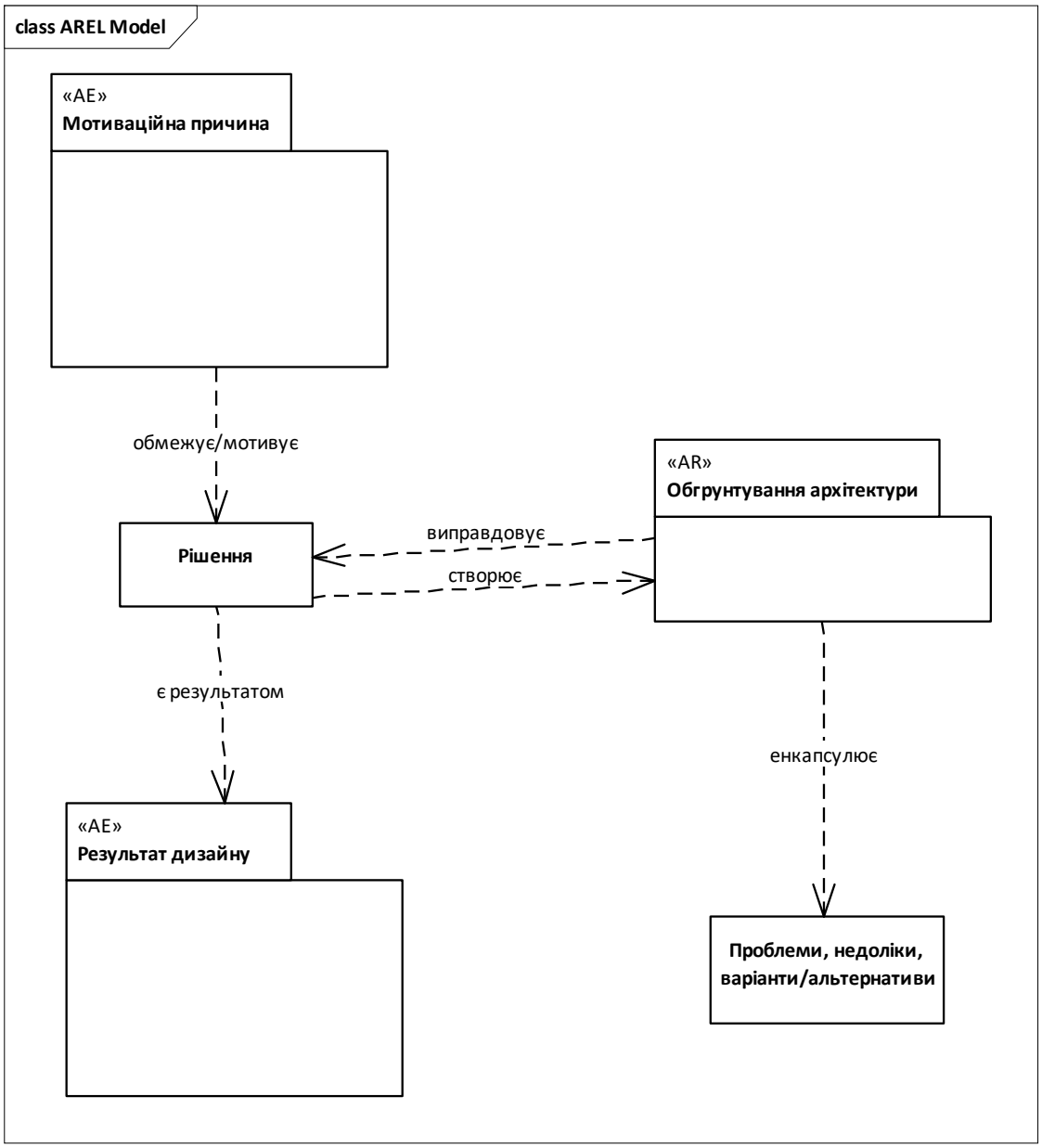

Рис. 12. Концептуальна модель AREL[5]

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 пошуковим 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 такі 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 неризиків.

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 із 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

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

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 нижче, ніж їх представлення взаємозв'язків із 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 із таких заходів є аналіз впливу змін. Методи обґрунтування дизайну повинні підтримувати аналіз ефектів пульсації (хвильовий вплив), використовуючи можливість простежувати обґрунтування конкретних проектних рішень із 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 в підходах, які можна

використати для проектування та обґрунтування дизайну критичних IT-інфраструктур.

Проаналізуємо корисність методів, що існують, 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) (таблиця 1).

Таблиця 1

Аналіз корисності методів обґрунтування дизайну

Метод / **Критерій**	**gIBIS**	**PHI**	**REMAP**	**DRL**	**SeuTAT**	**QOC**	**V&B**	**ADDT**	**ARL**
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
Усього	**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 завданням і вимагає зазвичай залучення багатьох варіантів із 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 під час аналізу моделі, чия складність швидко зростає у випадку ускладнення 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. Із 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 вищенаведених методів обґрунтування дизайну, але також не може бути використаний у вихідному стані для проектування та обґрунтування проектних рішень критичної IT-інфраструктури.

[23] Burge J. Software Engineering Using design RATionale, Ph.D. dissertation. 2005.

Мета-модель компенсаційно-декомпенсаційного підходу до проектування та обґрунтування проектних рішень щодо критичної ІТ-інфраструктури

Для побудови мета-моделі компенсаційно-декомпенсаційного підходу (КДК-підходу) до проектування та обґрунтування проектних рішень щодо критичної ІТ-інфраструктури з урахуванням недоліків і переваг методів, що розглянуті вище, потрібно також розглянути існуючі стратегії прийняття рішень та чинники, які впливають на вибір стратегії.

Стратегії прийняття рішень. Стратегії прийняття рішень зазвичай діляться на дві основні категорії: компенсаційні та некомпенсаційні.

У процесі прийняття компенсаційних рішень альтернативи оцінюються вичерпно, беручи до уваги всі критерії та їх компроміси. Критерії з високими пріоритетом компенсують критерії з більш нижчим. Нарешті, обирається альтернатива з найвищим балом. Компенсаційні стратегії спрямовані на забезпечення найкращих можливих рішень, беручи до уваги дані про рішення. Проте компенсаційні стратегії вимагають повної інформації про те, як альтернативи оцінюються за всіма критеріями, і для оцінювання потребують багато часу.

З іншого боку, некомпенсаційні стратегії узгоджуються з концепцією обмеженої раціональності. Це означає, що обґрунтування прийняття рішення обмежується такими факторами, як важкі обмеження, часові обмеження та когнітивне навантаження особи, що приймає рішення. Таким чином, некомпенсаційні стратегії оцінюють альтернативи евристично, використовуючи лише обмежену кількість критеріїв та компромісів.

Основними характеристиками некомпенсаційних стратегій є те, що вони:

– зменшують зусилля щодо прийняття рішень;

– не вимагають інформації, необхідної для прийняття рішення.

Таким чином, для осіб, які приймають рішення, загальною практикою є застосування некомпенсаційних стратегій у ситуаціях, обмеження яких впливають на процес прийняття рішень. Прикладами

таких ситуацій можуть бути часовий стрес або важкі обмеження. Однак особи, що приймають рішення, не враховують усі критерії під час застосування стратегій некомпенсаційних рішень.

І на останок, у деяких випадках вимагається використання комбінації компенсаційних та некомпенсаційних стратегій прийняття рішень[24 25 26]. Наприклад, дизайнер, який приймає рішення, починає свій процес оцінювання, виключаючи альтернативи, які не відповідають певним некомпенсаційним критеріям, і лише після цього оцінює залишкові альтернативи за допомогою компенсаційної стратегії.

Чинники вибору стратегії прийняття рішень. Рішення, зокрема на вибір стратегії прийняття рішень, впливають такі чинники, як час, повнота інформації, когнітивне навантаження особи, що приймає рішення, та багато іншого[27].

Коротко підсумуємо ці чинники і як вони впливають на процес прийняття рішень.

– *Часовий стрес*: одна з найпоширеніших ситуацій у процесі прийняття рішень – часовий стрес. Приймачі рішень під час тиску повинні приймати критичні рішення. Як правило, ці рішення приймаються в останню мить, адхоком та без достатньої раціоналізації.

– *Неструктуровані проблеми*: проблема рішення часто є складною з точки зору причинно-наслідкових зв'язків, кореляцій та петлі зворотного зв'язку між відповідними факторами. Таким чином, проблеми вирішення важко зрозуміти, а також оцінити їх із точки зору наслідків та результатів, які вони мають.

---

[24] Jeffreys I. The use of compensatory and non-compensatory multi-criteria analysis for small-scale forestry. *Small-scale Forestry*. 2004. Vol. 3. P. 99–117.

[25] Elrod T., Johnson R., White J. A new integrated model of noncompensatory and compensatory decision strategies. *Organizational Behavior and Human Decision Processes*. 2004. Vol. 95. P. 1–19.

[26] Rothrock L., Yin J. Integrating compensatory and noncompensatory decision-making strategies in dynamic task environments. *Decision Modeling and Behavior in Complex and Uncertain Environments*. 2008. P. 125–141.

[27] Alenljung B., Persson A. Portraying the practice of decision-making in requirements engineering: a case of large scale bespoke development. *Requirements engineering*. 2008. Vol. 13. P. 257–279.

– *Неповнота інформації*: це означає, що на практиці інформація може бути неоднозначною або навіть відсутньою.

– *Зміщені, погано визначені чи конкуруючі цілі*: в процесі прийняття рішень люди, що приймають рішення, можуть мати суперечливі цілі. Орган, який приймає рішення, повинен правильно визначати кожну із цих цілей під час прийняття рішення.

– *Дії та зворотній зв'язок*: прийняття рішень містить ряд циклів, які дизайнер повинен враховувати[28]. Ранні помилки та неякісна інформація генерують рішення, які слід переглянути. Наприклад, порушення в архітектурному проекті важко визначити на ранньому етапі, і це означає, що процес прийняття рішень повинен бути повторений знову після їх виявлення.

– *Ситуація з кількома гравцями*: коли до процесу прийняття рішень залучають декілька зацікавлених сторін, ситуація ускладнюється. Зацікавлені сторони мають різні інтереси, цілі та очікування від конкретного рішення. Ще однією поширеною проблемою є відсутність спільного розуміння між зацікавленими сторонами стосовно конкретної проблеми. У багатьох ситуаціях така ситуація може призвести до затримок у прийнятті рішень, що, можливо, призведе до перегляду рішення.

– *Організаційні цілі та норми*: організації діють за конкретними цілями та нормами[28]. Особи, що приймають рішення, повинні приймати рішення в контексті цих цілей та норм, а також повинні уникати прийняття рішення на основі лише їхніх особистих уподобань. Особи, які приймають рішення, незалежно від їхніх особистих переваг, повинні оцінювати альтернативи критеріям, які встановлює організація.

Беручи до уваги вищенаведене, побудована мета-модель компенсаційно-декомпенсаційного підходу до проектування та обґрунтування проектних рішень щодо критичної ІТ-інфраструктури, яка представлена на рисунку 13.

[28] Orasanu J., Connolly T. The reinvention of decision making. *Decision making in action: Models and methods*. 1993. P. 3–20.

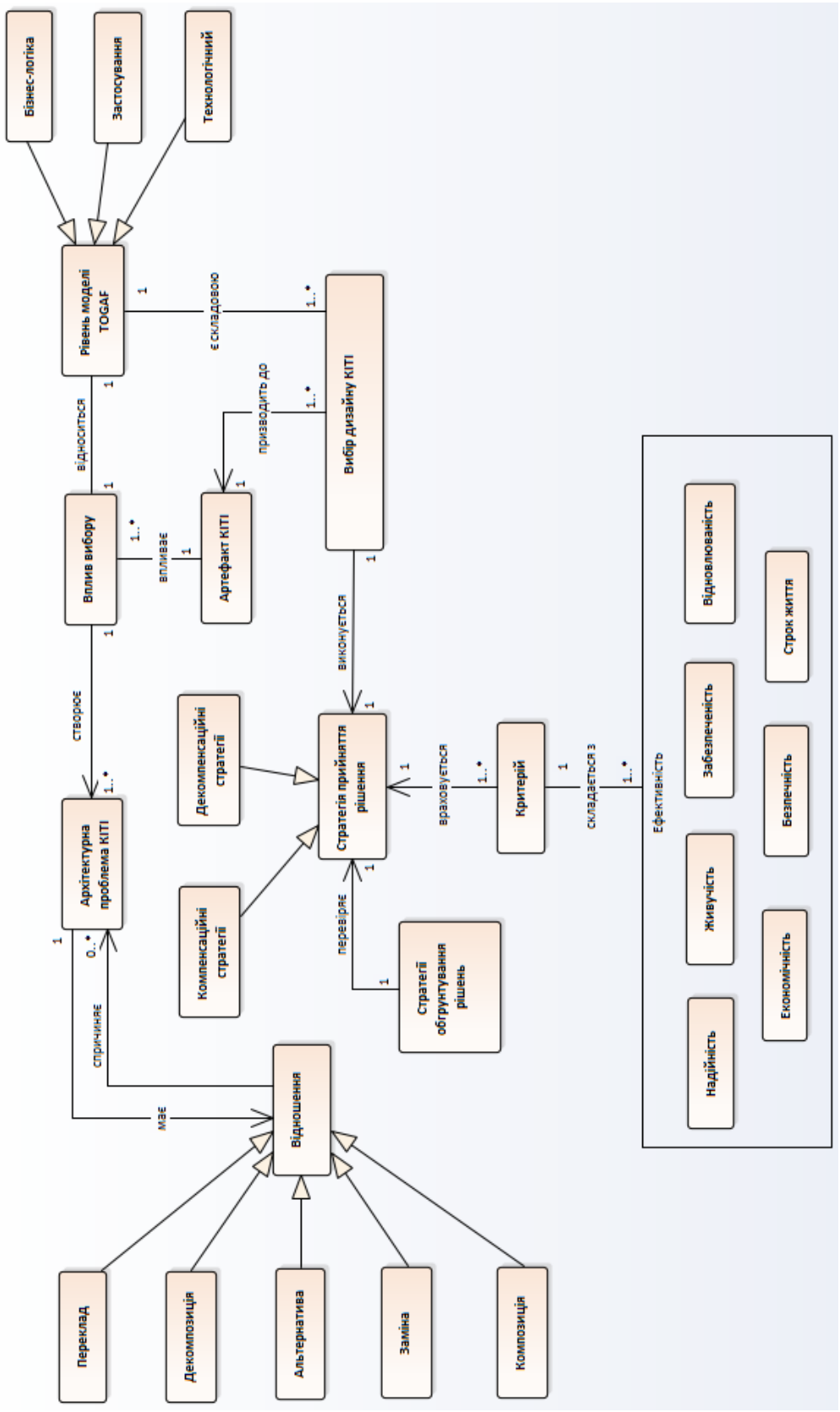

Рис. 13. Мета-модель КДК-підходу

Основними елементами моделі є:

– *вибір дизайну КІТІ* – являє собою фактичне рішення щодо проектування, яке було зроблено після процесу оцінки;

– *архітектурна проблема КІТІ* – являє собою архітектурну проблему, яку архітектори критичної ІТ-інфраструктури повинні вирішувати;

– *артефакт КІТІ* – є або прямим результатом, отриманим із набору виконаних виборів дизайну КІТІ, або представленням цього результату. Наразі артефакт КІТІ використовується для позначення архітектурних уявлень. Зокрема, він використовується як концепція-міст для мови моделювання АП ArchiMate, завдяки якій артефакт КІТІ дозволяє нам зв'язати вибір дизайну КІТІ з елементами «концепція» в ArchiMate;

– *рівень моделі TOGAF* – відповідно до мови ArchiMate[29] та моделі TOGAF[30] підприємство визначається на трьох рівнях: Business, Application та Technology. Використовуючи ці три рівні, можна описати підприємство цілісно, показуючи не тільки програмне забезпечення та фізичну ІТ-інфраструктуру (виражені через рівні застосування та технології), але також і яким чином ІТ впливає/вплинув на діяльність підприємства та його бізнес-стратегію і процеси;

– *вплив вибору* – цей елемент визначає непередбачуваний наслідок вже прийнятого рішення на артефакт КІТІ. Він протиставляється передбачуваним наслідкам, про які свідчать такі відносини, як композиція або переклад. У поточній повсякденній практиці архітектори моделюють передбачувані наслідки у вигляді сценаріїв на правилах типу «ЯКЩО, ТО». Але, на жаль, не можна передбачити всі можливі впливи прийнятих рішень щодо АП. Це особливо стосується критичної ІТ-інфраструктури, де розглядається вплив на всю критичну інфраструктуру, а не на окремі (наприклад, технічні) її частини. Деякі наслідки вибору дизайну КІТІ виявляються на етапі реалізації або під

[29] The Open Group. ArchiMate® 3.0.1 Specification 2017. (дата звернення: 27.10.2018).

[30] TOGAF 9.1 URL: http:/pubs.opengroup.org/architecture/togaf9-doc/arch/ (дата звернення: 18.10.2018).

час підтримки існуючого проекту архітектури. Ці непередбачені наслідки сприймаються точно елементом «Вплив вибору». Для нас головна корисність представлення непередбачених впливів вибору полягає в тому, що вони можуть бути використані архітекторами для уникнення рішень з негативними наслідками в майбутніх проектах архітектури.

Концепції процесу прийняття рішень. Концепції процесу прийняття рішень мета-моделі зосереджені на ухваленні стратегій прийняття рішень, які використовувались під час процесу архітектурного проекту для конкретного рішення вибору дизайну КІТІ, обґрунтуванні вибору стратегії прийняття рішень та наявних альтернатив та критеріїв. Опис відповідних елементів мета-моделі наведені нижче.

– *Стратегія прийняття рішень.* Ця концепція охоплює стратегію прийняття рішень, яку використовує архітектор підприємства, щоб оцінити альтернативи та зробити фактичне рішення щодо вибору дизайну КІТІ. Як вже згадувалося раніше, стратегії прийняття рішень характеризуються як компенсаційні, некомпенсаційні, або як гібрид цих двох.

– *Компенсаційна стратегія:*

✓ *Зважена сума* (WADD): у стратегіях WADD критерії оцінки альтернатив мають різні ваги. Оцінка кожної альтернативи обчислюється шляхом множення кожного критерію на його вагу, а потім беручи суму цих значень. Обирається альтернатива з найвищим балом.

✓ *Рівна вага*: оцінка кожної альтернативи обчислюється таким же чином, як і в стратегіях WADD. Різниця полягає в тому, що критерії мають однакову вагу.

– *Некомпенсаційна стратегія:*

✓ *Кон'юнктивна:* в кон'юнктивних стратегіях альтернативи, які не відповідають одному або декільком критеріям, негайно виключаються з вибору.

✓ *Диз'юнктивна*: в цій стратегії альтернатива обирається, якщо вона відповідає конкретному критерію, незалежно від значень інших критеріїв.

Некомпенсаційні стратегії є доцільними для оцінки альтернатив у ситуаціях неповноти інформації та відсутності числових даних. Наприклад, критерій «потужний комутатор» є суб'єктивним, і його важко обчислити. Альтернативи, які не відповідають цьому критерію, можуть бути усунені з вибору.

Слід також зазначити, що в застосуванні додаткових стратегій прийняття рішень немає обмежень.

Елемент «Критерій» д– це атрибут, який у більшій чи меншій мірі повинен бути задоволений протягом процесу прийняття рішень. Критерії використовуються як для компенсаційних, так і для некомпенсаційних стратегій прийняття рішень. Наприклад, якщо була застосована диз'юнктивна стратегія, приймається рішення про дотримання одного чи декількох критеріїв. Крім того, поняття *вартості* та *ваги* критерію також враховуються в моделі. Концепція вартості являє собою значення, яке дизайнер визначає під час процесу оцінки і яке вказує на те, наскільки добре артефакт відповідає певному критерію. Концепція ваги являє собою важливість цього критерію і зазвичай використовується в стратегіях WADD. До критеріїв оцінки критичної IT-інфраструктури відносяться[31]:

– некомпенсаторні:

1) надійність – показник надійності критичної IT-інфраструктури в період експлуатації;

2) живучість – можливість виконувати свої функції за умов втрати ресурсів, підсистем і т. ін.;

3) безпечність – показник неможливості виконання несанкціонованих дій, спрямованих на порушення роботи критичної IT-інфраструктури чи її частин;

– компенсаторні:

4) забезпеченість – показник максимальної кількості процесів та сервісів, що обслуговуються;

5) відновлюваність – тривалість відновлення готовності до експлуатації;

[31] Dorogyy Y.Y. MANAGEMENT OF CRITICAL IT-INFRASTRUCTURES. *Information and Telecommunication Sciences*. 2014. № 1. P. 10–15.

6) строк життя;

7) економічність – витрати різноманітних ресурсів на забезпечення функціонування критичної IT-інфраструктури;

– гібридного типу (можуть бути і компенсаторними, і некомпенсаторними):

1) ефективність – поєднання вищезгаданих параметрів у кожному окремому випадку під визначену задачу;

2) вартість проекту.

Елемент «Стратегії обґрунтування рішень»: у процесі прийняття рішень архітектору потрібно не лише вибрати деякі альтернативи (фактичний процес прийняття рішень), але також вибрати стратегію прийняття рішень, що задовольняє його поточні потреби в оцінці. Фактично цей елемент включає в себе обґрунтування стратегії прийняття рішень, яка була обрана для процесу оцінки.

Як вже визначалось вище, різні фактори впливають на процес прийняття рішень, і відповідальним особам слід відповідно адаптувати свою стратегію прийняття рішень. Поняття стратегічного обґрунтування дає змогу особі, що приймає рішення, задовольнити його поточні потреби в оцінці.

Важливим елементом моделі є *відношення*. Роль концепцій відносин полягає в тому, щоб різноманітні типи зв'язків між елементами «Вибір дизайну КІТІ» були явними. Виходячи з онтологій проектних рішень архітектури програмного забезпечення, в пропонованій моделі визначені п'ять типів відносин:

– *переклад* – відношення даного типу ілюструють зв'язки між елементами «*Вибір дизайну КІТІ*» та «*Архітектурна проблема КІТІ*», що належать до різних рівнів або різних артефактів КІТІ. Відповідно до мета-моделі відношення перекладу застосовуються для зв'язку елементів «*Вибір дизайну КІТІ*» між собою або для зв'язку з елементами «*Архітектурна проблема КІТІ*». Відношення перекладу базуються на типі загального відношення «is related to», тільки адаптованого до основної термінології архітекторів критичних IT-інфраструктур. Одне з ключових завдань для архітектора – шляхом комунікації перекласти вимоги, які нові артефакти КІТІ накладають

(*Архітектурна проблема КІТІ*) для рішень, які будуть підтримувати ці вимоги за допомогою іншого артефакту КІТІ;

– *декомпозиція* – відношення декомпозиції означають, як загальні рішення про АП (*Вибір дизайну КІТІ*) розкладаються на більш детальні проектні рішення;

– *альтернатива* – тип альтернативних відношень ілюструють рішення, які були відхилені (альтернативи), для вирішення конкретної архітектурної проблеми КІТІ;

– *заміна* – відношення заміни пояснюють, як одне рішення щодо архітектури заміщає негативний результат іншого рішення;

– *композиція* – відношення композиції означають, як детальні проектні рішення складаються в загальне рішення про АП (*Вибір дизайну КІТІ*).

Обговорення результатів дослідження

У даній роботі представлений детальний компаративний аналіз підходів до обґрунтування проектних рішень, визначені їх недоліки. Як результат запропонована мета-модель компенсаційно-декомпенсаційного підходу до проектування та обґрунтування проектних рішень щодо критичної IT-інфраструктури та відповідна візуалізація для відображення стратегій обґрунтування рішень, стратегій прийняття рішень стосовно архітектури критичної IT-інфраструктури. Разом інтегрована модель мета-моделі та виробничої моделі дозволяє порівнювати процес прийняття рішень із спостереженими результатами рішення. Таке порівняння прийняття рішень із спостережуваним впливом призводить до кращого розуміння існуючої архітектури критичної IT-інфраструктури. 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 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 із головних завдань є вивчення *способів* зменшення зусиль під час проектування з використанням запропонованого підходу.

ВИСНОВКИ

1. Проаналізовано основні існуючі підходи до обґрунтування проектних рішень щодо архітектури підприємства, визначені їх недоліки та можливість їх подальшого застосування в проектуванні архітектури критичної IT-інфраструктури.

2. Запропоновано мета-модель компенсаційно-декомпенсаційного підходу до проектування та обґрунтування проектних рішень щодо критичної IT-інфраструктури та шляхи її подальшого використання.

АНОТАЦІЯ

У статті розглядаються питання, пов’язані з проблемою обґрунтування проектних рішень щодо архітектури критичної IT-інфраструктури.

По-перше, проведено порівняльний аналіз підходів для обґрунтування і представлення архітектури критичної IT-інфраструктури на базі запропонованих критеріїв, визначені їх недоліки та переваги, можливість їх використання для обґрунтування проектних рішень щодо критичної IT-інфраструктури. Зроблено висновок про необхідність створення нового підходу до обґрунтування рішень щодо архітектури критичної IT-інфраструктури через неможливість застосування розглянутих методів та підходів.

По-друге, в статті наведено огляд існуючих стратегій прийняття рішень щодо архітектури підприємства та чинників, які впливають на процес прийняття цих рішень. Описана необхідність врахування наведених чинників під час проектування та побудови складних архітектур, таких як архітектура критичної IT-інфраструктури.

Запропоновано новий компенсаційно-декомпенсаційний підхід, побудований із використанням гібридної стратегії прийняття

проектних рішень на базі введених критеріїв. Описано мета-модель підходу, її основні елементи та їх взаємодія. Визначені подальші кроки дослідження та розвитку запропонованого підходу.

Результати, викладені в статті, допоможуть дослідникам і розробникам архітектури критичних IT-інфраструктур вибрати найбільш придатний варіант стратегії прийняття рішень під час проектування власної архітектури, а також ознайомитися з критеріями їх проектування та оцінки для проведення власного дослідження. Варто мати на увазі, що не існує універсального підходу, який би підійшов для розробки будь-якої архітектури, тому так важливо визначити слабкі та сильні сторони кожного з них на етапі проектування системи, а також критерії проектування. Від того, наскільки вірно було визначено ці критерії, буде залежати вибір і конкретних технологій, і програмного забезпечення, а також успішність проектування.

ЛІТЕРАТУРА

1. Kunz W. Issues as Elements of Information Systems. Berkeley, 1970.

2. Enterprise architecture: creating value by informed governance / M. Op't Land and etc. Springer, 2008.

3. Aier S., Winter R. Virtual decoupling for it/business alignment-conceptual foundations, architecture design and implementation example. *Business & Information Systems Engineering*. 2009. Vol. 1. P. 150–163.

4. Jansen A., Bosch J. Software architecture as a set of architectural design decisions. *Software Architecture*. 2005. P. 109–120.

5. Tang A., Jin Y., Han J. A rationale-based architecture model for design traceability and reasoning. *Journal of Systems and Software*. 2007. Vol. 80. P. 918–934.

6. Plataniotis G., de Kinderen S., Proper H.A. Capturing decision making strategies in enterprise architecture – a viewpoint. *Enterprise, Business-Process and Information Systems Modeling*. 2013. Vol. 147. P. 339–353.

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

8. Toulmin S. The Uses of Argument, 2nd ed. Cambridge University Press, 2003.

9. Extending the Potts and Bruns Model for Recording Design Rationale. *13th International Conference on Software Engineering*. 1991. P. 114–125.

10. Tyree J., Akerman A. Architecture decisions: Demystifying architecture. *Software, IEEE*. 2005. Vol. 22. P. 19–27.

11. Documenting Software Architectures: Views and Beyond, 1st ed. / *P. Clements and etc.* Addison Wesley, 2002.

12. Asundi J. Using Economic Considerations to Choose Amongst Architecture Design Alternatives. 2001.

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

14. PHI: A conceptual foundation for design hypermedia. *Design Studies*. 1991. Vol. 12. № 1. P. 30–41.

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

16. Conklin J., Begeman M. IBIS: A hypertext tool for exploratory policy discussion. *Proceedings ACM Conference on Computer-Supported Cooperative Work*. 1988.

17. Ramesh B., Dhar V. Supporting Systems Development by Capturing Deliberations During Requirements Engineering. *IEEE Transactions on Software Engineering*. 1992. Vol. 18. № 6. P. 498–510.

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

19. Lee J. SIBYL: A Tool for Managing Group Decision Rationale. *Proceedings of the Conference on on Computer-Supported Coopoerative Work*. 1990. P. 77–92.

20. Maclean A., Young R., Bellotti V., Moran T. Questions, Options and Criteria *Design Rationale: Concepts, Techniques and Use*. Ch. 3. Lawrence Erlbaum Associates, 1996. P. 53–106.

21. Dutoit A.H., Paech B. Rationale-Based Use Case Specification. *Requirements Engineering*. 2002. Vol. 7. № 1. P. 3–19.

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

23. Burge J. Software Engineering Using design RATionale, Ph.D. dissertation. 2005.

24. Jeffreys I. The use of compensatory and non-compensatory multi-criteria analysis for small-scale forestry. *Small-scale Forestry*. 2004. Vol. 3. P. 99–117.

Elrod T., Johnson R., White J. A new integrated model of noncompensatory and compensatory decision strategies. *Organizational Behavior and Human Decision Processes*. 2004. Vol. 95. P. 1–19.

25. Rothrock L., Yin J. Integrating compensatory and noncompensatory decision-making strategies in dynamic task environments. *Decision Modeling and Behavior in Complex and Uncertain Environments*. 2008. P. 125–141.

26. Alenljung B., Persson A. Portraying the practice of decision-making in requirements engineering: a case of large scale bespoke development. *Requirements engineering*. 2008. Vol. 13. P. 257–279.

27. Orasanu J., Connolly T. The reinvention of decision making. *Decision making in action: Models and methods*. 1993. P. 3–20.

28. The Open Group. ArchiMate® 3.0.1 Specification 2017. (дата звернення: 27.10.2018).

29. TOGAF 9.1 URL: http://pubs.opengroup.org/architecture/togaf9-doc/arch/ (дата звернення: 18.10.2018).

30. Dorogyy Y.Y. MANAGEMENT OF CRITICAL IT-INFRASTRUCTURES. *Information and Telecommunication Sciences*. 2014. № 1. P. 10–15.

Information about authors:

Dorogyy Ya. Yu.,

Candidate of Technical Sciences, Docent,

Associate Professor in Department of Automation

and Control in Technical Systems

National Technical University of Ukraine
“Igor Sikorsky Kyiv Politechnic Institute”
37, Peremohy av., Kyiv, 03056, Ukraine
Tsurkan V. V.,
Candidate of Technical Sciences,
Associate Professor in Department of Cybersecurity and Application of Automated Informational Systems and Technologies
Institute of Special Communication and Information Security of
National Technical University of Ukraine
“Igor Sikorsky Kyiv Politechnic Institute”
4, Verchnokluchova st., 4, Kyiv, 03056, Ukraine