УДК: 004.891:005.8:330.4 DOI 10.33111/mise.105.2

Варава І.А.,
Кандидат технічних наук, доцент кафедри інженерії програмного забезпечення в енергетиці, НТУУ «Київський політехнічний інститут ім. Ігоря Сікорського»
Гагарін О.О.,
Кандидат технічних наук, доцент кафедри інженерії програмного забезпечення в енергетиці, НТУУ «Київський політехнічний інститут ім. Ігоря Сікорського»
Гайдраржи В.І.,
Завідувач лабораторії комп'ютерного моделювання в енергетиці, НТУУ «Київський політехнічний інститут ім. Ігоря Сікорського»
Корольов А.П.,
кандидат технічних наук, професор кафедри системного аналізу та кібербепеки, КНЕУ імені Вадима Гетьмана

Ivan A. Varava,
Cand.tech.sc., assoc.prof. of Software Engineering in Energy Department, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Oleksander O. Haharin,
Cand.tech.sc., assoc.prof. of Software Engineering in Energy Department, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Volodymyr I. Gaidarzhy,
Head of the Laboratory of Computer Modeling in Energy, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Koroliov A.P.,
Cand.tech.sc., Prof. of Department of Systems Analysis and Cybersecurity Kyiv National Economic University named after Vadym Hetman

ОНТОЛОГІЧНО-ОРІЄНТОВАНЕ ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМ ПІДТРИМКИ ПРИЙНЯТТЯ ІННОВАЦІЙНИХ РІШЕНЬ ДЛЯ ВАРТІСНОЇ ОПТИМІЗАЦІЇ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНИХ СИСТЕМ

ONTOLOGY-ORIENTED INFORMATION SUPPORT FOR DECISION SUPPORT SYSTEMS FOR COST OPTIMIZATION OF THE SOFTWARE SYSTEMS LIFE CYCLE

***Анотація**. У статті досліджено методологію побудови онтолого-орієнтованого інформаційного забезпечення систем підтримки прийняття рішень для оптимізації загальної вартості володіння програмними*

системами протягом їх життєвого циклу. Запропоновано використання онтологій та графів знань для інтеграції даних, забезпечення семантичної узгодженості та автоматизації процесів управління. Розроблено модель онтології, що охоплює повний цикл розробки та експлуатації і включає проєкти, артефакти, показники якості, ресурси, витрати та ризики. На її основі сформовано структуру системи, яка об'єднує дані з різних інструментів у єдиний граф знань і забезпечує логічне виведення, аналіз впливу змін та формування рекомендацій. Запропоновано підхід до оптимізації витрат, що включає оцінювання трудовитрат, автоматичне формування робіт, управління технічним боргом і ризикорієнтоване тестування. Практична цінність полягає у підвищенні обґрунтованості рішень, зменшенні витрат життєвого циклу та повторному використанні знань.
***Ключові слова**: граф знань, OWL, RDF, система прийняття рішень, управління проєктами, життєвий цикл програмного забезпечення, TCO, онтологія, оптимізація вартості, технічний борг.*

***Abstract**. The article investigates the methodology for constructing ontology-oriented information support for decision support systems aimed at optimizing the total cost of ownership (TCO) of software systems throughout their life cycle. The use of ontologies and knowledge graphs is proposed to enable data integration, ensure semantic consistency, and automate management processes. An ontology model covering the full development and operation life cycle is developed, including projects, artifacts, quality attributes, resources, costs, and risks. Based on this model, a system structure is formed that integrates data from various tools into a unified knowledge graph and supports logical reasoning, change impact analysis, and recommendation generation. A cost optimization approach is proposed, incorporating effort estimation, automated task generation, technical debt management, and risk-based testing. The practical value lies in improving decision justification, reducing life cycle costs, and enabling knowledge reuse.*
***Keywords:** knowledge graph, OWL, RDF, decision support system, project management, software life cycle, TCO, ontology, cost optimization, technical debt.*

Постановка наукової проблеми та її значення. Сучасне управління життєвим циклом програмних комплексів здійснюється в умовах високої складності, багатокомпонентності та постійних змін вимог, архітектурних рішень, середовищ розгортання й експлуатації. Численна кількість інструментів опосередковано використовується на різних стадіях життєвого циклу програмного забезпечення. Такий підхід ускладнює прийняття обґрунтованих рішень по управлінню проектами у зв'язку із відсутністю єдиного інформаційного простору.

Існуючі системи управління проектами переважно виконують операційний облік та трасування. Проте вони недостатньо використовують знання про предметну область, не використовують залежності між артефактами життєвого циклу та не автоматизують визначення наслідків управлінських рішень для вартості володіння програмним забезпеченням.

Аналіз останніх досліджень і публікацій. Життєвий цикл програмних систем у сучасній інженерії програмного забезпечення

регламентується міжнародним стандартом ISO/IEC/IEEE 12207:2017 [1]. В ньому описуються процеси розробки, експлуатації та супроводу програмного забезпечення. Керування проєктами у сфері ПЗ є процесом, що інтенсивно використовує знання, тому перспективним є застосування онтологічного підходу. Дослідники наголошують, що впровадження онтологій сприяє покращенню управління знаннями та підвищує рівень повторного використання артефактів [8].

Вартісні моделі оцінювання і планування представлені класичними підходами COCOMO II [6], Function Point Analysis [4], Cost of Quality[5], що був адаптований для програмного забезпечення. Також застосовуються практики керування технічним боргом.

Галузь онтологій та графів знань також розвивається досить швидко, тому існує багато спільного між тим, як ми описуємо речі за допомогою онтології, та тим, як ми будуємо цю онтологію в графі (тобто стандарти OWL/RDF використовуються для формального опису термінів стосовно артефактів, а також їхніх зв'язків один з одним, та підтримують простежуваність та/або логічний висновок).

Автори статті [3] приходять до висновку, що «об'єднання онтології управління… і онтології підтримки прийняття рішень дає змогу… знаходити рішення в проблемних ситуаціях».

Розглядаючи в роботі [2] онтологію, автори ставлять вимогу «щоби програмний засіб був у змозі самостійно класифікувати нові дані, що одержуються при аналізі, і пропонувати користувачеві найбільш доцільні кроки сценарію, виходячи з накопичених знань».

Метою статті є дослідження методології застосування онтолого-орієнтованої системи прийняття рішень по оптимізації загальної вартості володіння (ТСО) програмним забезпеченням в системах управління проектами. В роботі розробляється онтологічне дерево класів життєвого циклу програмного забезпечення на основі якого проводиться оптимізація управління розробкою та експлуатацією програмних систем. До структури інформаційної підтримки DSS входять графа знань, правила, процеси оптимізації та можливість інтеграції з відомими системами керування проектами.

Процедури оптимізації витрат були обґрунтовані (як процедури) та включають формалізаційні процедури (з використанням формалізації DoD/gate), а також процедури аналізу змін, а також методи, що перевіряють витрати відповідно до прийнятих ризиків, а також враховують процеси управління технічним боргом та інцидентами.

Поширені відомі системи керування проектами фіксують події, що пов'язані з виконанням задач під час розробки програмного забезпечення. Рішення, які приймає менеджер проекту, все ж таки суб'єктивні, і успішність проекту безпосередньо може залежати від досвіду менеджера.

Ключовими класами онтології являються проект, інженерні артефакти, показники якості, ресурси, фінансові показники та ризики. Фрагмент онтології представлений на рис. 1.

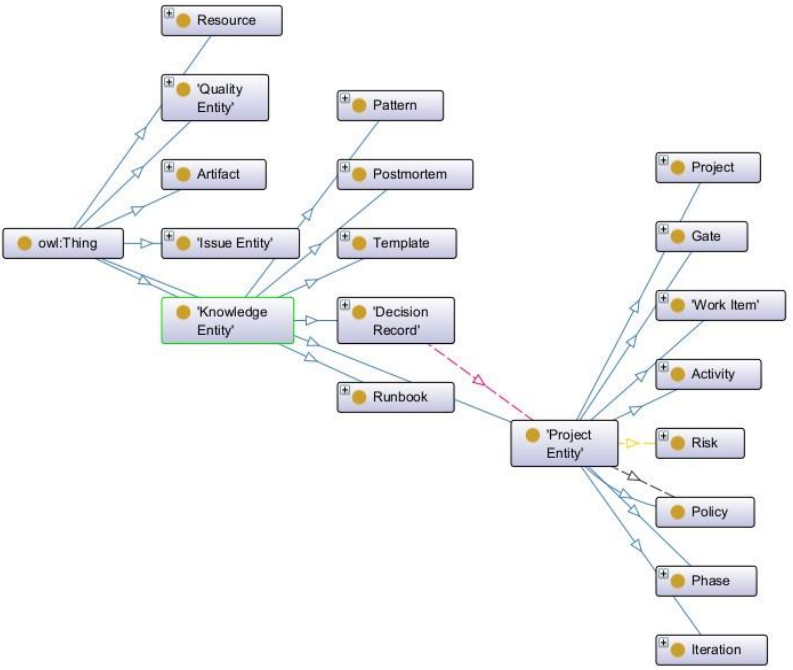

Рис. 1. Фрагмент онтології життєвого циклу розробки програмної системи

В запропонованому підході базу знань накопиченого досвіду складається із документів різного типу. До бази знань включаються записи про прийняті рішення у форматі ADR, сформовані шаблони для створення нових артефактів, патерни проектування, інструкції по реагуванню на конкретні інциденти. Необхідним компонентом бази знань також є аналіз фаз проекту, на основі якого можна отримати знання, що трансформуються у нові правила чи ризики. База

знань стає джерелом причинно-наслідкових зв'язків: від варіанта реалізації до прийнятого рішення.

Окрім класів важливо визначити їх властивості, що представляються у триплетах RDF предикатами і дозволяють перетворюють статичну ієрархію понять на динамічний граф знань, що дозволяє системі автоматично відстежувати залежності, виявляти суперечності та розраховувати інтегральні показники проєкту.

Дана онтологія дозволяє вирішити ряд оптимізаційних задач як функцій СППР. Ці задачі вирішуються на основі правил та SPARQL-запитів. В процесі оптимізації витрат розглядаються семантична оцінка трудовитрат та бюджету, автогенерація шаблонів робіт, перевірка критеріїв готовності, оптимізація призначення виконавців на основі їх компетенцій, пріоритезація технічного боргу, прогнозування додаткових витрат, оптимізація експлуатації та інші.

Готовність виконання конкретних задач при розробці ПЗ визначається на основі онтологічних правил, що виявляють наявність потрібних артефактів, результатів перевірок та виконаних критеріїв приймання. Наприклад, для функціоналу переказу коштів між рахунками користувача у вебсистемі інтернет-банкінгу необхідно мати: модель загроз з можливими атаками та механізмами захисту, докази перевірки безпеки за допомогою таких інструментів як OWASP ZAP, перелік матеріалів програмного забезпечення (SBOM) для складових якого відсутні CVE з високим рівнем.

В інтелектуальних системах управління великими проєктами застосування онтології робить більш гнучким процес виявлення заборони прийняття рішень, генерування на основі шаблонів відсутніх задач та оцінювання ризику релізу програми із відомими багами.

Відомо [9, 10], що найдорожче дефекти виправляються після інтеграції та впровадження.

Одним із факторів, що суттєво вливає на вартість розробки є зміни вимог. Зміна вимог пов'язана з іншими артефактами життєвого циклу: вимогою, програмним компонентом, набором тестів, маніфестом розгортання. Наприклад, якщо зміна стосується безпеки, то СППР повинна додати обов'язкові елементи: модель загроз, перелік компонентів, перевірки безпеки. Також на основі онтології автоматично мають згенеруватися необхідні задачі: створення контрактних тестів, оновлення інструкцій та документації, перевірка плану відкату. Поряд з цим виконується аналіз впливу, наприклад, оцінка трудовитрат чи оцінка вартості простою.

Таким чином рішення про прийняття чи відкладення змін стає прозорим і економічно обґрунтованим.

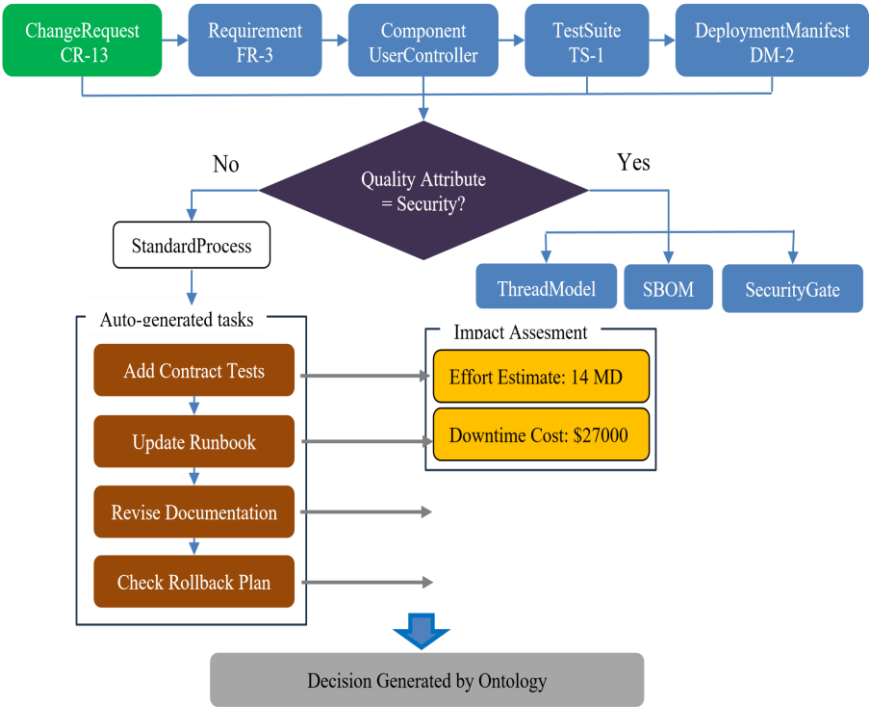

Рис. 2. Онтологічний вивід та аналіз впливу для запиту на зміни

У системах із підвищеними вимогами (надійність, безпека, продуктивність) “економія на тестах” часто веде до зростання операційних витрат (OPEX). Тому необхідно зосередитись на тих компонентах програмного забезпечення, де ризик від потенційних збитків найбільший. Онтологія допомагає зрозуміти пріоритет виконання тестів і сформувати оптимальні сценарії подальшого управління проектом.

СППР може оптимізувати набір тестів, мінімізуючи витрати на перевірку при обмеженні на ризик, а також запропонувати набір рішень по застосуванню автоматизованого тестування, статичного аналізу, рефакторингу тощо.

Розробка складного програмного забезпечення, що орієнтоване на довгострокове використання, потребує чіткого розуміння причин виникнення та оцінки технічного боргу. Таку оцінку можна

представити, використовуючи запропоновану Вордом Каннінгемом у 1992 році концепцію «відсотків боргу»[7]. Однією із функцій СППР є мінімізація довгострокового ТСО та формування сценарію погашення технічного боргу.

В роботі пропонується архітектура інформаційного забезпечення СППР (рис. 3):

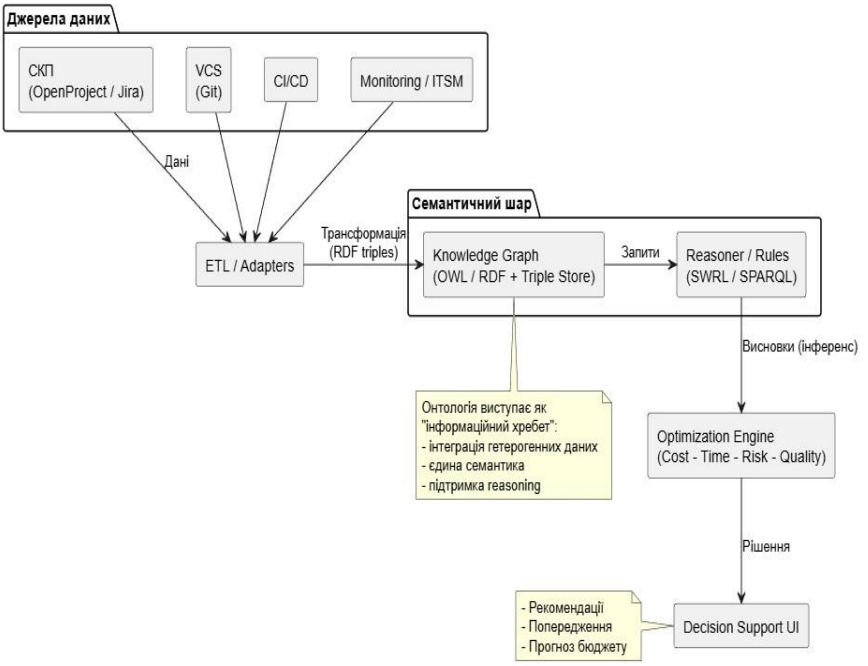

Рис. 3. Архітектура онтологічно-орієнтованого інформаційного забезпечення СППР.

Рівень прийому даних включає адаптери для існуючих систем управління проектами, систем керування версіями, систем CI/CD, систем моніторингу та інструментів ITSM. Граф знань базується на сховищі RDF-триплетів та OWL-онтології, які разом зберігають інформацію, яка зібрана з різних системних адаптерів.

Інтелектуальний шар системи, який відповідає за логічне виведення нових знань із зібраних даних. Для його складу входять резонер, правила SWRL та механізм виявлення нових знань на основі запитів SPARQL CONSTRUCT (див. Прог. 1). На основі правил реалізуються конкретні механізми управління проектами, наприклад

визначення умов зміни статусу задач чи підбір набору робіт в залежності від типу проекту.

Прог. 1. Приклад побудови семантичного графа впливу PREFIX ex: <http://example.org/ontology#>

```
CONSTRUCT {
 ex:CR_17 ex:affectsRequirement ?req .
 ?req ex:implementedBy ?comp .
 ?comp ex:verifiedBy ?testSuite .
 ?comp ex:deployedBy ?manifest .
}
WHERE {
 ex:CR_17 ex:affectsRequirement ?req .
 ?req ex:implementedBy ?comp .
 OPTIONAL { ?comp ex:verifiedBy ?testSuite . }
 OPTIONAL { ?comp ex:deployedBy ?manifest . } }
```

В модулі оптимізації реалізується багатокритеріальна оптимізація на основі даних графу знань. Результатом є визначене найкраще рішення, що враховує показники вартості, часу, ризику, якості та інші.

Елементами інтерфейсу підтримки рішень є інтерактивні панелі рекомендацій з пропозиціями, попередженнями, розрахунками вартості змін, прогнозуванням бюджету та SLO технічного боргу.

Висновки. У статті досліджено можливості застосування онтологічно-орієнтованого підходу для побудови систем підтримки прийняття рішень, спрямованих на оптимізацію загальної вартості володіння (TCO) програмними системами протягом усього життєвого циклу. Розроблено модель онтології життєвого циклу програмної системи, що містить основні сутності. На основі онтології запропоновано структуру інформаційного забезпечення СППР для багатокритеріальної оптимізації.

Бібліографічні посилання

1. ISO/IEC/IEEE 12207:2017. Systems and software engineering — Software life cycle processes.

2. Додонов О. Г., Коваль О. В., Сенченко В. Р., Шпурик В. В. Автоматизована система формування сценарію аналітичної діяльності. *Реєстрація, зберігання і обробка даних*. 2019. Т. 21, № 1. С. 11–22.

3. Ткаченко, К., & Байдак, А. (2021). Онтологічне моделювання інтелектуальної системи підтримки прийняття рішень під час аналізу ризиків у інноваційноінвестиційній сфері. *Цифрова платформа: інформаційні технології в соціокультурній сфері*, *4*(1), 66–78.

4. J. Albrecht and J. E. Gaffney, “Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation,” IIEEE Trans. Software Eng., vol. SE-9, no. 6, pp. 639–648, Nov. 1983 doi: 10.1109/TSE.1983.235271.

5. Boehm, B. (1981). Software Engineering Economics. Prentice-Hall.

6. Boehm B., Abts C., Brown A. W. et al. Software Cost Estimation with COCOMO II. Upper Saddle River: Prentice Hall, 2000.

7. Cunningham W. The WyCash portfolio management system // *Addendum to the Proceedings on Object-oriented Programming Systems, Languages, and Applications*. New York : ACM, 1992. P. 29–30.

8. Fitsilis, P., Gerogiannis, V. and Anthopoulos, L. (2014) Ontologies for Software Project Management: *A Review. Journal of Software Engineering and Applications*, 7, 1096-1110

9. Jones C. Applied Software Measurement: Global Analysis of Productivity and Quality. 3rd ed. New York: McGraw-Hill, 2008.

10. Pressman, R. S. Software Engineering: A Practitioner’s Approach.