DOI: 10.32347/2412-9933.2023.54.69-76
УДК 004:728+827

Гончаренко Тетяна Андріївна
Кандидат технічних наук, доцент, доцент кафедри інформаційних технологій проєктування та прикладної математики, *https://orcid.org/0000-0003-2577-6916*
Київський національний університет будівництва і архітектури, Київ

АРХІТЕКТУРА ПРОГРАМНОЇ СИСТЕМИ НА ОСНОВІ КОНЦЕПЦІЇ РЕФЛЕКСИВНОЇ АДАПТАЦІЇ

Анотація. *Проведене дослідження обґрунтовує актуальність застосування та необхідність розроблення адаптивного програмного забезпечення для вирішення проблем використання ІТ у будівельній галузі. Розглянуто базові принципи рефлексивної адаптації, а також використання технології розроблення лінійки програмних продуктів (англ. Software Product Lines Engineering (SPLE)) для створення адаптивних програмних систем. Запропоновано узагальнену архітектуру адаптивної програмної системи, яка заснована на моделях варіативності та рівнях рефлексії. Здійснено формальний опис рефлексивної програмної системи для реалізації варіантів адаптивної поведінки прикладних програм. Практичне значення дослідження полягає в тому, що надання прикладній програмній системі адаптивних властивостей дасть змогу їй здійснювати модифікацію своєї структури на якісно новому рівні, що своєю чергою підвищить її надійність, стійкість до відмов, гнучкість, знизить вартість супроводу та продовжить термін експлуатації. Така система зможе розширювати клас розв'язуваних завдань протягом усього життєвого циклу, а також виконувати такі операції, які на сьогодні вважаються частиною обов'язків певних спеціалістів, наприклад, зможе виконувати функції самоадміністрування.*

Ключові слова: адаптивні програмні системи; Software Product Lines Engineering (SPLE); концепція рефлексивної адаптації; модель варіативності; технологічна лінійка програмних продуктів; програмна інженерія

Актуальність та аналіз проблеми

Забезпечення інформаційних потреб будівельної галузі залежить від якісного використання арсеналу програмно-апаратних комплексів та комп'ютерних технологій. Інтенсивність та динаміка інформаційних процесів накладають певні вимоги до програмного забезпечення (ПЗ). Є низка прикладних завдань будівельного проєкту, для яких модернізація обслуговуючого ПЗ для внесення оновлень у математичні та інформаційні моделі, структури даних, алгоритми, вихідний код, інтерфейсні оболонки тощо пов'язана з великими труднощами або зовсім неможлива. І єдиним виходом у цьому випадку є адаптація програми в режимі реального часу, без зупинок на перепроєктування та перекомпіляцію. Такі програмні компоненти мають вміти самостійно відстежувати критичні зміни, адекватно пристосовуватися до них у фоновому режимі, не перестаючи виконувати свої основні корисні функції.

Адаптація прикладного ПЗ є однією з принципово важливих задач у сфері сучасної програмної інженерії, яка має як теоретичне значення, так і практичне значення, і є особливо актуальною для вітчизняного ІТ-сектору, оскільки в найближчі роки очікується підвищений інтерес з боку урядових структур, бізнесу та науки до питань інтеграції будівельно-технологічних та програмно-кібернетичних процесів. Це пов'язано з пріоритетною програмою переходу країни до моделей цифрової економіки. За прогнозами експертів, найбільший розвиток до 2030 р. мають досягнути саме інформаційні системи, що працюють з великими масивами даних (Big Data).

Аналіз наукових праць [1 – 3] засвідчив, що і досі існують певні проблеми застосування ІТ у будівельній галузі, які потребують негайного вирішення. По-перше, це зростаюча структурна складність сучасних прикладних програмних систем, яка призводить до збільшення часу та витрат на розроблення ПЗ, ускладнень супроводу, скорочення тривалості життєвого циклу, зниження стійкості, надійності та гнучкості. По-друге, необхідність інтеграції декількох програмних комплексів та систем для реалізації будівельного проєкту протягом життєвого циклу, що потребує швидкого відновлення дієздатності ПЗ при виникненні колізій та може призвести до втрат великого обсягу даних.

По-третє, сталі інформаційні системи не можуть оперативно адаптуватися до динамічних змін і вимог цифровізації та швидко застарівають.

Проблеми, пов'язані з питаннями адаптивної поведінки інформаційних систем, успішно вирішувалися в рамках теорії автоматичного управління (ТАУ), що підтверджується висновками досліджень в роботах [4 – 6]. Однак практичний досвід останніх років показав, що механізми зворотного зв'язку в програмному забезпеченні і досі погано формалізовані, а методи реалізації адаптивної поведінки, запропоновані класичною теорією управління, не є актуальними для програмних систем через їх інформаційну природу.

Проблематика системної поведінки на основі зворотних зв'язків, включаючи питання адаптації, є традиційним предметом вивчення кібернетики. За понад півстоліття розвитку цієї науки накопичено багатий досвід концептуальних, математичних, інформаційних та технічних рішень подібних завдань. Термін "програмна кібернетика" (software cybernetics) було введено відомим фахівцем у галузі програмної інженерії К. Кає у роботі [11]. Він спочатку означав спробу застосувати методи класичної кібернетики і теорії управління до програмних систем з метою оптимізації тривалості життєвого циклу та скорочення витрат на розроблення складних програмних систем.

Останнім часом програмна кібернетика пройшла значні зміни і на сьогодні включає не лише спроби реалізації основних кібернетичних принципів щодо структури програмного забезпечення, а й доволі оригінальні концепції, пов'язані з питаннями розподілених та хмарних обчислень, створенням та експлуатацією кіберфізичних систем, зокрема, мережевих систем та Інтернету речей, процесів розроблення, тестування та супроводження програм. Проте, незважаючи на великий інтерес до цього напряму, тема створення універсальних (іншими словами, застосовних до широкого класу прикладних програмних систем) механізмів адаптації та адаптації практично не досліджена. Наявні підходи, які були викладені, зокрема у роботах [8 – 10], не дають змоги вирішити весь клас проблем, пов'язаних з адаптацією. Системи, побудовані з їх використанням, не є достатньо гнучкими та надійними.

Особливо перспективними є ідеї адаптації програмних систем, задачі і результати застосування якої окреслено в роботах [11; 12]. Сучасні прикладні програмні системи та інформаційні технології, які зараз застосовуються в будівельній галузі, опрацьовують великі обсяги складно структурованої та неструктурованої інформації, мають дуже складну архітектуру, повинні мати відповідну поведінкову складність і різноманітність без порушення встановленої функціональної компактності.

Сучасні програмні системи перебувають у такому стані, що неможливо заздалегідь, на етапі проєктування, врахувати всі можливі обставини функціонування. Багато речей стають очевидними лише в процесі експлуатації, що змушує розробників ПЗ виділяти більше ресурсів процесу супроводу системи, регулярно випускати оновлення, які не містять кардинально нової функціональності.

Розв'язанням проблеми недостатності знань може стати надання системам здатності до самостійного аналізу власного стану. У цьому випадку розробнику достатньо закласти в програму лише базові механізми аналізу та пошуку закономірностей, припускаючи, що деякі моменти на етапі розроблення залишилися неврахованими, а виявленням відсутніх знань система буде займатися самостійно протягом свого життєвого циклу. Такий підхід допоможе не тільки знизити витрати на супровід програмної системи, але й виявити ті особливості інформаційного забезпечення будівельного проєкту, які були невідомі навіть експертам.

Для вирішення зазначених проблем доцільно використовувати концепцію рефлексивної адаптації. Базові принципи та вимоги до її впровадження у сфері програмної інженерії зазначено в роботі [13] і наведено на рис. 1.

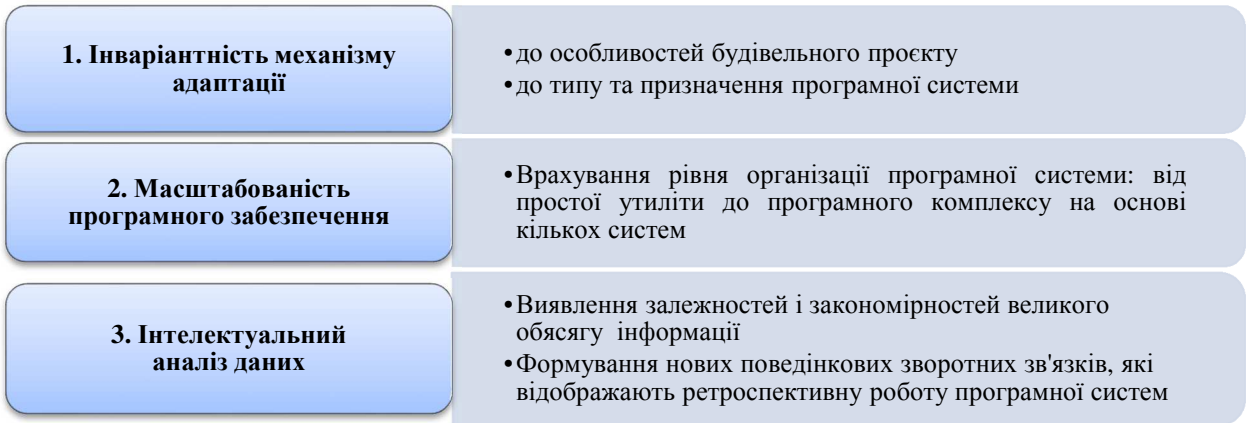

Рисунок 1 – Базові принципи концепції рефлексивної адаптації

Мета статті

Метою статті є дослідження актуальності та вирішення проблеми адаптації програмних компонентів на основі концепції поведінкової рефлексії, що дасть змогу розробити архітектуру програмної системи на основі концепції рефлексивної адаптації для забезпечення інформаційних потреб будівельної галузі.

Виклад основного матеріалу

Опис принципів узагальненої архітектури програмної системи на основі концепції рефлексивної адаптації

Зазначений вище принцип масштабованості (рис. 1) висуває певні вимоги до архітектурної організації програмної системи. Адаптивна архітектура повинна легко модифікуватися, і це стосується не тільки перебудови програми на основі результатів рефлексивного аналізу, а й модифікації програми ззовні (наприклад, самими розробниками) та подальшого врахування змін у процесі експлуатації. Отже, новий модуль, підключений до системи, має відразу ж долучатися до загального процесу рефлексії, під час якого буде виявлено його зв'язок з іншими модулями та їх взаємний вплив один на одного. Крім цього, певні закономірності мають бути виявлені у функціонуванні самого модуля.

На основі проведеного аналізу робіт [14; 15] щодо розроблення систем з модульною архітектурою можна сформулювати кілька важливих принципів та визначень розробки програмних систем з рефлексивною адаптацією функціоналу:

– мінімальними компонентами програмної системи, які підлягають адаптації, є блоки адаптації. Кожен блок адаптації, що входить до складу системи, відповідає за можливу реалізацію якого-небудь атомарного компонента програми (функції, класу, моделі об'єкта предметної області тощо);

– сукупність логічно пов'язаних блоків адаптації, що належать до однієї предметної області і призначені для розвязання однієї задачі, називається рефлексивним компонентом;

– рефлексивна програмна система може складатися з одного або кількох рефлексивних компонентів (їх кількість буде визначати адаптивне різноманіття та рефлексивну складність програми);

– рефлексивні компоненти об'єднані в єдину систему за допомогою загального інтерфейсу і початково можуть не мати даних один про одного. Однак під час рефлексивного аналізу буде виявлятися ступінь та характер впливу одних компонентів на інші. Для цього необхідно передбачити канали інформаційного обміну даними між компонентами.

Важливо розуміти, що рефлексивний компонент – це не обов'язково програмний модуль у класичному розумінні. У деяких випадках корисно мати компоненти, реалізовані у формі сервісів у рамках концепції сервіс-орієнтованої архітектури (англ. service-oriented architecture). До суттєвих переваг сервіс-орієнтованого рефлексивного компонента належить:

– низька зв'язність. Сервіси, що входять до складу системи, можуть бути реалізовані незалежно від інших її служб, а також можуть входити до складу інших систем. Єдине вимога – це знання відповідних протоколів взаємодії;

– принципова допустимість використання функцій одного сервісу сторонніми додатками і системами. Якщо виникне потреба в модифікації певної функціональності, то достатньо буде змінити її тільки в одному сервісі, а не в кожній системі, що його використовує;

– відкриті стандарти взаємодії, які значно зменшують час підключення нового сервісу до наявної системи.

Отже, підбиваючи підсумок, можна виокремити (рис. 2) три рівні рефлексії адаптивної програмної системи на основі визначених принципів узагальненої архітектури.

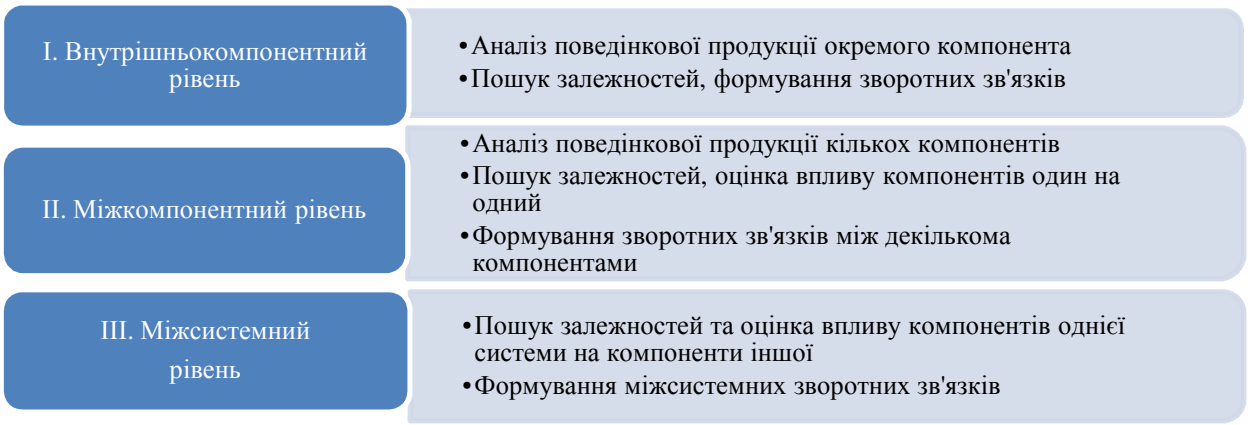

Рисунок 2 – Рівні рефлексії адаптивної програмної системи

Використання технології розроблення лінійки програмних продуктів (англ. Software Product Lines Engineering (SPLE)) для створення адаптивних програмних систем

Розроблення лінійки програмних продуктів (Software Product Lines Engineering (SPLE)) – це концепція повторного використання компонентів програмного забезпечення, яка допомагає розробляти сімейства (лінійки) продуктів зі скороченням часу виходу на ринок і підвищенням якості [16].

Центральним поняттям концепції SPLE є моделі варіативності (англ. variability model). Модель варіативності – це деякий формалізований опис множини можливих конфігурацій програмної системи, доповнений деякими обмеженнями і правилами, які стосуються питань сумісності окремих компонентів системи один з одним.

У SPLE використовуються такі типи моделей варіативності:

– *моделі характеристик* – моделі, які найчастіше використовуються на практиці. Вони графічно відображаються у вигляді модифікованого дерева І/АБО, відомого як діаграма характеристик (рис. 3), та можуть мати різні формалізовані представлення (гіперграф, математична формула, алгебраїчна нотація тощо);

– *ортогональні моделі змінності*. Схожі на моделі характеристик, вони також графічно відображаються у формі діаграм. Основна відмінність полягає в тому, що ортогональні моделі показують лише наявність змінності у програмній системі, тоді як моделі характеристик надають більш конкретний опис як предметної області, так і точок змінності;

– *моделі рішень*. Модель рішень включає такі компоненти: питання з предметної області, на які потрібно отримати відповіді під час розроблення програмного продукту; множини можливих відповідей на питання; посилання на використовувані компоненти (активи) або інші рішення; опис наслідків прийняття рішення (відповіді на певне питання або вибір певного активу).

Узагальнена архітектура адаптивної програмної системи, заснована на моделях варіативності та рівнях рефлексії, представлена на рис. 4. Адаптивна програмна система охоплює три рівні рефлексивного зворотного зв'язку: рівень окремих компонентів, рівень груп компонентів та загальносистемний рівень. Кожен контур рефлексії забезпечується своєю моделлю варіативності, що відповідає за вибір поточних конфігурацій компонентного та системного рівнів. Загальна координація міжрівневої сукупності моделей варіативності та потоків поведінкових даних здійснюється модулем аналізу поведінкової інформації. Цей модуль не завжди є підсистемою на системному рівні (як зображено на рис. 4). Можливі і більш гнучкі архітектурні варіації, коли кожен окремий компонент має свою власну підсистему поведінкового аналізу.

Формальний опис адаптивної поведінки рефлексивної програмної системи

Концепція Dynamic Software Product Lines Engineering (DSPLE) [17] розвиває ідею SPLE. Моделі варіантивності динамічних лінійок програмних продуктів (англ. Dynamic Software Product Lines (DSPL)) доцільно використовувати для формування оптимальної конфігурації програмної системи під час її виконання, тому DSPL-системи можна вважати адаптивними.

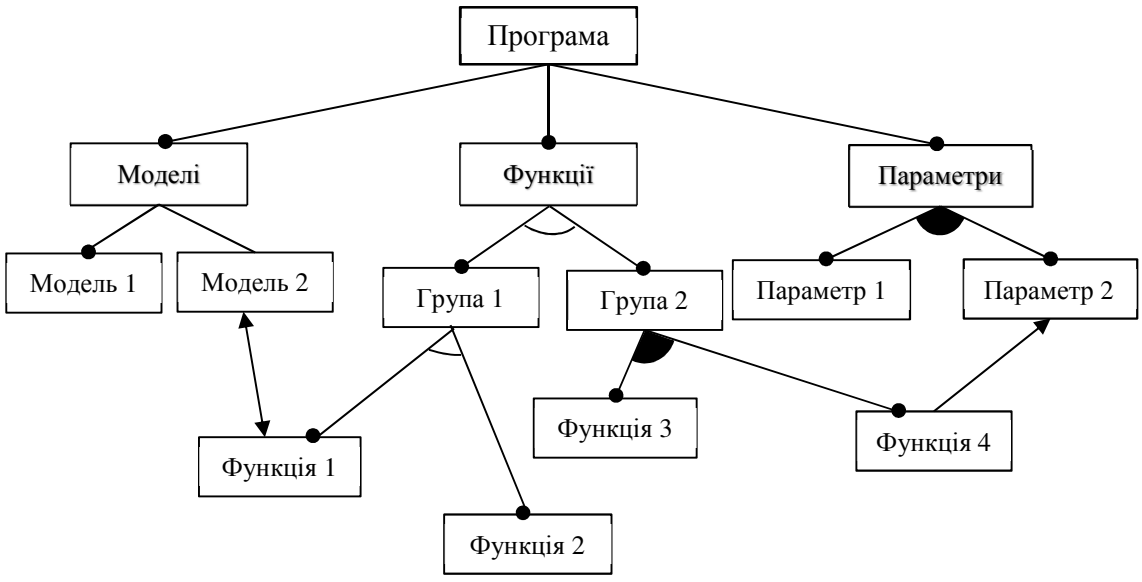

Рисунок 3 – Приклад моделі варіативності концепції SPLE

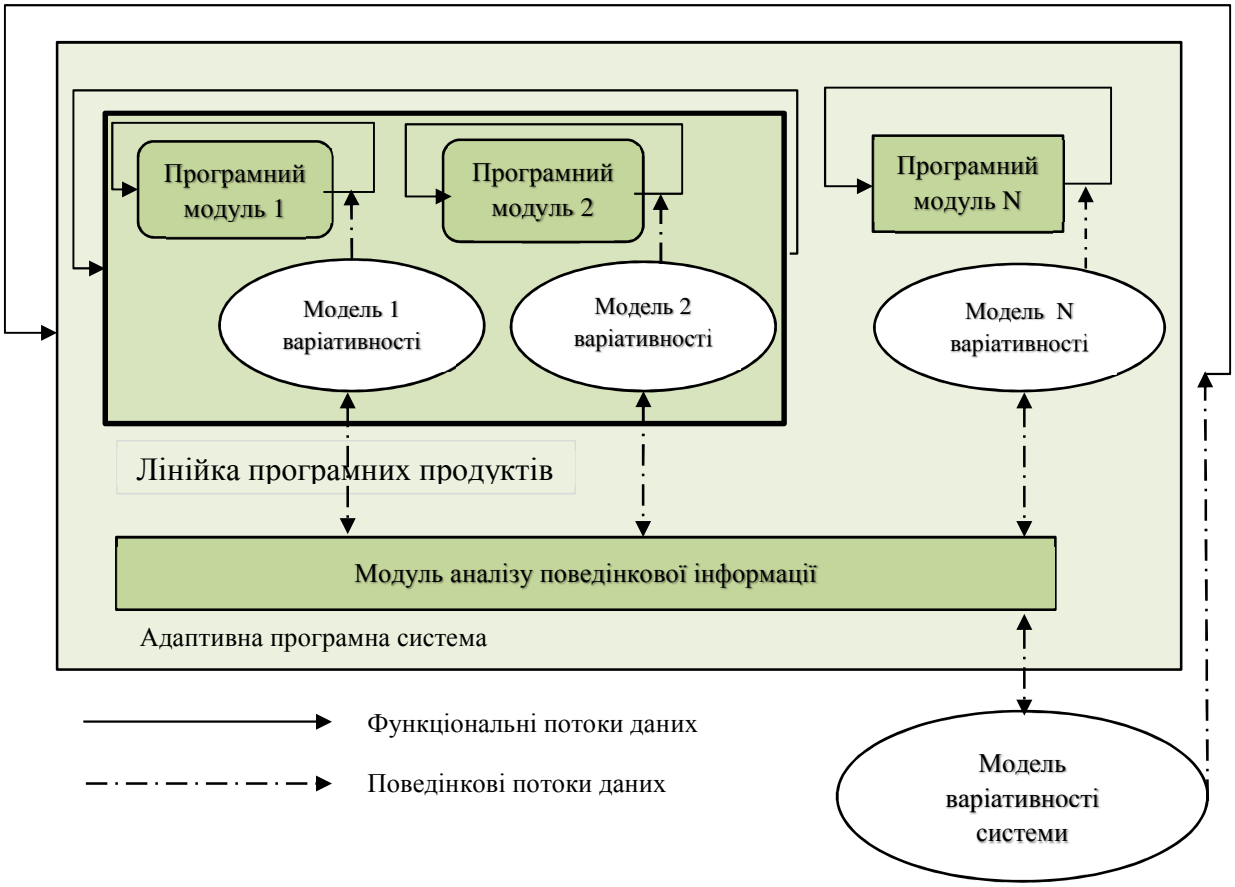

Рисунок 4 – Загальна архітектура адаптивної програмної системи, заснована на моделях варіативності та рівнях рефлексії

Одним із перспективних підходів до реалізації механізму програмної рефлексії є інтеграція принципів DSPLE і технології Data Mining. Першим кроком до створення технології синтезу адаптивних систем має стати розробка математичної моделі варіативності, яка встановлює формальні правила структурних і функціональних змін для програмного забезпечення. Для цих цілей пропонується використовувати модель характеристик (feature model), яка має форму наочної діаграми для формалізації. У таблиці представлені основні типи відношень у моделях характеристик. Кожна з характеристик служить абстракцією певного програмного компонента. Відношення між характеристиками визначають потенційний набір конфігурацій системи в цілому. Крім того, за допомогою відношень буде задаватися ступінь сумісності різних компонентів у рамках однієї лінійки програмних продуктів.

Для формального опису модель характеристик $Model$ задається у вигляді орієтованого прямого гіперграфа таким відношенням:

$$Model = (V, E, R, F), \quad (1)$$

де $V = \{V_1, V_2, \dots, V_i \dots, V_n\}$ – множина вершин графа, кожна з яких представляє характеристику діаграми; $E = \{E_1, E_2, \dots E_i \dots, E_m\}$ – множина гіперребер графа, які описують відношення моделі характеристик); $R \in V$ – вершина, яка представляє кореневу характеристику діаграми; $F: E \to M, M \subset N \times N$ – функція значення потужності

$$mv = E_i(\min, \max) \in M,$$

де M – множина значень потужності вершин гіперграфа, кожному орієнтованому гіперберу E_i. Значення потужності – це пара мінімального та максимального числа вершин основної множини гіперребра, які можуть бути включені до конфігурації діаграми характеристик, що відповідає гіперграфу.

За допомогою відношень виключаючого АБО та множинного АБО визначаються блоки варіативності – структурні одиниці програмної системи, які підлягають адаптивним процесам. Блок варіативності може використовуватись для визначення множини станів окремого компонента програмної системи, а також для правила вибору того чи іншого стану в кожному конкретному випадку.

Згідно з принципом масштабованості блок варіативності може задавати стратегії адаптивної поведінки окремого компонента сервісу, всього сервісу, або навіть всієї системи в цілому.

Таблиця – ***Основні типи відношень у моделях характеристик***

№	Позначення	Найменування	Опис
1		Відношення обов'язкової дочірньої характеристики	Якщо батьківська характеристика включена в конфігурацію діаграми, то дочірня характеристика також повинна бути включена
2		Відношення опціональної дочірньої характеристики	Включення дочірньої характеристики в конфігурацію не є обов'язковим
3		Множинне АБО	Якщо батьківська характеристика включена в конфігурацію, то також має бути включене певне число дочірних
4		Виключаюче АБО	Тільки одна з представлених дочірніх характеристик має бути наявною в кінцевій конфігурації
5	включити	Відношення включення	Включення в конфігурацію характеристики А потребує включення характеристики В
6	виключити	Відношення виключення	Включення в конфігурацію характеристики А потребує виключення характеристики В

Пропонується блок варіативності формально описати такими характеристиками:

$$Block = (Model, Params, States, Rule), \quad (2)$$

де $Model -$ формула (1) опису моделі характеристик блока варіативності; $Params = \{Param_1, Param_2, \dots, Param_i, \dots, Param_k\}$ – множина параметрів, що впливають на стан блока; $States = \{States_1, States_2, \dots, State_i, \dots, States_z\}$ – множина станів блока,

де $State_i \subseteq F \; \forall \; i, \dots, z; Rule: Param_i \rightarrow States$ – правило, яке встановлює відповідність кожному елементу $Param_i$ деякий елемент множини $States$.

Множину параметрів впливу $Params$ можна представити так:

$$BlockParams = ExtParams \cup InnerParams \cup TargetParams, \quad (3)$$

де $ExtParams = \{ExtParam_1, ExtParam_2, \dots, ExtParam_{k1},\}$ – множина зовнішніх по відношенню до розглянутого блоку параметрів;

$$InnerParams = \{InnerParam_1, InnerParam_2, \dots, InnerParam_{k2},\}$$

– множина внутрішніх, змінних параметрів блока;

$$TargetParams = \{TargetParam_1, TargetParam_2, \dots, TargetParam_{k3}\}$$

– множина цільових параметрів.

Висновки

Проведене дослідження доводить актуальність застосування та необхідність розроблення адаптивного програмного забезпечення для вирішення проблем використання ІТ у будівельній галузі. Як наукову новизну запропоновано загальну архітектуру адаптивної програмної системи, яка заснована на моделях варіативності та рівнях рефлексії. Практичне значення полягає в тому, що надання прикладній програмній системі адаптивних властивостей дасть змогу їй здійснювати модифікацію своєї структури на якісно новому рівні, що своєю чергою підвищить її надійність, стійкість до відмов, гнучкість, знизить вартість супроводу та продовжить термін експлуатації. Така система зможе розширювати клас розв'язуваних завдань протягом усього життєвого циклу, а також виконувати такі операції, які на сьогодні вважаються частиною обов'язків певних спеціалістів. Наприклад, зможе виконувати функції самоадміністрування.

Для подальших досліджень передбачається розроблення методу рефлексивної адаптаці, який полягає у формуванні функції $Rule$, що встановлює відповідність між елементами множини параметрів впливу $Params$ та елементами множини $States$ блока варіативності $Block$.

Список літератури

1. E Carvalho M. B. et al.The journey: a service-based adaptive serious game on probability // Serious games analytics / edited by C. S. Loh, Y. Sheng, D. Ifenthaler. Cham: Springer International Publishing, 2015. P. 97–106.

2. Torbeyns J., Lehitinen E., Elen J. Describing and stu dying domain-specific serious games. Cham: Springer International Publishing, 2015. 250 p.

3. Colledanchise M., Ogren P. How behavior trees modularize robustness and safety in hybrid systems // 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. Chicago. 2014. P. 53–62.

4. Berinstein P. et al. Game development tool essentials. Heidelberg: Springer Berlin Heidelberg, 2014. 467 p.

5. Park J. S. Essence-based, goal-driven adaptive software engineering // EEE/ACM 4th SEMAT Workshop on General Theory of Software Engineering (GTSE). Washington: IEEE Computer Society, 2015. P. 33–38.

6. Choi, T., Chan, H., Yue, X. Recent development in Big Data Analytics for business operations and risk management // IEEE Transactions on Cybernetics. Washington: IEEE Computer Society, 2016. P. 1–12.

7. Losif – Lazar A. F., Schaef I., Wasowski A. A Core Language for Separate Variability Modeling // Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change / edited by T. Margaria, B. Steffen. Heidelberg: Springer Berlin Heidelberg, 2014. P. 257–272.

8. Гончаренко, Т. А. Кластерний метод формування метаданих багатовимірних інформаційних систем для розв'язання задач генерального планування. *Управління розвитком складних систем*. Київ, 2020. № 42. С. 93 – 101, dx.doi.org\10.32347/2412–9933.2020.42.93–101.

9. Chernyshev D., Dolhopolov S., Honcharenko T., Sapaiev V. and Delembovskyi M. "Digital Object Detection of Construction Site Based on Building Information Modeling and Artificial Intelligence Systems", CEUR Workshop Proceedings, 2022, 3039, стр. 267–279, 1st International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2022, http://ceur–ws.org/Vol–3039/paper16.pdf

10. Гончаренко, Т. А. Інтеграційна модель життєвого циклу території будівлі на основі BIM. *Управління розвитком складних систем*. Київ, 2020. № 43. С. 83 – 90, dx.doi.org\10.32347/2412–9933.2020.43.83–90.

11. Oti – Sarpong K., Pärn E. A., Burgess G., Zaki M. Transforming the construction sector: An institutional complexity perspective. *Constr. Innov*. 2021, 22, 361–387

12. Гончаренко Т. А. Верифікація інформаційних моделей об'єктів будівництва. *Управління розвитком складних систем.* Київ, 2019. № 39. С. 69 – 74; dx.doi.org\10.6084/m9.figshare.11340656.

13. Arefazar Y., Nazari A., Hafezi M. R., Hossain Maghool S. A. Prioritizing agile project management strategies as a change management tool in construction projects, *International Journal of Construction Management*, 2019, DOI: 10.1080/15623599.2019.1644757

14. Smith K., Sepasgozar S., Governance, Standards and Regulation: What Construction and Mining Need to Commit to Industry 4.0. *Buildings,* 2022, 12, 1064

15. R. Akselrod, A. Shpakov, G. Ryzhakova, T. Honcharenko, I. Chupryna, H. Shpakova. «Integration of Data Flows of the Construction Project Life Cycle to Create a Digital Enterprise Based on Building Information Modeling», International Journal of Emerging Technology and Advanced Engineering, 2022, 1, pp. 40–50.

16. Honcharenko, K. Kyivska, O. Serpinska, V. Savenko, D. Kysliuk and Y. Orlyk. «Digital transformation of the construction design based on the Building Information Modeling and Internet of Things», CEUR Workshop Proceedings, 2021, 3039, стр. 267–279, 1st International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2021 http://ceur–ws.org/Vol–3039/paper16.pdf

17. Younis O., Ghoul S., Alomari M. H. Systems variability modeling: a textual model mixing class and feature concepts. *International Journal of Computer Science & Information Technology*. 2013. N. 5. P. 127–139.

Стаття надійшла до редколегії 20.05.2023

Honcharenko Tetiana
PhD (Eng.), Assistant Professor of the department of information technologies of design and applied mathematics,
https://orcid.org/0000-0003-2577-6916
Kyiv National University of Construction and Architecture, Kyiv

SOFTWARE SYSTEM ARCHITECTURE BASED ON THE CONCEPT OF REFLEXIVE ADAPTATION

Abstract. *The conducted research substantiates the relevance of the application and the need to develop adaptive software to solve the problems of using IT in the construction industry. The basic principles of reflexive adaptation, as well as the use of software product lines engineering (SPLE) technology to create adaptive software systems, are considered. A generalized architecture of an adaptive software system based on models of variability and levels of reflection is proposed. A formal description of the reflexive software system for the implementation of variants of the adaptive behavior of application programs has been carried out. The practical significance of the research is that giving the applied software system adaptive properties will enable it*

to modify its structure at a qualitatively new level, which in turn will increase its reliability, resistance to failures, flexibility, reduce the cost of maintenance and extend the life of the system. Such a system will be able to expand the class of solvable tasks throughout the entire life cycle, as well as perform operations that are currently considered part of the duties of certain specialists, for example, it will be able to perform self-administration functions.

Keywords: adaptive software systems; Software Product Lines Engineering; SPLE; the concept of reflexive adaptation; variability model; technological line of software products; Software Engineering

References

1. Carvalho, M. B. et al.(2015). The journey: a service-based adaptive serious game on probability. Serious games analytics / edited by C. S. Loh, Y. Sheng, D. Ifenthaler. Cham: Springer International Publishing, 97–106.

2. Torbeyns, J., Lehitinen, E., Elen, J. (2015). Describing and stu dying domain-specific serious games. Cham: Springer International Publishing, 250.

3. Colledanchise, M., Ogren, P. (2014). How behavior trees modularize robustness and safety in hybrid systems.Procc. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. Chicago, Pp. 53–62.

4. Berinstein, P. et al. (2014). Game development tool essentials. Heidelberg: Springer Berlin Heidelberg, 467.

5. Park, J. S. (2015). Essence-based, goal-driven adaptive software engineering. Procc. EEE/ACM 4th SEMAT Workshop on General Theory of Software Engineering (GTSE). Washington: IEEE Computer Society, Pp. 33–38.

6. Choi, T., Chan, H., Yue, X. (2016). Recent development in Big Data Analytics for business operations and risk management. Procc. IEEE Transactions on Cybernetics. Washington: IEEE Computer Society, Pp. 1–12.

7. Losif–Lazar, A. F., Schaef, I., Wasowski, A. (2014). A Core Language for Separate Variability Modeling. Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change / edited by T. Margaria, B. Steffen. Heidelberg: Springer Berlin Heidelberg, 257–272.

8. Honcharenko, Tetyana. (2020). Cluster method of forming metadata of multidimensional information systems for solving general planning problems. *Management of Development of Complex Systems*, 42, 93–101. dx.doi.org\10.32347/2412-9933.2020.42.93-101.

9. Chernyshev, D., Dolhopolov, S., Honcharenko, T., Sapaiev, V. and Delembovskyi, M. (2022). Digital Object Detection of Construction Site Based on Building Information Modeling and Artificial Intelligence Systems. Procc. CEUR Workshop Proceedings, 2022, 3039, стр. 267–279, *1st International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP* 2022, http://ceur–ws.org/Vol–3039/paper16.pdf

10. Honcharenko, Tetyana. (2020). Integration model of the life cycle of the building area based on BIM. *Management of Development of Complex Systems*, 43,83–90. dx.doi.org\10.32347/2412-9933.2020.43.83-90.

11. Oti–Sarpong, K., Pärn, E. A., Burgess, G., Zaki, M. (2021). Transforming the construction sector: An institutional complexity perspective. *Constr. Innov*., 22, 361–387

12. Honcharenko, Tetyana. (2019). Structural analysis of the territory for construction as a complex spatially distributed system. *Management of Development of Complex Systems*, 39, 69–74; dx.doi.org\10.6084/m9.figshare.11340656.

13. Arefazar, Y., Nazari, A., Hafezi, M. R., Hossain, Maghool S. (2019). Prioritizing agile project management strategies as a change management tool in construction projects. *International Journal of Construction Management*. DOI: 10.1080/15623599.2019.1644757

14. Smith, K., Sepasgozar, S. (2022). Governance, Standards and Regulation: What Construction and Mining Need to Commit to Industry 4.0. *Buildings,* 12, 1064

15. Akselrod, R., Shpakov, A., Ryzhakova, G., Honcharenko, T., Chupryna, I., Shpakova, H. (2022). Integration of Data Flows of the Construction Project Life Cycle to Create a Digital Enterprise Based on Building Information Modeling. *International Journal of Emerging Technology and Advanced Engineering,* 1, 40–50.

16. Honcharenko, T, Kyivska, K., Serpinska, O., Savenko, V., Kysliuk D. and Orlyk, Y. (2021). Digital transformation of the construction design based on the Building Information Modeling and Internet of Things. CEUR Workshop Proceedings, 2021, 3039, 267–279, *1st International Workshop on Information Technologies: Theoretical and Applied Problems,* ITTAP 2021 http://ceur–ws.org/Vol–3039/paper16.pdf

17. Younis, O., Ghoul, S., Alomari, M. H. (2013). Systems variability modeling: a textual model mixing class and feature concepts. *International Journal of Computer Science & Information Technology,* 5, 127–139.

Посилання на публікацію

APA Honcharenko, Tetyana. (2023). Software system architecture based on the concept of reflexive adaptation. *Management of Development of Complex Systems*, 54, 69–76, dx.doi.org\10.32347/2412-9933.2023.54.69-76.

ДСТУ Гончаренко Т. А. Архітектура програмної системи на основі концепції рефлексивної адаптації. *Управління розвитком складних систем.* Київ, 2023. № 54. С. 69 – 76, dx.doi.org\10.32347/2412-9933.2023.54.69-76.