Issue 6

https://doi.org/10.23939/sisn2019.02.030

УДК 004.415

СИСТЕМА ІДЕНТИФІКАЦІЇ ПРОБЛЕМНИХ СИТУАЦІЙ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Є. В. Буров[1], Х. І. Микіч[2], О. М. Верес[3], В. В. Литвин[4]

[1–4] Національний університет "Львівська політехніка"
[1] Yevhen.V.Burov@lpnu.ua, ORCID: 0000-0001-8653-1520
[2] Khrystyna.I.Mykich@lpnu.ua, ORCID: 0000-0002-4324-2080
[3] Oleh.M.Veres@lpnu.ua, ORCID: 0000-0001-9149-4752
[4] Vasyl.V.Lytvyn@lpnu.ua, ORCID: 0000-0002-9676-0180

Досліджено та розроблено методи та засоби ідентифікації проблемних ситуацій на базі онтологій із використанням механізмів логічного виведення, які застосовано в інтелектуальних системах підтримки прийняття рішень для завдань тестування програмного забезпечення.

Розглянуто актуальну проблему тестування програмного забезпечення із використанням онтологічного моделювання для своєчасного виявлення помилок та поліпшення якості розроблюваного програмного продукту.

Використання онтологічного моделювання для подання та ідентифікації ситуацій створює додаткові можливості для розв'язання задачі ідентифікації та обмеження. Перевагою є здатність застосування логічного виведення та використання аксіом під час міркувань про ситуації. Це забезпечує перспективу розроблення методів ідентифікації ситуацій, що ґрунтуються на логічному виведенні на основі інформації про поточний стан предметної області та знань про цю область.

Використана модель завдань дає змогу не лише автоматизувати виконання деяких простих завдань, але й на основі наявних знань про ситуації здійснювати логічне міркування у системах тестування.

Онтологічне подання знань про предметну область дало змогу формалізувати знання про проблемні ситуації, що виникають у проєкті, а застосування розроблених методів ідентифікації ситуацій у системі забезпечило вчасне виявлення загрозливих ситуацій та формування рекомендацій щодо їх уникнення. Всі ці фактори сприяють поліпшенню якості програмного продукту під час його розроблення.

У роботі подано онтологію галузі тестування програмного забезпечення, а також наведено алгоритм роботи системи та здійснено моделювання на базі UML.

Розроблено архітектуру системи ідентифікації ситуацій та програмний комплекс для аналізу і моделювання проблемних ситуацій на прикладі систем підтримки прийняття рішень галузі тестування, центральним компонентом яких є інструментальний засіб для онтологічного моделювання – Protégé.

Для розширення функціональних можливостей редактора Protégé використано два плагіни, за допомогою яких здійснено моделювання за допомогою мов SWRL, SQWRL.

Результати роботи доцільно використовувати для розв'язування задач виявлення критичних ситуацій під час розроблення та тестування програмного забезпечення, повторного використання інформації в базах знань організацій з розроблення програмного забезпечення, що поліпшить якість створюваного програмного забезпечення.

Ключові слова: онтологічне моделювання, проблемна ситуація, тестування програмного забезпечення, логічне міркування.

Вступ

Стрімкий розвиток галузі програмного забезпечення та інформаційних технологій, створення нових та вдосконалення наявних технологій щодо побудови програмних систем, розширення сфери застосування та використання автономних систем підтримки прийняття рішень у сучасному світі зумовлюють паралельний розвиток усіх частин процесу побудови та впровадження програмного забезпечення. Підвищена складність та багатокомпонентність сучасних програмних систем вимагають спеціалізованого підходу до їх створення. Сьогодні основною вимогою до автоматизованих систем є досягнення високого рівня надійності, що дає змогу зробити їх ефективним інструментом у світі сучасних технологій.

Постановка проблеми

Галузь тестування програмного забезпечення безпосередньо пов'язана із його проблемами, адже вартість адміністрування програмного продукту і розмір збитків, пов'язаних із неякісним програмним забезпеченням, у рази перевищує вартість програмного забезпечення. Тобто сьогодні тестування програмного забезпечення є одним із найвитратніших етапів життєвого циклу продукту, оскільки становить значну частину від загальних витрат, виділених на розроблення програмного забезпечення. Тестування ґрунтується на вимогах до програмного забезпечення: саме із вимог одержують тестові дані та тестові приклади, що є результатом аналізу предметної області.

У роботі наведено методи ідентифікації проблемних ситуацій під час онтологічного моделювання знань з метою поліпшення якості процесу тестування програмного забезпечення за рахунок ідентифікації та вчасного реагування на ситуації, що виникають під час тестування.

Онтологічне подання знань про предметну область дає змогу формалізувати знання про ситуації в проєкті, а застосування методів ідентифікації ситуацій забезпечить вчасне виявлення загрозливих ситуацій та формування рекомендацій щодо їх усунення, що сприятиме поліпшенню якості як процесу розроблення, так і самого програмного продукту.

Аналіз останніх досліджень та публікацій

Тестування програмного забезпечення – це процес технічного дослідження, призначений для виявлення інформації щодо якості продукту відносно певного контексту, в якому його будуть використовувати.

Будь-який дефект розробленої системи може спричинити серйозні наслідки та втрати, що недопустимо у сучасному світі конкуренції [1]. Тому саме тестування програмного продукту є одним із методів забезпечення якості цього продукту [2, 3].

Жодна сфера діяльності в сучасному світі не може обійтися без комп'ютерів з різноманітним програмним забезпеченням. Комп'ютери та мобільні пристрої увійшли в життя кожної сучасної людини. Програмне забезпечення стрімко розвивається з кожним днем, зростають можливості новоствореного програмного забезпечення. Разом з цим, стрімко підвищуються і вимоги користувачів до якості розробленого програмного забезпечення. Це демонструють світові лідери IT-індустрії, такі як Microsoft, Apple, Google тощо. У цих компаніях для забезпечення високої якості програмного забезпечення вже працює більше тестувальників, ніж самих розробників програмного продукту.

У сучасних умовах стрімкого розвитку IT ринку тенденція забезпечення якості тільки посилюватиметься. Тестування програмного забезпечення є важливою складовою розроблення програмного забезпечення, оскільки спрямоване на перевірку створюваного програмного продукту і якнайшвидше виявлення помилок, що виникають під час його розроблення. Галузь тестування програмного забезпечення безпосередньо пов'язана із його проблемами, адже вартість

адміністрування самого програмного продукту і розмір збитків, спричинених неякісним програмним забезпеченням, у рази перевищує саму вартість програмного забезпечення [1].

Отже, сьогодні тестування програмного забезпечення є одним із найвитратніших етапів життєвого циклу продукту, оскільки близько 50–65 % загальних витрат від вартості програмного забезпечення становить його тестування [1].

Перспективним напрямом у вирішенні проблеми щодо якісного тестування програмної системи є використання підходів інженерії знань, зокрема онтологічного моделювання. На відміну від модельних підходів, у підході онтологічного моделювання будують формальну модель предметної області (онтологію), з урахуванням її особливостей та обмежень. Така модель може бути повторно використана для побудови інших програмних систем для цієї ж предметної області, а також дасть змогу повторно використати отримані знання.

Основна мета розробленої у роботі програмної системи полягає у покращенні тестування програмного забезпечення з використанням онтологічного моделювання для подання предметної області, що дає змогу в межах єдиної системи поєднувати різні методи ідентифікації ситуацій. Основна її перевага – гнучкість, оскільки поєднання різних методів у межах єдиної системи дає змогу розглядати різні типи ситуацій, що виникають у середовищі.

Формулювання цілі статті

У статті запропоновано нові методи ідентифікації проблемних ситуацій під час онтологічного моделювання знань з метою поліпшення якості тестування шляхом ідентифікації та вчасного реагування на ситуації, що виникають під час тестування. Онтологічне представлення знань про предметну область дає змогу точніше визначити вимоги до програмного забезпечення, покращити їх якість і, отже, забезпечити зростання якості програмного забезпечення [4–7]. Використання онтологічного моделювання для подання та ідентифікації ситуацій також створює додаткові можливості для вирішення завдання ідентифікації й обмеження. Перевагою є здатність застосування логічного виведення та використання аксіом під час міркувань про ситуації [8]. Це відкриває перспективу розроблення методів ідентифікації ситуацій, що ґрунтуються на логічному виведенні на основі інформації про поточний стан предметної області та знань про цю область.

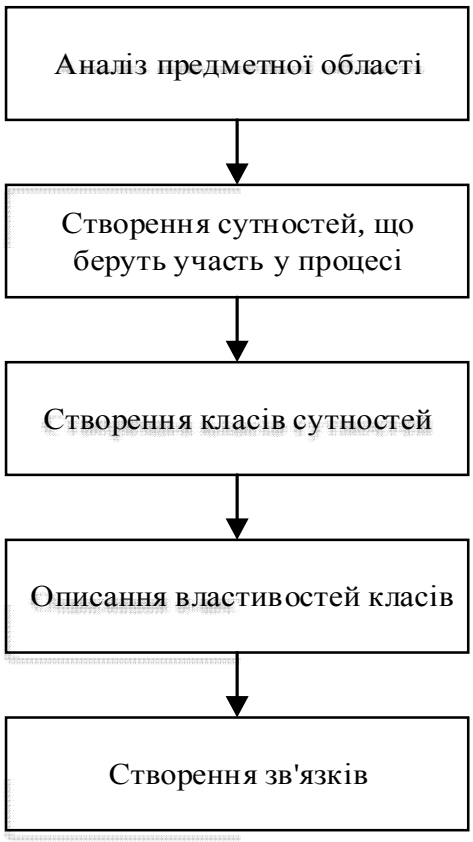

Рис. 1. Процес створення онтології предметної області

Виклад основного матеріалу

Онтологія предметної області використовується для збереження сутностей та відношень галузі тестування програмного забезпечення та для побудови правил щодо виявлення ситуацій. Онтологія містить модель домена, подану як таксономія класів [9]. Це уможливлює однозначну інтерпретацію усіх об’єктів предметної області з інформаційної бази, визначення для них загальних атрибутів та властивостей. Система реагує на певний діапазон подій (ситуацій, що виникають) у зовнішньому світі, ідентифікує їх та створює нові або змінює відомі факти про неї (здійснює логічне міркування). Процес створення онтології розпочинається з аналізу предметної області та складається із певних етапів [10], які подано на рис.1.

Для створення онтології предметної області використано редактор онтологій Protégé [10]. Отже, після аналізу предметної області важливим є виділення основних концептів (класів) та створення ієрархії.

У результаті створено онтологію предметної області галузі тестування (рис. 2).

Ця онтологія містить такі сутності:

· Employee – у цій сутності зберігається інформація про працівників фірми;

· Information (database) – у сутності зберігається інформація щодо відповідних баз даних;

· Defect tracking database (база даних відстеження помилок) – це база даних, що містить інформацію про виявлені помилки (дефекти);

· Product modules database (база даних модулів продукту) – інформація про модулі системи, що тестується;

· Information (document)

· Bug report (баг репорт) – технічний документ, який містить повний опис дефекту;

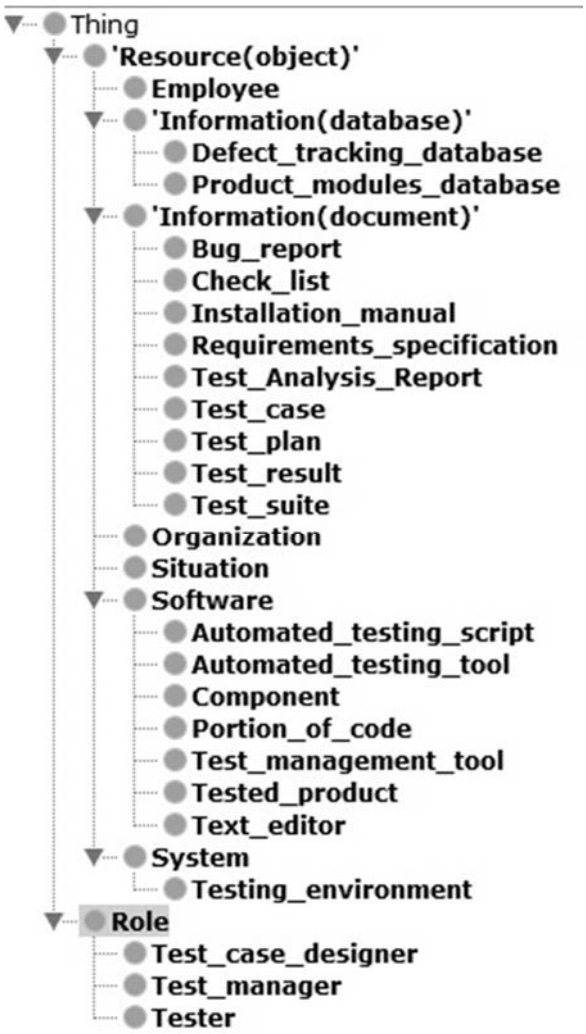

Рис. 2. Онтологія предметної області (тестування програмного забезпечення)

· Check list (контрольний список) – документ, який описує, що необхідно протестувати;

· Installation manual – інструкція щодо того, як інсталювати продукт;

· Requirements specification (специфікація вимог) – спеціальний документ, що містить інформацію про те, як повинна поводитись система, які функції виконувати, яке навантаження витримувати тощо;

· Test Analysis Report – документ, що містить аналіз отриманих результатів;

· Test case (тестовий випадок) – містить інформацію (сукупність кроків, конкретних умов і параметрів), необхідну для перевірки реалізації продукту, що тестується;

· Test plan (тест план) – це певний документ-стратегія, який складає керівник проєкту та який містить весь обсяг роботи з тестування програмного забезпечення;

· Test result – міститься інформація про результат тестування тест-кейса;

· Test suite – це набір тест-кейсів;

· Organization;

· Situation – використовується для збереження ситуацій, які ідентифіковано;

· Software;

§ Automated testing script – це набір інструкцій для автоматичної перевірки певної частини програмного забезпечення;

§ Automated testing tools – програмне забезпечення, за допомогою якого фахівець з автоматизованого тестування здійснює створення, налагодження, виконання та аналіз результатів прогону тест-скриптів;

§ Component – комплекс компонентів (модулів), з яких складається система;

§ Portion of code – використовується для посилання на фрагмент, що містить помилку;

§ Test management tools – це інструменти керування тестуванням програмного забезпечення;

§ Tested product – певний програмний комплекс, який тестується;

§ Text editor – це певні текстові редактори, які використовують для збереження різної інформації, наприклад тестового плану тощо;

§ System;

· Testing environment – середовище, в якому здійснюватиметься перевірка коректної роботи продукту;

· Role;

· Test case designer – створює тест-кейси;

· Test manager (керівник проєкту) – складає тест-план і визначає, що і як тестувати;

· Tester – тестує тест-кейси.

Система ідентифікації ситуацій використовує онтологічне моделювання предметної області та на його основі, за допомогою логічних правил, робить висновки про поточні ситуації. Для створення онтології предметної області використано відомий редактор онтологій Protégé. Protégé – це інструмент для конструювання онтології певної галузі, а також за його допомогою можна створювати логічне міркування [4, 11–13].

Знання про ситуації зберігаються у формі ситуаційних моделей, а сутність “ситуація” є частиною онтології. Формально модель ситуації подається кортежем, який містить сутності Ts та відношення Rls, релевантні для ситуації, умови або інформацію для ідентифікації ситуації ds та множину дій Acs, які треба виконати, якщо ситуація буде виявлена:

$$Tmds = (Ts, Rls, Ids, Acs). \quad (1)$$

Умови ідентифікації ситуації використовують значення фактів, типи яких подано у моделі ситуації.

Розв’язання задачі ідентифікації ситуації зводиться до розпізнавання у базі фактів умов та залежностей, специфікованих у ситуаційній моделі. У цьому плані ця задача подібна до задачі розпізнавання образів.

База фактів BF є множиною окремих фактів: $BF = \{bf\}$ із онтології On:

$$BF = \{bf\ i | \forall i(bf\ i \in T \vee bf\ i \in Rl)\}. \quad (2)$$

Множина моделей ситуацій, в якій кожен елемент належить до типу ситуації Ts з онтології *On*:

$$MdSi = \{mds_{ij} | \forall j(mds_{ij} \in Ts)\}. \quad (3)$$

Для розв'язання задачі ідентифікації ситуації побудувано таку функцію Fid відображення стану бази фактів у множину моделей ситуацій, що мінімізує неточність розпізнавання.

$$Fid{:}BF \rightarrow MdSi. \quad (4)$$

Функцію втрат побудували, порівнюючи результат виконання функції ідентифікації із результатом деякої ідеально точної функції ідентифікації Fzr над тією самою базою фактів BF. Отже, функція втрат, що визначає неточність ідентифікації, має вигляд:

$$Flos = min\Delta(Fzr(BF), Fid(BF)). \quad (5)$$

Функцію відображення побудовано як функцію відстані (подібності) поточного стану до моделі:

$$D(BF, mds_{ij}). \quad (6)$$

Ідентифікувати ситуацію − означає знайти модель $mdsi_{min}$, для якої відстань між поточним станом BF мінімальна:

$$mdsi_{min} = Minj\ (D(BF, mds_{ij})). \quad (7)$$

Архітектура розробленої системи дає змогу краще зрозуміти, як відбувається ідентифікація ситуацій у системі та як взаємодіють у ній використані програмні модулі та плагіни (рис. 3). Центральним компонентом архітектури системи є інструментальний засіб для онтологічного моделювання – Protégé. Цей програмний засіб підтримується мовою програмування Java та використовує інтерфейс прикладного програмування (API – Application Programming Interface), який дає змогу розробникам вбудовувати компоненти, що доповнюють та змінюють функціональні можливості редактора. Також за користувацький інтерфейс та його додаткові можливості відповідає компонента Protégé GUI. Програмний засіб Protege містить онтології, що були створені, а також OWL файли (OWL – Web Ontology Language – мова опису Web-онтологій).

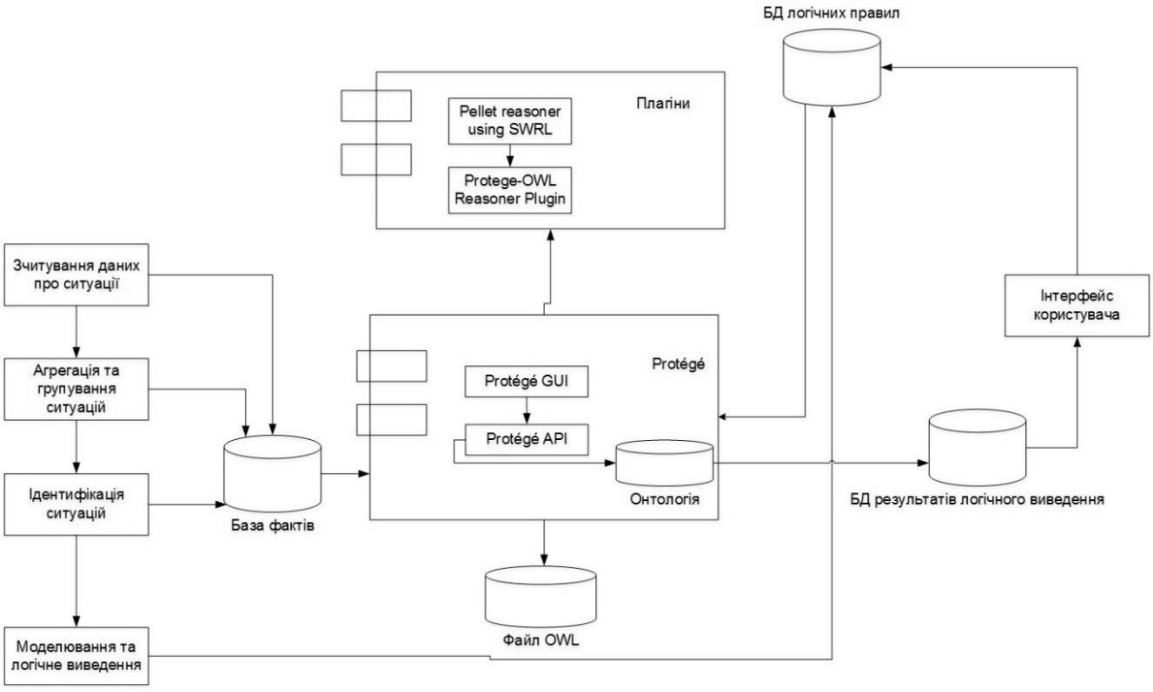

Рис. 3. Архітектура системи ідентифікації ситуацій

Система містить базу фактів, у якій зберігаються:

· дані про ситуацію, яку потрібно ідентифікувати;

· агрегацію та групування. Оскільки, як зазначено вище, у реальному масштабі часу існують різні типи ситуацій, система після ідентифікації поточної ситуації повинна зарахувати її до певного класу ситуацій;

· ідентифікації ситуацій. База даних повинна зберігати ситуації, які відбувались, щоб повторно їх використовувати та уникнути помилок в майбутньому.

Також у системі наявні бази даних:

· логічних правил: це база даних, що містить логічні правила для моделювання ситуацій;

· результатів логічного виведення. Ця база даних зберігає результати здійснення логічних міркувань над ситуаціями.

Робота системи починається з того, що наявна певна ситуація, яка виникає під час тестування програмного забезпечення. Система зчитує цю ситуацію та зараховує до певного класу. Після цього, за допомогою логічних правил, система робить певний висновок про те, чи існує така ситуація. Всі дані вносять в базу даних.

Блок-схему алгоритму роботи системи подано на рис. 4.

Розроблення будь-якого програмного забезпечення починається із аналізу вимог до його функціональності [14–17]. Під час такого аналізу виявляють користувачів розроблюваного програмного забезпечення і перелік основних аспектів його поведінки в процесі взаємодії із користувачами та системою.

Рис. 4. Блок-схема алгоритму роботи системи

Для створення програмної системи ідентифікації ситуацій використано метод моделювання на основі UML: подано діаграму використання (use case) розробленої системи, корисну для визначення переліку можливостей, які повинна мати проєктована система (рис. 5).

Як показано на рис. 5, акторами є:

· експерт, що аналізує предметну область;

· програма ідентифікації ситуацій.

Прецедентами (варіантами використання) для експерта є:

· аналіз предметної області – експерт аналізує дані предметної області;

· створення структури контенту, тобто ієрархічної структури предметної області, та встановлення зв'язків між ними;

· побудова онтології – на підставі наявних даних експерт створює онтологію за допомогою редактора онтологій Protégé.

Прецедентами (варіантами використання) для програми ідентифікації ситуацій є:

· отримання даних із середовища – система одержує дані, які визначив експерт;

· подання даних за допомогою правил – у прийнятному для онтологічного моделювання вигляді, тобто за допомогою правил, що дасть змогу здійснювати логічне міркування;

· здійснення логічного міркування – виконано логічне міркування із використанням SWRl;

· ідентифікація ситуацій – за допомогою міркування система ідентифікує поточну ситуацію та зараховує до певного класу;

· запис інформації про ситуацію у базу даних – вся інформація про ситуації зберігається в базі даних, що дає змогу повторно використовувати дані.

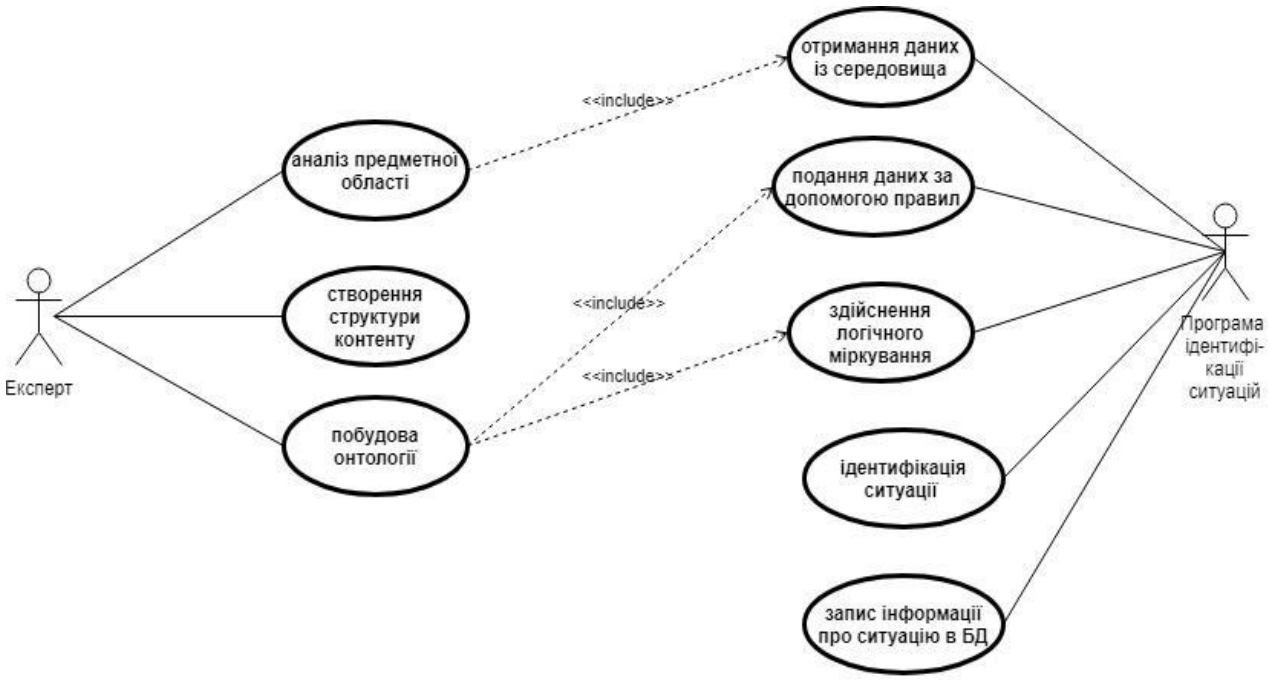

Рис. 5. Діаграма використання системи ідентифікації ситуацій

Для створення онтології предметної області використано формальну мову OWL, яка є розширенням RDF/RDFS. Формальною основою мови опису OWL-онтологій є дескриптивна логіка, яка дає змогу здійснювати логічне виведення [18, 19].

Висновки

Сьогодні тестування програмного забезпечення є одним із найважливіших засобів перевірки його надійності. Проте ручне тестування є працемістким процесом, неефективним без автоматизації, потребує багато часу та ресурсів, а також не дає змоги виявити усі недоліки створюваного програмного продукту. Важливим завданням під час розроблення програмного забезпечення є автоматизація процесу тестування, ефективна побудова сценаріїв тестування, що дасть змогу мінімізувати часові та економічні затратити.

Отже, тестування програмного забезпечення – складний та один із найважливіших етапів його розроблення. А оцінювання ефективність роботи таких програмних систем є необхідною частиною забезпечення якості програмного забезпечення. Відсутність ефективних методів оцінювання негативно впливає на якість розроблюваного програмного забезпечення. Запропонований у роботі прототип програмної системи дає змогу запобігти виникненню критичних ситуацій у проєкті та підвищити ефективність як процесу тестування, так і самого розроблення програмного забезпечення.

Список літератури

1. RTI Study Finds. (n.d.). Software Bugs Cost U. S. Economy $59.6 Billion Annually. Retrieved from http://www.nist.gov/director/prog-ofc/report02-3.pdf.

2. Руда, О. А., & Моденов, Ю. Б. (2013). Методи та моделі тестування програмного забезпечення. *Проблеми інформатизації та управління*, 2(42), 93–98.

3. Liu, Y., Wu, J., Liu, X., & Gu, G. (2009, July). Investigation of knowledge management methods in software testing process. *International Conference on Information Technology and Computer Science* , 2, 90–94.

4. Литвин, В. В. (2009). Мультиагентні системи підтримки прийняття рішень, що базуються на прецедентах та використовують адаптивні онтології. *Радіоелектроніка, інформатика, управління*, 2 (21), 120–126.

5. Guo, S., Zhang, J., Tong, W., & Liu, Z. (2011, August). An application of ontology to test case reuse. *International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, 775–778.

6. Botzenhardt, A., Maedche, A., & Wiesner, J. (2011, August). Developing a domain ontology for software product management. *Fifth International Workshop on Software Product Management (IWSPM)*, 7–16.

7. OWL Web Ontology Language. (n.d.). Retrieved from https://www.w3.org/TR/owl-guide/

8. Kitchenham, B. A., Travassos, G. H., Von Mayrhauser, A., Niessink, F., Schneidewind, N. F., Singer, J., ... & Yang, H. (1999). Towards an ontology of software maintenance. *Journal of Software Maintenance: Research and Practice*, 11(6), 365–389.

9. Ikeda, M., Seta, K., Kakusho, O., & Mizoguchi, R. (1998). Task ontology: Ontology for building conceptual problem solving models. *Environment*, 126–133.

10. Raubal, M., & Kuhn, W. (2004). Ontology-based task simulation. *Spatial Cognition and Computation*, 4(1), 15–37.

11. Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *International journal of human-computer studies*, 43(5–6), 625–640.

12. Johnson, P., Johnson, H., Waddington, R., & Shouls, A. (1988, October). Task-related knowledge structures: analysis, modelling and application. *In BCS HCI* , 35–62.

13. Буров, Є. В. (2009). Опрацювання знань у когнітивній інформаційній системі, керованій моделями. *Східно-Європейський журнал передових технологій*, 6(7 (42)), 40–49.

14. Dargie, W., Mendez, J., Möbius, C., Rybina, K., Thost, V., & Turhan, A. Y. (2013, March). Situation recognition for service management systems using OWL 2 reasoners. *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 31–36.

15. Chen, P. P., & Wong, L. Y. (Eds.). (2007). *Active conceptual modeling of learning*. Springer Verlag.

16. Фаулер, М., & Скотт, К. (1999). *UML в кратком изложении*. М.: Мир.

17. Буч, Г. (2006). *Язык UML. Руководство пользователя*. 2-е изд.: пер. с англ. Мухин Н. М.: ДМК Пресс.

18. O'Connor, M. J., & Das, A. K. (2009, October). SQWRL: A Query Language for OWL. *Proceedings of the 6th International Conference on OWL: Experiences and Directions (OWLED'09)*, 529, 208–215.

19. Fudholi, D. H., Maneerat, N., Varakulsiripunth, R., & Kato, Y. (2009, January). Application of Protégé, SWRL and SQWRL in fuzzy ontology-based menu recommendation. *International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2009)*, 631–634.

References

1. RTI Study Finds. (n.d.). Software Bugs Cost U. S. Economy $59.6 Billion Annually. Retrieved from http://www.nist.gov/director/prog-ofc/report02-3.pdf.

2. Ruda, O. A., & Modenov, Y. B. (2013). Software Testing Methods and Models. *Problems of informatization and management*, 2(42), 93–98.

3. Liu, Y., Wu, J., Liu, X., & Gu, G. (2009, July). Investigation of knowledge management methods in software testing process. *International Conference on Information Technology and Computer Science* , 2, 90–94.

4. Lytvyn, V. V. (2009). Multi-agent case-based decision support systems that use adaptive ontologies. *Radio electronics, computer science, management*, 2 (21), 120–126.

5. Guo, S., Zhang, J., Tong, W., & Liu, Z. (2011, August). An application of ontology to test case reuse. *International Conference on Mechatronic Science, Electric Engineering and Computer* (*MEC*), 775–778.

6. Botzenhardt, A., Maedche, A., & Wiesner, J. (2011, August). Developing a domain ontology for software product management. *Fifth International Workshop on Software Product Management* (*IWSPM*), 7–16.

7. OWL Web Ontology Language. (n.d.). Retrieved from https://www.w3.org/TR/owl-guide/

8. Kitchenham, B. A., Travassos, G. H., Von Mayrhauser, A., Niessink, F., Schneidewind, N. F., Singer, J., ... & Yang, H. (1999). Towards an ontology of software maintenance. *Journal of Software Maintenance: Research and Practice*, 11(6), 365–389.

9. Ikeda, M., Seta, K., Kakusho, O., & Mizoguchi, R. (1998). Task ontology: Ontology for building conceptual problem solving models. *Environment*, 126–133.

10. Raubal, M., & Kuhn, W. (2004). Ontology-based task simulation. *Spatial Cognition and Computation*, 4(1), 15–37.

11. Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *International journal of human-computer studies*, 43(5–6), 625–640.

12. Johnson, P., Johnson, H., Waddington, R., & Shouls, A. (1988, October). Task-related knowledge structures: analysis, modelling and application. *In BCS HCI* , 35–62.

13. Burov, EV (2009). Knowledge development in a model-driven cognitive information system. *Eastern European Journal of Advanced Technology*, 6(7 (42)), 40–49.

14. Dargie, W., Mendez, J., Möbius, C., Rybina, K., Thost, V., & Turhan, A. Y. (2013, March). Situation recognition for service management systems using OWL 2 reasoners. *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 31–36.

15. Chen, P. P., & Wong, L. Y. (Eds.). (2007). *Active conceptual modeling of learning*. Springer Verlag.

16. Fowler, M., & Scott, K. (1999). UML is a summary. M.: Mir.

17. Butch, G. (2006). UML language. User Manual. 2nd ride: Trans. with english N. Mukhin. M.: DMK Press.

18. O'Connor, M. J., & Das, A. K. (2009, October). SQWRL: A Query Language for OWL. *Proceedings of the 6th International Conference on OWL: Experiences and Directions (OWLED'09)*, 529, 208–215.

19. Fudholi, D. H., Maneerat, N., Varakulsiripunth, R., & Kato, Y. (2009, January). Application of Protégé, SWRL and SQWRL in fuzzy ontology-based menu recommendation. *International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2009)*, 631–634.

SITUATION IDENTIFICATION SYSTEM IN THE SOFTWARE TESTING

Eugene Burov[1], Khrystyna Mykich[2], Oleh Veres[3], Vasyl Lytvyn[4]

[1–4] Lviv Polytechnic National University
[1] Yevhen.V.Burov@lpnu.ua, ORCID 0000-0001-8653-1520
[2] Khrystyna.I.Mykich@lpnu.ua, ORCID 0000-0002-4324-2080
[3] Oleh.M.Veres@lpnu.ua, ORCID 0000-0001-9149-4752
[4] Vasyl.V.Lytvyn@lpnu.ua, ORCID 0000-0002-9676-0180.

The paper is devoted to the research and development of methods and tools for identifying problematic situations on the basis of ontologies using the mechanisms of logical inference that are used in intellectual decision support systems for software testing problems.

The important problem of software testing using ontological modeling for timely detection of errors and improvement of quality of the developed software is considered.

Using ontological modeling to represent and identify situations creates additional opportunities and constraints to solving the identification problem. The advantage is the ability to use logical inference and use axioms in the process of reasoning about situations. This opens up the prospect of developing methods for identifying situations based on logical inference based on information about the current state of the subject area and knowledge about the subject area.

The used situation models allows not only automate the execution of some simple tasks, but as well as to make logical reasoning in the testing systems based on your existing knowledge of situations.

The ontological knowledge presentation of domain made possible to formalize knowledge about the problematic situations that arise on the project. Application of the developed methods of situation identification in the system ensured timely identification of threatening situations and formation of recommendations for their elimination. All these factors help to increase the quality of the software product during its development.

The ontology of the software testing industry is presented in the paper, as well as the algorithm of the system operation and modeling based on UML.

The architecture of the situation identification system was developed, as well as the software for the analysis and modeling of problem situations on the example of decision support systems of the field of software testing. The central component of software is the ontological modeling tool – Protégé.

The plugins SWRL and SQWRL were used to extend the functionality of the Protégé, and provide necessary modeling functionality.

It is advisable to use the results of the work to solve the problems of identifying critical situations during the development and testing of software, reuse the software quality control information in knowledge bases and thus increase the quality of created software.

Key words: ontological modeling, problematic situation, software testing, logical reasoning.