РАДІОТЕХНІКА, ЕЛЕКТРОНІКА ТА ТЕЛЕКОМУНІКАЦІЇ

УДК 004.9: 004.89

Т.О. ГОВОРУЩЕНКО, О.В. ПОМОРОВА, О.О. ПАВЛОВА
Хмельницький національний університет

МОДЕЛЮВАННЯ ДІЯЛЬНОСТІ ІНТЕЛЕКТУАЛЬНОГО АГЕНТА НА ОСНОВІ ОНТОЛОГІЧНОГО ПІДХОДУ ДЛЯ ОЦІНЮВАННЯ СПЕЦИФІКАЦІЙ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Метою даного дослідження є моделювання діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікації вимог до ПЗ. У статті вперше запропоновані теоретико-множинні, базові онтологічні моделі нефункційних характеристик ПЗ, онтологічні моделі нефункційних характеристик конкретного ПЗ, а також модель діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ.

Ключові слова: програмне забезпечення (ПЗ), програмний проект, специфікація вимог до програмного забезпечення, нефункційні характеристики програмного забезпечення, онтологія, інтелектуальний агент.

T. O. HOVORUSHCHENKO, O. V. POMOROVA, O. O. PAVLOVA
Khmelnitsky National University

MODELING THE ACTIVITY OF ONTOLOGY-BASED INTELLIGENT AGENT FOR EVALUATING THE SOFTWARE REQUIREMENTS SPECIFICATIONS

The aim of this study is the modeling the activity of the ontology-based intelligent agent for evaluating the software requirements specifications. The conducted analysis of the known ontology-based intelligent agents has shown that they do not solve the problem of quantifying the sufficiency of information in the software requirements specification for determining the software non-functional characteristics. In this paper, the set-theoretical models of software non-functional characteristics, base (universal) ontological models of software non-functional characteristics, and ontological models of non-functional characteristics of the concrete software are first time developed. These models are based on the considering the requirements of the standards of ISO 25010:2011 and ISO 25023:2016, and provide the basis for choosing the sufficient amount of information for evaluating the non-functional characteristics of software. The proposed ontology-based intelligent agent during its activity uses the base ontologies of the software non-functional characteristics as known facts, with which the agent compares the information, obtained from the specification of the requirements for concrete software, in the form of real ontologies. On the basis of this comparative analysis, the ontology-based intelligent agent evaluates the information of the software requirements specification and decides on further actions. The intelligent agent provides: the conclusion about the sufficiency or insufficiency of information in the specification, numerical evaluations of the level of sufficiency of information for determining each software non-functional characteristics; numerical evaluation of the level of sufficiency of information for determining all software non-functional characteristics. The model of activity of the ontology-based intelligent agent for evaluating the software requirements specifications is first time developed in the paper. This model is based on the comparative analysis of ontologies and is the theoretical basis for the development of methods of activity of the ontology-based intelligent agent.

Keywords: software, software project, software requirements specification, non-functional software characteristics, ontology, intelligent agent.

Вступ

Особливої уваги у напрямку розроблення та впровадження ефективних інформаційних технологій на сьогодні потребує галузь інженерії програмного забезпечення (ПЗ), оскільки помилки та відмови ПЗ загрожують катастрофами, які призводять до людських жертв, екологічних катаклізмів, значних часових та фінансових втрат.

На сьогодні в галузі інженерії ПЗ реалізована велика кількість програмних проектів, наявна статистика успішних та неуспішних рішень та результатів, але відсутні ефективні інформаційні технології, які б дали можливість отримати всю корисну інформацію з наявної статистики. Відомі інформаційні технології передбачають людинно-машинну взаємодію на всіх етапах опрацювання інформації, тобто всю інформацію інтерпретує людина, що для галузі інженерії ПЗ часто призводить до втрат істотної інформації і до виникнення помилок на ранніх етапах життєвого циклу. Відтак *актуальною задачею* є розроблення для галузі інженерії ПЗ інформаційних технологій нової генерації, в яких людина усувається з процесів опрацювання інформації та здобуття знань.

Наразі людство все частіше покладається на ПЗ при вирішенні складних задач, стрімко зростає кількість програмних проектів з високою вартістю. Але, як показує статистика [1, 2], частка проблемних програмних проектів складає порядку половини всіх програмних проектів, а частка провальних програмних проектів складає порядку 1/5 всіх програмних проектів, тобто успішними програмними проектами, на які можна покладатись і на які варто витрачати кошти, є лише 1/3 всіх програмних проектів. Значна кількість помилок вноситься у ПЗ на початкових етапах життєвого циклу ПЗ. Переважна більшість аварій, пов'язаних із ПЗ, виникли через помилки у специфікації вимог [3]. Отже, успішність реалізації програмного проекту (як вчасне виконання програмного проекту в рамках виділеного бюджету та з реалізацією всіх необхідних можливостей та функцій) суттєво залежить від ранніх етапів життєвого циклу ПЗ, тому *актуальною та важливою задачею* є автоматизоване оцінювання рівня відпрацювання початкових етапів життєвого циклу ПЗ на основі аналізу специфікацій (зокрема, оцінювання достатності інформації у специфікації вимог до ПЗ

як ключової складової успішності програмних проектів), причому особливої уваги потребують нефункційні характеристики ПЗ (наприклад, якість, надійність, функційна придатність, ефективність, сумісність, супроводжуваність, можливість переносу, захищеність, зручність використання).

Актуальність та важливість задачі автоматизованого оцінювання початкових етапів життєвого циклу ПЗ на основі аналізу специфікацій та необхідність інформаційних технологій нової генерації для галузі інженерії ПЗ обумовлює необхідність розроблення агентно-орієнтованої інформаційної технології оцінювання початкових етапів життєвого циклу ПЗ на основі онтологічного підходу. *Задачею даного дослідження* є моделювання діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікації вимог до ПЗ.

Огляд інтелектуальних агентів на основі онтологічного підходу для галузі інженерії програмного забезпечення

Перевагами використання онтологій є системний підхід до вивчення предметної галузі, можливість цілісного подання відомої інформації предметної галузі, виявлення дублювань та прогалин у знаннях на основі візуалізації відсутніх логічних зв'язків. Онтології визначаються як ключова технологія для розвитку семантичного вебу, оскільки надають можливість доступу, розуміння та аналізу інформації інтелектуальними агентами [4].

Інтелектуальний агент (ІА) – система, котра спостерігає за навколишнім середовищем, взаємодіє з ним, а її поведінка раціональна в тому сенсі, що агент розуміє суть власних дій, і ці дії скеровані на досягнення певної мети [5]. Таким агентом може виступати програмна система, бот, сервіс. Інтелектуальний агент використовує під час свого функціонування інформацію, отриману з навколишнього середовища, аналізує її, зіставляючи з уже відомими йому фактами і, на основі результатів аналізу, приймає рішення про подальші дії.

Ідеї використання онтологій для галузі інженерії ПЗ присвячено ряд робіт: запропоновано методи та засоби побудови програмних систем на основі онтологічних моделей задач [6]; розроблено онтологічні моделі та методи формування нормативного профілю при сертифікації ПЗ [7]; запропоновано онтологічний підхід до специфікування властивостей програмних систем та їх компонентів [8]; розроблено підходи трасування вимог до ПЗ на основі зважених онтологій [9-11]; запропоновано використання доменної онтології для аналізу ПЗ та засобів реінжинірингу [12]; розроблено онтологічну модель для опису та визначення предметних і операційних знань щодо забезпечення якості ПЗ [13]; розроблено онтології та зважені онтології предметної галузі «Інженерія програмного забезпечення» (частини «Якість ПЗ», «Якість ПЗ. Метричний аналіз», «Специфікація вимог до ПЗ») [14].

Розв’язанню задачі розроблення інтелектуальних агентів на основі онтологічного підходу для галузі інженерії ПЗ також присвячено ряд досліджень. Так, автори [15] досліджують застосування онтологій для агентно-орієнтованої програмної інженерії, пропонують інструмент, який використовує існуючі онтологічні конструкції для створення програмного коду, а також експериментально підтверджують переваги застосування агентів на основі онтологій для галузі інженерії ПЗ. Робота [16] присвячена усуненню невизначеності у вимогах до ПЗ та покращенню спілкування зацікавлених сторін шляхом впровадження інтелектуальних агентів на основі онтологічного підходу. Автори [17] пропонують розроблення інтелектуального агента на основі онтології для мінімізації існуючої семантичної невизначеності при розробленні специфікації вимог до ПЗ природньою (іспанською) мовою, для автоматичного отримання основних елементів специфікації та для автоматичної побудови діаграми цілей. У роботі [18] онтологічні агентно-орієнтовані моделі використовують для формалізації первинних вимог до ПЗ з метою зниження витрат на прикладі розроблення додатків для Ambient Assisted Living для пацієнтів із хворобою Паркінсона. Автори [19] пропонують завдання-орієнтовану архітектуру на основі агентно-орієнтованої парадигми та онтологічного дизайну для систем підтримки прийняття рішень (на прикладі клінічної системи для відділення невідкладної допомоги), яка дозволяє ітеративно переносити функційні вимоги в архітектурні компоненти. В роботі [20] надано основу для формального подання та перевірки вимог і забезпечення функційної правильності обмежених ресурсами контекстних систем критичного застосування у вигляді інтелектуальних агентів на основі онтологічного підходу.

Проведений аналіз відомих інтелектуальних агентів на основі онтологічного підходу показав, що вони не розв’язують задачу кількісного оцінювання рівня відпрацювання початкових етапів життєвого циклу ПЗ на основі аналізу специфікацій (зокрема, оцінювання достатності інформації у специфікації вимог до ПЗ). Єдиним рішенням в галузі оцінювання достатності інформації у специфікаціях вимог до ПЗ є робота [14], в якій запропоновано теоретичні та прикладні засади інформаційної технології оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ та розроблено онтології предметної галузі «Інженерія програмного забезпечення» (частини «Якість ПЗ», «Якість ПЗ. Метричний аналіз», «Специфікація вимог до ПЗ»).

Моделі нефункційних характеристик програмного забезпечення

Відповідно до стандарту ISO 25010:2011, такі нефункційні характеристики ПЗ, як функційна придатність (Fs), ефективність (Pe), зручність використання (Ub), надійність (Rb), сумісність (Cb), захищеність (Scr), супроводжуваність (Mb), можливість переносу (Pb) є функціями від декількох підхарактеристик, тоді *представимо кожну з вищевказаних характеристик як функцію від підхарактеристик*:

$$Fs = f_1(FCom, FCor, FAppr), \quad (1)$$

де $FCom$ – функційна повнота, $FCor$ – функційна коректність, $FAppr$ – функційна доцільність;

$$Pe = f_2(Tb, Ru, Cc), \quad (2)$$

де Tb – поведінка у часі, Ru – поведінка ресурсів, Cc – ємність (місткість);

$$Ub = f_3(Ar, Lb, Ob, Uep, Uia, Ab), \quad (3)$$

де Ar – розпізнавання доцільності, Lb – можливість вивчення, Ob – керованість, Uep – захист від помилок користувача, Uia – естетичність інтерфейсу користувача, Ab – доступність;

$$Rb = f_4(Mat, Avb, Ft, Rcv), \quad (4)$$

де Mat – зрілість, Avb – наявність (доступність), Ft – відмовостійкість, Rcv – відновлюваність;

$$Cb = f_5(Ce, Ib), \quad (5)$$

де Ce – співіснування, Ib – взаємодія;

$$Scr = f_6(Conf, Int, Nr, Acb, Auth), \quad (6)$$

де $Conf$ – конфіденційність, Int – цілісність, Nr – невідхилюваність, Acb – підзвітність, $Auth$ – ідентичність;

$$Mb = f_7(Mod, Rub, Anb, Mdfb, Tsb), \quad (7)$$

де Mod – модульність, Rub – повторне використання, Anb – аналізованість, $Mdfb$ – модифікованість, Tsb – тестованість;

$$Pb = f_8(Adb, Inb, Rpb), \quad (8)$$

де Adb – адаптованість, Inb – можливість інсталяції, Rpb – можливість заміни.

Тоді *множини підхарактеристик нефункційних характеристик ПЗ* мають вигляд:

$A = \{FCom, FCor, FAppr\}$ – множина підхарактеристик функційної придатності;

$B = \{Tb, Ru, Cc\}$ – множина підхарактеристик ефективності;

$C = \{Ar, Lb, Ob, Uep, Uia, Ab\}$ – множина підхарактеристик зручності використання,

$D = \{Mat, Avb, Ft, Rcv\}$ – множина підхарактеристик надійності;

$E = \{Ce, Ib\}$ – множина підхарактеристик сумісності;

$F = \{Conf, Int, Nr, Acb, Auth\}$ – множина підхарактеристик захищеності;

$G = \{Mod, Rub, Anb, Mdfb, Tsb\}$ – множина підхарактеристик супроводжуваності;

$H = \{Adb, Inb, Rpb\}$ – множина підхарактеристик можливості переносу.

Водночас, кожна підхарактеристика нефункційних характеристик ПЗ є функцією певних атрибутів, описаних у стандарті ISO 25023:2016. *Залежності підхарактеристик нефункційних характеристик від атрибутів* представимо в наступному вигляді:

$$FCom = \varphi_1(Nof, Ficn, Faq, Fic), \quad (9)$$

де Nof – кількість функцій, $Ficn$ – повнота функційної реалізації, Faq – функційна адекватність, Fic – покриття функційної реалізації;

$$FCor = \varphi_2(Ot, Nic, Ndi, Ca, Pc), \quad (10)$$

де Ot – час роботи, Nic – кількість неточних обчислень, з якими стикаються користувачі, Ndi – кількість елементів даних, Ca – обчислювальна точність, Pc – точність (відповідність);

$$FAppr = \varphi_3(Ot, Nof, Ficn, Faq, Fic, Pc); \quad (11)$$

$$Tb = \varphi_4(Ot, Nmot, Rt, Noe, Tnt, Tskt, Mathr), \quad (12)$$

де $Nmot$ – кількість задач, Rt – час реакції, Noe – кількість оцінок, Tnt – час обробки, $Tskt$ – час задачі, $Mathr$ – середнє значення пропускної здатності;

$$Ru = \varphi_5(Ot, Nofl, Noe, Niore, Uwt, Nmre, Ntre, Tcc, Iou, Nolcd, Ioll, Mmu, Mtu, Mote), \quad (13)$$

де $Nofl$ – кількість відмов, $Niore$ – кількість помилок введення-виведення, Uwt – час очікування користувача під час використання пристрою введення-виведення, $Nmre$ – кількість помилок пам'яті, $Ntre$ – кількість помилок передачі даних, Tcc – потужність (ємність) передачі даних, Iou – кількість буферів при використанні введення-виведення, $Nolcd$ – безпосередня кількість рядків коду, $Ioll$ – ліміт завантаження пристроїв введення-виведення, Mmu – максимум використовуваної пам'яті, Mtu – максимум використовуваної передачі даних, $Mote$ – середня поява помилки передачі;

$$Cc = \varphi_6(Ndi, Ncu, Cbw, Mathr, Sdb), \quad (14)$$

де Ncu – кількість одночасних користувачів, Cbw – ширина смуги комунікації, Sdb – розмір бази даних;

$$Ar = \varphi_7(Nof, Nott, Niodi, Cnd, Fua, Uaio), \quad (15)$$

де $Nott$ – кількість посібників, $Niodi$ – кількість елементів даних введення-виведення, Cnd – повнота описів, Fua – зрозумілість функціоналу, $Uaio$ – зрозумілість входів та виходів;

$$Lb = \varphi_8(Nof, Ot, Efl, Nmot, Hfq, Eudhs, Haa, Cudh), \quad (16)$$

де Efl – простота вивчення функціоналу, Hfq – частота звертань до довідки, $Eudhs$ – ефективність документації користувача та системи допомоги, Haa – доступність довідки, $Cudhf$ – повнота документації користувача та довідкового фонду;

$$Ob = \varphi_9(Nof, Ot, Ecr, Nsf, Nuec, Nac, Niore, Nop, Niwccvd, Nmi, Nie, Pha, Neum), \quad (17)$$

де Ecr – частота корекції помилок, Nsf – кількість екранів або форм, $Nuec$ – кількість помилок або змін користувача, Nac – кількість спроб налаштування, Nop – кількість операцій, $Niwccd$ – кількість елементів, які можна перевірити на наявність дійсних даних, Nmi – кількість виконаних повідомлень, Nie – кількість елементів інтерфейсу, Pha – фізична доступність, $Neum$ – кількість легко зрозумілих повідомлень;

$$Uep = \varphi_{10}(Nurs, Nuec, Otpdo, Nouheo, Niewusc, Nacie, Necwusc, Tnect, Nfiuet, Tnfrtc, Tniop), \quad (18)$$

де $Nurs$ – кількість невдало відновлених ситуацій, $Optdo$ – час роботи під час спостережень, $Nouheo$ – кількість випадків операційних помилок користувача, $Niewusc$ – кількість вхідних помилок, які користувач успішно виправляє, $Nacie$ – кількість спроб коригування вхідних помилок, $Necwusc$ – клькість помилкових умов, які користувач успішно виправляє, $Tnect$ – загальна кількість перевірених помилкових умов, $Nfiuet$ – кількість функцій, виконаних з толерантністю до помилки користувача, $Tnfrtc$ – загальна кількість функцій, що вимагають можливості толерантності, $Tniop$ – загальна кількість некоректних шаблонів операцій;

$$Uia = \varphi_{11}(Nie, Nige, Dipu, Disu, Dea, Drwmu), \quad (19)$$

де $Nige$ – кількість графічних елементів інтерфейсу, $Dipu$ – ступінь збільшення задоволення користувача, $Disu$ – ступінь збільшення задоволеності потреб користувача, Dea – ступінь ергономічної привабливості, $Drwmu$ – ступінь використання метафор реального світу;

$$Ab = \varphi_{12}(Ewscbuusd, Ewusd, Ffrusd, Susd, Ppsa), \quad (20)$$

де $Ewscbuusd$ – обсяг, до якого програмним забезпеченням можуть користуватись користувачі з обмеженими можливостями, $Ewusd$ – ефективність роботи користувачів з обмеженими можливостями, $Ffrusd$ – свобода від ризику для користувачів із обмеженими можливостями, $Susd$ – задоволення потреб користувачів з обмеженими можливостями, $Ppsa$ – наявність властивостей, які підтримують доступність;

$$Mat = \varphi_{13}(Ot, Noft, Nofl, Ps, Ntc, Nrf, Ncf, Fdatc, Frn, Frl, Mtbf, Tmy, Elfd, Fdy), \quad (21)$$

де $Noft$ – кількість збоїв, Ps – розмір продукту, Ntc – кількість тестових випадків, Nrf – кількість фіксованих відмов, Ncf – кількість усунутих (виправлених) збоїв, $Fdatc$ – щільність відмов відносно до тестових випадків, Frn – роздільна здатність відмов, Frl – усунення збоїв, $Mtbf$ – середній час між відмовами, Tmy – випробувальний термін, $Elfd$ – орієнтовна щільність прихованих збоїв, Fdy – щільність збоїв;

$$Avb = \varphi_{14}(Ot, Ttdwsis, Nob, Tdt), \quad (22)$$

де $Ttdwsis$ – загальний час, протягом якого програма перебуває у стані зростання, Nob – кількість спостережуваних несправностей, Tdt – загальна тривалість простоїв;

$$Ft = \varphi_{15}(Nofl, Ntc, Nbd, Nof, Nio), \quad (23)$$

де Nbd – кількість несправностей, Nio – кількість недозволених операцій;

$$Rcv = \varphi_{16}(Ot, Nbd, Ttr, Dt, Nrs, Nrn, Ray), \quad (24)$$

де Ttr – час ремонту, Dt – тривалість простою, Nrs – кількість перезапусків, Nrn – кількість відновлень, Ray – відновлюваність (можливість перезапуску);

$$Ce = \varphi_{17}(Ot, Nofl, Nof, Ndi); \quad (25)$$

$$Ib = \varphi_{18}(Ot, Ndfrt, Ndfbe, Nip, Deay), \quad (26)$$

де $Ndfrt$ – кількість форматів даних, пов'язаних інструментом, $Ndfbe$ – кількість форматів даних для обміну, Nip – кількість протоколів інтерфейсу, $Deay$ – обмінність даних;

$$Conf = \varphi_{19}(Ot, Nio, Ntc, Nidc, Ndi, Nat, Ncr, Aca, Ndiced, Ndibred), \quad (27)$$

де $Nidc$ – кількість випадків пошкодження даних, Nat – кількість типів доступу, Ncr – кількість контрольованих вимог, Aca – керованість доступу, $Ndiced$ – кількість елементів даних, які правильно зашифровані та розшифровані, $Ndibred$ – кількість елементів даних, які потребують шифрування та розшифрування;

$$Int = \varphi_{20}(Ot, Nio, Ntc, Nidc, Ndi, Nat, Ncr, Aca); \quad (28)$$

$$Nr = \varphi_{21}(Nepuds, Nernrp), \quad (29)$$

де $Nepuds$ – кількість подій, оброблених за допомогою цифрового підпису, $Nernrp$ – кількість подій, що вимагають властивості невідхилення (неприпинення);

$$Acb = \varphi_{22}(Nasdrsl, Naao), \quad (30)$$

де $Nasdrsl$ – кількість доступів до системи та даних, записаних у системний журнал, $Naao$ – кількість дійсно набутих доступів;

$$Auth = \varphi_{23}(Npam), \quad (31)$$

де $Npam$ – кількість наданих методів перевірки автентичності;

$$Mod = \varphi_{24}(Ot, Nofl, Nrf, Nmm, Nv, Nof, Nm), \quad (32)$$

де Nmm – кількість внесених змін, Nv – кількість змінних, Nm – кількість модулів;

$$Rub = \varphi_{25}(Fcy, Nfcy, Vrn, Aay, Tay, Cra), \quad (33)$$

де Fcy – функційна спільність, $Nfcy$ – нефункційна спільність, Vrn – міцініть варіабельності, Aay – застосовність, Tay – пристосованість, Cra – заміна компонентів;

$$Anb = \varphi_{26}(Nofl, Ndi, Ert, Nirbl, Ndfr, Atc), \quad (34)$$

де Ert – час помилки, $Nirbl$ – кількість елементів, необхідних для запису в журналі, $Ndfr$ – кількість необхідних діагностичних функцій, Atc – можливість аудиту сліду;

$$Mdfb = \varphi_{27}(Ot, Nrv, Nrf, Ert, Nof, Ccca, Ntcpbm, Ntspam), \quad (35)$$

де Nrv – кількість переглянутих версій, $Ccca$ – можливість управління змінами, $Ntcpbm$ – гількість проблем протягом певного періоду до модифікації, $Ntspam$ – гількість проблем за той же період після модифікації;

$$Tsb = \varphi_{28}(Ot, Ntc, Nrf, Nbtfr, Ntdos, Ncp), \quad (36)$$

де $Nbtfr$ – кількість необхідних вбудованих тестових функцій, $Ntdos$ – кількість тестових залежностей на інших системах, Ncp – кількість контрольних точок;

$$Adb = \varphi_{29}(Nof, Ot, Noft, Puf, Ndi, Nds, Aads, Hea, Sea, Noftwnca, Tnfwtde), \quad (37)$$

де Puf – портативна дружність користувача, Nds – кількість структур даних, $Aads$ – адаптивність структур даних, Hea – адаптивність апаратного оточення, Sea – адаптивність програмного оточення, $Noftwnca$ – кількість операційних функцій, завдання яких не були виконані або неадекватні, $Tnfwtde$ – загальна кількість функцій, які були випробувані за різних умов;

$$Inb = \varphi_{30}(Noft, Nso, Nis, Eoi), \quad (38)$$

де Nso – кількість операцій налаштування, Nis – кількість кроків інсталяції, Eoi – простота інсталяції;

$$Rpb = \varphi_{31}(Nof, Ndi, Net), \quad (39)$$

де Net – кількість сутностей.

Тоді *множини атрибутів для підхарактеристик нефункційних характеристик ПЗ* мають вигляд:

$I = \{Nof, Ficn, Faq, Fic\}$ – множина атрибутів для функційної повноти; $J = \{Ot, Nic, Ndi, Ca, Pc\}$ – множина атрибутів для функційної коректності; $K = \{Ot, Nof, Ficn, Faq, Fic, Pc\}$ – множина атрибутів для функційної доцільності;

$L = \{Ot, Nmot, Rt, Noe, Tnt, Tskt, Mathr\}$ – множина атрибутів для поведінки у часі; $M = \{Ot, Nofl, Noe, Niore, Uwt, Nmre, Ntre, Tcc, Iou, Nolcd, Ioll, Mmu, Mtu, Mote\}$ – множина атрибутів для поведінки ресурсів; $N = \{Ndi, Ncu, Cbw, Mathr, Sdb\}$ – множина атрибутів для ємності (місткості);

$O = \{Nof, Nott, Niodi, Cnd, Fua, Uaio\}$ – множина атрибутів для розпізнавання доцільності; $P = \{Nof, Ot, Efl, Nmot, Hfq, Eudhs, Haa, Cudhf\}$ – множина атрибутів для можливості вивчення; $Q = \{Nof, Ot, Ecr, Nsf, Nuec, Nac, Niore, Nop, Niwccvd, Nmi, Nie, Pha, Neum\}$ – множина атрибутів для керованості; $R = \{Nurs, Nuec, Otpdo, Nouheo, Niewusc, Nacie, Necwusc, Tnect, Nfiuet, Tnfrtc, Tniop\}$ – множина атрибутів для захисту від помилок користувача; $S = \{Nie, Nige, Dipu, Disu, Dea, Drwmu\}$ – множина атрибутів для естетичності інтерфейсу; $T = \{Ewscbuusd, Ewusd, Ffrusd, Susd, Ppsa\}$ – множина атрибутів для доступності;

$U = \{Ot, Noft, Nofl, Ps, Ntc, Nrf, Ncf, Fdatc, Frn, Frl, Mtbf, Tmy, Elfd, Fdy\}$ – множина атрибутів для зрілості; $V = \{Ot, Ttdwsis, Nob, Tdt\}$ – множина атрибутів для наявності (доступності); $W = \{Nofl, Ntc, Nbd, Nof, Nio\}$ – множина атрибутів для відмовостійкості; $X = \{Ot, Nbd, Ttr, Dt, Nrs, Nrn, Ray\}$ – множина атрибутів для відновлюваності;

$Y = \{Ot, Nofl, Nof, Ndi\}$ – множина атрибутів для співіснування; $Z = \{Ot, Ndfrt, Ndfbe, Nip, Deay\}$ – множина атрибутів для взаємодії;

$AA = \{Ot, Nio, Ntc, Nidc, Ndi, Nat, Ncr, Aca, Ndiced, Ndibred\}$ – множина атрибутів для конфіденційності; $AB = \{Ot, Nio, Ntc, Nidc, Ndi, Nat, Ncr, Aca\}$ – множина атрибутів для цілісності;

$AC = \{Nepuds, Nernrp\}$ – множина атрибутів для невідхилюваності; $AD = \{Nasdrsl, Naao\}$ – множина атрибутів для підзвітності; $AE = \{Npam\}$ – множина атрибутів для ідентичності;

$AF = \{Ot, Nofl, Nrf, Nmm, Nv, Nof, Nm\}$ – множина атрибутів для модульності; $AG = \{Fcy, Nfcy, Vrn, Aay, Tay, Cra\}$ – множина атрибутів для повторного використання; $AH = \{Nofl, Ndi, Ert, Nirbl, Ndfr, Atc\}$ – множина атрибутів для аналізованості; $AI = \{Ot, Nrv, Nrf, Ert, Nof, Ccca, Ntcpbm, Ntspam\}$ – множина атрибутів для модифікованості; $AJ = \{Ot, Ntc, Nrf, Nbtfr, Ntdos, Ncp\}$ – множина атрибутів для тестованості;

$AK = \{Nof, Ot, Noft, Puf, Ndi, Nds, Aads, Hea, Sea, Noftwnca, Tnfwtde\}$ – множина атрибутів для адаптованості; $AL = \{Noft, Nso, Nis, Eoi\}$ – множина атрибутів для можливості інсталяції; $AM = \{Nof, Ndi, Net\}$ – множина атрибутів для можливості заміни.

Враховуючи отримані множини атрибутів та підхарактеристик нефункційних характеристик ПЗ, отримані функції залежностей характеристик від підхарактеристик і підхарактеристик від атрибутів, а також відношення між поняттями онтології «залежить від», розробимо *базові (універсальні) онтологічні моделі нефункційних характеристик ПЗ* (принципи розроблення базових онтологічних моделей представлені у [21]). Тоді:

- базова онтологічна модель функційної придатності:

$$O_{Fs} = \{\{Fs, A, I, J, K\}, "depends\ \ on", \{f_1, \varphi_1, \varphi_2, \varphi_3\}\} = \{\{fsa_1, ..., fsa_{19}\}, "depends\ \ on", \{f_1, \varphi_1, \varphi_2, \varphi_3\}\}, \quad (40)$$

де $fsa_1 = Fs$, $\{fsa_2, ..., fsa_4\} \in A$, $\{fsa_5, ..., fsa_8\} \in I$, $\{fsa_9, ..., fsa_{13}\} \in J$, $\{fsa_{14}, ..., fsa_{19}\} \in K$;

- базова онтологічна модель ефективності:

$$O_{Pe} = \{\{Pe, B, L, M, N\}, "depends\ \ on", \{f_2, \varphi_4, \varphi_5, \varphi_6\}\} = \{\{pea_1, ..., pea_{30}\}, "depends\ \ on", \{f_2, \varphi_4, \varphi_5, \varphi_6\}\}; \quad (41)$$

- базова онтологічна модель зручності використання:

$$O_{Ub} = \{\{Ub, C, O, P, Q, R, S, T\}, "depends\ \ on", \{f_3, \varphi_7, \varphi_8, \varphi_9, \varphi_{10}, \varphi_{11}, \varphi_{12}\}\} = \\ = \{\{uba_1, ...uba_{56}\}, "depends\ \ on", \{f_3, \varphi_7, \varphi_8, \varphi_9, \varphi_{10}, \varphi_{11}, \varphi_{12}\}\}; \quad (42)$$

- базова онтологічна модель надійності:

$$O_{Rb} = \{\{Rb, D, U, V, W, X\}, "depends\ \ on", \{f_4, \varphi_{13}, \varphi_{14}, \varphi_{15}, \varphi_{16}\}\} = \\ = \{\{rba_1, ...rba_{35}\}, "depends\ \ on", \{f_4, \varphi_{13}, \varphi_{14}, \varphi_{15}, \varphi_{16}\}\}; \quad (43)$$

- базова онтологічна модель сумісності:

$$O_{Cb} = \{\{Cb, E, Y, Z\}, "depends\ \ on", \{f_5, \varphi_{17}, \varphi_{18}\}\} = \{\{cba_1, ..., cba_{12}\}, "depends\ \ on", \{f_5, \varphi_{17}, \varphi_{18}\}\}; \quad (44)$$

- базова онтологічна модель захищеності:

$$O_{Scr} = \{\{Scr, F, AA, AB, AC, AD, AE\}, "depends\ \ on", \{f_6, \varphi_{19}, \varphi_{20}, \varphi_{21}, \varphi_{22}, \varphi_{23}\}\} = \\ = \{\{scra_1, ..., scra_{29}\}, "depends\ \ on", \{f_6, \varphi_{19}, \varphi_{20}, \varphi_{21}, \varphi_{22}, \varphi_{23}\}\}; \quad (45)$$

- базова онтологічна модель супроводжуваності:

$$O_{Mb} = \{\{Mb, G, AF, AG, AH, AI, AJ\}, "depends\ \ on", \{f_7, \varphi_{24}, \varphi_{25}, \varphi_{26}, \varphi_{27}, \varphi_{28}\}\} = \\ = \{\{mba_1, ..., mba_{39}\}, "depends\ \ on", \{f_7, \varphi_{24}, \varphi_{25}, \varphi_{26}, \varphi_{27}, \varphi_{28}\}\}; \quad (46)$$

- базова онтологічна модель можливості переносу:

$$O_{Pb} = \{\{Pb, H, AK, AL, AM\}, "depends\ \ on", \{f_8, \varphi_{29}, \varphi_{30}, \varphi_{31}\}\} = \\ = \{\{pba_1, ..., pba_{22}\}, "depends\ \ on", \{f_8, \varphi_{29}, \varphi_{30}, \varphi_{31}\}\}. \quad (47)$$

Онтології (онтологічні база знань), які розробляються за моделями (40)-(47), наповнюються на основі інформації, взятої зі стандартів ISO 25010:2011, ISO 25023:2016.

Тоді *онтологічні моделі нефункційних характеристик конкретного ПЗ* мають вигляд:

- онтологічна модель функційної придатності:

$$O_{Fs}^{real} = \{\{fsa_1, ..., fsa_m\}, "depends\ \ on", \{f_1, \varphi_1, \varphi_2, \varphi_3\}\}, \quad (48)$$

де $m \le 19$ – кількість атрибутів підхарактеристик функційної придатності, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик функційної придатності, які можна обчислити на основі наявних атрибутів;

- онтологічна модель ефективності:

$$O_{Pe}^{real} = \{\{pea_1, ..., pea_n\}, "depends\ \ on", \{f_2, \varphi_4, \varphi_5, \varphi_6\}\}; \quad (49)$$

де $n \le 30$ – кількість атрибутів підхарактеристик ефективності, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик ефективності, які можна обчислити на основі наявних атрибутів;

- онтологічна модель зручності використання:

$$O_{Ub}^{real} = \{\{uba_1, ...uba_k\}, "depends\ \ on", \{f_3, \varphi_7, \varphi_8, \varphi_9, \varphi_{10}, \varphi_{11}, \varphi_{12}\}\}; \quad (50)$$

де $k \le 56$ – кількість атрибутів підхарактеристик зручності використання, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик зручності використання, які можна обчислити на основі наявних атрибутів;

- онтологічна модель надійності:

$$O_{Rb}^{real} = \{\{rba_1,...rba_o\},"depends\ \ on",\{f_4,\varphi_{13},\varphi_{14},\varphi_{15},\varphi_{16}\}\}; \quad (51)$$

де $o \le 35$ – кількість атрибутів підхарактеристик надійності, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик надійності, які можна обчислити на основі наявних атрибутів;

- онтологічна модель сумісності:

$$O_{Cb}^{real} = \{\{cba_1,...,cba_p\},"depends\ \ on",\{f_5,\varphi_{17},\varphi_{18}\}\}; \quad (52)$$

де $p \le 12$ – кількість атрибутів підхарактеристик сумісності, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик сумісності, які можна обчислити на основі наявних атрибутів;

- онтологічна модель захищеності:

$$O_{Scr}^{real} = \{\{scra_1,...,scra_q\},"depends\ \ on",\{f_6,\varphi_{19},\varphi_{20},\varphi_{21},\varphi_{22},\varphi_{23}\}\}; \quad (53)$$

де $q \le 29$ – кількість атрибутів підхарактеристик захищеності, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик захищеності, які можна обчислити на основі наявних атрибутів;

- онтологічна модель супроводжуваності:

$$O_{Mb}^{real} = \{\{mba_1,...,mba_r\},"depends\ \ on",\{f_7,\varphi_{24},\varphi_{25},\varphi_{26},\varphi_{27},\varphi_{28}\}\}; \quad (54)$$

де $r \le 39$ – кількість атрибутів підхарактеристик супроводжуваності, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик супроводжуваності, які можна обчислити на основі наявних атрибутів;

- онтологічна модель можливості переносу:

$$O_{Pb}^{real} = \{\{pba_1,...,pba_l\},"depends\ \ on",\{f_8,\varphi_{29},\varphi_{30},\varphi_{31}\}\}. \quad (55)$$

де $l \le 22$ – кількість атрибутів підхарактеристик можливості переносу, наявних у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик можливості переносу, які можна обчислити на основі наявних атрибутів.

Онтології (онтологічні бази знань), які розробляються за моделями (48)-(55), наповнюються на основі інформації, взятої зі специфікації вимог до конкретного програмного забезпечення.

Моделювання діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до програмного забезпечення

Пропонований інтелектуальний агент на основі онтологічного підходу використовує під час свого функціонування базові онтології нефункційних характеристик ПЗ як відомі йому факти, з якими він зіставляє інформацію, отриману зі специфікації вимог до реального ПЗ, представлену у вигляді реальних онтологій, на основі чого оцінює інформацію у специфікації вимог до ПЗ та приймає рішення про подальші дії.

Отже, *процес оцінювання специфікації вимог інтелектуальним агентом полягає у*: 1) порівнянні онтологій нефункційних характеристик конкретного ПЗ з базовими онтологіями нефункційних характеристик ПЗ з метою виявлення атрибутів, відсутніх у специфікації вимог до ПЗ, за якою реальні онтології були побудовані, а також виявлення підхрактеристик та нефункційних характеристик ПЗ, які неможливо обчислити на основі наявних у специфікації вимог до конкретного ПЗ атрибутів; 2) формуванні висновку про достатність або недостатність інформації у специфікації вимог для визначення кожної нефункційної характеристики ПЗ окремо та для визначення всіх нефункційних характеристик ПЗ разом; 3) розрахунку числових оцінок рівня достатності наявної у специфікації вимог інформації для визначення кожної нефункційної характеристики ПЗ; 4) розрахунку числової оцінки рівня достатності наявної у специфікації вимог інформації для визначення всіх нефункційних характеристик ПЗ.

Нехай $SMM_{Fs} = O_{Fs} \setminus (O_{Fs} \cap O_{Fs}^{real})$, де: $SMM_{Fs} = \{fsa_1,...,fsa_{(19-m)}\}$ – множина атрибутів підхарактеристик функційної придатності, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик функційної придатності, які неможливо обчислити на основі наявних атрибутів; $SMM_{Pe} = O_{Pe} \setminus (O_{Pe} \cap O_{Pe}^{real})$, де: $SMM_{Pe} = \{pea_1,...,pea_{(30-n)}\}$ – множина атрибутів підхарактеристик ефективності, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик ефективності, які неможливо обчислити на основі наявних атрибутів; $SMM_{Ub} = O_{Ub} \setminus (O_{Ub} \cap O_{Ub}^{real})$, де: $SMM_{Ub} = \{uba_1,...,uba_{(56-k)}\}$ – множина атрибутів підхарактеристик зручності використання, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик зручності використання, які неможливо обчислити на основі наявних атрибутів; $SMM_{Rb} = O_{Rb} \setminus (O_{Rb} \cap O_{Rb}^{real})$, де: $SMM_{Rb} = \{rba_1,...,rba_{(35-o)}\}$ – множина атрибутів підхарактеристик надійності, відсутніх у специфікації

вимог до конкретного ПЗ, а також кількість підхарактеристик надійності, які неможливо обчислити на основі наявних атрибутів; $SMM_{Cb} = O_{Cb} \setminus (O_{Cb} \cap O_{Cb}^{real})$, де: $SMM_{Cb} = \{cba_1, ..., cba_{(12-p)}\}$ – множина атрибутів підхарактеристик суміності, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик сумісності, які неможливо обчислити на основі наявних атрибутів; $SMM_{Scr} = O_{Scr} \setminus (O_{Scr} \cap O_{Scr}^{real})$, де: $SMM_{Scr} = \{scra_1, ..., scra_{(29-q)}\}$ – множина атрибутів підхарактеристик захищеності, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик захищеності, які неможливо обчислити на основі наявних атрибутів; $SMM_{Mb} = O_{Mb} \setminus (O_{Mb} \cap O_{Mb}^{real})$, де: $SMM_{Mb} = \{mba_1, ..., mba_{(39-r)}\}$ – множина атрибутів підхарактеристик супроводжуваності, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик супроводжуваності, які неможливо обчислити на основі наявних атрибутів; $SMM_{Pb} = O_{Pb} \setminus (O_{Pb} \cap O_{Pb}^{real})$, де: $SMM_{Pb} = \{pba_1, ..., pba_{(22-l)}\}$ – множина атрибутів підхарактеристик можливості переносу, відсутніх у специфікації вимог до конкретного ПЗ, а також кількість підхарактеристик можливості переносу, які неможливо обчислити на основі наявних атрибутів.

Продукційні правила для формування висновку щодо достатності або недостатності інформації у специфікації вимог для визначення кожної нефункційної характеристики ПЗ:

- для функційної придатності:

$$\begin{array}{c} if\ SMM_{Fs} = \varnothing\ then\ "SRS\ information\ is\ sufficient\ for\ Functional\ Suitability" \\ else\ "SRS\ information\ is\ insufficient\ for\ Functional\ Suitability" \end{array}, \quad (56)$$

- для ефективності:

$$\begin{array}{c} if\ SMM_{Pe} = \varnothing\ then\ "SRS\ information\ is\ sufficient\ for\ Perfromance\ Efficiency" \\ else\ "SRS\ information\ is\ insufficient\ for\ Perfromance\ Efficiency" \end{array}, \quad (57)$$

- для зручності використання:

$$\begin{array}{c} if\ SMM_{Ub} = \varnothing\ then\ "SRS\ information\ is\ sufficient\ for\ Usability" \\ else\ "SRS\ information\ is\ insufficient\ for\ Usability" \end{array}, \quad (58)$$

- для надійності:

$$\begin{array}{c} if\ SMM_{Rb} = \varnothing\ then\ "SRS\ information\ is\ sufficient\ for\ Reliability" \\ else\ "SRS\ information\ is\ insufficient\ for\ Reliability" \end{array}, \quad (59)$$

- для сумісності:

$$\begin{array}{c} if\ SMM_{Cb} = \varnothing\ then\ "SRS\ information\ is\ sufficient\ for\ Compatibility" \\ else\ "SRS\ information\ is\ insufficient\ for\ Compatibility" \end{array}, \quad (60)$$

- для захищеності:

$$\begin{array}{c} if\ SMM_{Scr} = \varnothing\ then\ "SRS\ information\ is\ sufficient\ for\ Security" \\ else\ "SRS\ information\ is\ insufficient\ for\ Security" \end{array}, \quad (61)$$

- для супроводжуваності:

$$\begin{array}{c} if\ SMM_{Mb} = \varnothing\ then\ "SRS\ information\ is\ sufficient\ for\ Maintainability" \\ else\ "SRS\ information\ is\ insufficient\ for\ Maintainability" \end{array}, \quad (62)$$

- для можливості переносу:

$$\begin{array}{c} if\ SMM_{Pb} = \varnothing\ then\ "SRS\ information\ is\ sufficient\ for\ Portability" \\ else\ "SRS\ information\ is\ insufficient\ for\ Portability" \end{array}. \quad (63)$$

Продукційне правило для формування висновку щодо достатності або недостатності інформації у специфікації вимог для визначення всіх нефункційних характеристик ПЗ:

$$\begin{array}{c} if\ (SMM_{Fs} \cup SMM_{Pe} \cup SMM_{Ub} \cup SMM_{Rb} \cup SMM_{Cb} \cup SMM_{Scr} \cup SMM_{Mb} \cup SMM_{Pb}) = \varnothing \\ then\ "SRS\ information\ is\ sufficient"\ else\ "SRS\ information\ is\ insufficient" \end{array}. \quad (64)$$

Розроблена модель діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ відображає особливості оцінювання достатності інформації для визначення нефункційних характеристик ПЗ та є теоретичним підґрунтям для розроблення методів діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ.

Висновки

Проведений аналіз відомих інтелектуальних агентів на основі онтологічного підходу показав, що вони не розв'язують задачу кількісного оцінювання достатності інформації у специфікації вимог до ПЗ для визначення нефункційних характеристик ПЗ.

У статті вперше запропоновані теоретико-множинні, базові (універсальні) онтологічні моделі нефункційних характеристик ПЗ, а також онтологічні моделі нефункційних характеристик конкретного ПЗ, які ґрунтуються на врахуванні вимог стандартів ISO 25010:2011 та ISO 25023:2016 і забезпечують підґрунтя для вибору достатнього обсягу інформації для оцінювання нефункційних характеристик ПЗ.

Вперше розроблено також модель діяльності інтелектуального агента на основі онтологічного підходу для оцінювання специфікацій вимог до ПЗ, яка ґрунтується на порівняльному аналізі онтологій та є теоретичним підґрунтям для розроблення методів діяльності інтелектуального агента на основі онтологічного підходу.

Література

1. Hastie Shane. Standish Group 2015 Chaos Report – Q&A with Jennifer Lynch [Electronic resource] / Shane Hastie, Stéphane Wojewoda. – Electronic data. – Mode of access: http://www.infoq.com/articles/standish-chaos-2015 (viewed on June 20, 2018). – Title from the screen.

2. A Look at 25 Years of Software Projects. What Can We Learn? – Electronic data. – Mode of access: https://speedandfunction.com/look-25-years-software-projects-can-learn/ (viewed on June 20, 2018). – Title from the screen.

3. McConnell S. Code complete /S. McConnell. – Microsoft Press, 2013. – 896 p.

4. Андон Ф. И. Semantic Web как новая модель информационного пространства Интернет / Ф. И. Андон, И. Ю. Гришанова, В. А. Резниченко // Проблеми програмування. – 2008. – №2-3. Спеціальний випуск. – С. 417-430.

5. Wooldridge M. Intelligent agents – theory and practice / M. Wooldridge, N. R. Jennings // Knowledge Engineering Review. – 1995. – Vol. 10. – Issue 2. – Pp. 115-152.

6. Burov E. Complex ontology management using task models / E. Burov // International Journal of Knowledge-Based and Intelligent Engineering Systems. – 2014. – Vol. 18. – No. 2. – Pp. 111-120.

7. Shostak I. Ontology approach to realization of information technology for normative profile formimg at critical software certification / I. Shostak, I. Butenko // Збірник наукових праць Військового інституту КНУ ім. Т. Г. Шевченка. – 2012. – № 38. – С. 250–253.

8. Бабенко Л. Онтологический подход к спецификации свойств программных систем и их компонентов / Л. Бабенко // Кибернетика и системный анализ. – 2009. – № 1. – С. 180-187.

9. Assawamekin N. Ontology-based multiperspective requirements traceability framework / N. Assawamekin, A. Namfon, T. Sunetnanta, C. Pluempitiwiriyawej // Knowledge Information Systems. – 2010. – No. 3 – Pp. 493-522.

10. Zhang Y. G. Ontological approach for the semantic recovery of traceability links between software artefacts / Y. G. Zhang, R. Witte, J. Rilling, V. Haarslev // IET Software. – 2008. – Vol. 2. – Issue 3. – Pp. 185-203.

11. Leonid K. Ontology and model alignment as a means for requirements validation / K. Leonid, R. Gacitua, M. Rouncefield, P. Sawyer // The 4-th IEEE International Conference on Semantic Computing, September 22-24, 2010 : Proceedings. – Pittsburgh (USA), 2010. – Pp. 46-51.

12. Jin D. Ontology-based software analysis and reengineering tool integration: The OASIS service-sharing methodology / D. Jin, J. R. Cordy // The 21-st IEEE International Conference on Software Maintenance, September 25-30, 2005 : Proceedings. – Budapest (Hungary), 2005. – Pp. 613-616.

13. Bajnaid N. O. An ontological approach to model software quality assurance knowledge domain / N. O. Bajnaid, R. Benlamri, A. Pakstas, Sh. Salekzamankhani // Lecture Notes on Software Engineering. – 2016. – Vol. 4. – No. 3. – Pp. 193-198.

14. Hovorushchenko T. Information Technology for Assurance of Veracity of Quality Information in the Software Requirements Specification / T. Hovorushchenko // Advances in Intelligent Systems and Computing. – 2018. – Vol. 689. – Pp. 166-185.

15. Freitas A. Model-driven engineering of multi-agent systems based on ontologies / A. Freitas, R. H. Bordini, R. Vieira // Applied Ontology. – 2017. – Vol. 12. – Issue 2. – Pp. 157-188.

16. Ossowska K. Exploring an Ontological Approach for User Requirements Elicitation in the Design of Online Virtual Agents / K. Ossowska, L. Szewc, P. Weichbroth, I. Garnik, M. Sikorski // Information Systems: Development, Research, Applications, Education. – 2017. – Vol. 264. – Pp. 40-55.

17. Lezcano-Rodriguez L. A. Ontological characterization of basics of KAOS chart from natural language / L. A. Lezcano-Rodriguez, J. A. Guzman-Luna // Revista Iteckne. – 2016. – Vol. 13. – Issue 2. – Pp. 157-168.

18. Garcia-Magarino I. An ontological and agent-oriented modeling approach for the specification of intelligent Ambient Assisted Living systems for Parkinson patients / I. Garcia-Magarino, J. J. Gomez-Sanz // Hybrid Artificial Intelligent Systems. – 2013. – Vol. 8073. – Pp. 11-20.

19. Wilk S. A task-based support architecture for developing point-of-care clinical decision support systems for the emergency department / S. Wilk, W. Michalowski, D. O'Sullivan, K. Farion, J. Sayyad-Shirabad, C. Kuziemsky, B. Kukawka // Methods of Information in Medicine. – 2013. – Vol. 52. – Issue 1. – Pp. 18-32.

20. Rakib A. A formal approach to modelling and verifying resource-bounded context-aware agents / A. Rakib, R. U. Faruqui // Lecture Notes of the Institute for Computer Sciences Social Informatics and Telecommunications Engineering. – 2013. – Vol. 109. – Pp. 86-96.

21. T. Hovorushchenko. Models and Methods of Evaluation of Information Sufficiency for Determining the Software Complexity and Quality Based on the Metric Analysis Results // Central European Researchers Journal. –2016. – Vol. 2. – Issue 2. – Pp.42-53.

References

1. Shane Hastie, Stéphane Wojewoda. Standish Group 2015 Chaos Report – Q&A with Jennifer Lynch [Electronic resource]. – Electronic data. – Mode of access: http://www.infoq.com/articles/standish-chaos-2015 (viewed on June 20, 2018). – Title from the screen.

2. A Look at 25 Years of Software Projects. What Can We Learn? – Electronic data. – Mode of access: https://speedandfunction.com/look-25-years-software-projects-can-learn/ (viewed on June 20, 2018). – Title from the screen.

3. S. McConnell. Code complete. – Microsoft Press, 2013. – 896 p.

4. F. I.Andon, I. Yu. Grishanova, V. A. Reznichenko. Semantic Web kak novaya model informacionnogo prostranstva Internet. – Problemy Programuvannya. – 2008. – No. 2-3. Special issue. – Pp. 417-430.

5. M. Wooldridge, N. R. Jennings. Intelligent agents – theory and practice // Knowledge Engineering Review. – 1995. – Vol. 10. – Issue 2. – Pp. 115-152.

6. E.Burov. Complex ontology management using task models // International Journal of Knowledge-Based and Intelligent Engineering Systems. – 2014. – Vol. 18. – No. 2. – Pp. 111-120.

7. Shostak, I. Butenko. Ontology approach to realization of information technology for normative profile formimg at critical software certification // Zbirnyk naukovykh prats Viyskovogo instytutu KNU im. T. G. Shevchenka. – 2012. – № 38. – C. 250–253.

8. Babenko L. Ontologicheskiy podhod k specifikacii svoistv programmnih sistem i ih komponentov. – Kibernetyka i systemnyi analiz. – 2009. – No. 1. – Pp. 180-187.

9. N. Assawamekin, A. Namfon, T. Sunetnanta, C. Pluempitiwiriyawej. Ontology-based multiperspective requirements traceability framework // Knowledge Information Systems. – 2010. – No. 3 – Pp. 493-522.

10. Y. G. Zhang, R. Witte, J. Rilling, V. Haarslev. Ontological approach for the semantic recovery of traceability links between software artefacts // IET Software. – 2008. – Vol. 2. – Issue 3. – Pp. 185-203.

11. K. Leonid, R. Gacitua, M. Rouncefield, P. Sawyer. Ontology and model alignment as a means for requirements validation // The 4-th IEEE International Conference on Semantic Computing, September 22-24, 2010 : Proceedings. – Pittsburgh (USA), 2010. – Pp. 46-51.

12. D. Jin, J. R. Cordy. Ontology-based software analysis and reengineering tool integration: The OASIS service-sharing methodology // The 21 IEEE International Conference on Software Maintenance, September 25-30, 2005 : Proceedings. – Budapest (Hungary), 2005. – Pp. 613-616.

13. N. O. Bajnaid, R. Benlamri, A. Pakstas, Sh. Salekzamankhani. An ontological approach to model software quality assurance knowledge domain // Lecture Notes on Software Engineering. – 2016. – Vol. 4. – No. 3. – Pp. 193-198.

14. T. Hovorushchenko. Information Technology for Assurance of Veracity of Quality Information in the Software Requirements Specification // Advances in Intelligent Systems and Computing. – 2018. – Vol. 689. – Pp. 166-185.

15. Freitas, R. H. Bordini, R. Vieira. Model-driven engineering of multi-agent systems based on ontologies // Applied Ontology. – 2017. – Vol. 12. – Issue 2. – Pp. 157-188.

16. K. Ossowska, L. Szewc, P. Weichbroth, I. Garnik, M. Sikorski. Exploring an Ontological Approach for User Requirements Elicitation in the Design of Online Virtual Agents // Information Systems: Development, Research, Applications, Education. – 2017. – Vol. 264. – Pp. 40-55.

17. L. A. Lezcano-Rodriguez, J. A. Guzman-Luna. Ontological characterization of basics of KAOS chart from natural language // Revista Iteckne. – 2016. – Vol. 13. – Issue 2. – Pp. 157-168.

18. Garcia-Magarino, J. J. Gomez-Sanz. An ontological and agent-oriented modeling approach for the specification of intelligent Ambient Assisted Living systems for Parkinson patients // Hybrid Artificial Intelligent Systems. – 2013. – Vol. 8073. – Pp. 11-20.

19. S. Wilk, W. Michalowski, D. O'Sullivan, K. Farion, J. Sayyad-Shirabad, C. Kuziemsky, B. Kukawka. A task-based support architecture for developing point-of-care clinical decision support systems for the emergency department // Methods of Information in Medicine. – 2013. – Vol. 52. – Issue 1. – Pp. 18-32.

19.1. Rakib, R. U. Faruqui. A formal approach to modelling and verifying resource-bounded context-aware agents // Lecture Notes of the Institute for Computer Sciences Social Informatics and Telecommunications Engineering. – 2013. – Vol. 109. – Pp. 86-96.

20. T. Hovorushchenko. Models and Methods of Evaluation of Information Sufficiency for Determining the Software Complexity and Quality Based on the Metric Analysis Results // Central European Researchers Journal. –2016. – Vol. 2. – Issue 2. – Pp.42-53.

Рецензія / Peer review: 22.06.2018
Надрукована / Printed:
Рецензент: д.т.н., завідувач кафедри ТКІТ ХНУ, В.В.Мартинюк