УДК 681.518:658.51

Р. О. ГАМЗАЄВ, мол.н.с. каф. «АСУ», НТУ «ХПІ»;
М. В. ТКАЧУК, д-р техн. наук, проф., каф. «АСУ», НТУ «ХПІ»

МОДЕЛЬ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПОБУДОВИ АДАПТИВНОЇ МАТРИЦІ ТРАСУВАННЯ ВИМОГ У ГНУЧКИХ ПРОЦЕСАХ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розглянуто деякі особливості управління вимогами в гнучких процесах розробки програмного забезпечення (ПЗ) та обґрунтована актуальність проблеми розробки відповідних засобів трасування вимог для забезпечення ефективності їх реалізації. Запропоновано підхід на основі застосування алгоритмічної моделі процесу побудови адаптивної матриці трасування вимог і інформаційної технології накопичення та опрацювання даних щодо вимог, проектних артефактів та виконавців проекту, що взаємодіють в ітераційному процесі розробки ПЗ.

Ключові слова: програмне забезпечення, адаптивна розробка, трасування вимог, функція ступеню інтересу

Рассмотрены некоторые особенности управления требованиями в гибких процессах разработки программного обеспечения (ПО) и обоснована актуальность проблемы разработки соответствующих средств трассировки требований для обеспечения эффективности их реализации. Предложен подход на основе использования алгоритмической модели процесса построения адаптивной матрицы трассировки требований и информационной технологии сбора и обработки данных о требованиях, проектных артефактах и об исполнителях проекта, которые взаимодействуют в итерационном процессе разработки ПО.

Ключевые слова: программное обеспечение, адаптивная разработка, трассировка требований, функция степени интереса

Some features of requirements management in agile software development are considered, and problem actuality to elaborate respective requirements traceability tools for their effective implementation is defined. An approach is proposed, which is based on an algorithm-centered model to build adaptive traceability matrix and on usage of information technology to collect and to process data related to requirements, project artifacts and project's stakeholders participated in iterative software development.

Keywords: software, adaptive software development, requirements traceability, degree of interest function

1. Актуальність проблеми створення модельно-технологічного інструментарію для гнучких процесів розробки програмного забезпечення. Мета дослідження. На сучасному етапі розвитку індустрії програмної інженерії вже фактично є доведеними на практиці переваги гнучких (agile) підходів до розробки програмного забезпечення (ПЗ), до яких належать Scrum, Extreme Programming (XP), Adaptive Software Development (ASD), Feature Driven Development (FDD), Dynamic Systems Development Method (DSDM), Kanban Software Development, Lean Software Development, та деякі інші [1].

Головною причиною такого поширення agile-підходів є невідповідність традиційних методологій розробки ПЗ, таких, як, наприклад, каскадна та

спіральна, або, за висловом Дж. Хайсміта (J. Heighsmith), так званих "монументальних" методологій [2], такими новими чинникам сучасних процесів розробки ПЗ як

- постійне зростання складності, змінюваності та непередбачуваності вимог замовників щодо ПЗ, особливо в таких сферах як розробка Web-застосувань, вбудованих програмні системи (embedded software system), мобільні програмні комплекси (mobile software) та деякі ін.,
- поява «швидких» інструментальних засобів розробки (Rapid Application Development – RAD), які поєднують в собі як можливості безпосередньої розробки програмного коду, так і певні функції управління вимогами та завданнями, що виконуються окремими розробниками проекту (див., напр., в [3–4]).

Спроби вирішити ці проблеми шляхом безпосереднього застосування в програмній інженерії методів проектування та реалізації складних технічних систем виявилися неефективними [2], тому що процес розробки ПЗ є менш формалізованими, із присутністю великого впливу людського чиннику. Все це й викликало потребу в появі саме гнучких методологій проектування ПЗ, основі загальні принципи яких містить документ "The Manifesto for Agile Software Development" [2], який було опубліковано міжнародним співтовариством розробників ПЗ. В ньому міститься 12 базових ознак agile-підходів, серед яких, зокрема, є наступні:

- коректно спроектована та працююча програмна система (ПС) є більш важливою, ніж наявність повного комплекту її документації;
- адекватна та швидка реакція на зміни вимог користувачів важливіше, ніж жорстке виконання попередньо визначеного плану розробки ПС,
- постійне співробітництво проектної групи із замовниками ПС є важливішим, ніж формальне виконання календарного плану проекту;
- особисті професійні якості розробників (тобто їх знання та досвід) та ступінь ефективності організації їх взаємодії є більш важливим чинником успіху виконання проекту, ніж самі технологічні процеси та засоби розробки ПЗ, та ряд інших положень.

Незважаючи на загальну більшу, у порівнянні із іншими, ефективність гнучких методологій, однією із найбільш складних та витратних задач в них залишається управління вимогами (requirements management). Для підвищення ефективності цього процесу застосовуються різні моделі та технології (див., в [3–6]), серед яких досить значне місце займає трасування вимог (requirements traceability), яка, згідно визначення проф. К. Л. Лаврищевої [6], є *"... невід'ємною складовою процесу управління вимогами… для відстежування правильності визначення та реалізації вимог*

до системи ... і зворотний процес відстежування від отриманого програмного продукту до вимог".

Вочевидь, що трасування вимог у гнучких процесах розробки ПЗ має бути також таким, що враховує їх особливості та, в кінцевому рахунку, має сприяти підвищенню ефективності їх застосування, зокрема, шляхом зменшення трудомісткості та скорочення часу, що є необхідним для виконання процедури трасування вимог. Тому метою цього дослідження є розробка моделей та технологічних засобів, які можуть забезпечити вирішення цих задач.

2. Деякі існуючи підходи до трасування вимог та їх особливості в сучасних процесах розробки програмного забезпечення. В семантичному плані трасування є тим чи іншим видом відображення множини вимог у відповідну множину проектних артефактів (фрагментів коду, моделей даних, елементів графічного інтерфейсу та ін.). Традиційно для формалізації цього процесу використовується поняття матриці трасування (*Traceability Matrix – TM*) [7], яка розглядається як відношення

$$TM : (R \times F \to \{0,1\}) \tag{1}$$

де R – це множина вимог;

F – множина програмних артефактів.

Такий підхід дозволяє лише визначити підмножину тих артефактів, що пов'язані з певними вимогами, із застосуванням бінарних значень: "0" – "зв'язок є", "1" – "зв'язку немає", але при цьому не враховує питому вагу або важливість того чи іншого артефакту у проекті, і він був типовим для традиційних методологій розробки ПЗ (див. розділ 1). В роботі [8] проаналізовані деякі інші варіанти побудови *TM*, а також альтернативні технології трасування вимог, такі, як, наприклад, використання асоціативних правил (association rules mining) та патернів трасування (traceability patterns). Також зроблено висновок щодо доцільності побудови так званої розвинутої *TM (Advanced Traceability Matrix – ATM)*, яка на відміну від виразу (1), подається у наступному вигляді

$$ATM : (R \times F \to [0,1]) \tag{2}$$

Тобто елементи цієї матриці відображають зв'язок певного проектного артефакту із кожною з вимог до ПЗ шляхом подання відповідного дійсного числа, що належить до закритого інтервалу їх значень $[0,1]$.

Для обчислення елементів матриці (2) в [8] пропонується застосовувати кількісні значення відповідної функції ступеня інтересу (*Degree-Of-Interest – DOI)* розробника ПЗ (див. в [9]), яка в загальному випадку може бути визначена як:

$$DOI(a_i) = \sum_{j=1}^{N} (e_j^{a_i}) \cdot k_j \qquad (3)$$

де: $a_i \in A$, A – множина артефактів проекту; $e_j \in E$, E – множина подій в процесі розробки; $k_j \in K$, K – множина вагових коефіцієнтів для кожної події (визначається експертним шляхом).

Це, в свою чергу, дозволяє реалізувати у відповідному інструментальному середовищі так званий сфокусований інтерфейс розробника завдань (*task-focused developer's interface*), які мають бути виконані для реалізації вимог щодо кінцевого програмного продукту, тобто: в інтерфейсі розробника проекту (наприклад, в середовищі Eclipse – див. його приклад на рис. 1) будуть відображені тільки ті файли проекту, що мають ступень інтересу вищий за певне значення DOI, яке може бути визначена евристичним шляхом.

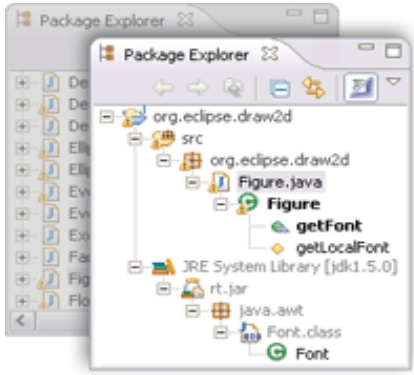

Рис. 1 – Приклад сфокусованого інтерфейсу розробника ПЗ у середовищі Eclipse

На технологічному рівні застосування моделі *DOI* передбачає наявність відповідних програмних засобів, таких, як наприклад, Eclipse Mylyn, огляд деяких з них також наведено в роботі [8].

Запропонований підхід дозволяє, в цілому, отримати значення елементів матриці *ATM* згідно виразу (2), але при цьому він не враховує таких досить важливих особливостей гнучких процесів розробки ПЗ як:

(a) ітеративний характер усього процесу, причому саме із збільшенням числа ітерацій проекту уточняються вимоги користувачів щодо кінцевого програмного продукту;

(b) наявність в групі розробників різних за видами діяльності осіб, наприклад: адміністратор баз даних (database admin), програміст застосувань (developer), дизайнер інтерфейсу користувача (user interface designer) тощо, які: по-перше, можуть змінювати свої ролі

(робочі профілі) і, по-друге, мають різні вагові коефіцієнти відносно ступеню свого інтересу до вимог та відповідних артефактів проекту.

Розглянемо в подальшому один із можливих підходів до побудови так званої адаптивної матриці трасування вимог (*ADaptive Traceability Matrix – ADTM*), значення елементів якої, на відміну від елементів матриці *ATM* із виразу (2), мають враховувати як саму наявність додаткових чинників впливу (a)–(b) так і можливість їх поточних змін при виконанні певного проекту по розробці ПЗ за однією з гнучких методологій (див. розділ 1).

3. Алгоритмічна модель процесу побудови адаптивної матриці трасування вимог Запропонований нижче підхід спирається на основні результати, отримані в роботі [8] та узагальнює їх шляхом розробки *алгоритмічної моделі* (див., напр., в [10]) процесу побудови матриці ADTM. У формалізованому вигляді ця модель *AM* може бути подана у вигляді наступного кортежу

$$AM = \langle InfBase, ToM, Workflow(ADTM) \rangle, \quad (4)$$

де $InfBase$ – інформаційний базис моделі, ToM (Time-oriented Metric) – час-орієнтована метрика для визначення ступеню інтересу розробника щодо певного проектного артефакту, $Workflow(ADTM)$ – алгоритм побудови матриці ADTM.

Інформаційний базис моделі AM визначається кортежем множин

$$InfBase = \langle D, R, F, S, P \rangle, \quad (5)$$

де $D = \{d_i\}, i = \overline{1, I}, I = |D|$ – множина розробників проекту;

$R = \{r_j\}, j = \overline{1, J}, J = |R|$ – множина вимог до ПС;

$F = \{f_k\}, k = \overline{1, K}, K = |F|$ – множина файлів або артефактів проекту,

$S = \{s_l\}, l = \overline{1, L}, L = |S|$ – множина проектних сесій (ітерацій);

$P = \{p_m\}, p = \overline{1, M}, M = |P|$ – множина робочих профілів, або ролей розробників ПС у проекті.

Слід зауважити, що під метрикою ToM у виразі (4) мається на увазі саме метрика програмного забезпечення (software metrics), тобто певна міра, що дозволяє отримати чисельні значення деякої властивості програмного продукту або його специфікації (див., напр., в [6]). Один із найпростіших, але в той же час досить логічно-обґрунтований, спосіб визначити таку метрику для оцінки питомої ваги окремих елементів матриці ADTM полягає в тому, щоб взяти до уваги те, що певні файли (проектні артефакти) вимагають більше часу при роботі над ними, коли мають бути реалізовані відповідні вимоги до ПС. Така час-орієнтована метрика дозволяє відрізнити такі дії розробників ПС, що вносять лише невеликі (“косметичні”) проектні зміни від

тих, що забезпечують значну проектну доробку. Графічно це ілюструє часова діаграма, що наведена на рис. 2.

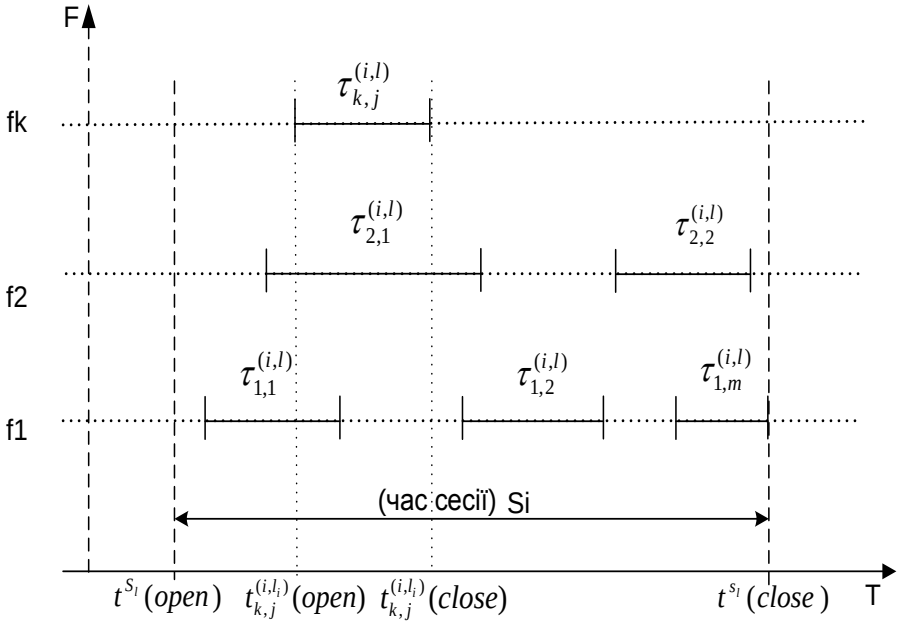

Рис. 2 – Час-орієнтована метрика ступеню інтересу (графічна інтерпретація)

Виходячи з діаграми на рис. 2, кількісно ступень інтересу певного розробника проекту до окремого файлу при реалізації відповідної вимоги може бути розраховано з використанням наступного виразу:

$$\tau_{k,j}^{(i,l_i)} = \frac{t_{k,j}^{(i,l_i)}(close) - t_{k,j}^{(i,l_i)}(open)}{t^{S_l}(close) - t^{s_l}(open)} \qquad (6)$$

де $\tau_{k,j}^{(i,l_i)}$ – інтервал часу виконання розробником $d_i \in D$ проекту відповідних дій над *k*-м файлом $f_k \in F$ під час реалізації вимоги $r_j \in R$ протягом сесії $s_l \in S$; $t_{k,j}^{(i,l_i)}(open)$ – момент часу початку роботи над *j*-ю вимогою *i*-м розробником; $t_{k,j}^{(i,l_i)}(close)$ – момент часу для закінчення роботи над *j*-ю вимогою *i*-*м* розробником; $t^{s_l}(close)$ – час закінчення *li* - проектної сесії; $t^{S_l}(open)$ – час початку *li* - сесії.

Таким чином, метрика ToM із виразу (4), в кінцевому рахунку, є множиною значень, отриманих за допомогою формули (6), а саме:

$$ToM = \{\tau_{k,j}^{(i,l_i)}\}, \tag{7}$$

де визначення відповідних індексних змінних вже були наведені у поясненнях до виразу (5).

Слід зазначити, що метрика (7) також має певні недоліки, а саме: вона не працює у ситуації, коли деякі проектні файли були активовані розробником, але при цьому корисна робота над ними не здійснювалась. Але, з огляду на її логічну обґрунтованість та можливість практичного обчислення, пропонується використовувати саме її для принципової перевірки працездатності запропонованого підходу.

Безпосередньо, сам алгоритм $Workflow(ADTM)$ із виразу (4), який на підставі інформаційного базису (5) із використанням метрики (7) забезпечує побудову матриці *ADTM*, виконується у наступний спосіб:

Крок 1. Для кожного розробника $\forall d_i \in D$, що приймає участь в проекті, будується його локальна матриця $ADTM_{ijk} = ||\tau_{i,j,k,l}||\cdot\gamma$, де $\gamma = \{\gamma_l\}_{l=1}^{L}$, $\gamma_l = \frac{1}{L}$ $l = \overline{1,L}$, елементи якої розраховуються за формулою:

$$\tau_{j,k}^{(i)} = \frac{\sum_{l=1}^{|S|}(\tau_{j,k}^{(i)})_l}{|S|} \tag{8}$$

Крок 2. На підставі локальних агрегованих локальних матриць всіх розробників, шляхом осереднення значень їх елементів будується агрегована сумарна матриця $ADTM_{jk} = ||\tau_{i,j,k,l}||\cdot\lambda$, де $\lambda = \{\lambda_i\}_{i=1}^{I}$ $\lambda = \{\lambda_i\}_{i=1}^{I}$ $i = \overline{1,I}$, коефіцієнти якої можуть бути обчислені за формулою

$$\hat{\tau}_{k,j} = \frac{\sum_{i=1}^{|D|}\tau_{k,j}^{(i)}}{|D|} \tag{9}$$

Крок 3. Відповідно до множини можливих робочих профілей учасників проекту $P = \{p_m\}$, із отриманої на попередньому етапі колекції матриць відбираються їх окремі підмножини: $D = D(p_1) \cup D(p_2) \cup ..D(p_M)$, та на підставі цього будуються окремі локальні агреговані матриці для кожного профілю розробника ПЗ .

$$ADTM_{k,j,D(p_i)} = \frac{\sum_{j=1}^{|D(p_i)|} (ADTM[F,R,D(p_i)])_j}{|D(p_i)|} \quad (10)$$

При цьому для спрощення моделі зроблено припущення, що кожний розробник може належить тільки до однієї групи робочих профілей, тобто: $D(p_i) \cap D(p_j) = \varnothing$.

Крок 4. На підставі отриманих на попередніх кроках результатів, із урахуванням того, що в процесі розробки ПЗ за технологією Scrum один і той же виконавець проекту може змінювати свої робочі профілі, тобто мати різну ступінь інтересу до певних артефактів, остаточно значення елементів матриці *ADTM* знаходяться за наступною формулою

$$ADTM'_{j,k,D(\bar{p}_i)} = \frac{ADTM_{j,k,D(\bar{p}_i)} \cdot K_{role} + ADTM_{j,k,i}}{K_{role} + 1} \quad (11)$$

Таким чином, в результаті виконання алгоритму, визначеного за формулами (8)–(11), побудовано адаптивну матрицю трасування вимог *ADTM*, елементи якої обчислюються за формулою (11), що враховує вплив чинників (a)–(b) (див. розділ 2) на особливості опрацювання вимог та відповідних проектних артефактів у гнучкому процесі розробки ПЗ.

4. Інформаційна технологія для реалізації та застосування адаптивної матриці трасування вимог. Для реалізації розробленої у попередньому розділі алгоритмічної моделі побудови адаптивної матриці трасування вимог пропонується відповідна інформаційна технологія, концептуальна схема якої наведена на рис. 3 в нотації IDEF0 (див., напр., в [3]).

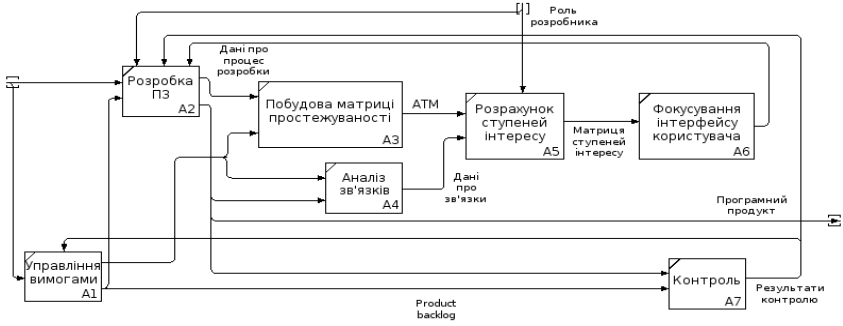

Рис. 3 – Схема інформаційної технології трасування вимог (IDFX0)

Для програмної реалізації цієї технології пропонується інтегрований CASE-засіб, який поєднує в собі функціональні можливості

інструментального середовища Eclipse IDE, системи управління вимогами (це компонент Issue Tracker) та розробленої підсистеми ReqMIT, яка імплементує запропонований у розділі 3 алгоритм побудови матриці ADTM. На рис. 4 показана компонентна програмна архітектура цього CASE-засобу в нотації UML 2.0.

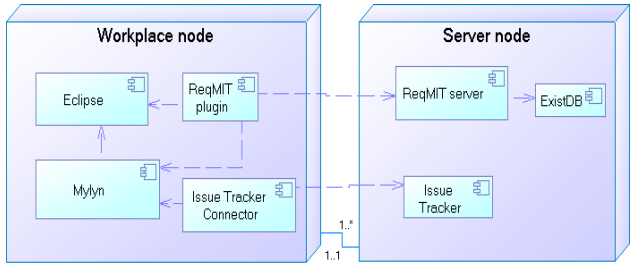

Рис. 4 – Компонентна архітектура інтегрованого CASE-засобу

Клієнтський компонент *ReqMIT_plugin* використовується для побудови локальної матриці трасування для кожного розробника проекту, яка потім передається на сервер. Крім того, цей компонент, в залежності від ролі розробника та його проектного інтерфейсу, фокусує останній саме на тих артефактах, які є релевантними для роботи з відповідними вимогами. Серверний компонент *ReqMIT_server* використовується для зберігання даних щодо вимог, прив'язки вимог до завдань, а також обчислює ступінь інтересу розробника до відповідних проектних артефактів.

Запропонована модель побудови адаптивної матриці трасування та відповідна інформаційна технологія, що забезпечує можливість реалізації сфокусованого інтерфейсу розробника, можуть бути застосовані в загальній схемі автоматизованого управління гнучкою розробкою ПЗ із застосуванням методології Scrum, яка представлена на рис. 5.

У ній, додатково та у контексті двох вже існуючих та добре апробованих на практиці *організаційних циклів* виконання Scrum-проекту, а саме:

1. цикл побудови загального каталогу вимог щодо програмного продукту (*Product Backlog* – PB) та його подальшого опрацювання шляхом формування каталогу вимог для однієї проектної ітерації або одного спринту (*sprint*) у термінології Scrum (*Sprint Backlog* – SB). Тривалість такого спринту часто сягає від двох до чотирьох тижнів в залежності від типи проекту;
2. цикл виконання послідовності щоденних завдань, які є необхідними в рамках однієї проектної ітерації.

Ведено 2 нових технологічних контури управління із зворотним зв’язком, а саме

3. контур управління процедурами пріоритезації та оцінки якості вимог (*requirements priority and quality estimation*), що уможливлює

формування так званого динамічного каталогу вимог РВ, загальна процедура для реалізації цих методів представлена в [11] ;

4. контур управління трасуванням вимог у процесі безпосереднього програмування (виконання завдань проекту), з метою забезпечення ефекту сфокусованого інтерфейсу розробника ПЗ із застосовуванням запропонованої адаптивної матриці трасування ADTM.

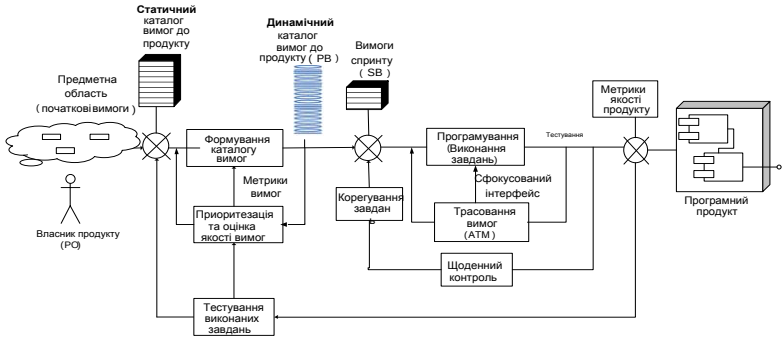

Рис. 5 – Схема автоматизованого управління гнучким процесом розробки ПЗ

Результатом функціонування цієї схеми є програмний продукт, який має задовольняти певним значення метрик якості ПЗ (див. рис. 5). Запропонована схема управління виконанням Scrum-проекту дозволяє застосовувати знання-орієнтовані методи розробки ПЗ, тобто такі, що накопичують і використовують в подальшому для прийняття проектних рішень кількісні оцінки щодо стану виконання окремих етапів проекту та якості отриманих при цьому результатів. Тестування цього підходу, повний опис методики проведення якого та отримані при цьому результати наведено в [8], підтвердило його ефективність, зокрема для цього використовувалися такі показники як точність (precision) та повнота (recall) [12] при визначенні зв'язків між вимогами та проектними артефактами в процесі трасування .

Точність p – це відсоток коректно визначених взаємозв`язків між вимогами та артефактами із загальної кількості знайдених, тоді як повнота $r$ – це відсоток визначених взаємозв`язків, із загальної їх кількості, які дійсно були імплементовані в проекті. Ці значення можуть бути обчислені у наступний спосіб:

$$p = \frac{tp}{tp + fp} \quad , \quad r = \frac{tp}{tp + fn} \quad , \tag{12}$$

де tp – істинно позитивні, розглядаються як цікаві і що використовуються;
fp – хибно позитивні, розглядаються як цікаві, але не використовуються;
fn – хибно негативні, розглядаються як не цікаві, але використовуються.

Середні значення цих показників склали відповідно 0.43 та 0.61, що є цілком прийнятним для позитивної оцінки працездатності запропонованого підходу.

5. Висновки та напрямки подальших досліджень. Таким чином, у даній статті розглянуті деякі особливості управління вимогами в гнучких методологіях розробки ПЗ та запропоновано новий підхід до розробки відповідних засобів трасування вимог з метою забезпечення підвищення ефективності цього процесу. Цей підхід базується на застосуванні алгоритмічної моделі для побудови адаптивної матриці трасування вимог та інформаційній технології накопичення та опрацювання даних щодо вимог, проектних артефактів та виконавців проекту, які взаємодіють в ітераційному процесі розробки ПЗ за гнучкою методологією Scrum.

Для розвитку цього підходу в подальшому планується розробити моделі та експериментально дослідити ефективність застосування інших (у порівнянні із час-орієнтованими) метрик визначення ступеню інтересу розробника ПЗ у контексті запропонованої адаптивної моделі трасування вимог.

Список літератури: **1.** *Амблер С.* Гибкие технологии: экстремальное программирование и унифицированный процесс разработки. / *С. Амблер.* – СПб.: Питер, 2005. – 412 с. **2.** *Highsmith J.* Agile Project Management. / *Highsmith J.* – Addison-Wesley, 2004 – 277 p. **3.** *Соммервил И.* Инженерия программного обеспечения. : пер. с англ. / *И. Соммервил*. – М.: Изд. Дом «Вильямс», 2002. – 624 с. **4.** *Коваль Г. І.* Удосконалення процесу розробки сімейств програмних систем елементами гнучких технологій / *Г. І. Коваль, А. Л. Колесник, К. М. Лавріщева* // Проблемы программирования. – К.: НАН України. – 2010. – № 2–3 (спец. выпуск). – С. 261–270. **5**. *Вигерс К.* Разработка требований к программному обеспечению.– М. ИТД «Русская редакция», 2004. – 576 с. **6.** Електронний підручник "Програмна інженерія" Київського національного університету ім. Т. Г. Шевченка – http://www.programsfactory.univ.kiev.ua/content/books/2 – переглянута 03.09.2012 **7.** *Ramesh B.* Toward Reference Model for Requirements Traceability / *Ramesh B., Jarke* M. // IEEE Transactions on Software Engineering, 2001, 27(1), pp. 58–93. **8.** *Tkachuk M. V.* Models and Tools for Effectiveness Increasing of Requirements Traceability in Agile Software Development / *Tkachuk M. V., Gamzayev R. O., Mayr H. C., Bolshutkin V. O.* // Проблемы программирования. – К.: НАН України. – 2012. – № 2–3 (спец. выпуск). – С. 252 – 260. **9.** *Kersten M.* Mylar: a Degree-of-Interest Model for IDEs / Kersten, M., Murphy, G. C. // Proceedings of AOSD'05, New York, USA, 2005, pp. 159–168. **10.** *Сергиенко А. М.* Алгоритмические модели обработки потоков данных / *А.М. Сергиенко, В. П. Симоненко* // Электронное моделирование. – 2008 – Т. 30, №6. – С. 49–60. **11.** *Ткачук Н. В.* Процедура построения динамического каталога требований при разработке программного обеспечения по методологии Scrum / *Ткачук Н. В., Гнатенко М. Г., Гамзаев Р. А.* // Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: Тези доповідей XX міжнародної науково-практичної конференції, Ч.I (15–17 травня 2012р., Харків). / за ред. проф. *Товажнянського Л. Л.* – Харків, НТУ «ХПІ». – С. 30. **12.** *David L.Olson,* Advanced Data Mining Techniques / *David L. Olson //* Springer, 2008. – 180 p.

Надійшла до редколегії 07.12.2012