ТЕОРЕТИЧНІ ОСНОВИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 004.415.2 (045)

Сидоров Є.М.
Національний авіаційний університет

МЕТОД ІДЕНТИФІКАЦІЇ ВТРАТ У БЕРЕЖЛИВІЙ РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Частою є ситуація, коли у розробника є декілька активних проектів. Наприклад, прагнення використати колектив на усіх етапах agile розробки веде до того, що одні і ті ж люди будуть задіяні в декількох проектах. Тоді, способом організації роботи, може бути одночасна реалізація проектів перемикаючись між ними. Пропонується метод ідентифікації втрат у бережливій розробці програмного забезпечення. Метод ґрунтується на гіпотетично-дедуктивної моделі наукового методу і тому містить дві головних складових. Перша, гіпотези щодо втрат у бережної розробці програмного забезпечення. Друга, експеримент щодо перевірки гіпотез. В ході експериментів виконується огляд, який дозволяє ідентифікувати втрати. Застосування методу розглядається на прикладі втрат при перемиканні завдань. Наведено види діяльності, які ведуть до втрат та типи втрат. Запропонований в роботі метод може застосовуватися не тільки для дослідження втрат переміщення (до яких відносяться розглянути перемикання між завданнями) але і до інших видів втрат.

Частой является ситуация, когда у разработчика есть несколько активных проектов. Например, стремление использовать коллектив на всех этапах agile разработки ведет к тому, что одни и те же люди будут задействованы в нескольких проектах. Тогда, способом организации работы, может быть одновременная реализация проектов переключаясь между ними. Предлагается метод идентификации потерь в щадящей разработке программного обеспечения. Метод основан на гипотетически-дедуктивной модели научного метода и поэтому содержит две главных составляющих. Первая, гипотезы, о потерях в бережной разработке программного обеспечения. Вторая, эксперимент, по проверке гипотез. В ходе экспериментов выполняется уход, который позволяет идентифицировать потери. Применение метода рассматривается на примере потерь при переключении задач. Приведены виды деятельности ведущих к потерям и типы потерь. Предложенный в работе метод может применяться не только для исследования потерь перемещения (к которым относятся рассмотреть переключения между задачами) но и к другим видам потерь.

Often there is a situation where the developer has several active projects. For example, the desire of the staff in all phases of agile development is the fact that the same people will be involved in several projects. Then, the way of organizing works, can be both designs of switching between them. The method of identification of wastes in lean software development is proposed. The method is based on the hypothetical-deductive model of scientific method and therefore has two main components. First, the hypotheses regarding wastes in lean software development. Second, an experiment to verify the hypothesis. In experiments is performed identification of wastes. Application of the method is considered for example on switching of problems. Types of activities leading to wastes and types of wastes is presented. The method can be applied not only to study the wastes of movement (which include consider switching between tasks) but also to other types of wastes.

Ключові слова: *сталий розвиток, програмне забезпечення, бережлива розробка програмного забезпечення, втрата.*

В контексті сталого розвитку має місце бережлива розробка програмного забезпечення, основне завдання якої зменшити втрати або відходи [1, 2, 3]. Перевірку припущень про появу відходів в процесах розробки можна здійснити, шляхом експериментального дослідження in vivo. Проте, подібне дослідження, але in vitro

поліпшить розуміння причин, що призводять до появи втрат, дозволить їх ідентифікувати, дасть можливість оцінити втрати кількісно. У статті пропонується метод щодо таких досліджень. Метод демонструється на прикладі втрат від перемикання між завданнями [2].

Вступ

Частою є ситуація, коли у розробника є декілька активних проектів. Наприклад, прагнення використати колектив на усіх етапах agile розробки веде до того, що одні і ті ж люди будуть задіяні в декількох проектах. Тоді, способом організації роботи, може бути одночасна реалізація проектів перемикаючись між ними. Організація роботи колективу, використовуючи перемикання між проектами критикується, оскільки терміни виявляються більшими, з'являються втрати [2].

Робіт, що вивчають вплив перемикання завдань на процес розробки програмного забезпечення немає, проте, схожі дослідження існують в інших галузях. Так, в роботі [4], проблема перемикання між завданнями досліджується в цілому. У роботі [5], розглядаються аспекти перемикання (середовище, причини, стратегії) в контексті взаємодії «людина-комп'ютер». У роботі [6], у рамках експерименту представлено результати дослідження перемикання між задачами під час робочого дня офісного робітника (зустрічі, телефонні дзвінки, робота з електронною поштою). У роботі [7], запропоновано поняття графа діяльностей як засобу представлення і розуміння мікроструктури комплексної роботи. І, хоча подібні дослідження підтверджують існування втрат, викликаних перемиканням, в інженерії програмного забезпечення є необхідність їх ідентифікувати і оцінювати.

Метод

Метод ґрунтується на гіпотетично-дедуктивної моделі научного методу і тому містить дві головних складових. Перша, гіпотези, щодо втрат у бережної розробці програмного забезпечення. Друга, експеримент, щодо перевірки гіпотез. В ході експериментів виконується догляд, який дозволяє ідентифікувати втрати.

Гіпотези

Робочі та програмні продукти, що розробляються, характеризуються значним обсягом інформації (програмний код, специфікації, прийняті рішення, плани розробки і тестування). При перемиканні з одного проекту на інший, робітникові знадобиться час, щоб освоїти (згадати, відновити) цю інформацію. Сформулюємо три гіпотези.

Гіпотеза 1 – перемикання з проекту на проект вимагає час, а, отже, розробка проектів на перемиканнях, при інших рівних, умовах закінчиться пізніше, ніж розробка цих же проектів без перемикань. Тоді $T(n) > T((n)$, де T(n) – це час, витрачений учасником на виконання завдань з перемиканням; n – категорія складності завдань; T'(n) – це час, витрачений на виконання тих же завдань, але послідовно. Різниця T(n) – T'(n), це затримка (часова втрата), викликана перемиканням. Оцінка кількості часових втрат в процентному співвідношенні має вид:

$$S(n) = \frac{T(n) - T'(n)}{T(n)} \mathbf{100.}$$

Гіпотеза 2 – обсяг робіт, необхідний для реалізації проекту залежить від його складності, тому існує залежність між втратами від перемикання між проектами і складністю проектів.

Гіпотеза 3 – існує кореляція між складністю завдань і часовими втратами, викликаними перемиканнями, а саме, якщо . Із зростанням складності завдань росте час необхідній на виконання перемикання.

Експеримент

Цілі експерименту. Схему експерименту наведено на рис. 1. Цілями експерименту в контексті перемикань є наступні: виявлення часових втрат, ідентифікація і оцінка втрат. Перша мета досягається шляхом порівняння послідовної роботи над декількома проектами і роботи з перемиканнями проектів. Досягнення другої мети здійснюється на основі результатів експериментів по досягненню першої мети, шляхом догляду та кількісної і якісної оцінки втрат.

Учасники. Для виконання експерименту in vitro, за для перевірки гіпотези та ідентифікації і оцінки часових втрат в ролі учасників можуть виступати, наприклад, студенти старших курсів, знайомі з програмуванням. Рівень підготовки учасників виявляється заздалегідь і має бути приблизно однаковий. Оцінка рівня підготовки учасників може бути зроблена за допомогою тестування. Бажано, щоб кількість учасників була кратна двом.

Підготовка завдань. Перед початком експериментатор готує завдання, які відрізнятимуться по категоріях складності, але

повинні не занадто відрізнятися у рамках категорії. Мінімальна кількість категорій три, наприклад, легка, середня, важка. Складність завдань оцінюється експертно. У легкій категорії можуть бути завдання, виконання яких займе 1-3 години (реалізація окремої функції програми). Наприклад, реалізація алгоритму збереження рядка у файлі. У середню категорію входять завдання, що вимагають для виконання половину дня або день (реалізація окремої функції застосунку). Наприклад, реалізація програми побудови діалогового вікна збереження, з можливістю вибору і розміщення кодування і файлу. У важкій категорії завдання виконуються впродовж декількох днів (невелике застосування). Наприклад, написання програми текстового редактора типу "Блокнот".

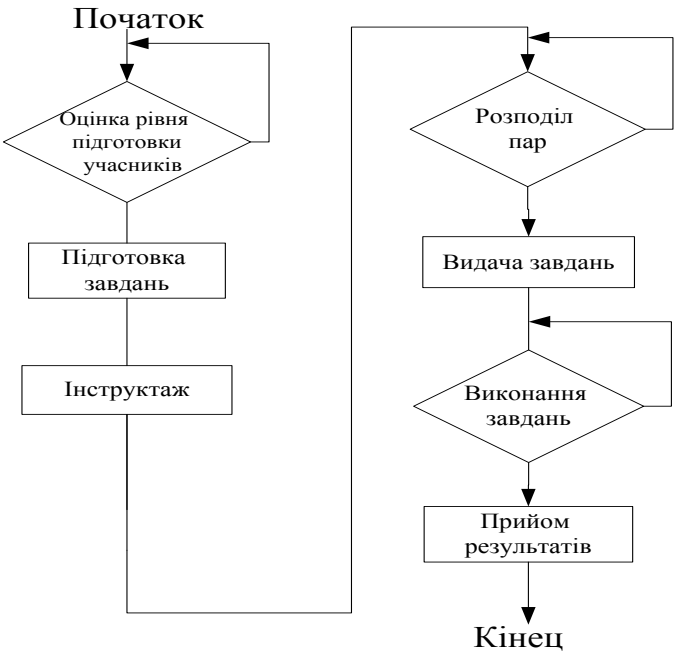

Рис. 1. Схема експерименту

Інструктаж. Учасники повинні дотримуватися схеми експерименту і виконувати його вимоги, тому вони інформуються, наприклад, про те, як правильно рахувати чистий час роботи і чому важливо чітко дотримуватися схеми експерименту. Завдання, виконані за нереальний час, не розглядаються. Можна попросити учасників оцінити при обліку часу похибку, що припустилася. Це підвищить якість оцінки. Учасники також мають бути проінформовані про те, як їм слід організувати процес. Наприклад, учасник, використовуючий метод Test - Driven Development може отримати якісніший результат шляхом написання тестів, але це, ймовірно, позначиться на загальному часі виконання, і спотворить результати, якщо інший учасник в парі не робив тестування взагалі.

Розподіл пар. Учасники експерименту діляться на пари. Важливо, щоб до початку дослідження, експериментатор мав інформацію щодо вмінь та досвіду кожного учасника. Якщо в одну пару будуть об'єднані початкуючий програміст і програміст з деяким практичним досвідом, швидше за все у досвідченого програміста час виконання буде менше, навіть якщо він перемикатиметься із завдання на завдання. Тому, слід забезпечити рівний рівень підготовки учасників, як в парі, так в категорії складності, і розподіляти учасників випадковим чином.

Видача завдань. Кожна пара отримує по два завдання відповідної складності. Одна людина

в парі виконуватиме завдання послідовно, а інша, з певним інтервалом перемикатиметься між завданнями. Інтервал перемикання обирається, виходячи із загальної тривалості завдання. Бажано, щоб кількість таких перемикань була однакова для кожної категорії складності. При виборі тривалості інтервалу слід врахувати тривалість завдання легкої складності.

Виконання завдання. Учасникам слід вести облік часу витраченого на виконання завдань.

Прийом результатів. Перевірка якості завдань може бути виконана за допомогою формалізованих приймальних тестів, працюючих по методу чорного ящика або оцінкою експериментатором. Завдання, якість яких не відповідає певному мінімальному рівню, відкидаються.

Представлення діяльностей і ідентифікація втрат

Для представлення діяльностей можна використати граф діяльностей [7], який пропонується розширити додатковим виміром проекти (рис.2). За допомогою графу можливе представлення і аналіз складних діяльностей, на приклад, довжина діяльності, складність, частота переривань за видом діяльності.

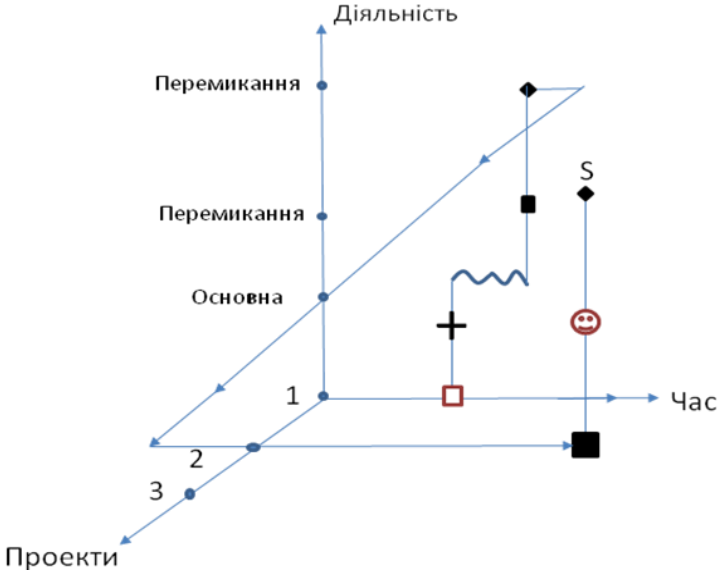

Рис.2 Граф діяльності

За результатами експериментів ідентифіковані деякі види діяльності, які ведуть до втрат у будь-яких процесах при розробці програмного забезпечення (табл.1) [3].

Проведені експерименти дозволили ідентифікувати втрати (табл. 2) [1, 8].

Таблиця 1

Види діяльності що ведуть до втрат

№ п/п	Позначення на графі	Вид діяльності	Зміст діяльності
1	□	Календарь	Моніторинг, заповнення
2	■	Email	Читання, написаннння
3	☺	Розмова	Діалог, конференція
4	S	Самоперемикання	За розкладом, випадково
5	E	Зовнішні перериванння	Пристрій, коллеги, відвідувач, діяльність
6	◆	Переривання	Будь-яке переривання
7	—	Зустріч (конференція)	На місті або з виїздом
8	+	Папір	Підготовка тексту
9	≀	Читання	Читання документації

Наведені в табл.1 види діяльності безпосередньо ведуть до втрат часу, але вони також пов'язані з іншими типами втрат (табл.2).

Таблиця 2
Типи втрат

№ п/п	Тип втрат	Характер втрат
1	Часові	Затримки, очікування, зрив графіків
2	Матеріальні	Папір, картріджи, паливо, вода.
3	Енергетичні	Електроенергія, тепло.
4	Фінансові	Витрати на діяльність, яка веде до втрат (табл. 1)

Висновки

На сьогодні ідентифіковано сім видів втрат, які можуть мати місце в процесах розробки програмного забезпечення [2]. Але жоден з цих видів не досліджено. Запропонований в роботі метод може застосовуватися не тільки для дослідження втрат переміщення (до яких відносяться розглянути перемикання між завданнями) але і до інших видів втрат.

Список використаних джерел

1. Сидоров Н.А. Экология программного обеспечения/ Н.А. Сидоров/. – Інженерія програмного забезпечення . – № 1. – 2010. – С. 53 – 61.
2. Попендик М. Бережливое производство программного обеспечения/ Попендик Т/ – М.: Вильямс, 2010. – 256 с.
3. Сидоров Є.М. Класифікація відходів для розробки програмного забезпечення. //Сидоров Є.М./ – Вісник Східноукраїнського національного університету імені В. Даля. – №3. – 2013. – С.20 – 23.
4. Monsell S. Task switching.-TRENDS in Gognitive Sciences/ S.Monsell /. – V.7. – N.3.- 2003. – P.134 – 140.
5. Bondarenko O. Task switching in Detail: Task Identification, Switching Strategies and Influencing Factors/ O.Bondarenko /. – Eindhoven Univ.of Techn. – 2004. – 11p.
6. Czerwinski M. A Diary Study of Task Switching and Interruptions. – Microsoft Research /E.Horvitz, S.Wilhite /. – 2001. – 12p.
7. Muller M.J. Activity Graphs of the Microstructure of Complex Work. IBM Research /M.J. Muller/. – 2004. – 3p.
8. Mark G. The Cost of Interrupted Work: More Speed and Stress/ D.Gudith., U.Klocke/ Universiti of California. – 2005. – 5p.

Відомості про автора:

Сидоров Євген Миколайович – кандидат технічних наук, доцент кафедри комп'ютерних інформаційних технологій Інституту комп'ютерних інформаційних технологій Національного авіаційного університету. Наукові інтереси: інженерія програмного забезпечення.

E-mail: eugen.sidorov@live.com