УДК 004.023

А.Й. Савицький, А.С. Петров

ДОСЛІДЖЕННЯ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРИ ВИКОРИСТАННІ ПАРАЛЕЛЬНО ПОСЛІДОВНОГО ПІДХОДУ

Анотація: Стаття присвячена питанню використання паралельного послідовного підходу до розробки програмного забезпечення (ПЗ). Досліджена гнучка методологія розробки ПЗ. Запропоновано використання паралельно послідовного підходу до проектів з розробки ПЗ. Модифіковано алгоритм Кернігана-Ліна балансованого розбиття графу. Побудована імітаційна модель гнучкого процесу розробки ПЗ, на якій перевірено запропоновану модифікацію.

Ключові слова: розробка ПЗ, паралельна розробка, балансоване розбиття графу, імітаційне моделювання.

Вступ

На сьогодні література з управління інформаційними системами наводить безліч прикладів провалених у часі проектів з розробки ПЗ [1]. Велика швидкість зміни ринку ПЗ призводить до корегування вимог та створення нових у процесі розробки, що відбивається на оцінках часу виконання проектів. До того ж складність ПЗ постійно зростає і унеможливлює точні оцінки, що ґрунтуються на попередньому досвіді. Все це висуває час розробки ПЗ на перше місце серед інших обмежень проекту [2].

Метою статті є демонстрація можливості зменшення тривалості проекту з розробки ПЗ за рахунок використання послідовно паралельних підходів на імітаційній моделі.

Об'єктом дослідження є процес розробки програмного забезпечення. Предметом дослідження є тривалість процесу розробки ПЗ, як проекту. Результати дослідження можуть бути використані в галузі управління проектами з розробки ПЗ.

Огляд існуючих підходів до вирішення задачі

Гнучка розробка програмного забезпечення (англ. Agile software development) показала себе, як ефективний інструментарій для швидкого реагування на зміни та зменшення часу розробки ПЗ [3]. Саме тому дослідження, що представлене у даній статті базується на даній методології.

Одним з методів скорочення тривалості розробки ПЗ, що добре себе зарекомендував, є послідовно паралельний підхід [4]. Основою

для використання паралельно послідовного підходу є розбиття даної програмної системи на частини, що можуть бути розроблені паралельно за умови мінімізації затрат на розбиття. Якщо представити програмну систему, як сукупність зв'язаних модулів, то видно, що задача зводиться до балансованого розбиття графу.

На міжнародному симпозіумі з математики у Словаччині 2012 року було показано, що при умові зважених вершин (у нашому випадку – часу розробки модуля) задачі балансованого розбиття графу відносяться до NP-повних та не мають ефективного рішення за поліноміальний час [5].

Існує ряд евристичних алгоритмів (Кернігана-Ліна, Фідуччіа-Маттеєса, спектральний, багаторівневий), які дають прийнятний результат за поліноміальний час. Алгоритми розбивають граф на задану кількість підграфів з однаковою кількістю вершин. У нашому випадку необхідна модифікація, що буде розбивати з умовою однакової ваги підграфів.

Постановка задачі

Розглянемо ПЗ, як систему, що складається з N модулів та залежностей між ними. Існує ряд базових модулів системи, що не мають залежності від інших, і повинні бути розроблені в першу чергу. Всі ж інші приймають на вхід інформацію з деякої кількості модулів K. Кожен модуль системи має один вихід U, що є входом Vj $(0 < j \leqslant K)$ іншого модуля, або ж виходом системи.

Отже, систему, що складається із залежних модулів можна представити у вигляді орієнтованого ациклічного графу. Кожен модуль можна представити у вигляді елемента M з набором входів V та виходом U, як показано на рисунку 1.

Базові модулі системи можна розробляти паралельно з самого початку, інші ж модулі потребують інформації для входів. Це може бути інформація з виходу розробленого і протестованого модуля, або, як було зазначено, інформація з виходу заглушки.

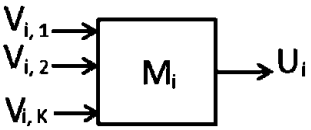

Рис. 1 – Схематичне зображення модуля системи

Для ефективного паралельного виконання робіт з розробки та тестування, система повинна бути розбита на повністю незалежні частини в залежності від кількості наявних людських ресурсів.

Зв'язки між частинами будуть розриватися за рахунок заглушок. Теоретично будь-який модуль можна замінити заглушкою з константним виходом. Проте, якщо від замінюваного модуля будуть залежить багато інших, то затрати часу на їх тестування і

налагодження при оберненій заміні на практиці будуть перевищувати виграш паралельної розробки. Тобто, мінімізація кількості входів системи, що будуть отримувати інформацію від заглушок є одним із критеріїв розбиття.

Кількість частин системи, що повинні бути створені внаслідок розбиття, - це величина, що залежить від кількості наявних людських ресурсів, так як робота одного розробника над двома паралельними частинами не приносить виграшу часу. Так само велика кількість розробників, що працюють над однією частиною, буде призводити до простоїв робочою сили за рахунок зв'язності робіт.

Кожен модуль характеризується оціночним часом, що треба затратити для його розробки та тестування. Для ефективної паралельної розробки сума всіх оцінок часу по частинам системи повинна буде приблизно рівною або кратною. В такому випадку робота над усіма гілками паралельного процесу буде завершена одночасно.

Отже, при розбитті системи повинно бути враховано три критерії:

- кількість входів модулів системи, що отримують інформацію від заглушок (позначимо, як Z);
- кількість підсистем для розбиття (L);
- оціночна трудомісткість розробки підсистеми (T_i).

Як було зазначено, перший критерій повинен бути мінімізований. Кількість підсистем для розбиття задається константою в залежності від наявності людських ресурсів. Оціночна трудомісткість розробки підсистем задається для кожної підсистеми, як сума оціночних трудомісткостей розробки всіх модулів, що входять до підсистеми.

Отже, задачу розбиття системи на підсистеми можна представити наступним чином:

$$\bigcup_{i}^{N} M_i = M. \tag{1}$$

$$\forall i \neq j \; m_i \cap M_j = \emptyset. \tag{2}$$

$$\min \sum_{i=1}^{N} \left| \frac{\sum_{j=1}^{L} T_j}{L} - T_i \right|. \tag{3}$$

$$\min Z. \tag{4}$$

Експериментально-дослідне вирішення задачі

Відмінністю від оригінальної задачі про розбиття графа на підграфи є те, що граф орієнтований (важливими є дуги, що входять до підграфу), а також вершини мають ваги.

Була зроблена модифікація алгоритму Кернігана-Ліна. Оцінка приросту за обмін була розрахована з урахуванням тільки вхідних зв'язків підграфа. До того ж, окрім обміну вершинами додалася операція перенесення вершини з урахуванням її ваги.

Модифікований алгоритм був протестований на найпоширеніших та крайніх випадках структури програмної системи:

- послідовність модулів;
- несильно зв'язані гілки;
- середня за розміром система з приблизно однаковою трудомісткістю модулів;
- середня за розміром система з модулями, що значно відрізняються по трудомісткості;
- сильно зв'язаний граф.

Для демонстрації і перевірки можливості зменшення тривалості проекту з розробки ПЗ за рахунок використання послідовно паралельного підходу з наведеним вище алгоритмом була розроблена імітаційна модель. Її структура наведена на рисунку 2.

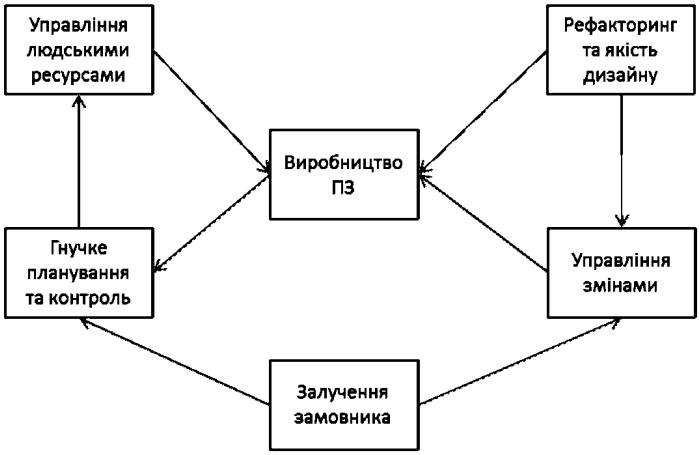

Рис. 2 – Структура імітаційної моделі

Модель, що розроблена в цьому дослідженні, інтегрує основні практики Agile розробки, такі як: гнучкого планування, коротких ітерацій, залучення клієнтів, рефакторинга, модульного тестування і парного програмування. Ці практики моделюються в чотирьох секторах: залучення клієнтів, управління змінами, гнучке планування і контроль, рефакторинг і якість дизайну. Підсистеми управління людськими ресурсами та виробництва програмного забезпечення, які необхідні для підтримки швидкої розробки, були

створені шляхом адаптації та розширення роботи Абдель Хаміда та Лейді [6].

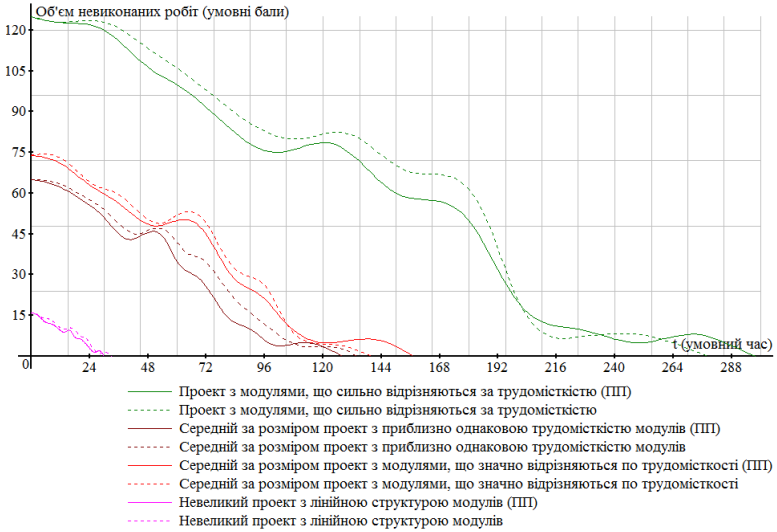

Рис. 3 – Залежність об’єму невиконаних робіт від часу

На рисунку 3 показано графіки залежності об’єму невиконаних робіт від часу (іншими словами – прогрес проекту) для проектів з різним типом структури програмної системи без використання послідовно паралельного підходу та з ним. Як видно з графіка час зменшення об’єму невиконаних робіт (тривалість проекту) помітно зменшується для систем, структура яких представляє собою послідовність модулів та для несильно зв’язаних систем. Це пояснюється тим, що розбиття з великою кількістю заглушок приносить додаткові затрати часу на їх розробку та повторне тестування.

Висновки

Дана стаття демонструє можливість зменшення тривалості проектів з розробки за рахунок використання послідовно паралельної розробки, зокрема заміни модулів, що будуть розроблені, пізніше заглушками, задля розбиття система на частини, що можуть бути розроблені паралельно.

Експеримент на імітаційній моделі розробки ПЗ за гнучкою методологією показав, що при помірній зв’язності модулів системи паралельно послідовний підхід зменшує тривалість проекту на час до 10% від початкової тривалості, при умові, що існує достатня кількість людських ресурсів для ведення паралельної розробки.

Список використаних джерел

1. *Лешек А.М.* Анализ требований и проектирование систем. Разработка информационных систем с использованием UML / Лешек А.М. – Вильямс, 2002 – 432 с.
2. *Duggal J.S.* Next Level Up: How Do You Measure Project Success? / Project Management Institute // [електронний ресурс] Режим доступу:
3. http://www.pmi.org/Knowledge-Center/Next-Level-Up-How-Do-You-Measure -Project-Success.aspx
4. *Mayer T.* The People's Scrum: Agile Ideas for Revolutionary Transformation / Dymaxicon , 2012 – 170 C.
5. *Andrew Funk та ін.* Analysis of Parallel Software development using the
6. relative development Time Productivity Metric. / CTWatch Quarterly – November 2006, C. 46-51.
7. *Feldmann A.E.* Fast Balanced Partitioning is Hard, Even on Grids and Trees / Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science (Bratislava, Slovakia) // 2012, C. 372 – 382.
8. *Abdel-Hamid T. K., Leidy F.H. An expert simulator for allocating the quality assurance effort in software development / Simul – 1991, C. 233–240.*

Отримано 09.09.2014 р.