УДК 004.89

СЕМЧЕНКО А.О.,
ОЛІЙНИК Ю.О.

МОНІТОРИНГ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ

Публікація присвячена проблемі ефективного вимірювання та моніторингу процесу розробки в програмній інженерії. Метою є покращення процесу моніторингу розробки програмного забезпечення. Запропоновано метод моніторингу процесу розробки програмного забезпечення на основі аналізу текстових даних, що виникають в процесі розробки. Результати проведених експериментів показують, що використання запропонованого методу дозволяє пришвидшити моніторинг процесу розробки програмного забезпечення на 22%.
Ключові слова: програмні метрики розробки, моніторинг процесу розробки, керування проєктом, сентимент-аналіз.

The publication is dedicated to the problem of adequate measurement and monitoring of the development process in software engineering. The goal is to improve monitoring of the software development process. It is proposed that the method of monitoring the software development process is based on the textual data analysis of the data emerging during the development process. The results of the experiments show that the use of the proposed method can speed up the monitoring of the software development process by 22%.
Keywords: software development metrics, development process monitoring, project management, semantic analysis, sentiment analysis.

1. Вступ

Наразі, широкої популярності набули так звані «гнучкі» методології розробки, засновані на Agile-принципах. Активно впроваджуються механізми зворотного зв'язку, призначені для корегування процесу розробки та покращення розуміння того, що відбувається на кожній конкретній ітерації. Розроблена обширна наукова теоретична база, що описує підходи до проведення вимірювань процесів розробки та продуктів [1] [2]. Зазвичай, для вимірювання програмних процесів або продуктів вводяться певні метрики, значення яких відстежують час від часу. Однак, окрім метрик, ще одним цікавим джерелом інформації про розробку є численні артефакти, що виникають в її процесі. Наприклад, задачі та їх обговорення у вигляді коментарів, що зберігаються в системі керування задачами. Наразі наукова база по методам моніторингу розробки програмного забезпечення на основі аналізу текстових артефактів розроблена дуже слабко. Саме цій проблемі присвячена дана публікація. Проблемі вимірювання та моніторингу в програмній інженерії присвячено багато досліджень та публікацій. В роботі [3] проведено огляд основних принципів та підходів до проведення вимірювань у програмній інженерії. Наведено модель програмних вимірювань, описано класифікацію програмних метрик та запропоновано принципи, якими потрібно керуватись при проведенні вимірювань в програмній інженерії.

Метод GQM [4] (Goal-Question-Metric) активно використовує агенція NASA для свої проєктів. Це є метод створення нових програмних метрик, при проведені вимірювань програмних проєктів. У книзі [5] наведено підхід до контролювання процесу розробки та проведення аналізу використовуючи програмні метрики. Публікації [6-8] містять опис способів статистичної обробки результатів вимірювання програмних метрик, а саме агрегація та нормалізація накопичених даних. У статтях [9-10] наведено інформацію, про використання семантичних методів для обробки текстових даних та підходи до збирання таких даних (text mining).

Проте наразі не існує методів, що дозволяють здійснювати моніторинг процесу розробки програмного забезпечення базуючись на аналізі текстових артефактів, що виникають в процесі розробки. Тому виникла необхідність розробити такий метод.

2. Запропоноване рішення

Для забезпечення кращого огляду процесу розробки програмного забезпечення пропонується використати обсяг текстових даних, що накопичується в процесі розробки. Текстові дані виникають, наприклад, в процесі обговорень задач в системах керування задачами TMS (Task Management System) [11] у вигляді опису та коментарів.

Розроблений метод моніторингу процесу розробки програмного забезпечення складається з наступних основних етапів:

- Оцінка ризику порушення термінів виконання задачі;
- Виявлення основних обговорюваних тем;
- Виявлення категорій та підкатегорій серед обговорюваних термінів.

3. Оцінка ризику порушення термінів виконання задачі

У процесі виконання задачі працівник може стикнутися зі складнощами, що призводять до неможливості успішного виконання завдання або ж порушення термінів його виконання. Або ж у процесі обговорення задачі між працівниками може виникнути конфліктна ситуація, що знову ж таки створює ризик порушення термінів виконання. В обох випадках для обговорення такої задачі характерні аномальні емоційні показники.

Завчасне втручання проєктного менеджера у процес виконання такої задачі дає можливість зекономити робочий час та зменшити ризики, пов'язані зі зривом термінів виконання задачі. Тобто, існує необхідність оцінювати ризик порушення термінів виконання задачі. Для виявлення таких проблемних задач пропонується використати аналіз тональності тексту (сантимент-аналіз).

У процесі обговорення задачі загалом тональність коментарів може змінюватися в залежності від успішності її виконання. Причому для оцінки ризиків більш релевантними є останні коментарі, аніж перші. Наприклад, на початку виконання задачі працівники стикались зі складнощами, через що текст коментарів мав чітко виражене негативне забарвлення. Однак, через деякий час проблема була вирішена і тональність коментарів змінилась на нейтрально-позитивну. Оцінюючи ризики невчасного виконання задачі, слід врахувати даний аспект.

Тому, для оцінки ризику невчасного виконання задачі пропонується розраховувати зважену інтегральну оцінку тональності кожного текстового документа, що відноситься до задачі.

$$R = \frac{2}{N * (N+1)} \sum_{i=1}^{N} i * c_i,$$

де $c_i -$ показник compound i
$-$ ого документа,
$N -$ кількість документів,
$R -$ інтегральний показник ризику

Далі необхідно від числового значення перейти до якісної оцінки ступеня ризику. Для цього пропонується таблиця переходу (табл.1).

Табл. 1. Якісна шкала ступеня ризику

Рівень ризику	Опис	Значення інтегрального показника
Ймовірний	Невиконання або зрив термінів виконання задачі відбувається у більшості випадків	[-1; 0)
Звичайний	Зрив термінів виконання зазвичай не відбувається	[0; 0.35]
Низький	Невиконання або зрив термінів виконання практично не відбувається	(0.35; 1]

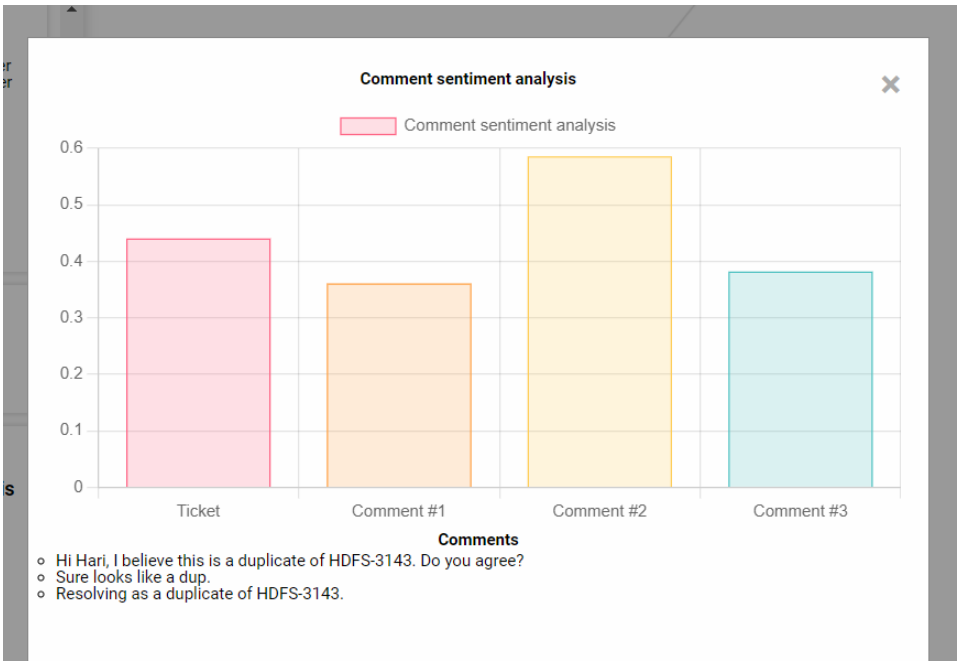

Рис. 1. Вікно з діаграмою сантимент-аналізу задачі

4. Виявлення основних обговорюваних тем

Виявлення основних обговорюваних тем дає можливість швидко оцінити про що йшлося в обговореннях поточних задач. Для цього пропонується використати латентно-семантичний аналіз. Латентно-семантичний аналіз (LSA) – це метод обробки інформації на природній мові, що дозволяє проводити аналіз взаємозв'язку між бібліотекою документів та термінами, що в них зустрічаються, а також дозволяє виявляти тематики, що властиві документам. Цей метод досить успішно застосовується для представлення баз знань [12] і побудови когнітивних моделей [13,16].

Метод латентно-семантичного аналізу складається з декількох кроків: підготовка корпусу документів, побудова терм-документної матриці, виявлення залежностей. Підготовка корпусу документів полягає у розбитті тексту на лексеми, видаленні «шумових» лексем- сполучників, пунктуаційних знаків та приведенні слів до початкової форми (нормалізація). Далі будується терм-документна матриця, рядками якої є документи, а стовпцями – слова, які в них зустрічаються. Елементом матриці є певна вага слова в даному корпусі документів. Зазвичай для обчислення ваги використовується метрика TF-IDF.

$$tf - idf(r, d, D) = tf(t, d) \times idf(t, D)$$

$$tf(t, d) = \frac{n_t}{\sum_k n_k} \qquad idf(t, D) = \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|}$$

На наступному кроці проводиться сингулярний розклад [14] отриманої матриці, що дозволяє виявити приховані залежності між документами. Такий підхід дає можливість визначити семантичний зв'язок між термінами в документах та кластеризувати терміни – тобто визначити основні теми документів.

Використовуючи такий підхід, стає можливим виявляти основні теми, що були актуальними на поточному етапі процесу розробки програмного забезпечення, виділяючи цю інформацію з текстів issues та коментарів до них. Приклад виявлених тем наведено на рис. 2.

Рис. 2. Виявлені методом LSA теми

5. Виявлення категорій та підкатегорій серед обговорюваних термінів

При обговоренні задач практично завжди використовуються специфічні для програмної інженерії слова, що можна віднести до певного більш ширшого поняття.

Наприклад, поняття “SQL” є частковим випадком більш ширшого поняття “Декларативні мови програмування”, яке в свою чергу є частковим випадком поняття “Мови програмування”. Розуміння цієї специфіки дає можливість створити довідник термінів, що визначає до яких родових понять відноситься той або інший обговорюваний термін.

Табл. 2. Приклад довідника термінів

Категорія	Підкатегорія	Приклади даних
Мови програмування	Об’єктно-орієнтовані	C#, Java, C++
	Процедурні	Pascal, C, Go, GoLang, Kotlin
	Декларативні	SQL, HTML, CSS
	Функціональні	Lisp, Erlang, Scala, F#
Стандарти програмної інженерії	Протоколи	HTTP, FTP, HTTPS, POP, SMTP
	Формати представлення даних	XML, JSON, JPG, PNG
	Архітектури та архітектурні підходи	MVC, MVP, MVVM, Factory, Builder
Програмні засоби	Середовища розробки	Idea, PyCharm, VisualStudio, ReSharper
	Засоби тестування	JProfiler, Postman, Firebug
	Фреймворки	Maven, Spring, Angular, Jooq
	Бібліотеки	Mockito, JAXB, Guava, Log4j

Етап виявлення тем на основі довідника сутностей полягає у фільтруванні тексту документу та заміні відомих термінів на назви їх категорій та підкатегорій.

У подальшому це дає можливість побудувати хмару тем, де замість безпосередньо приведених у тексті задачі слів будуть показані категорії та підкатегорії, до яких належать ці слова.

Використання такого підходу дає можливість виявляти взаємозв’язки між задачами не за термінами, що в них описуються, а за категоріями та підкатегоріями сутностей. Тобто, маємо можливість виявляти зв’язки між задачами на більш концептуальному рівні.

Приклад застосування такого підходу наведено на рис. 3.

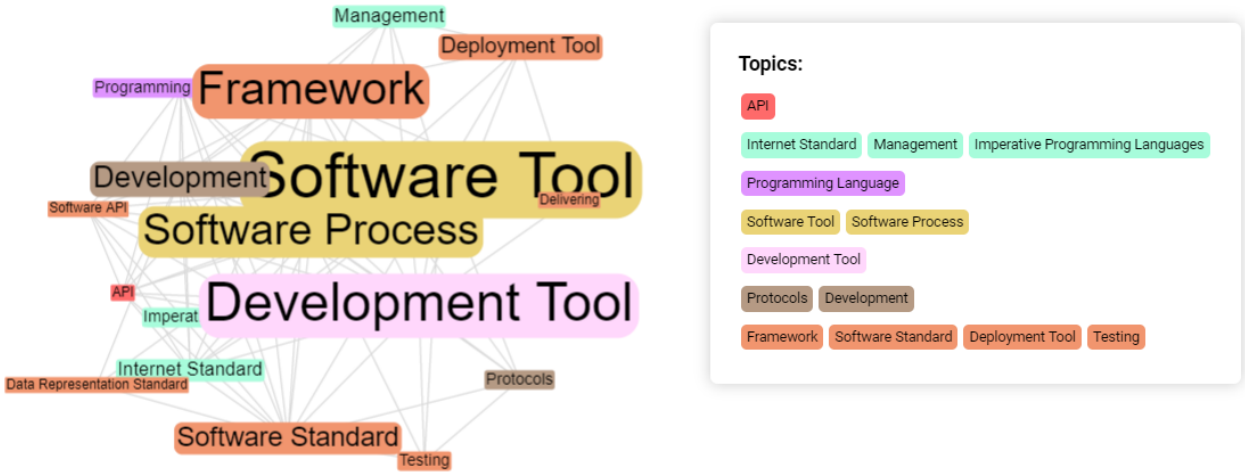

Рис. 3. Виявлені категорії та підкатегорії термінів

6. Дослідження ефективності

З метою оцінки ефективності було розроблено наступні метрики:

– швидкість виявлення проблемних задач;
– швидкість виявлення пов'язаних задач.

Для оцінки поточного стану розробки проєкту було запрошено 2 експерта у сфері проєктного менеджменту компанії «Неткрекер», що мають багаторічний (7 та більше років) досвід керування проєктами.

Проведено серію з двох експериментів, в рамках кожного з яких експерт у ролі проєктного менеджера спробує оцінити поточний стан розробки проєкту, використовуючи:

– звичайну Kanban-дошку з переліком задач;
– розроблений метод для моніторингу процесу розробки на основі аналізу текстових артефактів.

Далі порівняємо обидва способи за розробленими вище критеріями оцінювання.

Для першого експерименту було взято 63 задачі з проєкту [15], що загалом містять 1245 коментарів. Середня кількість коментарів до задачі – 19.7. З результатів експерименту видно, що використання розробленого методу пришвидшило оцінку ризику невиконання задачі у 2.7 рази для першого експерта та у 2.3 рази для другого експерта.

Для другого експерименту було взято 78 задач з проєкту [15], що загалом містять 1064 коментарі. Таким чином, середня кількість коментарів до задачі – 13.6. З результатів проведення експерименту видно, що використання розробленого методу пришвидшило оцінку ризику невиконання задачі у 2.8 рази для першого експериментатора та у 2.2 рази для другого експериментатора.

Табл. 3. Результати експерименту оцінки швидкості виявлення проблемних задач

Експеримент	Експерт	Спосіб огляду	Час, витрачений на огляд
Проєкт 1	Експерт 1	Kanban-дошка	145 хвилин
		Розроблений метод	53 хвилини
	Експерт 2	Kanban-дошка	163 хвилини
		Розроблений метод	68 хвилин
Проєкт 2	Експерт 1	Kanban-дошка	138 хвилин
		Розроблений метод	49 хвилин
	Експерт 2	Kanban-дошка	155 хвилин
		Розроблений метод	69 хвилин

Проведемо аналогічний експеримент, в рамках якого виміряємо час, необхідний проєктному менеджеру для того, щоб виявити пов’язані задачі (такі, що мають схожий зміст, або ж, в яких обговорюється одна й та сама тема).

В рамках даного експерименту експерт у ролі проєктного менеджера переглядає список задач та їх вміст (текст опису та коментарів) з метою виявлення задач, що є взаємопов’язаними між собою. Кінцевою метою є розуміння проєктним менеджером структури взаємозв’язків між задачами та перелік тем, до яких належить кожна задача. Спершу, експеримент з виявлення взаємопов’язаних задач проводиться за допомогою класичної Kanban-дошки. Потім, аналогічний експеримент було проведено за допомогою розробленого методу.

Табл. 4. Результати експерименту оцінки швидкості виявлення пов’язаних задач

Експеримент	Експерт	Спосіб огляду	Час, витрачений на огляд
Проєкт 1	Експерт 1	Kanban-дошка	179 хвилин
		Розроблений метод	136 хвилини
	Експерт 2	Kanban-дошка	190 хвилин
		Розроблений метод	153 хвилини
Проєкт 2	Експерт 1	Kanban-дошка	213 хвилин
		Розроблений метод	180 хвилин
	Експерт 2	Kanban-дошка	243 хвилини
		Розроблений метод	209 хвилин

У рамках першого експерименту бачимо, що використання розробленого методу пришвидшило процес огляду та виявлення взаємопов'язаних задач і їх тематики у 1.3 рази для першого експериментатора та у 1.24 рази для другого експериментатора. У рамках другого експерименту бачимо, що використання розробленого методу пришвидшило процес огляду та виявлення взаємопов'язаних задач і їх тематики у 1.18 рази для першого експерта та у 1.16 разів для другого експерта.

Підсумовуючи, розрахуємо інтегральну оцінку часу, витраченого на оцінку ступеня ризику задачі та виявлення взаємопов'язаних задач з їх темами для двох експериментів сумарно.

Табл. 5. Інтегральна оцінка часу, витраченого на огляд

Використаний засіб	Інтегральна оцінка
Kanban-дошка	825 хвилин
Розроблений метод	678 хвилин

Висновки

Запропоновано метод моніторингу процесу розробки програмного забезпечення на основі аналізу текстових артефактів. Запропоновано застосування математичних методів обробки текстів на природній мові для покращення моніторингу процесу розробки програмного забезпечення. Застосування лінгвістичних методів обробки природніх текстів дає можливість виявляти тенденції (теми), які з'являються у текстових артефактах, що виникають у процесі розробки (issues-тікети та коментарі до них).

Використання сантимент-аналізу дає можливість оцінювати ризик порушення термінів виконання задач. Використання таких методів та засобів дає можливість накопичувати та аналізувати дані процесу розробки програмного забезпечення, що сприяє подальшому розвитку наукових досліджень у напрямку керування процесом розробки. Результати проведених експериментів показують, що використання розробленого методу пришвидшує процес моніторингу розробки в середньому на 22% в порівнянні з використанням звичайної Kanban-дошки.

Список літератури

1. Simon Alexandre. Software Metrics An Overview [Online] / Simon Alexandre // University of Namur Software Quality Lab Belgium. – 2002. – URL: https://www.cetic.be/IMG/pdf/Software_Metrics_Overview.pdf.
2. Software Measurement: A Necessary Scientific Basis [Online] // IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. – 1994. – URL: https://www.ipd.kit.edu/mitarbeiter/padberg/lehre/sqs07/FentonTSE1994.pdf.
3. UNIQUE FUNDAMENTALS OF SOFTWARE MEASUREMENT AND SOFTWARE METRICS IN SOFTWARE ENGINEERING [Online] // International Journal of Computer Science & Information Technology. – 2015. – URL: http://www.airccse.org/journal/jcsit/7415ijcsit03.pdf.
4. Victor R. Basili. THE GOAL QUESTION METRIC APPROACH [Online] / Victor R. Basili, Gianluigi Caldiera, H. Dieter Rombach // FB Informatik Universität Kaiserslautern Kaiserslautern, Germany – URL: https://www.cs.umd.edu/users/mvz/handouts/gqm.pdf.
5. Alan J. Perlis. Software Metrics: An Analysis and Evaluation / Alan J. Perlis, Frederick Sayward, Mary Shaw., 1983.
6. A Study of the Effect of Data Normalization on Software and Information Quality Assessment [Online] / Morgan Ericsson, Welf Löwe, Tobias Olsson та ін.] – URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.497.3097&rep=rep1&type=pdf.
7. Normalization [Online] // Wikipedia. – 2022. – URL: https://en.wikipedia.org/wiki/Normalization_(statistics).
8. Karine Mordal. Software quality metrics aggregation in industry [Online] / Karine Mordal, Nicolas Anquetil, Jannik Laval // JOURNAL OF SOFTWARE: EVOLUTION AND PROCESS.

– 2012. – URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.700.305&rep=rep1&type=pdf.

9. Thomas K Landauer. An Introduction to Latent Semantic Analysis [Online] / Thomas K Landauer, Peter W. Foltz, Darrell Laham. – 1998. – URL: http://lsa.colorado.edu/papers/dp1.LSAintro.pdf.
10. Yu. Oliynik. Review and analysis of algorithms TEXT MINING / O. Gavrilenko, Yu. Oliynik, H. Hanko. // Project management, systems analysis and logistics. – K .: NTU, 2017. - Vol., pp32-41
11. Mostafa Taha. Task Management System (TMS) [Online] / Mostafa Taha. – 2016. – URL: https://repository.najah.edu/handle/20.500.11888/12232.
12. Thomas K. Landauer. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge [Online] / Thomas K. Landauer, Susan T. Dumais – URL: http://www.welchco.com/02/14/01/60/96/02/2901.HTM.
13. B. Lemaire, G. Denhière. Cognitive Models based on Latent Semantic Analysis (неопр.) // Tutorial given at the 5th International Conference on Cognitive Modeling (ICCM'2003), Bamberg, Germany, April 9 2003.. — 2003
14. Singular Value Decomposition (SVD) tutorial [Online] – URL: https://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm.
15. Apache HADOOP issues [Online]. – 2022. – URL: https://issues.apache.org/jira/projects/HADOOP/issues/
16. Мигаль Д. С., Олійник Ю. О. СУЧАСНІ ЗАСОБИ ТА МЕТОДИ СЕМАНТИЧНОГО АНАЛІЗУ УКРАЇНОМОВНИХ ТЕКСТІВ // Topical issues of the development of modern science. Abstracts of the 10th International scientific and practical conference. Publishing House “ACCENT”. Sofia, Bulgaria. 2020. Pp. *554-557*. URL: *https://sci-conf.com.ua/wp-content/uploads/2020/06/TOPICAL-ISSUES-OF-THE-DEVELOPMENT-OF-MODERN-SCIENCE_4-6.06.20.pdf*