AWERProcedia Information
Technology & Computer Science

Vol 03 (2013) 1157-1162
3rd World Conference on Information Technology (WCIT-2012)

Webification of Software Development: General Outline and the Case of Enterprise Application Development

Mária Bieliková *, Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, Bratislava, 842 16 Slovakia.
Pavol Návrat, Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, Bratislava, 842 16 Slovakia.
Daniela Chudá, Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, Bratislava, 842 16 Slovakia.
Ivan Polášek, Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, Bratislava, 842 16 Slovakia.
Michal Barla, Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, Bratislava, 842 16 Slovakia.
Jozef Tvarožek, Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, Bratislava, 842 16 Slovakia.
Michal Tvarožek, Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, Bratislava, 842 16 Slovakia.

Suggested Citation:
Bieliková, M., Návrat, P., Chudá, D., Polášek, I., Barla, M., Tvarožek, J. & Tvarožek, M. Webification of Software Development: General Outline and the Case of Enterprise Application Development, *AWERProcedia Information Technology & Computer Science*. [Online]. 2013, 3, pp 1157-1162. Available from: http://www.world-education-center.org/index.php/P-ITCS *Proceedings of 3rd World Conference on Information Technology* (WCIT-2012), 14-16 November 2012, University of Barcelon, Barcelona, Spain.

Received 11 April, 2013; revised 4 July, 2013; accepted 12 August, 2013.
Selection and peer review under responsibility of Prof. Dr. Hafize Keser.

Abstract
We present a view on the changes in software processes activated by the evolution of the World Wide Web, which shifts the software development more and more towards on-line practices. This includes not only the shift towards the use of web-based resources in software processes, but also and more importantly it opens a space for using approaches originally devised for the Web (as a network of interconnected content) to support the software development process. We envision the concept of collaborative software development to improve software quality and development efficiency by using both implicit and explicit user (software developer)

* ADDRESS FOR CORRESPONDENCE: **Mária Bieliková,** Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, Bratislava, 842 16 Slovakia, *E-mail address*: maria.bielikova@stuba.sk / Tel.: +421-2-6029-4173

feedback, which creates rich interconnections between software artifacts and enables employment of adapted methods originally devised for information processing on the Web.

Keywords: Collaborative programming; information tags; implicit feedback; social aspects; web of software artifacts;

1. Introduction

Ubiquitous access to networking has made the Web an integral part of our daily lives both professionally and in private. The underlying principle of the Web is that of a huge interconnected set of information artifacts. However, the Web also includes footprints of users (information and services consumers) as their activity is now an integral part of the Web. The impact and wide-reaching availability of the Web triggered the "webification" of many particular domains. The use of web-based interfaces can be seen as the simplest form of "webification". However, there is much more in the World Wide Web concept that can be transferred to other domains.

Looking at software development, and specifically at enterprise application development (EAD), which has historically been a closed in-house process, we observe that the used information space is rich in interconnected information artifacts (e.g., documentation, source code, developer blogs or help forums). Many useful methods devised for the Web can be used to search, browse and navigate within this web-like structure.

Software projects are realized by teams that create source code and documentation in an integrated development environment (IDE) optionally with additional support tools. Most frequent problems involve delivery delays, quality assurance issues, being over-budget [1]. A project manager may miss the day-to-day performance drop of a particular developer, a solution architect may fail to track all module dependencies and a developer may struggle alone to solve a deadlock.

Main issue presented in this paper is that by leveraging experience from rapidly evolving field of information processing on the Web and employing web-based methods in the EAD domain one could improve software quality and development efficiency, e.g., support the creation of better code, improve progress visibility or help developers be more efficient. This can be facilitated by providing users (both developers and managers) with the *right information in the right context*.

Software engineers use software metrics, based on well-defined indicators obtained by analysis of software artifacts (e.g., finding patterns), their properties (e.g., number of errors) or analysis of the software process itself (e.g., effort analysis) [2]. Thus the challenge is to devise and compute the right metrics to identify particular problems and their causes quickly while recommending corrective actions. We see a new opportunity in this important task of defining accurate software metrics in considering the software developer interaction monitoring for inferring important properties of software artifacts or software process itself. It is based on considering a web of software artifacts as the web of information artifacts and employing (adapting) known methods and techniques of information processing to the domain of software engineering.

2. Information Artifacts and Metadata

The software development domain includes several highly interlinked actors and information artifacts: developers who design, write and maintain source code, analysts who create specifications, requirements analysis, or testers who provide feedback ensuring that the resulting product meets design requirements.

Key information artifacts include the source code and supporting documentation that the various actors make use of in different stages of the software development process. Furthermore, in an

enterprise context, the software process is often formalized and supported by a variety of enterprise-class information systems.

In our approach we model the content and actor-related artifacts that affect and support day-to-day work of software developers in these categories:

Source codes, consisting of events in source control repositories and their relations to bug reports, customer change requests, persons, projects.

Knowledge documents that the developers work with (e.g., best practices, guidelines, notes stored in a company's intranet, blogs and bookmarks).

Developer interactions, consisting of meta-data about email and instant messaging, and emergent interactions in the source code.

Similarly to the highly interlinked nature of the Web, these artifacts contain a significant number of underused interconnections both *explicit* and *implicit* that lend themselves for further graph-based analysis that is successfully used on the web of software artifacts, which in many of its parts is characterized by scale-free, power law distributions [3,4]. Examples of explicit links are source code changes in response to bug reports, or instant messages between developers. Implicit links are on the level of software artifacts, such as source code to code snippet similarities, or on the level of software engineers activity related to the software artifacts such as follow-up actions after commits.

We model these heterogeneous software artifacts using a lightweight semantics which is based on a homogeneous underlying metadata representation: *information tags* – metadata that describe an aspect of an artifact [5]. The information tags are an extension to the basic concept of a tag as a simple keyword or term assigned to an artifact; in that an information tag is any metadata that adds additional value to the artifact itself; e.g., explicit and implicit feedback generated by developers working on source code. In our case, the information tags contain explicit and implicit feedback generated by software developers working on the project. The information tags are both

user generated (keywords describing code snippets, source code or other comments, explicit evaluations of artifacts, etc.) and

machine generated (statistics of time spent working on artifacts, similarity with code smells, crowdsourced artifact descriptions, etc.).

Information tags provide source for useful information to software engineers, similarly as metadata do in the Web information space, such as identification of bad practices, evaluation of source code quality based on an estimation of the current user state followed his activity (e.g., tired), recommendation of good programming practices and tricks/ snippets used by colleagues, source code & dependency visualization (Fig. 1). They also serve as an input for reasoning on properties of software artifacts such as similarity with code smells, estimation of developer skill and proficiency.

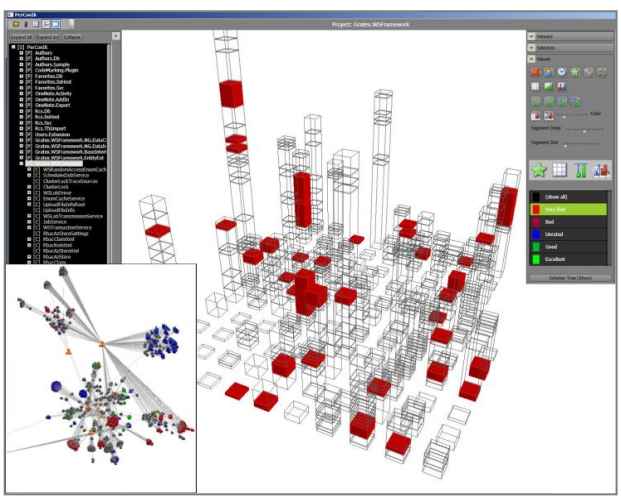

Fig. 2. GUI visualizing the graph of source codes. Columns show modules filtered by quality – highlighted parts are candidates for refactoring. Another graph shows authorship of project modules with color-coded layers of apps (bottom left).

3. Collaborative Software Development

People form the key element in the software development. The analysis of developer interaction with the (software) content as well as between users themselves provides valuable insights into

developer's characteristics as well as additional information related to the content of the Web-like infrastructure of software artifacts (including connections within source codebases). The scope of the observed content interactions includes: browsing within the content (software) and on the Web, content creation and other highly interactive activities less prevalent on the Web, where passive content consumption still prevails and other (complex) domain-dependent activities (e.g., testing content, which is unusual in the Web).

While we distinguish several kinds of explicit and implicit links between users on the Web (friends that visit the same content, have similar interests, etc.), the web of software artifacts involves much more refined links between various types of actors with different semantics. Two developers could have e.g. worked together on a project, used a piece of code produced by another, tested another one's code, or use similar coding styles.

We represent the user (software development actors) relations via information tags and provide a common infrastructure for both content and user interaction representation (see Fig. 2). A mixed approach of direct and indirect monitoring of user (software developer) interaction amongst themselves and with the information artifacts can be employed via integrated development environment plug-ins, proxy logs, support systems logs (see Fig. 2, right).

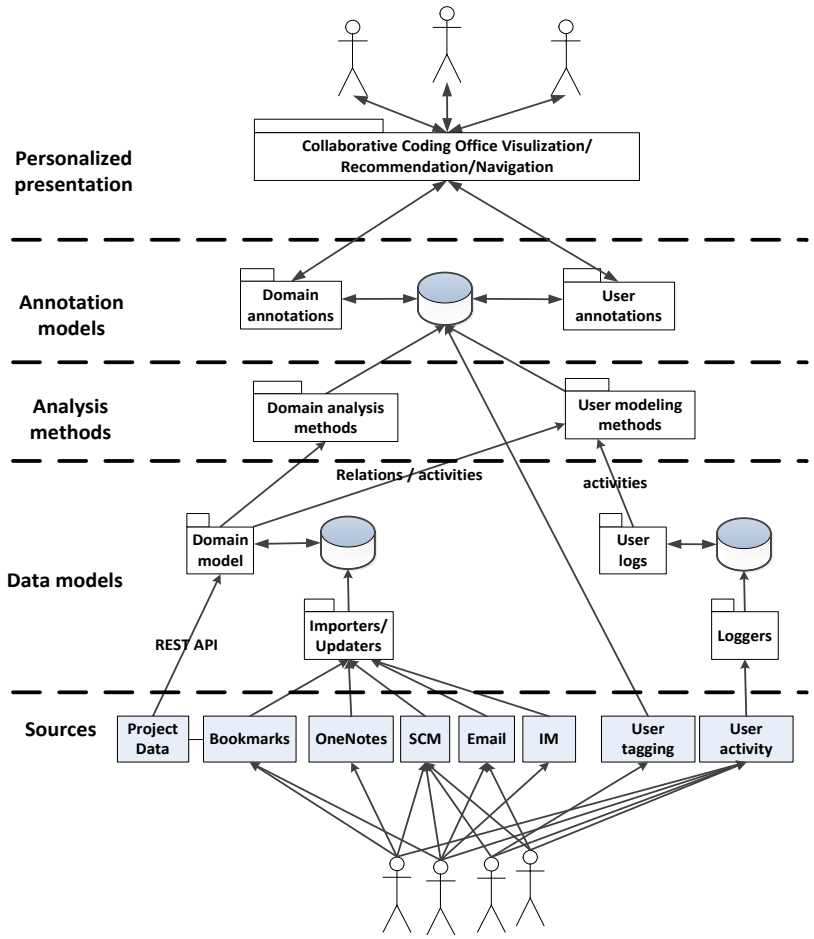

Fig. 2. Collaborative software development architecture. Data sources include source code, documentation and user interaction (bottom). These are processed into knowledge tags, which are used to provide added value to users via recommendation and personalization (top).

Using the acquired data, we can e.g., discover new communities within a company, or communities which do not respect the traditional organizational structure of a software company. Similarly to the Web, implicit user feedback can be successfully used. Surfers on the Web implicitly evaluate the usefulness and interestingness of content or accuracy of recommendations received on the Web. For instance, it was shown that the user's interest on the Web correlates with the time spent on a page [6].

In contrast to standard Web-based approaches, we have the opportunity to acquire the implicit feedback in both directions of interaction with the code: from a perspective of a developer writing the code as well as from a perspective of a developer trying to work on that code further on or to call it from other place. Implicit feedback in software development, i.e. observing developers during writing, reading and modifying a source code, gives us much more valuable information about the content than in the Web, where the monitoring of content creation is often too impractical if at all possible. Moreover, unlike the Web, the enterprise application development domain is much friendlier towards user monitoring and user interaction acquisition. Larger companies often have formalized policies regarding employee privacy and already monitor developers and the usage of software company resources.

The model of software artifacts based on information tags serves as an integration platform for (meta)data acquisition methods (Fig. 2, bottom) and for personalized and/or collaborative ranking, search, browse, recommendation and visualization methods that provide benefits to end users (Fig. 2, top). We have realized information tags store in flexible Open Annotation model [7].

4. Conclusions

We propose novel extensions of conventional software metrics. The evaluation of (software) metrics allows us to give aggregated information to developers who can then better understand information artifacts in the web of software, be it their own source code or code created by others.

There are benefits of information tags for managers too. E.g., they can see aggregated overviews of developer performance, reliability; aggregated overviews of software metrics for source code to observe progress; identify critical source code or developers based on knowledge tags (e.g., bug, bad code, deprecated code); observe knowledge and experience transfer between developers (e.g., who uses whose code snippets), or track capabilities and workload of developers.

The idea of viewing software repositories as webs is not new. Already Knuth in 1984 presented the idea that “a program is best thought of as a web…” [8]. The novel aspect lies in considering not only software artifacts but also users (developers) together with their explicit and implicit feedback, which brings a new view on software metrics. It helps developers be more efficient and can enrich them with the experience and knowledge of their colleagues while managers or senior developers can get advantage of improved planning and decision support via aggregation of statistical data for individual developers.

“Webification”' of software development considers not only software artifacts but also developers and their explicit and implicit feedback thus helping developers be more efficient and learn from others even more as we practice today in software teams. It opens many possibilities. E.g., repeated 'work' with source code can be supported by Games with a purpose originally devised for the Web [9]. Such a game can be focused to review source code fragments and identify potential problems by developers [10]. As a side effect, developers became familiar with more code and possibly learn others' tricks.

Acknowledgements. This contribution is the partial result of the Research & Development Operational Programme for the project Research of methods for acquisition, analysis and personalized conveying of information and knowledge - PerConIK, ITMS 26240220039, co-funded by the ERDF.

References

[1] Cerpa, N., & Verner, J. M. Why did Your Project Fail? *Commun. ACM,* 2009, 52 (12), pp. 130-134.
[2] Lincke, R., Lundberg, J., & Lowe, W. Comparing Software Metrics Tools. *Proc. of the Int. Symp. on Soft. Testing and Analysis.* ACM, New York, NY, 2008, pp. 131-142.
[3] Lindsay, J., Noble, J., & Tempero, E. Does Size Matter? A Preliminary Investigation of the Consequences of Powerlaws in Software. *Proc. of the ICSE Workshop on Emerging Trends in Soft. Metrics.* ACM, 2010, pp. 16-23.
[4] Louridas, P., Spinellis, D., & Vlachos, V. Power Laws in Software. ACM Trans. *Softw. Eng. Methodol.,* 2008, 18 (1), pp. 1-26.
[5] Bieliková, M., Barla, M., & Šimko, M. Lightweight Semantics for the "Wild Web". *Proc. of the Int. Conf. WWW/ Internet,* IADIS Press, 2011, pp. 25-27.
[6] Hofgesang, P. I. Methodology for Preprocessing and Evaluating the Time Spent on Web Pages. *Proc. of the IEEE/WIC/ACM Int. Conf. on Web Intelligence. IEEE Comp. Soc.,* Washington, DC, 2006, pp. 218-225.
[7] Bieliková, M., & Rástočný, K. Lightweight Semantics over Web Information Systems Content Employing Knowledge Tags. *Proc. of Int. Conf. on Conceptual Modeling Workshops, LNCS 7518,* Springer, 2012, pp. 327-336.
[8] Knuth, D. E. Literate Programming. *Comput. J.*, 1984, 27, pp. 97-111.
[9] von Ahn, L., & Dabbish, L. Designing Games with a Purpose. *Commun. ACM,* 2008, 51 (8), pp. 58-67.
[10] Šimko, J., Tvarožek, M., & Bieliková, M. Semantic Discovery via Human Computation Games. *Int. J. on Semantic Web and Inf. Sys.,* 2011, 7 (3), pp. 23-45.