PerConIK: Aplikovanie znalostných technológií v doméne tvorby softvéru

Mária Bieliková, Pavol Návrat, Michal Barla, Ivan Polášek,
Daniela Chudá, Peter Lacko

Ústav informatiky a softvérového inžinierstva, Fakulta informatiky a informačných technológií,
Slovenská technická univerzita v Bratislave, Ilkovičova 2, 842 16 Bratislava, Slovakia

{name.surname}@stuba.sk

Abstrakt. Projekt PerConIK sa zaoberá výskumom v oblasti nových spôsobov získavania a spracovania informácií a znalostí v prostredí softvérovej firmy. Tieto vychádzajú z chápania informačného priestoru softvérového projektu ako pavučiny prepojených komponentov, pričom prepojenia nachádzame nielen v obsahu, ale aj v aktivitách vývojárov. V príspevku predstavujeme vybrané výsledky projektu dosiahnuté riešiteľmi projektu v spoločnosti Gratex International a na Fakulte informatiky a informačných technológií STU v Bratislave.

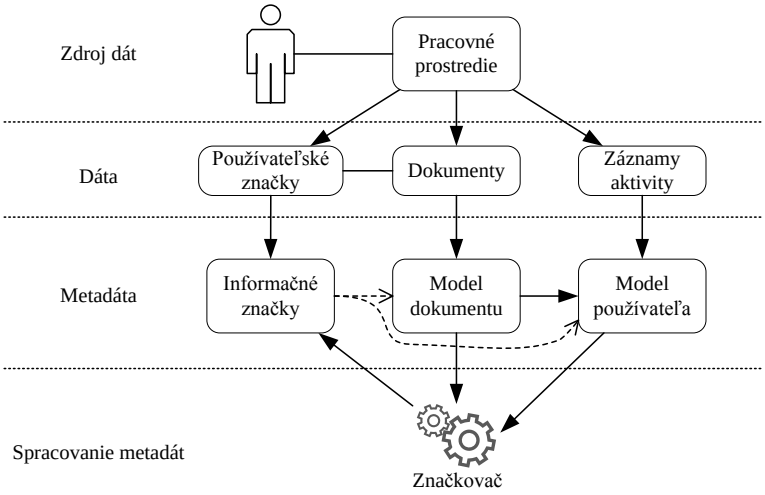
1 Zameranie a ciele projektu

Projekt PerConIK[1] (Personalized Conveying of Information and Knowledge) sa zameriava na výskum procesov tvorby softvéru na nízkej úrovni granularity, s dôrazom na automatizovanú podporu tých činností, v ktorých poskytnutie dodatočných informácií a/alebo znalostí môže pozitívne ovplyvniť výslednú kvalitu týchto procesov.

Základným cieľom projektu je definovať modely v doméne tvorby softvéru so zohľadnením aktivít, ktoré v tejto doméne prebiehajú a ktoré nepriamo definujú model používateľa (typicky programátora). Nad týmito modelmi navrhujeme metódy, ktoré zefektívňujú jednotlivé procesy vývoja softvéru – riešia podporu navigácie v zdrojových kódoch, spolupráce a interakcie (napr. cez prehliadky kódu). Niektoré metódy poskytujú dodatočné znalosti pre špecifických účastníkov procesu vývoja softvéru, napr. pre manažéra vývoja, softvérového architekta alebo manažéra kvality.

Primárnym zdrojom dát v projekte je programátor a pracovné prostredie, na ktorom si prezerá a edituje zdrojové kódy programov. Na obrázku 1 je znázornené spracovanie týchto dát do podoby tzv. informačných značiek [2], ktoré viažu semištruktúrovanú informáciu o dokumente ku konkrétnemu miestu v danom dokumente. Značkovač predstavuje realizáciu metódy, ktorej vstupom sú záznamy o aktivite používateľa a/alebo zdrojové súbory, resp. softvérové artefakty spolu s históriou ich vývoja, ktorá na výstupe označí používateľa a/alebo niektoré softvérové artefakty príslušnou značkou.

[1] http://perconik.fiit.stuba.sk/



Obr. 1. Horizontálne vrstvy spracovávania dát z prostredia, spracované z [3]

Samotné informačné značky môžu byť zobrazené dvoma rôznymi spôsobmi:

- priamo pri dotknutom zdrojovom kóde, ktorého sa týkajú (napr. táto trieda je veľmi nestabilná, jej *public* metódy sa často menia)
- V špecializovaných nástrojoch, ktoré poskytujú možnosť prehliadať si a navigovať v informačnom priestore značiek a označených zdrojových kódov [8]

2 Dosiahnuté výsledky a diskusia

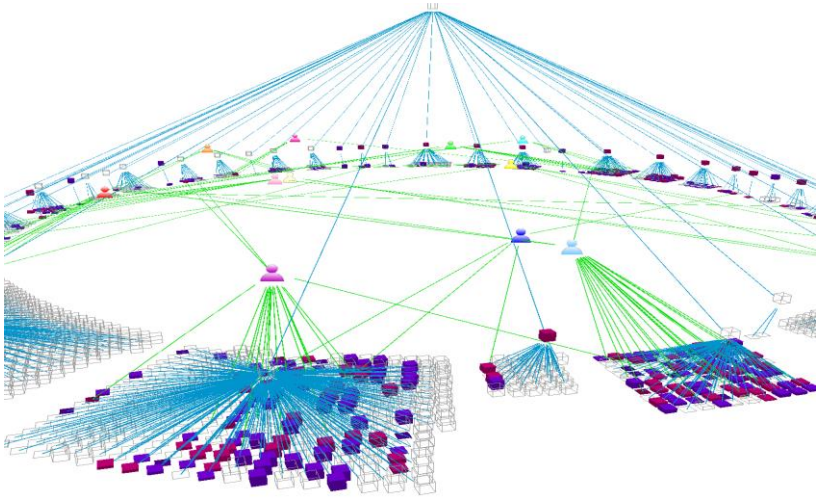
2.1 Generovanie a udržiavanie informačných značiek

V rámci projektu sme riešili problematiku automatizovaného generovania informačných značiek na základe preddefinovaných vzorov aktivít a ich kotvenie na dotknuté zdrojové súbory, resp. ich fragmenty [10]. Pri značkovaní jednotlivých fragmentov sme museli riešiť aj problematiku udržiavania aktuálnosti kotvenia pri jednotlivých značkách, ako aj problematiku platnosti samotnej značky, keďže zdrojový kód pod značkami sa neustále vyvíja a mení [11]. Keďže značky môžu byť vo svojej štruktúre veľmi heterogénne, museli sme riešiť aj spôsob ich uloženia, ´ktorý by bol škálovateľný, poskytoval dostatočnú flexibilitu pre definovanie nových typov značiek a zároveň umožňoval aspoň základné možnosti odvodzovania nad značkami [2].

2.2 Analýza zdrojových kódov.

V projekte sme sa sústredili aj na analýzu zdrojového kódu ex post. Základným zdrojom dát je kód spolu s históriu jeho vývoja uložený v systéme na riadenie revízií (Revision Control System). Kód ďalej prevádzame do formy abstraktných syntaktických stromov v jeho jednotlivých revíziách a získavame nový formát AST.RCS [8]. Tento

formát je vhodný nielen pre uchovanie a neskoršie zobrazenie samotnej štruktúry kódu (Obr. 2), jeho používateľov a autorov, ale využili sme ho aj pri našich metódach, ktoré detegujú jednotlivé témy obsahu kódu [7] či vzory a pachy [6].



Obr. 2. Príklad vizualizácie zdrojového kódu, antivzorov a vývojárov

Využitie abstraktných syntaktických stromov a ich rozvinutie v čase nám umožňuje efektívne vyhľadávanie duplicít v zdrojových kódoch projektu [9],či automatizovane vyhodnocovať evolúciu zdrojových kódov [1, str. 486]. V spojení so záznamami o aktivite programátorov dokážeme identifikovať skryté, implicitné prepojenia medzi softvérovými artefaktmi [1, str. 474], ktoré môžu pomôcť rýchlejšiemu zorientovaniu sa v zdrojových kódoch, či poslúžiť pri prevencii chýb.

2.3 Odhaľovanie charakteristík programátora

V rámci projektu sa venujeme aj metódam, ktorých cieľom je identifikácia charakteristík programátora [4,5], ako je napríklad úroveň jeho expertízy pre vybrané softvérové technológie alebo úroveň jeho vedomostí o projekte alebo jeho časti. Tieto znalosti môžu pomôcť manažérovi projektu napríklad pri rozhodovaní sa o tom, komu priradí konkrétnu úlohu. Iným druhom znalostí, ktoré extrahujeme z aktivít programátora je jeho aktuálny kontext [1 str. 492, 1, str. 468], ktorý môžeme využiť napr. pre efektívne prispôsobovanie procesu vyhľadávania vo webových, či iných zdrojoch aktuálnym potrebám programátora.

Viaceré z navrhnutých metód boli rozpracované a overené v rámci prototypu, ktorý je aktuálne nasadený na reálnych softvérových projektoch v spoločnosti Gratex International ako aj na menších študentských a výskumných projektoch, ktoré sa realizujú na Fakulte informatiky a informačných technológií (napr. projekty v rámci predmetu

Tímový projekt). Nasadenie prototypu na reálne softvérové projekty si na jednej strane vyžaduje odladenú implementáciu, ktorá nijakým spôsobom neobmedzuje vývojárov v ich práci, na druhej strane poskytuje výborný základ pre ďalší výskum, keďže v porovnaní s offline experimentami máme priamy kontakt s vývojármi a ich manažmentom a môžeme ich priamo konfrontovať s čiastkovými výsledkami.

Poďakovanie. *Táto publikácia vznikla vďaka podpore v rámci OP Výskum a vývoj pre projekt ITMS: 26240220039, spolufinancovaný zo zdrojov ERDF.*

Literatúra

1. Bieliková, M (Ed.): Student Research Conference 2014, Proc. in Informatics and Information Technologies, STU, 2014, p. 670.
2. Bieliková, M., Rástočný, K.: Lightweight Semantics over Web Information Systems Content Employing Knowledge Tags. In Advances in Conceptual Modeling : ER, LNCS 7518, Springer, 2012, pp. 327-336.
3. Bieliková, M. et al.: Platform independent software development monitoring: design of an architecture.In: *SOFSEM 2014 : theory and practice of computer science*, LNCS 8327, Springer, 2014, pp. 126-137.
4. Holub, M., Kuric, E., Bieliková, M.: Modelovanie znalostí programátora s využitím ľahkej sémantiky. In: *ZNALOSTI 2012* ,Matfyzpress, 2012, pp. 21-30
5. Kuric. E, Bieliková, M.: Estimation of Student's Programming Expertise. In *Int. Symp. on Empirical Software Engineering and Measurement, ESEM 2014*, ACM, 2014, *accepted*.
6. Polášek, I., Snopko, S., Kapustík, I.: Automatic Identification of the Anti-patterns Using the Rule-based Approach. In: *SISY 2012, Int. Symp. on Intelligent Systems and Informatics*, IEEE, 2012. pp. 283-286
7. Polášek, I., Uhlár, M.: Extracting, Identifying and Visualisation of the Content, Users and Authors in Software Projects. In: *Trans. on Comp. Sci. XXI: Special Issue on Innovations in Nature-Inspired Computing and Applications*, LNCS 8160, Springer, 2013, pp. 269-295
8. Polášek, I. et al.: Information and Knowledge Retrieval within Software Projects and their Graphical Representation for Collaborative Programming. In: *Acta Polytechnica Hungarica. Vol. 10, No. 2*, 2013, pp. 173 – 192
9. Súkeník, P., Lacko, P.: Vyhľadávanie zduplikovaného kódu, In *Workshop on Int. and Knowl. Oriented Technologies, WIKT 2012*, Vydavateľstvo STU, 2012, pp. 189-192
10. Rástočný, K., Bieliková, M.: Enriching Source Code by Empirical Metadata. In *Int. Symp. on Empirical Softw. Engineering and Measurement, ESEM 2014*. ACM, 2014, *accepted*.
11. Rástočný, K., Bieliková, M.: Metadata anchoring for source code: Robust location descriptor definition, building and interpreting. In *Databases and Expert Systems Applications, DEXA 2013*, LNCS 8056, Springer, 2013, pp. 372-379.

English summary

PerConIK: Application of knowledge-oriented tech. in software engineering domain
Project PerConIK focuses on research on acquisition and processing of information and knowledge within a domain of a software house. The research is based on a web-like model of interconnected software artifacts with an additional layer of developers' interaction. This paper presents some of the results achieved in the project.