9th International Conference on Applied Informatics
Eger, Hungary, 2014.

Szoftver elemzés és program minőség javítása*

Király Roland

Eszterházy Károly Főiskola
kiraly.roland@aries.ektf.hu

Kivonat

A programok méretének és bonyolultságának növekedésével a fejlesztési költségek egyre nagyobb részét képezi a tesztelés [2, 3], és a tesztelés során felmerülő problémák megoldása, valamint a változtatást követően annak a bizonyítása, hogy a módosított programszöveg jelentése nem változott meg. A programok átalakításának a költségét nem kizárólag a méretük, hanem a forráskódjuk bonyolultsága is nagyban befolyásolja.

Statikus forrásszöveg elemzéssel [5] már a programfejlesztés kezdeti fázisaiban fényt deríthetünk a programban előforduló következetlenségekre, és segítségével megtalálhatjuk és javíthatjuk a kezelhetetlenül bonyolult programrészeket, ezáltal csökkentve a programtesztelés költségeit. A bonyolultsági mértékeken [1] alapuló elemzés magában foglalja a forrásszöveg kezelhetetlenül bonyolult részeinek lokalizálását, a rossz programozási stílusjegyek megtalálását, valamint a nem hatékony kódrészletek, és algoritmusok kiszűrését is. A forráskód karakterisztikája meghatározható minden egyes programozási paradigmában, és a legtöbb esetben más szempontok alapján határozzuk meg annak attribútumait. Fontos momentum a kód indentálása, és a kódsorok száma mellett az is, hogy az adott programban melyik vezérlési szerkezetből mennyi van, és mekkora ezek beágyazottsági mélysége.

Mivel a mértékrendszer, és annak egyes értékei, valamint az értékek egymáshoz viszonyított aránya eltérhet a különböző programok fejlesztésénél, lehetőséget kell biztosítanunk arra, hogy a program tervezője, és a programfejlesztők definiálni tudják a számukra megfelelőnek ítélt értékeket és arányokat.

A különböző program [6, 4, 7] transzformációs lépéseket követően megmérhető a forrásszöveget jellemző strukturális bonyolultsági mértékek változása, és ez alapján eldönthető, hogy milyen program transzformációkat alkalmazzunk a kódminőség javítására.

*A kutatási tevékenység a TÁMOP 4.2.4.A/2-11-1-2012-0001 azonosító számú Nemzeti Kiválóság Program – Hazai hallgatói, illetve kutatói személyi támogatást biztosító rendszer kidolgozása és működtetése országos program című kiemelt projekt keretében zajlik. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Ebben az írásban bemutatjuk azt a bonyolultsági mértékeken alapuló elemző, és optimalizáló algoritmust [12, 13, 14], amelyet *Erlang* [8, 9] nyelvű forrásszövegek készítése során, valamint korábban készült, de átalakításra váró programszövegek automatikus, vagy fél-automatikus javítására használhatunk.

Keywords: bonyolultság, szoftver metrikák, elemzés, forrásszöveg, erlang, refactorerl

Hivatkozások

[1] McCabe T. J. A Complexity Measure, *IEE Trans. Software Engineering, SE-2(4), pp.308-320 (1976)*

[2] Zoltán Porkoláb, Ádám Sipos, Norbert Pataki, Structural Complexity Metrics on SDL Programs. *Computer Science, CSCS 2006, Volume of extended abstracts, (2006)*

[3] Zoltán Porkoláb Programok Strukturális Bonyolultsági Mérőszámai. *PhD thesis Dr Töke Pál, ELTE Hungary, (2002)*

[4] Zoltán Horváth, Zoltán Csörnyei, Roland Király, Róbert Kitlei, Tamás Kozsik, László Lövei, Tamás Nagy, Melinda Tóth, and Anikó Víg.: Use cases *for refactoring in Erlang, To appear in Lecture Notes in Computer Science, (2008)*

[5] Csörnyei Zoltán *Fordítóprogramok* Typotex Kiadó, Budapest, 2006. 3

[6] R. Kitlei, L. Lövei, M Tóth, Z. Horváth, T. Kozsik, T. Kozsik, R. Király, I. Bozó, Cs. Hoch, D. Horpácsi.: Automated Syntax Manipulation in RefactorErl. *14th International Erlang/OTP User Conference. Stockholm, (2008)*

[7] Lövei, L., Hoch, C., Köllö, H., Nagy, T., Nagyné-Víg, A., Kitlei, R., and Király, R.: Refactoring Module Structure *In 7th ACM SIGPLAN Erlang Workshop, (2008)*

[8] Lövei, L., Horváth, Z., Kozsik, T., Király, R., Víg, A., and Nagy, T.: Refactoring in Erlang, a Dynamic Functional Language *In Proceedings of the 1st Workshop on Refactoring Tools, pages 45–46, Berlin, Germany, extended abstract, poster (2007)*

[9] Erlang - Dynamic Functional Language
http://www.erlang.org

[10] T. Kozsik, Z. Horváth, L. Lövei, T. Nagy, Z. Csörnyei, A. Víg, R. Király, M. Tóth, R. Kitlei.. Refactoring Erlang programs. *CEFP'07, Kolozsvár (2007)*

[11] Király Roland, *Funkcionális programozási nyelvek* EKF TTK TAMOP412 2010 120 oldal.

[12] Király, R., Kitlei R.: *Complexity measurments for functional code* 8th Joint Conference on Mathematics and Computer Science (MaCS 2010) refereed, and the proceedings will have ISBN classification July 14-17, 2010

[13] Király, R., Kitlei R.: *Implementing structural complexity metrics in Erlang.* '10 ICAI 2010 – 8th International Conference on Applied Informatics to be held in Eger, Hungary January 27-30, 2010

[14] Király, R. and Kitlei, R.: *Implementing structural complexity metrics for Erlang* Poster on the 8th International Conference on Applied Informatics, ICAI 2010, 2010

Király Roland
Eszterházy Károly Főiskola, TTK, Matematikai és Informatikai Intézet

9th International Conference on Applied Informatics
Eger, Hungary, 2014.

Source Code Analysation and Optimalization*

Roland Király

Eszterházy Károly Collage
kiraly.roland@aries.ektf.hu

Abstract

As code size and complexity grows, an ever increasing part of development costs [2, 3] is testing, and dealing with problems the arises during testing, and also proving that the changes made to the code will not alter its semantic meaning. The cost of code modification is not only dependent upon its size, but also significantly influenced by its level of sophistication. Static source analysis [5] could help to uncover code inconsistencies even in the early phases of program development, and to find and correct unmanageably complex code parts to lower the cost of code-testing. A complexity measure based analysis [1] includes finding unmanageably complex parts and pin-pointing bad coding style, ineffective or unoptimized code snippets and algorithms. The characteristics of the source code is definable in every programming paradigm, and in most cases its attributes are determined based on different criteria. Besides code indentation and the amount of code lines it is also an important factor how many programming constructs are used and how deeply nested those are. There can be differences in the scale of the measuring system, and the various values and the ratio between those during development so we should provide means to the program engineers and developers to be able to define the values and ratios they deemed appropriate. The various code transformation steps can cause measurable changes in the structural complexity of source codes, and by that one can decide upon which transformations are to be used to improve code quality. In this paper we would like to introduce an analyzing and optimizing algorithm based upon complexity measurement [6, 4, 7], with which one can automatically or semi-automatically use to correct during coding new, or already made (but awaiting transformation) source-codes written in Erlang [8, 9].

Keywords: software metrics, complexity metrics, source code, semantic graph, refactorerl

*This research was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP 4.2.4. A/2-11-1-2012-0001 'National Excellence Program'.

References

[1] McCabe T. J. A Complexity Measure, *IEE Trans. Software Engineering, SE-2(4), pp.308-320 (1976)*

[2] Zoltán Porkoláb, Ádám Sipos, Norbert Pataki, Structural Complexity Metrics on SDL Programs. *Computer Science, CSCS 2006, Volume of extended abstracts, (2006)*

[3] Zoltán Porkoláb Programok Strukturális Bonyolultsági Mérőszámai. *PhD thesis Dr Töke Pál, ELTE Hungary, (2002)*

[4] Zoltán Horváth, Zoltán Csörnyei, Roland Király, Róbert Kitlei, Tamás Kozsik, László Lövei, Tamás Nagy, Melinda Tóth, and Anikó Víg.: Use cases *for refactoring in Erlang, To appear in Lecture Notes in Computer Science, (2008)*

[5] Csörnyei Zoltán *Fordítóprogramok* Typotex Kiadó, Budapest, 2006. 3

[6] R. Kitlei, L. Lövei, M Tóth, Z. Horváth, T. Kozsik, T. Kozsik, R. Király, I. Bozó, Cs. Hoch, D. Horpácsi.: Automated Syntax Manipulation in RefactorErl. *14th International Erlang/OTP User Conference. Stockholm, (2008)*

[7] Lövei, L., Hoch, C., Köllö, H., Nagy, T., Nagyné-Víg, A., Kitlei, R., and Király, R.: Refactoring Module Structure *In 7th ACM SIGPLAN Erlang Workshop, (2008)*

[8] Lövei, L., Horváth, Z., Kozsik, T., Király, R., Víg, A., and Nagy, T.: Refactoring in Erlang, a Dynamic Functional Language *In Proceedings of the 1st Workshop on Refactoring Tools, pages 45–46, Berlin, Germany, extended abstract, poster (2007)*

[9] Erlang - Dynamic Functional Language *http://www.erlang.org*

[10] T. Kozsik, Z. Horváth, L. Lövei, T. Nagy, Z. Csörnyei, A. Víg, R. Király, M. Tóth, R. Kitlei.. Refactoring Erlang programs. *CEFP'07, Kolozsvár (2007)*

[11] Király Roland, *Funkcionális programozási nyelvek* EKF TTK TAMOP412 2010 120 oldal.

[12] Király, R., Kitlei R.: *Complexity measurments for functional code* 8th Joint Conference on Mathematics and Computer Science (MaCS 2010) refereed, and the proceedings will have ISBN classification July 14-17, 2010

[13] Király, R., Kitlei R.: *Implementing structural complexity metrics in Erlang.* '10 ICAI 2010 – 8th International Conference on Applied Informatics to be held in Eger, Hungary January 27-30, 2010

[14] Király, R. and Kitlei, R.: *Implementing structural complexity metrics for Erlang* Poster on the 8th International Conference on Applied Informatics, ICAI 2010, 2010

Roland Király
Eszterházy Károly Collage, Institute of Mathematics and Informatics