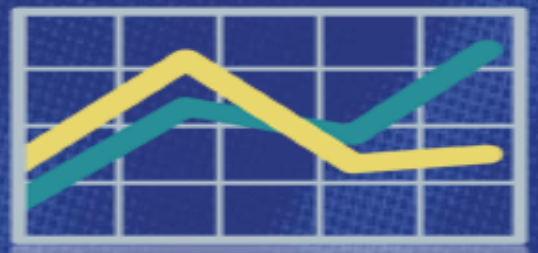
International Conference on
Computer Science and Education in Computer Science
CSECS'11

Facilitating Quality Assurance through a Source Code Metrics Framework



Nikolay Grozev Neli Maneva Delyan Lilov

Overview

- Introduction
- Preceding work
- Extended framework
- Prototype and validation
- Conclusion and future work
- Authors and Acknowledgements
- Questions

INTRODUCTION

Introduction

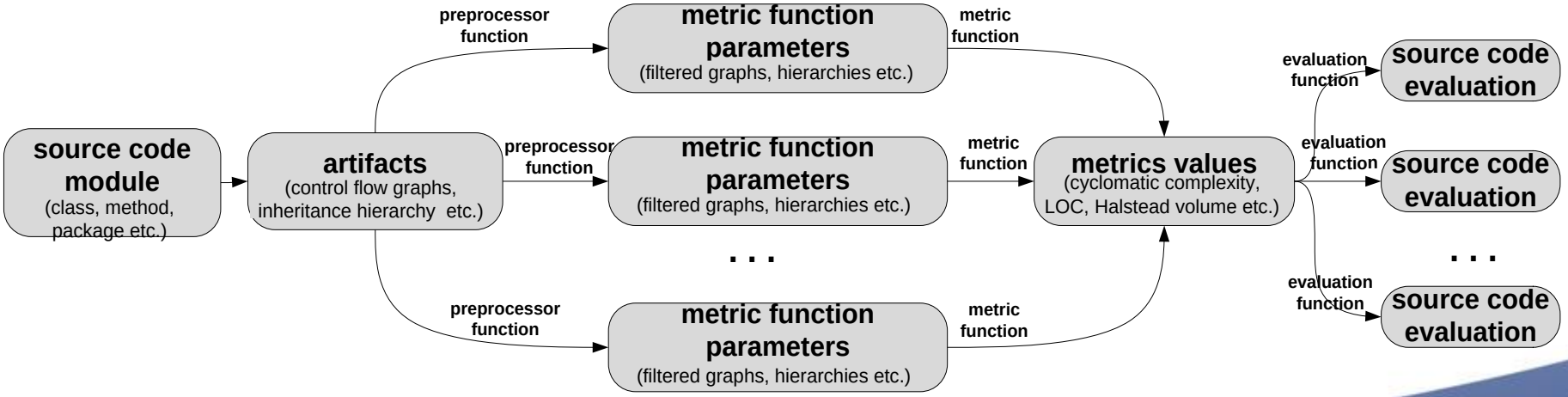
- Optimizing QA activities is a top priority, especially in times of economical hardship
- Static source code metrics present a promising approach to partially automating QA work
- We extend our previous research in the area by employing:
  - A CCC (Constant/Continuous/Correct) approach
  - An approach that can facilitate the so-called "umbrella" activities
- As a result our previous framework is technically and conceptually extended to better incorporate in practitioners' daily life.

PRECEDING WORK

Core source code framework

Design and structure

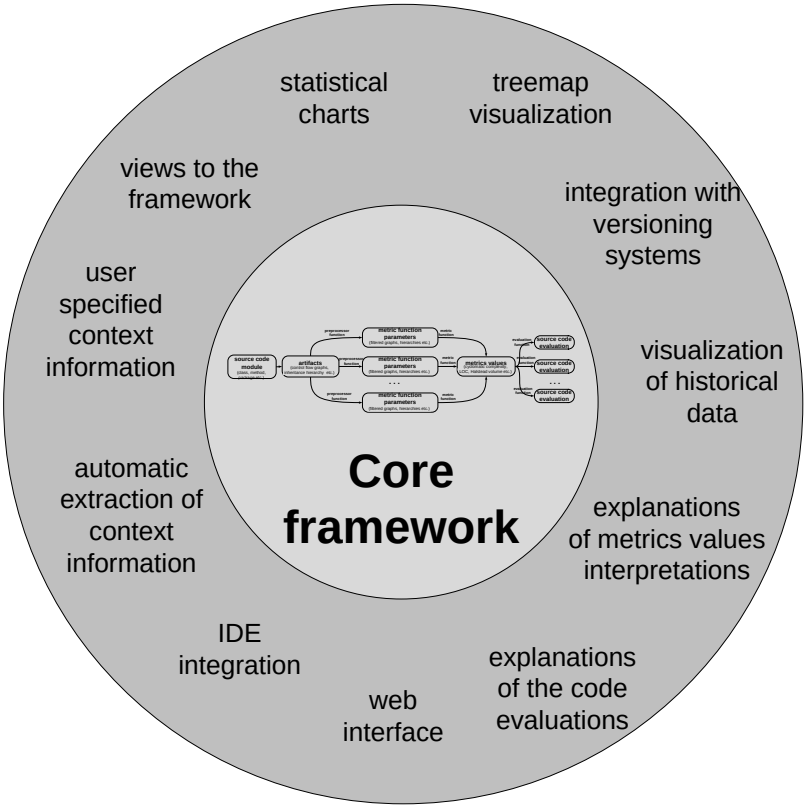
- An abstract framework allowing practitioners to consider the *context* of the code measurements.
- Comprises a number of separate modules modeled as functions
  - Metric functions – extract the metrics values based on some code artifacts;
  - Preprocessor functions – prepare the artifacts used by the metric functions;
  - Evaluation functions – combine metric values into meaningful code quality evaluations.
- A user can "hook" contextual logic through custom preprocessor and evaluation functions.



EXTENDED FRAMEWORK

Extended Framework

Overview



- The extended framework adds conceptual and technical extensions to the core one
- Its main focus is usability
- The extensions are “inspired” by practical experience gained while trying to incorporate a prototype of the core framework

Extended Framework

Extensions

- Interface of the framework
  - Should be accessible to both engineering and managerial staff
  - Web interface – integrated with popular web based project management and tracking systems used by managerial staff
  - IDE integration - for engineers
- Metrics values visualizations
  - Visualization techniques to spot code modules with poor quality - statistical charts, treemaps, polymetric views etc.
  - Visualization of historical trends of metrics values and assessments for continuous code quality monitoring.
  - Version control systems can be source of historical information.

Extended Framework

Extensions

- Explanation system
  - Engineers find it difficult to interpret metrics values. They should be provided with interactive info on metrics values and interpretations.
  - Aggregated evaluations should also be explained so to evaluate for "false positives". This can be achieved by an expert system.
- Framework settings
  - Core framework represents solely a source code evaluation scheme
  - Defining the correct functions may be a time consuming task requiring significant expert knowledge
  - A set of both preprocessor and evaluation functions can be predefined for often recurring contexts. A user has to specify the context of the application.
  - Automatic context detection based on project structure - binaries, import statements, naming conventions etc.

Extended Framework

Extensions

- Views to the framework
  - The framework should facilitate the different roles in the software lifecycle by providing different user account capabilities - *views*.
  - A view defines how a user accesses the system and his/her accessible functionalities.
  - Predefined views:
    - View for source code developers - IDE access only. Suitable for interactive feedback when programming and testing.
    - View for senior software engineering staff - access to all features. The only view with access to the framework settings.
    - View for managerial staff - web interface only. Historical information/trends, high-level visualizations etc.
  - Custom views can be defined as well.

PROTOTYPE AND VALIDATION

Prototype and validation

- Current prototype status:
  - Prototype of the core framework.
  - Eclipse IDE integration.
  - Metrics visualizations - statistical charts, treemap visualization etc.
  - Context settings are done manually.
  - No explanation engine, historical information and support for views yet.
- Early adopters' feedback:
  - The close integration of the visualizations within the IDE allowed for interactive and understandable code quality feedback. ✔
  - Visualizations saved a lot of time when conducting code reviews. ✔
  - Lack of explanations of the code evaluations. ✘
  - Absence of easy ways to tune contextually the framework. ✘
  - Managerial staff could not test due to lack of appropriate interface. ✘
- The Identified problems can be overcome by implementing the rest of the framework.

CONCLUSION AND FUTURE WORK

Conclusion and future work

- A framework supporting quality assurance was presented.
- The framework extends our previous rather abstract one with a set of concrete conceptual and technical features.
- These features/extensions are meant to increase its usability in the context of daily software development and management.
- The results from the experiments with a preliminary prototype were described.
- Ideas for further research:
  - To implement in a prototype all of the proposed in this study core framework extensions and to examine them;
  - To study more source code metrics so as to decide which of them can be added to the base set of metrics.

Authors and Acknowledgements

- **Nikolay Grozev**, nikolay.grozev@gmail.com
- **Assoc.Prof. PhD Neli Maneva**, Software Engineering Department, Institute of Mathematics and Informatics - BAS neli.maneva@gmail.com
- **Delyan Lilov**, Musala Soft Ltd, delyan.lilov@musala.com

This work has been partially supported by the Bulgarian National Science Research Fund (project ДМУ02/18 / 18.12.2009).

Questions

Thank you!