A Framework for Source Code Metrics

Neli Maneva, Software Engineering Department, IMI, BAS
Nikolay Grozev, Musala Soft Ltd
Delyan Lilov, Musala Soft Ltd
Overview

- Introduction
- Background and Analysis
- Source code framework
- Prototype and validation
- Conclusion and future work
- Authors and Acknowledgements
- Questions
INTRODUCTION
Introduction

• Why use static source code metrics?
  – Source code - an essential part of every software system
  – Static analysis is a useful best practice
  – Solid theoretical background
  – Monitoring and assessment through metrics

• Metrics are rarely used in practice – this is for a reason!

• Our goals:
  – Analyze metrics usage problems
  – Define requirements for a metrics framework
  – Design an abstract framework, meeting the requirements
  – Prototype and validate the framework
BACKGROUND AND ANALYSIS
Background and Analysis
Classification of existing tools

- Reporting tools
  - Compute metrics values and produce reports
  - The user must know details about all metrics

- Combining tools
  - Besides metrics values, produce “combined” evaluations
  - Removes the need to know all about the metrics
Background and Analysis
Problems with existing tools

- Both reporting and combining tools are “hardcoded” in nature
  - Almost no settings available
  - Everything is measured and interpreted in the same way
- Should we really measure everything with the same “scales”?
Background and Analysis
Context and its aspects

• Source code metrics tools should:
  – Extract metrics values with regards to *what is being evaluated*.
  – Combine metrics values with regards to *what is being evaluated*.

• Context – additional info about *what is being evaluated*:
  – Programming languages
  – Used technologies and libraries
  – Project size
  – Architecture
  – Application area
  – Etc.

Conclusion: A *successful source code metrics tool should be “context sensitive”*
Background and Analysis

Objectives

- Define an abstract/general framework for evaluating source code quality
- It should flexibly accommodate:
  - Contextual extraction of metrics values
  - Contextual combinations of metrics values
- It should provide a solid basis for practical tools through a set of extension points
- Provide some speculations about possible extensions of the ideas of the framework
- Validate the feasibility of the approaches in practice
SOURCE CODE FRAMEWORK
DESIGN AND STRUCTURE
Source code framework
Design and structure

• Base set of metrics
  – Toolbox of metrics
  – Basis for all evaluations

• A stepwise framework (evaluation scheme) that simulates an expert’s work of evaluating source code through metrics

• Modeling each step as a function:
  – Metric functions – extract the value of a single metric
  – Preprocessor functions – used to prepare the parameters for the metric functions
  – Evaluation functions – combine the values of metrics into a meaningful source code evaluation
Source code framework
Design and structure

- Contextual user specified logic is “hooked” by:
  - Preprocessor functions – filter the data for the metrics, less “noisy” values
  - Evaluation functions – combine the values of the metrics, benefits from the less “noisy” values
Source code framework
Design and structure – preprocessor functions

- Preprocessor functions:
  - Determine which artifacts are relevant for the computation of a metric
  - Usually such functions filter the irrelevant to a metric input elements
  - Using preprocessor functions results in more “accurate” metric values
Source code framework
Design and structure – evaluation functions

- Functions highlighting design problems in the code
  - Promising researches into the area of OOP design problem recognition using
  - Can be improved by the less “noise” after the usage of preprocessors
- Functions combining metrics values into a new numerical value
  - Can be modeled as real valued functions
  - Linear combinations vs. Machine learning techniques
PROTOTYPE AND VALIDATION
Prototype
Smart source code analyzer (SSA)

- Eclipse plug-in
- A tool, based on an implementation of the framework
- Also used for validation purposes
Practical validation

• Validation by prototype usage:
  – Used to analyze many open source projects
  – Used in real life development – code quality assessment, code reviews

• Validation results:
  – Users like getting aggregated information and being abstracted from the details of the different metrics
  – Quicker code reviews
  – Problems with setting the contextual information – false positives
  – Problems understanding “Why this is bad?”

• Problems can be overcome through additional functionalities, planned for the prototype.
CONCLUSION
AND FUTURE WORK
Conclusion

• An analysis of existing tools and approaches was briefed
• A general framework for evaluating source code through metrics was described.
• A prototype was built and used for validation

• Future work:
  – Methods for metrics preprocessing and combinations
  – Visualization techniques
  – Incorporation throughout the software lifecycle
Authors and Acknowledgements

Assoc.Prof. PhD Neli Maneva, Software Engineering Department, Institute of Mathematics and Informatics - BAS neli.maneva@gmail.com

Nikolay Grozev, Musala Soft Ltd, nikolay.grozev@musala.com

Delyan Lilov, Musala Soft Ltd, delyan.lilov@musala.com

This work was partially supported by the National Innovative Fund attached to the Bulgarian Ministry of Economy and Energy (project № 5ИФ-02-3 / 03.12.08).
Questions

Thank you!