	DATE	CHANGES
3	03.10.2013	Rule sets removed (i. e. moved to other document)
2	27.09.2013	References; minor plan changes
1	25.09.2013	(initial version)

Rule-based code analysis

PROJECT PLAN

Master Thesis

Period: 1. October 2013 – 1. April 2014

Student name: Stefan Zurfluh

Email: zurfluhs@student.ethz.ch

Supervisor name: Julian Tschannen

1. PROJECT DESCRIPTION

Overview

Static code analysis is a powerful means to improving code quality in different contexts and on different levels. Its ultimate goal is – using an automated (but often customizable) analysis tool – to help the programmer write better programs. Be it due to just a typing error (i. e. one that does not lead to a syntactic error), some other form of incautiousness, or perhaps a lack of knowledge, – a code analysis framework will be capable of issueing warnings, errors, and even correctional suggestions each referring to a part of the source code. Code analysis is a useful part in the world of software quality improving concepts[1].

In addition to the already established theoretical background in program analysis, implementations of rule-based code analysis tools exist for several programming languages and environments[2].

The task in this project is to implement a rule-based code analysis for Eiffel. These rules can for example be about coding style, code structure, or API usage. A basic set of rules helps to enforce a standard coding style, and users can change or extend the existing rules to fit their needs.

Scope of the work

---

[1] Other such concepts are (automated) testing, formal verification and Design by Contract, etc.

[2] See *section 2, related software*.

A tool for rule-based Eiffel code analysis will be implemented. The set of rules will be designed in a manner that allows extensibility and customizability (such as possibility of enabling/disabling rules or setting the warning level). The tool shall analyze an Eiffel System (in this context a project, respectively) and output a list of rule violations found in the code.

The tool will be designed as a semi-independent module. It will get the syntactic structure of the source code from the Eiffel compiler, which will be used for the analysis. The analysis result will then be output to the user. Further steps include a smooth and easy-to-use integration into the EiffelStudio UI, and integrating other verification tools (AutoTest, AutoProof; integration of code analysis into EVE) in order to be able to run a series of tools automatically on every compilation.

Intended results

The resulting software is intended to be fully usable, working, and well-integrated in its environment. It should also be a basis for further extensions and improvements.

2. BACKGROUND MATERIAL

Reading list

- [1], Good programming style and practices
- [1,10], Eiffel as a language
- [2,3,4], Compiler design in general
- [5,6,8], program analysis

Related Software

- *PMD*, http://pmd.sourceforge.net/.
- *JetBrains ReSharper*, http://www.jetbrains.com/resharper/.
- *Microsoft FxCop*, http://www.microsoft.com/en-us/download/details.aspx?id=6544.

3. PROJECT MANAGEMENT

Objectives and priorities

- Design and implementation of a rule-based code analysis module
- Command line functionality
- User interface / EiffelStudio integration

- EVE integration
- Complete documentation and report

Criteria for success

Minimum quality requirements

- Command line usage and output
- All predefined test cases working
- Rules according to rule set 1[1] implemented

Expected requirements

- Integration into EiffelStudio user interface
- Two thirds of the rules according to rule set 2 implemented

Requirements for a result that significantly exceeds expectations

- Integration into EVE
- All rules according to rule set 2 implemented
- All rules according to rule set 3 implemented
- Hints / enforcement of naming conventions

Quality management

Documentation

The comprehensive documentation will be part of the report. In addition, the source code will be commented briefly.

Validation steps

- In the weekly meetings, the current status will be observed and evaluated.
- Partial test-driven development: Test cases that cover at least the most important features will be run regularly during development.
- Optionally, alpha testing by users outside the project.

4. PLAN WITH MILESTONES

Project steps

1. Getting acquainted with the EVE and EiffelStudio source code
2. General literature study of program analysis and rule-based code analysis
3. Object-oriented design of analysis and rules
4. Basic implementation including command line usage and output
5. Complete Implementation of rule set 1

---

[1] The rule sets will be precisely defined at a later date.

6. EiffelStudio user interface integration
7. Implementation of rule set 2
8. EVE integration
9. *Implementation of rule set 3
10. *Implementation of naming convention rules
11. Testing
12. Write documentation
13. Write master thesis report

* optional

Deadline

1st April 2014

Tentative schedule

(corresponds to project steps above)

Milestone / Week		40	41	42	43	44	45	46	47	48	49	50	51	52	1	2	3	4	5	6	7	8	9	10	11	12	13
1	Source code study																										
2	General literature study																										
3	Design																										
4	Basic implementation																										
5	Rule set 1 implementation																										
6	UI integration																										
7	Rule set 2 implementation																										
8	EVE integration																										
9	Naming conventions																										
10	Rule set 3 implementation																										
11	Testing																										
12	Write documentation																										
13	Write thesis report																										

A. REFERENCES

[1] Bertrand Meyer: *Object-Oriented Software Construction, 2nd edition*, Prentice Hall, 1997.

[2] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, *Compilerbau, Teil 1*, Oldenbourg, 1999.

[3] Alfred V. Aho, Monica Lam, Ravi Sethi, Jeffrey D. Ullman, *Compilers: Principles, Techniques, and Tools (2nd Edition),* Addison-Wesley, 2007.

[4] Steven Muchnick, *Advanced Compiler Design and Implementation*, Morgan Kaufmann Publishers, 1997.

[5] *Data-flow analysis*, Wikipedia, http://en.wikipedia.org/wiki/Data-flow_analysis.

[6] *Static program analysis*, Wikipedia, http://en.wikipedia.org/wiki/Static_program_analysis.

[7] *Eiffel Verification Environment (EVE)*, http://se.inf.ethz.ch/research/eve/.

[8] Flemming Nielson, Hanne Riis Nielson, Chris Hankin: *Principles of Program Analysis,* Springer, 1999.

[9] EiffelStudio Developer Wiki, http://dev.eiffel.com.

[10] Eiffel Documentation, http://docs.eiffel.com.