Interpretation von Software Qualitätsmetriken aus automatisierter statischer Analyse

Andreas Gerstinger

Institut für Computertechnik

ICT

Institute of Computer Technology

IIR Konferenz Software Testen & Software Qualität

26.-27.2.2007, Holiday Inn Vienna South

Agenda

- I will try to answer the following questions

- Why a static analysis tool?
- What is internal and external quality?
- What are the requirements for a suitable tool?
- How can I find out which metrics are relevant?
- Which metrics really are relevant?

Motivation

- **Software Maintenance** is a major cost factor
- "80% of the lifetime cost of a piece of software goes to maintenance." (first sentence in Sun Java Code Conventions)
- → streamlining maintenance is essential
- Practical experience also shows that most SW Engineering work is maintenance
  - sometimes very, very, very old code...

More Motivation

- Static Analysis is required in standards
  - mandated by several international standards (especially in the safety critical systems community)
- Static Anlaysis streamlines code reviews
  - if performed before the review, it shows where most problems are expected
- Static Analysis useful for SWQM personell
  - gives quick overview and enables a preliminary assessment, also for subcontracted code

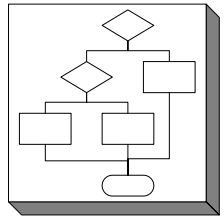
Maintenance Efficiency

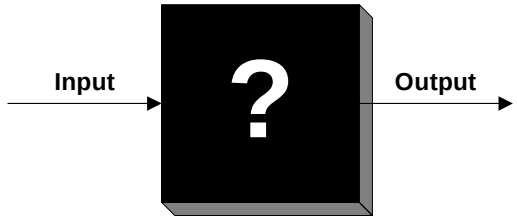
- Concerns all maintenance types
  - corrective, perfective, adaptive, preventive

- Mainteance efforts consist of:
  - reading code
  - understanding code
  - finding places where to edit
  - bug tracking
  - code analysis
  - improvements

Internal vs. External Quality

- Static Analysis measures the **internal** quality of a software system
  - it is a white-box analysis, opposed to black-box testing





- Internal Quality often correlates with external quality
  - Weak internal quality can cause bad external quality
  - → **the system is perceived as having "bad quality" by the users!**

Requirements for the Tool

- The objective was to reduce number of tools, and to ideally introduce one standard tool for the company
- The following requirements were established. The tool shall:
  - support all relevant platforms (Win/Linux)
  - support all relevant languages (C/C++/Java/C#)
  - be easy to use
  - have a GUI and command line interface
  - be extensible with custom rules (to adapt to company coding guidelines)
  - not force developer to change good code so the tool can run
  - output human readable and automatically analysable
  - calculate useful metrics
  - automatically verify as many of the Coding Guidelines as possible

Problems

- Which rules are good?
- What are the desired ranges for metrics?
- Which metric really influences maintainability?

- There are essentially three methods to set limits:
  (A) Take limits from literature (CC<10, one return point, no goto,...)
  (B) Measure many projects and take mean as a guideline
  (C) Correlate metrics with qualified opinion of expert developers

- → Method C selected

Outline of Method

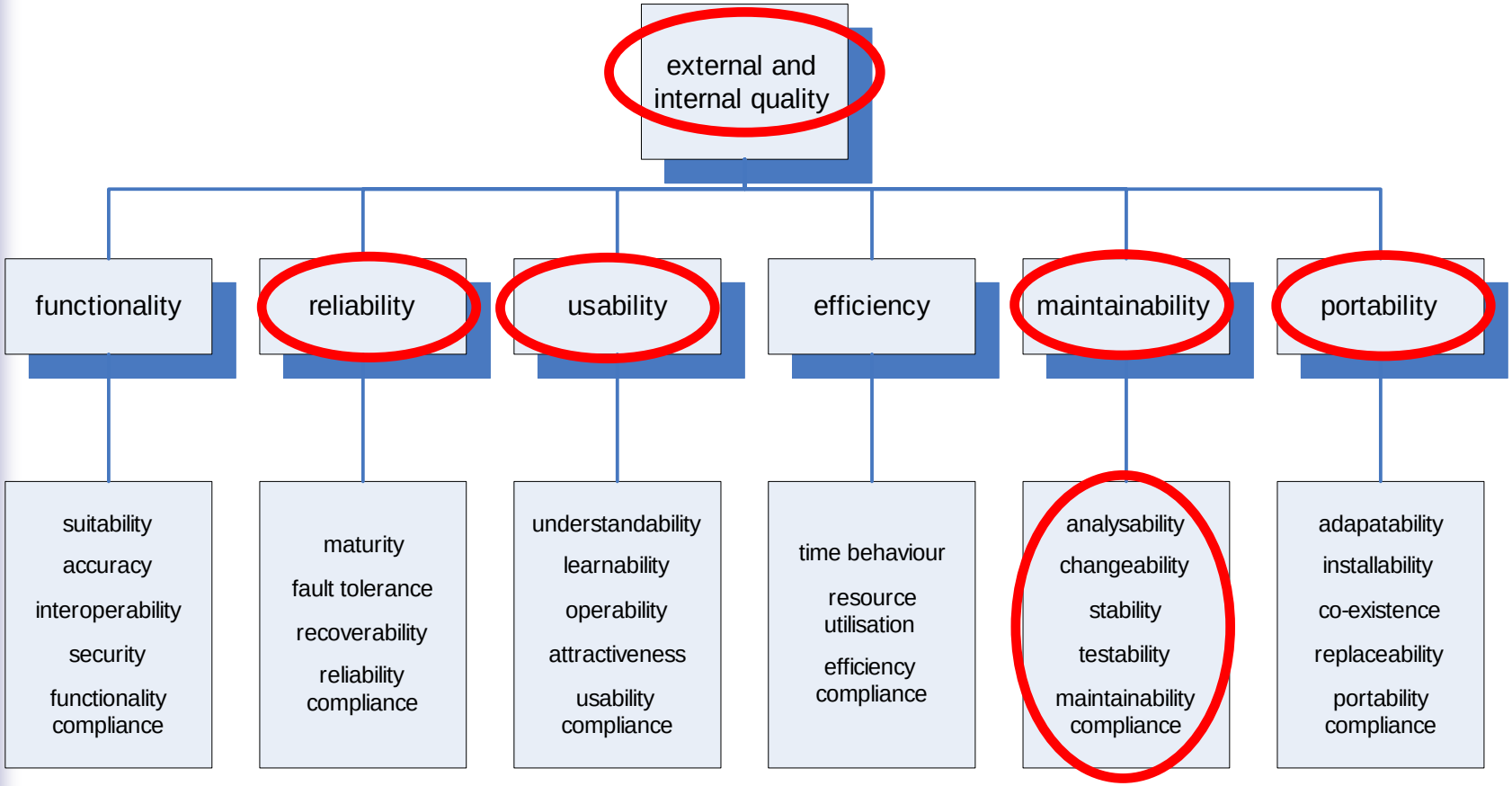
1. Create a questionnaire with relevant questions regarding software quality and get answers from expert developers for various software packages they work with

2. Automatically measure potentially interesting metrics of the software packages

3. Correlate questionnaire responses with the measured metrics to find out which metric correlates with which property

Step 1: Questionnaire

- Questionnaire based on ISO9126 Quality Model



Step 1: Questionnaire

- Consists of 12 questions, to be answered with Austrian school grades
  - 1 – best
  - 5 – worst

- Very easy to understand and quick to get responses
- Four questions about external quality, eight questions about internal quality

Step 1: Questionnaire

- Some questions regarding external quality

Question	Explanation	Grade
How would you rate the **overall externally perceived quality** of the system?	This includes all externally visible properties such as functionality, reliability, usability, efficiency,...	
How would you rate the **reliability** of the system?	Reliability is "the capability of the software product to maintain a specified level of performance when used under specified conditions". For your answer, consider unexpected crashes, the tolerance of the software to internal and external faults, the recoverability from faults, the perceived stability, and the number of faults during operation.	

Step 1: Questionnaire

- Some questions regarding internal quality

Question	Explanation	Grade
How would you rate the **overall internal quality** of the software?	This question is geared towards the overall maintainability and technical "beauty" of the internal quality.	
How would you rate the code **readability based on code clarity**?	Is the code (not the comments) easy to read? Are variable/function/class names clear? Is the code well structured? Can you read the code "like a book"?	

Step 2: Automated Static Analysis

- Analysis limited to C# Code
- All code in one directory, each module in one subdirectory
- Perl script running over all subdirectories
  - Execution of static analysis tool
  - Extraction of potential quality indicators from static analysis output files
- Process automated – easy to add more code

Step 2: Automated Static Analysis

Some Metrics that were measured:

- Average Cyclomatic Complexity (CC) per method
- Average Cyclomatic Complexity excluding methods with CC=1
- Average Cyclomatic Complexity excluding methods with CC=1 and CC=2
- Percentage of methods with Cyclomatic Complexity greater than 3
- Percentage of methods with Cyclomatic Complexity greater than 5
- Percentage of methods with Cyclomatic Complexity greater than 10
- Percentage of methods with Cyclomatic Complexity greater than 15
- Percentage of methods with Cyclomatic Complexity greater than 30
- Average Class Size (in effective Lines of Code (eLOCs))
- Average Method Size (eLOCs)
- Average number of public attributes per class
- Average number of comment lines per method
- Average number of return points per method
- Number of quality notices per KLOC (Kilo LOC)
- Number of severe quality notices per KLOC
- Number of quality notices pertaining to readability per KLOC
- Average file size (eLOCs)
- Ratio of comment lines to total lines
- Average interface complexity

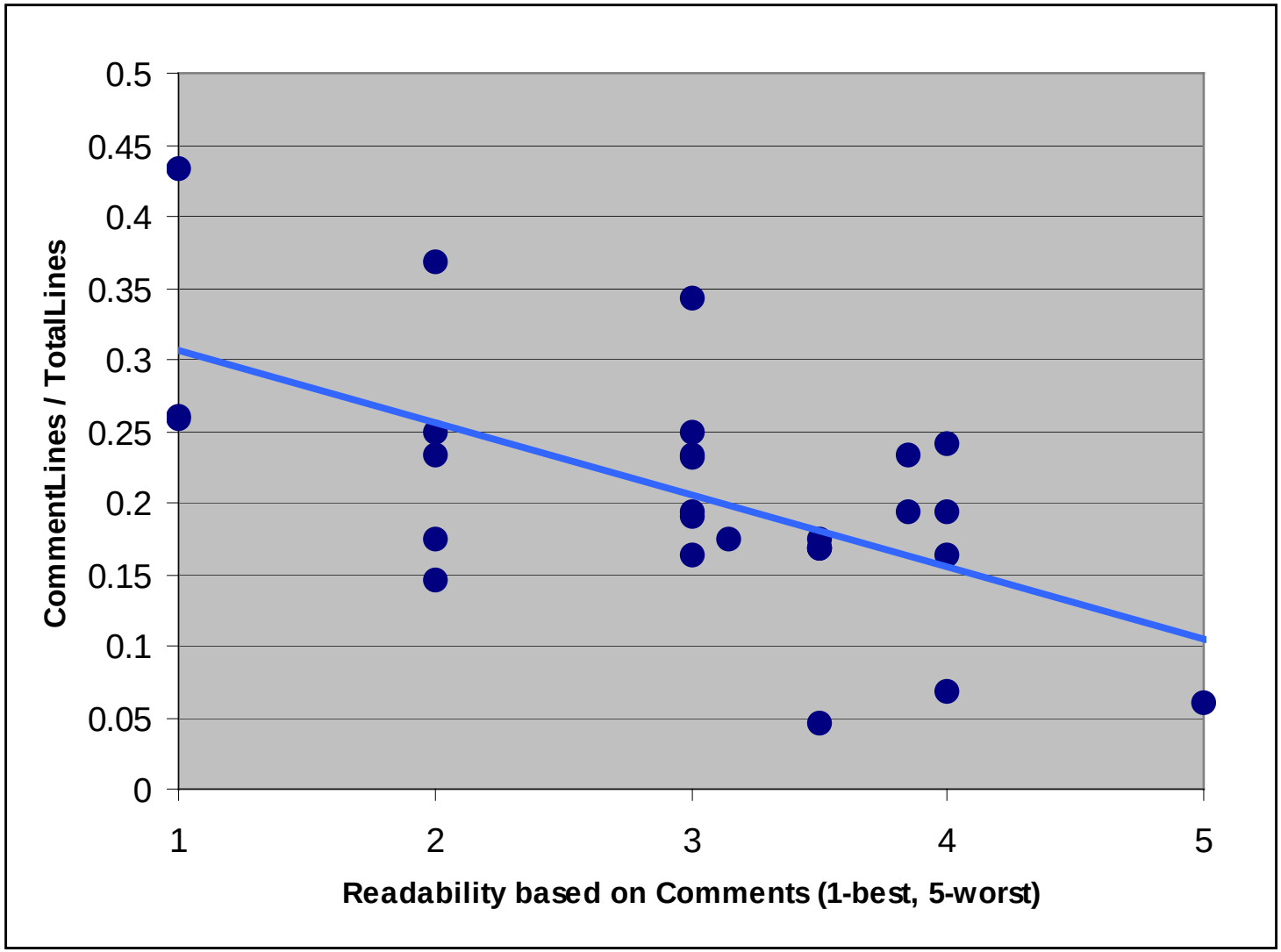
Step 3: Correlation

- Correlation between questionnaire reponses and potential quality indicators was analysed

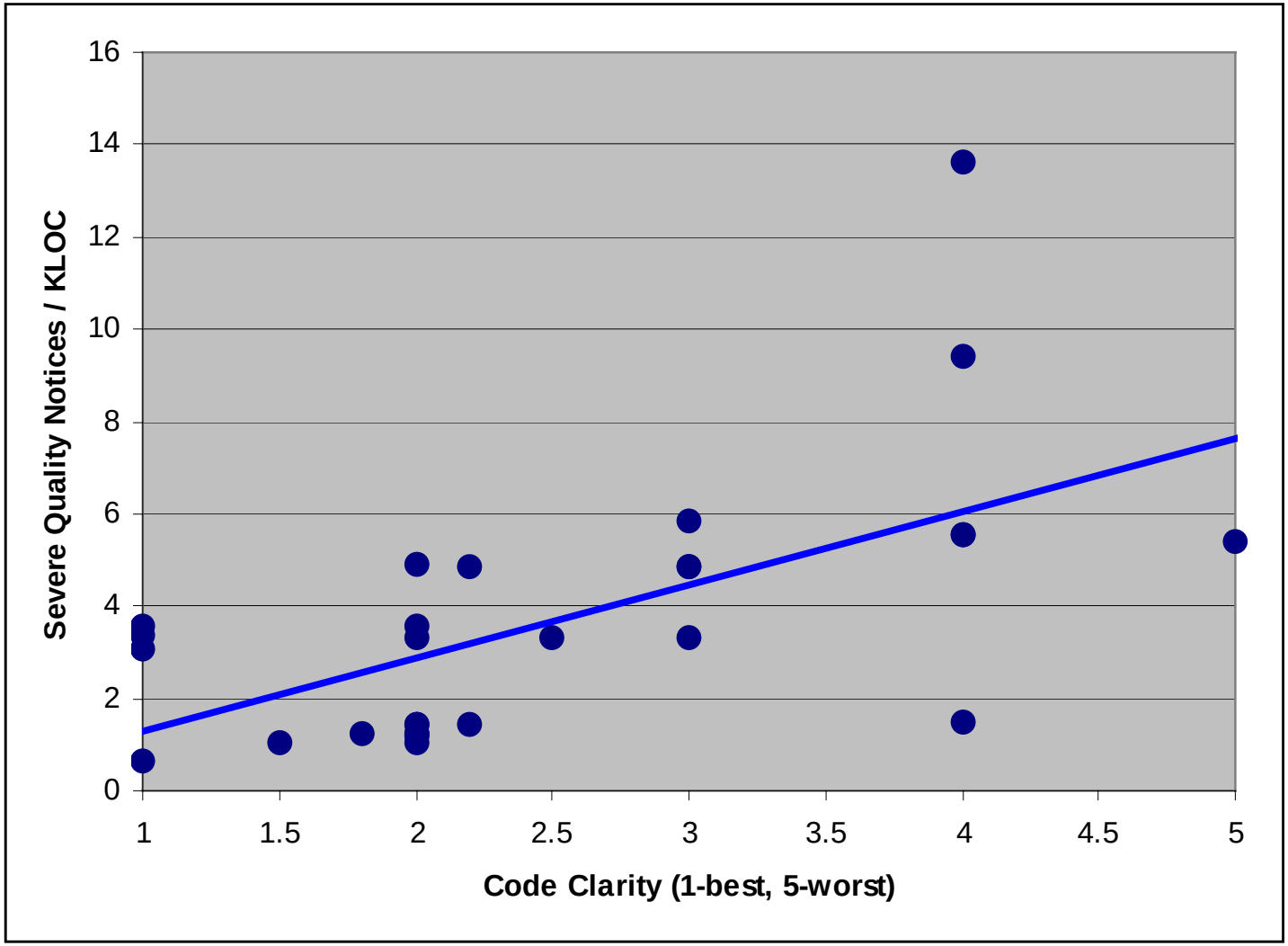
- XY Graphs to see correlation
- Trendlines added
- Strength of correlation was calculated

- Which metrics are relevant? Which are not?
  - Let's look at some graphs!

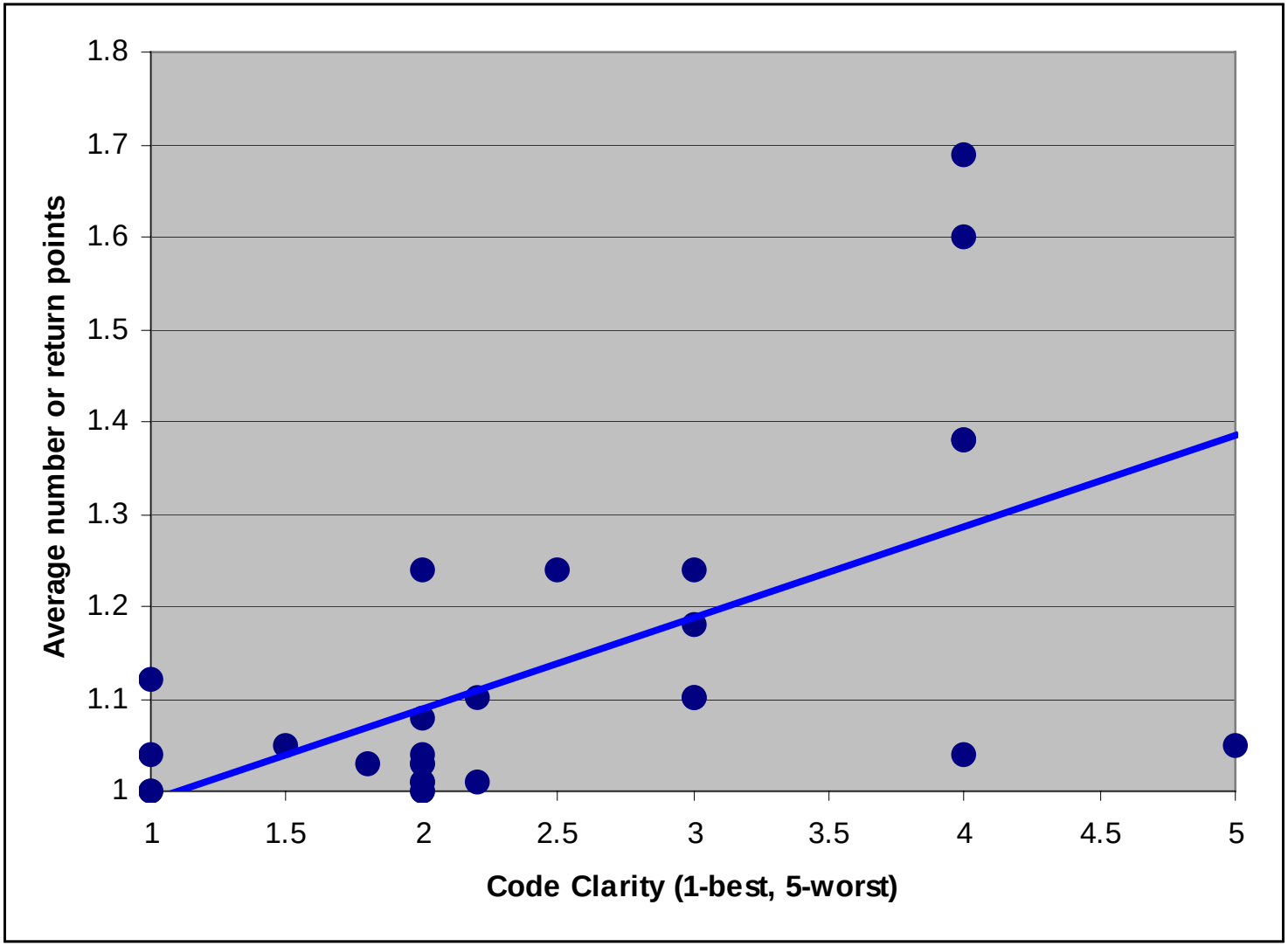
Graph 1: Readability vs. Comment Density



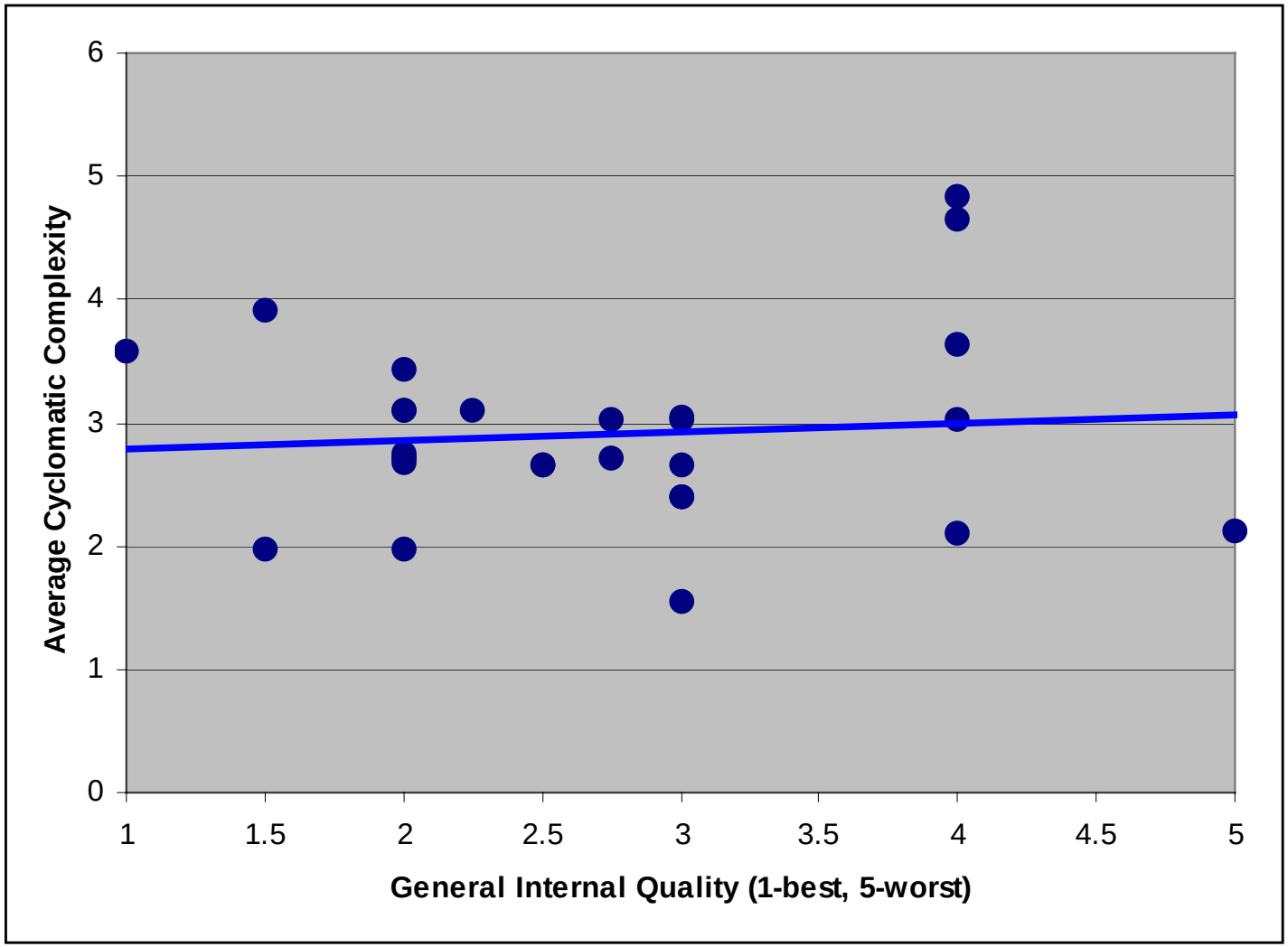
Graph 2: Code Clarity vs. Quality Notices



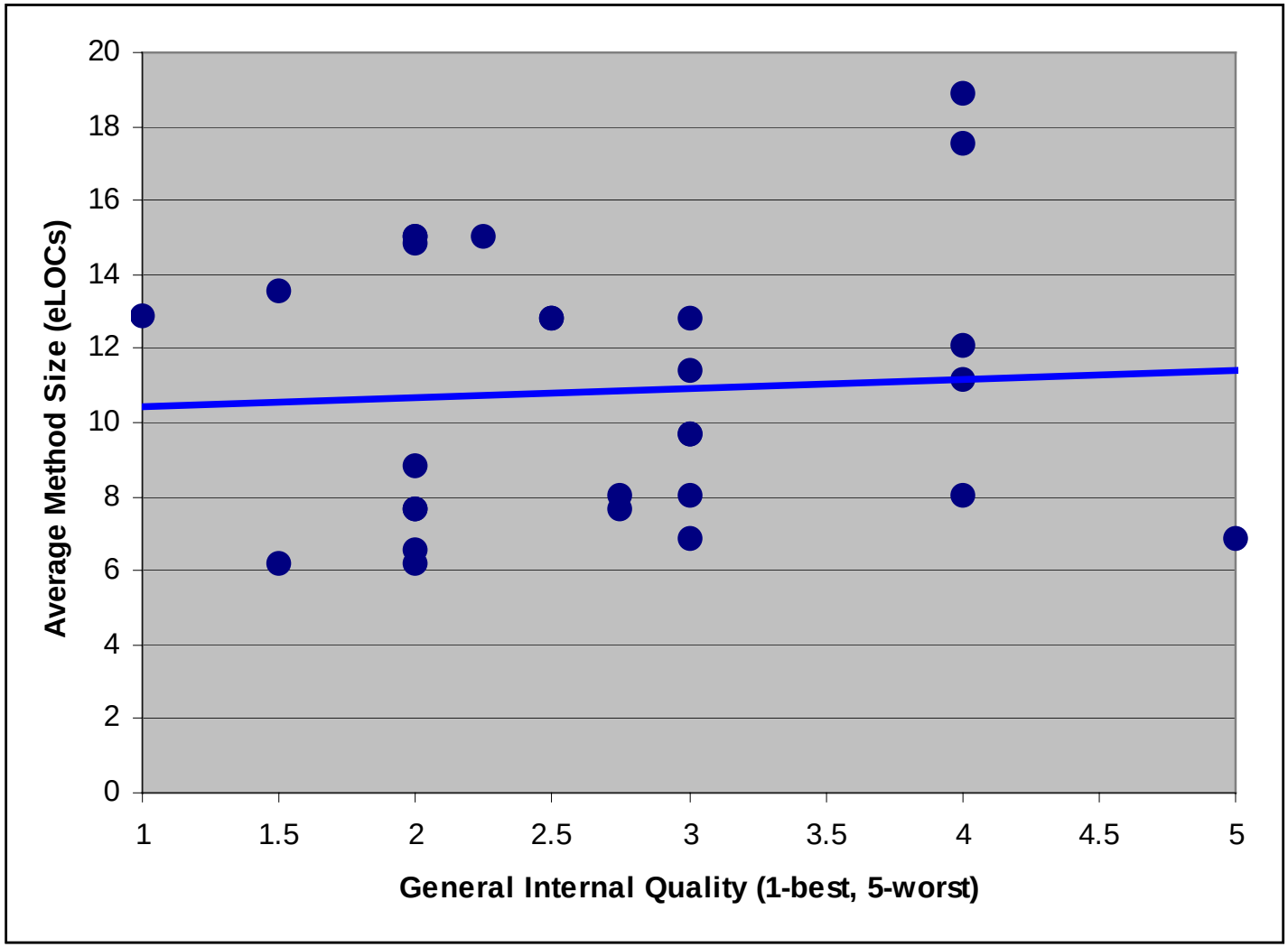
Graph 3: Code Clarity vs. Return Points



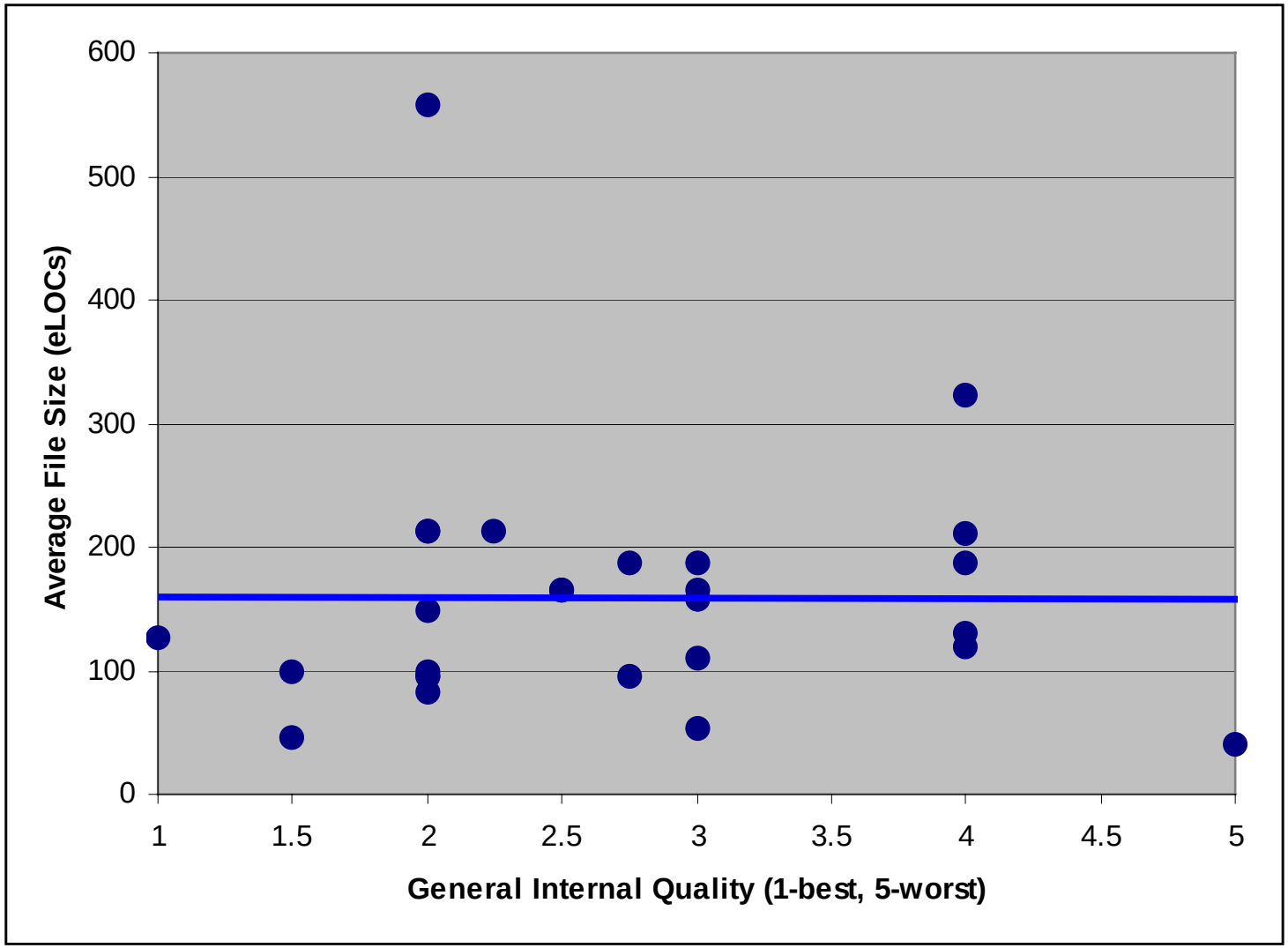
Graph 4: Internal Quality vs. CC



Graph 5: Internal Quality vs. Method Size



Graph 6: Internal Quality vs. File Size



Summary of Results

- Strongest correlation with perceived internal quality:
  - Comment density
  - Control Flow Anomalies
- No correlation with perceived internal quality:
  - Cyclomatic Complexity
  - Average Method Size
  - Average File Size
  - ...

Conclusions

- Static Analysis is essential for software quality
- Tool must be easy to use, and must give immediate feedback to developers
- Correlation of metrics with perceived quality gives surprising results
- But: "Correlation does not imply causation!"
- There is still a lot of work to be done...

☺ **Thank you!** ☺