An Overview of Software Code Quality and Connection to Business Process Quality

Yann-Gaël Guéhéneuc

1st IEEE International Workshop on Communicating Business Process and Software Models
(Quality, Understandability, and Maintainability)

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
I DON'T GET YOUR CODE
NEITHER DO I
BUT IT SEEMS TO WORK

THE ART OF PROGRAMING
We offer three kinds of service:

GOOD - CHEAP - FAST

You can pick any two

GOOD service CHEAP won’t be FAST
GOOD service FAST won’t be CHEAP
FAST service CHEAP won’t be GOOD
Outline

- Example
- Introduction
- Trail
- Definitions
- Following the Trail
- Conclusion
Introduction

- Software total cost of ownership (TCO) includes all the costs to acquire and keep a program running, from specifications to licenses to training

- Maintenance contributes between 50% and 90% of TCO [Erlikh, Lientz]
Introduction

- Four factors impact maintainers [ISO9126]
  - Software analysability
    • Their ability to identify and understand defects
  - Software Changeability
    • The effort needed to change the software
  - Software Stability
    • The sensitivity of the software to change
  - Software Testability
    • The effort needed to test the software
Introduction

- Two factors impact software analysability
  - Software quality characteristics
    - Software products are the output of software processes. Product quality is determined by the degree to which the developed software meets the defined requirements
  - Developers’ characteristics
    - Status, experience, education, gender
    - ...
Introduction

Three dimensions characterise software quality characteristics

- Functional vs. non-functional
  - At runtime vs. overall
- Internal vs. external
  - Maintainability vs. understandability
- Metric-based vs. practice-based
  - Objective vs. subjective
Introduction

- Three dimensions characterise software quality characteristics
  - Functional vs. non-functional
    - At runtime vs. overall
  - Internal vs. external
    - Maintainability vs. understandability
  - Metric-based vs. practice-based
    - Objective vs. subjective
Outline

- Example
- Introduction
- Trail
- Definitions
- Following the Trail
- Conclusion
In this talk, we will discuss how to measure the overall quality characteristics of maintainability and understandability objectively and subjectively.
Maintainability
Understandability

Quality Models → Models → Measures

Good Practices → Definition → Detection → Occurrences

Social Studies → Characteristics

Eye-tracking Studies → Behaviour

Metrics

Factors
Outline

- Example
- Introduction
- Trail
- **Definitions**
- Following the Trail
- Conclusion
Definitions

- **Maintainability**
  - Ease with which a software system can be modified [IEEE Standard Glossary of Software Engineering Terminology]

- **Understandability**
  - Ease with which a software system can be understood [Boehm]
Definitions

- Software code artefacts
  - Source code, such as Java
  - UML-like models
  - …

→ Labelled, weighted multi-graphs
Definitions

- Business process artefacts
  - Languages, such as BEPL
  - BPMN
  - ...

⇒ Labelled, weighted multi-graphs
Outline

- Example
- Introduction
- Trail
- Definitions
- Following the Trail
- Conclusion
Quality Models

Problem: how to measure software code artefacts?
Solution: define and use quality metrics and models
Quality Models

- Quality models
  - A model with the objective to describe, assess and—or predict quality [Deissenboeck]

- Metrics

- Models
Quality Models

- Metrics have been well researched since the early days of software engineering
  - Chidamber and Kemerer
  - Hitz and Montazeri
  - Lorenz and Kidd
  - McCabe
  - ...

(Don’t miss Briand et al.’s surveys on the metrics of cohesion and coupling)
Quality Models

- Metrics alone are meaningless
  - Size vs. IQ
  - ...
  - Hotness vs. IQ [Srivastava]

Quality Models

- Few quality models exist
  - Boehm’s
  - ISO9126
  - McCall’s
  - QMOOD
  - …
Quality Models

- Bansiya and Davis’ QMOOD
  - Hierarchical model for OO designs
  - Structural and behavioural design properties of classes, objects, and their relationships
    - Reusability, flexibility, and complexity
  - Object-oriented design metrics, such as
    - Encapsulation, modularity, coupling, and cohesion
  - Validated using empirical and expert opinion on several large commercial systems
Quality Models

- Difficulty to relate metric values with quality characteristics, such as maintainability and understandability

- Difficulty to identify relevant metrics and show that the metrics measure what they are expected to measure
Quality Models

- Metrics
  - 3QM-Framework [Overhage et al.]
  - ProM [Vanderfeesten et al.]

- Models

There probably exist more?
Good Practices

Maintainability
Understandability

Quality Models
Models
Metrics
Measures

Definition
Detection
Occurrences

Social Studies
Characteristics
Factors

Eye-tracking Studies
Behaviour
Good Practices

Problem: how to identify good/bad practices?
Solution: collect and detect models of the practices
Good Practices

- Software engineering development and maintenance are based on a few principles
  - SOLID
  - Do not reinvent the wheel
  - Beware of assumptions
  - …
Good Practices

- Martin and Feather’s SOLID
  - Single responsibility
  - Open/closed
  - Liskov substitution
  - Interface segregation
  - Dependency inversion

(Don’t miss Michael Feather’s keynote on Thursday on the software useful life)
Good Practices

Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such way that you can use this solution a million times over, without ever doing it the same way twice.

—Christopher Alexander, 1977
Good Practices

Important assumptions

– That patterns can be codified in such a way that they can be shared between different designers.
– That reuse will lead to “better” designs. There is an obvious question here of what constitutes “better”, but a key measure is maintainability.

—Zhang and Budgen, 2012
(With minor adaptations)
Good Practices
Good Practices

- **Design Patterns**
  - A general reusable solution to a commonly occurring problem within a given context in software design

- **Design Antipatterns**
  - A design pattern that may be commonly used but is ineffective/counterproductive in practice
Good Practices

- Pattern solution = Motif which connotes an elegant architecture
Good Practices

How to identify in the architecture of a program micro-architectures similar to design motifs to explain the problem solved?

To compose objects in a tree-like structure to describe whole–part hierarchies
Good Practices

- What motifs and what model for these motifs?

- What model for the program architecture?

- How to perform the identification?
Good Practices

- What motifs and what model for these motifs? Design motifs and a motif meta-model \( \mathcal{M}_{DM} \)

- What model for the program architecture? Design Meta-model \( \mathcal{M}_I \)

- How to perform the identification? \( \mathcal{M}_{\mu A} \)
Good Practices

- Multi-layer framework for design motif identification

\[ S \implies \mathcal{M}_S \implies \mathcal{M}_I \implies \mathcal{M}_D \quad (\subset \{\mathcal{M}_{\mu A}\}) \]

- Information retrieval
  - Search space \( \mathcal{M}_I \)
  - Query \( \mathcal{M}_{DM} \)
  - Results \( \mathcal{M}_{\mu A} \)
Good Practices

- Multi-layer framework for design motif identification
Good Practices

- Constraint satisfaction problem solved with explanation-based constraint programming (e-CP) to obtain $\mathcal{M}_{\mu A}$
  - No a priori descriptions of variations
  - Justification of the identified micro-architectures
  - Strong interaction with the developers
Good Practices – Example

Design motif ($\mathcal{M}_{DM}$)

- Component
  - operation()
  - 1..n

- Leaf
  - operation()

- Composite
  - add(Component)
  - remove(Component)
  - getComponent(int)
  - operation()

For each component
  - component.operation()
Good Practices – Example

- Example of JHotDraw
  - Erich Gamma and Thomas Eggenschwiler
  - 2D drawing
  - Design patterns
Good Practices – Example
Good Practices – Example

- Micro-architecture \( M_{\mu A} \)

- Maintainability
- Understandability
\( V = \{ \text{component, leaf, composite} \} \)

\( C = \{ \text{leaf} < \text{component, composite} < \text{component} \} \)

\( D = \{ \langle \text{DrawingEditor, DrawingView} \ldots \rangle \} \)
\[
V = \{ \text{component, leaf, composite} \} \\
C = \{ \text{leaf} \preceq \text{component, composite} \preceq \text{component, composite} \succeq \text{component} \} \\
D = \{ \langle \text{DrawingEditor, DrawingView} \ldots \rangle \} 
\]
Good Practices

- Search space $M_I$ can be very large and the efficiency in time of the search very low.
- Use metrics and topology to reduce the search space.

Diagram:

- Stage 1: Metrics Constraint Evaluation (D → R
- Stage 2: Structural Constraint Evaluation (R → S
- Stage 3: Delegation Constraint Evaluation (S → P

48/75
Good Practices

- “Design” patterns
  - Workflow patterns [van der Aalst]

- “Design” antipatterns
  - In BPEL [Palma et al.]

More? Detection? Impact?
Problem: how to characterise developers?
Solution: study developers’ social behaviours
Social Studies

- Developers’ characteristics
  - Gender
    - Males vs. females
  - Professional status
    - Practitioner vs. Students
  - Expertise
    - Experts vs. novices

(Don’t miss the sessions Developers, What are you Thinking? and Context)
Social Studies

- Such studies are typically experiments
  - Independent variable (few)
  - Dependent variables (many)
  - Statistical analyses

- Threats to validity
Social Studies

- For example, impact on identifiers on program understandability
  - Identifier styles [Sharif, Binkley]
  - Identifier quality [Lawrie]
  - Developers’ gender and identifiers [Sharafi]
  - ...

Social Studies

- Independent variables
  - Gender: male vs. female
  - Identifier: camel case vs. underscore

- Dependent variables
  - Accuracy
  - Time
  - Effort
Social Studies

■ Subjects

Subjects’ Demography
(24 Subjects)

<table>
<thead>
<tr>
<th>Academic background</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ph.D.</td>
<td>Male</td>
</tr>
<tr>
<td>M.Sc.</td>
<td>Female</td>
</tr>
<tr>
<td>B.Sc.</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
</tbody>
</table>

■ Conclusions

Accuracy
Time
Effort
Social Studies

- Threats to validity
  - Construct validity
    • Measurements represent the construct being studied
  - Internal validity
    • A causal conclusion based on the study is warranted
  - External validity
    • Extent to which the results of the study can be generalised to other contexts or developers
  - Conclusion validity
    • Whether the presumed cause and effect covary
Social Studies

- Few studies
  - Notations [Recker, Reijers]
    - Understandability
    - Personal factors
    - Model factors
  - Structuredness [Dumas et al.]
    - Understanding
  - Others?
Trail

Maintainability
Understandability

→ Quality Models → Models → Measures

→ Good Practices → Definition → Detection → Occurrences

→ Social Studies → Characteristics → Factors

→ Eye-tracking Studies → Behaviour
Eye-tracking Studies

Problem: how to study developers’ behaviour?
Solution: study developers’ thought processes
Eye-tracking Studies

- Developers’ thought processes
  - Cognitive theories
    - Brooks’
    - Von Mayrhauser’s
    - Pennington’s
    - Soloway’s
  - Memory theories
    - Kelly’s categories
    - Minsky’s frames
    - Piaget’s schema
    - Schank’s scripts

- Mental models
  - Gentner and Stevens’ mental models
Eye-tracking Studies

- Studying developers’ thought processes
  - Yarbus’ eye-movements and vision
  - Just and Carpenter’s eye-mind hypothesis
  - Palmer’s vision science
  - …
Eye-tracking Studies

- Picking into developers’ thought processes
Eye-tracking Studies

- Picking into developers’ thought processes

// This class calculates circle area based on runtime radius of a circle input

```java
public class CalculateArea {
    public void calculateCircleArea() {
        int radius = 0;
        System.out.println("Please enter radius of a circle");

        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            radius = Integer.parseInt(br.readLine());
        } catch (NumberFormatException ne) {
            System.out.println("Invalid radius value" + ne);
            System.exit(0);
        } catch (IOException ioe) {
            System.out.println("IO error :");
            System.exit(0);
        }

        double area = Math.PI * radius * radius;
        System.out.println("Area of a circle is: "+ area);
    }
}
```
Eye-tracking Studies

- Such studies are typically experiments
  - Independent variable (few)
  - Dependent variables (many)
  - Statistical analyses

- Threats to validity
Eye-tracking Studies

- For example, impact of design pattern notations
  - Strongly visual [Schauer and Keler]
  - Strongly textual [Dong et al.]
  - Mixed [Vlissides]
  - …
Eye-tracking Studies

- Independent variables
  - Design pattern notations
  - Tasks: Participation, Composition, Role

- Dependent variables
  - Average fixation duration
  - Ratio of fixations
  - Ration of fixation times
Eye-tracking Studies

Subjects
- 24 Ph.D. and M.Sc. students

Conclusions
- Stereotype-enhanced UML diagram [Dong et al.] more efficient for Composition and Role
- UML collaboration notation and the pattern-enhanced class diagrams more efficient for Participation
Eye-tracking Studies

- Threats to validity
  - Construct validity
  - Internal validity
  - External validity
  - Conclusion validity
Eye-tracking Studies

- One study
  - Relevant regions [Petrusel and Mendling]
    - Fixations
    - Fixation time

- Others?
Outline

- Example
- Introduction
- Trail
- Definitions
- Following the Trail
- Conclusion
Conclusion

- How to measure the overall quality characteristics of maintainability and understandability objectively and subjectively?

- Focus on source code artefacts modelled as graphs, business process can be modelled as graphs too
Conclusion

- Software code quality is based on the modelling of code as graphs and on
  - Quality metrics
  - Quality models
  - Good practices
and depends
  - Experiences
  - Human factors
Conclusion

- Business process quality could also be based on the modelling of business process code as graphs and follow similar research directions as source code?