Survey: Previous Experience

2018 vs. 2017

- Management: 2018
  - Median: 1
  - Average: 2.2069
- Management: 2017
  - Median: 1
  - Average: 1.531
- Requirements Engineering: 2018
  - Median: 1
  - Average: 2.284
- Requirements Engineering: 2017
  - Median: 3
  - Average: 3.9844
- Programming: 2018
  - Median: 3
  - Average: 3.9432
- Programming: 2017
  - Median: 1
  - Average: 1.9531
- Design Modelling: 2018
  - Median: 1
  - Average: 2.1932
- Design Modelling: 2017
  - Median: 2
  - Average: 2.625
- QA: 2018
  - Median: 1
  - Average: 2.5682
- QA: 2017
  - Median: 3
  - Average: 3.9432
1. **Survey:** Expectations on the Course
2. **Software Metrics**
3. **Motivation**
4. **Vocabulary**
5. **Requirements on Useful Metrics**
6. **Excursion: Scales**
7. **Some positive/negative examples**
8. **Example: LOC**
9. **Other Properties of Metrics**
10. **Base Measures vs. Derived Measures**
11. **Subjective and Pseudo Metrics**
12. **Example: McCabe**
13. **Discussion**

---

**Engineering vs. Non-Engineering**

*workshop (technical product)*

*studio (artwork)*

**Mental prerequisite**

the existing and available technical know-how, artist's inspiration, among others

**Deadlines**

can usually be planned with sufficient precision due to dependency on artist's inspiration

**Price**

oriented on cost, thus calculable determined by market value, not by cost

**Norms and standards**

exist, are known, and are usually respected are rare and, if known, not respected

**Evaluation and comparison**

can be conducted using objective, quantified criteria is only possible subjectively, results are disputed

**Author**

remains anonymous, often lacks emotions to the product considers the artwork as part of himself/herself

**Warranty and liability**

are clearly regulated, cannot be excluded are not defined and in practice hardly enforceable (Ludewig and Lichter, 2013)

---

**Example: Material Goods**

*Recall: “software that is reliable and works efficiently” (Bauer, 1971)*


**6.1 Functionality**

The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.

**6.1.1 Suitability**

The capability of the software product to provide an appropriate set of functions for specified tasks and user objectives.

**6.2 Reliability**

The capability of the software product to maintain a specified level of performance when used under specified conditions.

**6.2.2 Fault tolerance**

The capability of the software product to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.
Vocabulary

– 2 – 2018-04-19 – Svocabulary –

18/47

metric — A quantitative measure of the degree to which a system, component, or process possesses a given attribute.

IEEE 610.12 (1990)

quality metric —

(1) A quantitative measure of the degree to which an item possesses a given quality attribute.

(2) A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute.

IEEE 610.12 (1990)

Software Metrics: Motivation and Goals

– 2 – 2018-04-19 – Sgoals –

Important motivations and goals for using software metrics:

• specify quality requirements
• assess the quality of products and processes
• quantify experience, progress, etc.
• predict cost/effort, etc.
• support decisions

Software metrics can be used:

• prescriptive, e.g., “all procedures must not have more than \( N \) parameters”, or
• descriptive, e.g., “procedure \( P \) has \( N \) parameters”.

A descriptive metric can be

• diagnostic, e.g., “the test effort was \( N \) hours”, or
• prognostic, e.g., “the expected test effort is \( N \) hours”.

Note: prescriptive and prognostic are different things.

• Examples: support decisions by diagnostic measurements:

(i) Measure CPU time spent per procedure, then “optimize” most time consuming procedure.
(ii) Measure attributes which indicate architecture problems, then refactor accordingly.

Example: Material Goods

– 2 – 2018-04-19 – Sgoals –

For material goods, it is often pretty obvious what we want to evaluate for:

• and how to measure.

• And immaterial goods, like software?

Recall: “software that is reliable and works efficiently” (Bauer, 1971)


software related quality

product quality

functionality

suitability

accuracy

interoperability

security

reliability

fault tolerance

recoverability

usability

understandability

learnability

operability

attractiveness

efficiency

time behavior

resource utilization

maintainability

analysability

changeability

stability

testability

portability

adaptability

installability

coexistence

replaceability

6.1 Functionality

The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.

6.1.1 Suitability

The capability of the software product to provide an appropriate set of functions for specified tasks and user objectives.

Content

– 2 – 2018-04-19 – Scontent –

• Survey: Expectations on the Course
• Software Metrics
• Motivation
• Vocabulary
• Requirements on Useful Metrics
• Excursion: Scales
• Some positive/negative examples
• Example: LOC
• Other Properties of Metrics
• Base Measures vs. Derived Measures
• Subjective and Pseudo Metrics
• Example: McCabe
• Discussion

Recall: Vocabulary

– 2 – 2018-04-19 – Sreqonmetrics –

18/47

metric — A quantitative measure of the degree to which a system, component, or process possesses a given attribute.

IEEE 610.12 (1990)

quality metric —

(1) A quantitative measure of the degree to which an item possesses a given quality attribute.

(2) A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute.

IEEE 610.12 (1990)
There is a (natural) notion of difference between elements of $S \times \Delta$, but is not related to the measured information (thus not natural). Java

Software engineering example

• Minister (administrative ranks);
• Agreement: strongly agree
• disagree
• Nationality, gender, car manufacturer, geographic direction, train number, . . .

$100^\circ F$ is twice as warm as $50^\circ F$: . . . ? No. Note: the zero is arbitrarily chosen.

<table>
<thead>
<tr>
<th>Scale Type</th>
<th>Nominal Scale</th>
<th>Ordinal Scale</th>
<th>Interval Scale</th>
<th>Ratio Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>A scale which assigns to each $p \in P$ a label from $m \in M$.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examples</td>
<td>nationality, gender, car manufacturer, geographic direction, train number, . . .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operations</td>
<td>State the order of the labels: $\rho \in {&lt;,\leq,=,\neq,&gt;,\geq}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remarks</td>
<td>Cannot be converted to a ratio scale.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Discussion

• Example: McCabe

subvertible
developers cannot arbitrarily manipulate the yield; antonym: robust

Subjective and Pseudo Metrics

• pseudo-metric) → (plausible

Base Measures vs. Derived Measures

• metrics are not economical (if not available for free)

irrelevant

Other Properties of Metrics

• — at a high price;

worst case: doing the project gives a perfect prognosis of project duration

economical

Example: LOC

• wrt. overall needs

relevant

Some positive/negative examples

• valuation yields need to be in place when needed

available

Excursion: Scales

• Back From Excursion: Scales

Requirements on Useful Metrics

• multiple applications of a metric to the same proband should yield the reproducible

Vocabulary

• ordinal scale, better: rational (or absolute) scale

Motivation

• worst case: same valuation yield for all probands

differentiated

Software Metrics

• Expectations on the Course:

Survey

• In order to be useful, a (software) metric should be:

Content

Requirements on Useful Metrics

• Requirements on Useful Metrics

• Requirements on Useful Metrics

• Requirements on Useful Metrics

• Requirements on Useful Metrics

• Requirements on Useful Metrics

• Requirements on Useful Metrics
Some Positive and Negative Examples

- Characteristic ('Merkmal')
  - Positive example
  - Negative example

- Differentiated program length in LOC (→ in a minute)
- CMM/CMMI level below 2
- Comparable cyclomatic complexity (→ in two minutes)
- Review (text)
- Reproducible memory consumption
- Grade assigned by inspector
- Available number of developers
- Relevant number of errors in the code (not only known ones)
- Economical expected development cost; number of errors
- Plausible number of subclasses (NOC)
- Robust high detailed timekeeping

- Base measure — measure defined in terms of an attribute and the method for quantifying it.
  - ISO/IEC 15939 (2011)

- Examples:
  - Lines of code, hours spent on testing, ...
  - Average/median lines of code, productivity (lines per hour), ...

- Derived measure — measure that is defined as a function of two or more values of base measures.
  - ISO/IEC 15939 (2011)

- Examples:
  - Average lines of code, productivity (lines per hour), ...
  - Productivity; cost estimation by COCOMO (to a certain amount)

- Other Properties of Metrics

- Subjective and Pseudo Metrics

- Example: McCabe

- Discussion
Some of the most interesting aspects of software development projects are (today) hard or impossible to measure directly, e.g.:

- how maintainable is the software?
- how much effort is needed until completion?
- how is the productivity of my software people?
- do all modules do appropriate error handling?
- is the documentation sufficient and well usable?

Due to high relevance, people want to measure despite the difficulty in measuring. Two main approaches:

- differentiated
- comparable
- reproducible
- available
- relevant
- economical
- plausible
- robust

Expert review, grading (✔) (✔) (✘) (✔) (✔) (✘) (✔) (✘) (✔) (✔) (✘) (✔) (✘) (✔) (✔)

Pseudo-metrics, derived measures (✔) (✔) (✔) (✔) (✔) (✔) (✔) (✘) (✘)

Note: not every derived measure is a pseudo-metric:

- average LOC per module: derived, not pseudo → we really measure average LOC per module.
- measure maintainability in average LOC per module: derived, pseudo → we don't really measure maintainability; average-LOC is only interpreted as maintainability.

Not robust if easily subvertible (see exercises).

**Pseudo-Metrics Example**

Example: productivity (derived).

- Team T develops software S with LOC \( N = 817 \) in \( t = 310 \) h.
- Define productivity as \( p = \frac{N}{t} \), here: ca. \( 2.64 \) LOC/h.
- Pseudo-metric: measure performance, efficiency, quality, ... of teams by productivity (as defined above).
- Team may write \( x := y + z; \) instead of \( x := y + z; \) → 5-time productivity increase, but real efficiency actually decreased. → not (at all) plausible. → clearly pseudo.

**Can Pseudo-Metrics be Useful?**

- Pseudo-metrics can be useful if there is a (good) correlation (with few false positives and few false negatives) between valuation yields and the property to be measured:

<table>
<thead>
<tr>
<th>Valuation Yield</th>
<th>Low</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>False Positive</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>True Positive</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>True Negative</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>False Negative</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

- Usefulness may strongly depend on context information:
  - If LOC was (or could be made) non-subvertible (→ tutorials), then LOC/day could be a useful measure for, e.g., project progress.
Software Engineering


Software metrics are easier to understand than to propose new ones, (refutation) of the proposed metrics than to propose new ones, (agreed-upon limit).

Whether a software attribute should be directly measurable in a certain way depends.

There seems to be agreement that it is far more important to focus on empirical validation (or provide written explanation of why limit exceeded).