Lecture 3: More Metrics & Cost Estimation

Software Metrics
- Properties of Metrics
- Scales
- Examples

Cost Estimation
- "(Software) Economics in a Nutshell"
- Expert’s Estimation
- Algorithmic Estimation

Project Management
- Project Process and Process Modelling
  - Procedure Models
  - Process Models
- CMMI, Spice

Kinds of Metrics: by Measurement Procedure

<table>
<thead>
<tr>
<th>Objective Metric</th>
<th>Pseudo Metric</th>
<th>Subjective Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedure</td>
<td>Measurement, counting, possibly standardised</td>
<td>Computation (based on measurements or assessment)</td>
</tr>
<tr>
<td>Advantages</td>
<td>Exact, reproducible, can be obtained automatically</td>
<td>Yields relevant, directly usable statement on not directly visible characteristics</td>
</tr>
<tr>
<td>Disadvantages</td>
<td>Not always relevant, often subvertable, no interpretation</td>
<td>Hard to comprehend, pseudo-objective</td>
</tr>
<tr>
<td>Example, general</td>
<td>Body height, air pressure</td>
<td>Body mass index (BMI), weather forecast for the next day</td>
</tr>
<tr>
<td>Example in Software Engineering</td>
<td>Size in LOC or NCSI; number of (known) bugs</td>
<td>Productivity; cost estimation by COCOMO</td>
</tr>
<tr>
<td>Usually used for</td>
<td>Collection of simple base measures</td>
<td>Predictions (cost estimation); overall assessments</td>
</tr>
</tbody>
</table>

Recall: 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:
  - Valuation yield
    - Low
    - High
  - Quality
    - High
  - False positive
  - True positive
  - True negative
  - False negative

- This 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.
Subjective Metrics

Statement
“"The specification is available."
Terms may be ambiguous, conclusions are hardly possible. Allow only certain statements, characterize them precisely.

Assessment
“"The module is implemented in a clever way."
Not necessarily comparable. Only offer particular outcomes; put them on an (at least ordinal) scale.

Grading
“"Readability is graded 4.0."
Subjective grading not reproducible. Define criteria for grades; give examples how to grade; practice on existing artefacts (Ludewig and Lichter, 2013). The Goal-Question-Metric Approach

Information Overload!? Now we have mentioned nearly 60 attributes one could measure. Which ones should we measure? It depends. Relevant plausible available differentiated economical comparable reproducible robust One approach: Goal-Question-Metric (GQM). Goal-Question-Metric (GQM) (Basili and Weiss, 1984)
The three steps of GQM:
(i) Define the goals relevant for a project or an organisation.
(ii) From each goal, derive questions which need to be answered to check whether the goal is reached.
(iii) For each question, choose (or develop) metrics which contribute to finding answers.
Being good wrt. to a certain metric is (in general) not an asset on its own. We usually want to optimise wrt. goals, not wrt. metrics. In particular critical: pseudo-metrics for quality. Software and process measurements may yield personal data (“personenbezogene Daten”). Their collection may be regulated by laws.

Example: A Metric for Maintainability
• Goal: assess maintainability.
• One approach: grade the following aspects, e.g., with scale $S = \{0, \ldots, 10\}$.
  - Norm Conformance $n$: size of units (modules etc.)
  - Locality $l$: use of parameters
  - Readability $r$: data types
  - Testability $t$: test driver
  - Typing $y$: type differentiation
• Define:
  - $m = n_1 + \cdots + y_2$
  - $m_g = g_1 \cdot n_1 + \cdots + g_20 \cdot y_2$
  - $G = \sum_{i=1}^{20} g_i$.
• Procedure:
  - Train reviewers on existing examples.
  - Do not over-interpret results of first applications.
  - Evaluate and adjust before putting to use, adjust regularly. (Ludewig and Lichter, 2013)
And Which Metrics Should One Use?

• Investigate further (maybe using additional metrics).

• Unusual values over time. In case of CMMI, Spice etc.

• Process Metrics
  • Average effort for error (errors per LOC), error density per release, error rate
  • and buy time to take appropriate counter-measures. E.g., track early enough to indicate problems ahead.

• Process Models
  • Often useful: collect some basic measures (automatically provided by revision control software), e.g.:

• Procedure Models
  • In advance... at least errors recorded by customer reported during quality assurance, and errors found during verification, testing, review, and coding.

• Algorithmic Estimation
  • Examples
  • Goal-Question-Metric Approach
  • Subjective Metrics
  • Goal-Question-Metric Approach
  • Software Metrics
  • Software Metrics
  • Cost Estimation
  • Cost Estimation
  • Software Metrics
  • Cost Estimation
  • Subjective Metrics
  • Goal-Question-Metric Approach
  • Function Points
  •...
Software Metrics

• Software Economics in a Nutshell
• Goal-Question-Metric Approach

Subjective Metrics

• Cost Estimation

• "(Software) Economics in a Nutshell"
• Expert’s Estimation

• The Delphi Method
• Algorithmic Estimation

Costs

– Next to ‘Software’, ‘Costs’ is one of the terms occurring most often in this book. Ludewig and Lichter (2013)

A first approximation:

cost (‘Kosten ’) all disadvantages of a solution

benefit (‘Nutzen ’) all benefits of a solution.

Note: costs / benefits can be subjective — and not necessarily quantifiable in terms of money...

Super-ordinate goal of many projects:

• Minimize overall costs, i.e. maximize difference between benefits and costs.

(Equivalent: minimize sum of positive and negative costs.)

Costs vs. Benefits: A Closer Look

The benefit of a software is determined by the advantages achievable using the software; it is influenced by:

• the degree of coincidence between product and requirements,
• additional services, comfort, flexibility etc.

Some other examples of cost/benefit pairs: (inspired by Jones (1990))

<table>
<thead>
<tr>
<th>Costs</th>
<th>Possible Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labor during development (e.g., develop new test machinery)</td>
<td>Use of result (e.g., faster testing)</td>
</tr>
<tr>
<td>New equipment (purchase, maintenance, depreciation)</td>
<td>Better equipment (maintenance; maybe revenue from selling old)</td>
</tr>
<tr>
<td>New software purchases</td>
<td>(Other) use of new software</td>
</tr>
<tr>
<td>Conversion from old system to new</td>
<td>Improvement of system, maybe easier maintenance</td>
</tr>
<tr>
<td>Increased data gathering</td>
<td>Increased control</td>
</tr>
<tr>
<td>Training for employees</td>
<td>Increased productivity</td>
</tr>
</tbody>
</table>

Costs: Economics in a Nutshell

Distinguish current cost (‘laufende Kosten’), e.g.

• wages,
• (business) management, marketing, 
• rooms,
• computers, networks, software as part of infrastructure,

and project-related cost (‘projektbezogene Kosten’), e.g.

• additional temporary personnel,
• contract costs,
• expenses,
• hardware and software as part of product or system,

Software Costs in a Narrower Sense

software costs

net production

quality costs

error prevention costs

analyse-and-fix costs

error costs

error localisation costs

error removal costs

error caused costs

(in operation)

decreased benefit

maintenance (without quality)

quality assurance during and after development

Ludewig and Lichter (2013)

Software Engineering — the establishment and use of sound engineering principles to obtain economically software that is reliable and works efficiently on real machines.

F. L. Bauer (1971)

commons.wikimedia.org (CC-by-sa 3.0)
Discovering Fundamental Errors Late Can Be Expensive

Relative error costs over latency according to investigations at IBM, etc. By (Boehm, 1979); Visualisation: Ludewig and Lichter (2013).

The "Estimation Funnel"
than to estimate cost directly. The need for (expert's) estimation does not go away: one needs to estimate a function \( f \) of parameters that influence overall cost, like size in LOC.

\begin{align*}
\text{Estimate } f, \ldots, F_n, \ldots, F_1, \ldots, F_0 \text{ for the new project.}
\end{align*}

One approach: more general:

\begin{itemize}
  \item \textbf{Algorithmic Estimation: COCOMO II (Ludewig and Lichter 2013)}
  \item \textbf{Formulae which fit a huge set of archived project data (from the late 70's).}
\end{itemize}

COCOMO II takes as input the following factors:

\begin{itemize}
  \item \textbf{Cost factors (measurable)}
  \item \textbf{Adjust for the new project.}
  \item \textbf{Technical aspects}
  \item \textbf{Expertise/Leadership}.
\end{itemize}

\begin{align*}
\text{Factors like security requirements or experience of the project team.}
\end{align*}

\begin{align*}
\text{Estimate } f, \ldots, F_n, \ldots, F_1, \ldots, F_0 \text{ for the new project.}
\end{align*}
In the end, it's experience (and experience estimates cost, by estimating more technical aspects).


Recall: It's about the goal, not the metrics. Define goals, derive questions, choose metrics.

The IBM and VW curve for the conversion from AFPs to PM according to Boehm and Horowitz (1988), and Brainard (1986).

Try to identify factors, take notes on your projects. Estimate, document, estimate better.