Principles of Software Cost Estimation

In the end, it's experience, experience, experience.

"Estimate, document, estimate better" Ludewig and Lichter, 2013

Example:

- Assume these were the overall costs of previous, all similar projects:

<table>
<thead>
<tr>
<th>Project</th>
<th>Cost 1</th>
<th>Cost 2</th>
<th>Cost 3</th>
<th>Cost 4</th>
<th>Cost 5</th>
<th>Cost 6</th>
<th>Cost 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What could be an estimate of the new (also similar) Project 7?

- For a better estimate: analyse what costs are composed of. Maybe, Project 6 could re-use parts of Project 5; maybe Project 5 is the only one with a new customer. For Project 7 check: can we re-use parts? Is it a new customer?
A Classification of Software Costs

Distinguish current cost ('laufende Kosten'), e.g.
- fixed personnel,
- (business) management,
- marketing,
- rooms,
- computers,
- networks,
- software as infrastructure,
- ... and project-related cost ('projektbezogene Kosten'), e.g.
- additional temporary personnel,
- hardware and software as part of product or system,
- contract costs,
- ... The "Estimation Funnel"

4 × 2 × 1 × 0.5 × 0.25

effort estimated to real effort (log. scale)

Pre-Project
Analysis
Design
Coding & Test

Uncertainty with estimations (following (Boehm et al., 2000), p. 10).

Visualisation:
Ludewig and Lichter (2013)

Approaches to Software Cost Estimation

Expert's Estimation

- Delphi Method

Algorithmic Estimation
- COCOMO
- Function Points

One approach: the Delphi method.

- Step 1: write down your estimates!
- Step 2: show your estimates and explain!
- Step 3: estimate again!

Then take the median, for example.

Algorithmic Estimation: COCOMO

The "Estimation Funnel"
### Process maturity

- **new in COCOMO II**

### Team cohesion

- **project factors**
  - **experience with development environment**
    - **PLEX**
    - **0.00**
  - **experience with application domain**
    - **PCON**
    - **3.72**
  - **analyst capability**
    - **ACAP**
    - **4.24**

### Team factors

- **normal**
- **low**
- **very**

### Stability of development environment

- **amount of required documentation**: 
  - **function point (FP)**
    - **FP**
    - **0.00**

### Product factors

- **early design model**
  - **REVL**
  - **1**

### Functional Estimation

- **COCOMO II: Post-Architecture**
- **2000**, **Boehm et al.**

### COCOMO II Post-Architecture

<table>
<thead>
<tr>
<th>Function Point</th>
<th>REVL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program size</td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td></td>
</tr>
<tr>
<td>Deadlines/TIME</td>
<td></td>
</tr>
<tr>
<td>Complex HW/ENV</td>
<td></td>
</tr>
<tr>
<td>Stability</td>
<td></td>
</tr>
<tr>
<td>Level of effort</td>
<td></td>
</tr>
<tr>
<td>Complexity</td>
<td></td>
</tr>
<tr>
<td>Resource</td>
<td></td>
</tr>
<tr>
<td>Project factors</td>
<td></td>
</tr>
<tr>
<td>Team factors</td>
<td></td>
</tr>
<tr>
<td>Technology</td>
<td></td>
</tr>
<tr>
<td>Environment</td>
<td></td>
</tr>
</tbody>
</table>

### Cost Estimation

- **COCOMO 81**

### COCOMO II Examples

- **Basic COCOMO**
- **Complex COCOMO**
- **Algorithmic Estimation: COCOMO**

### COCOMO II Attributes and Characteristics

- **Tool**
- **Myriad of COCOMO tools**
- **Examples**
- **Application Example**
- **COCOMO II Features**
- **Benefits of COCOMO**
- **Limitations of COCOMO**
- **Comparisons with Other Estimation Models**
- **Case Studies**
- **Future Work**
- **Conclusion**

### COCOMO II Adjustments

- **Additional Cost Adjustments**
- **Cost Estimation Models**
- **COCOMO II vs. COCOMO I**
- **COCOMO II vs. Other Estimation Models**
- **Accuracy of COCOMO II**
- **COCOMO II vs. Traditional Estimation Methods**
- **COCOMO II vs. Modern Estimation Models**

### COCOMO II Conclusions

- **Summary of Key Findings**
- **Future Research Directions**
- **Implications for Practice**
- **Recommendations for Use**
- **Limitations of COCOMO II**
- **Future Work**
- **Conclusion**
Complexity

<table>
<thead>
<tr>
<th>Type</th>
<th>low</th>
<th>medium</th>
<th>high</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>· 3</td>
<td>· 4</td>
<td>· 6</td>
</tr>
<tr>
<td>output</td>
<td>· 4</td>
<td>· 5</td>
<td>· 7</td>
</tr>
<tr>
<td>query</td>
<td>· 3</td>
<td>· 4</td>
<td>· 6</td>
</tr>
<tr>
<td>user data</td>
<td>· 7</td>
<td>· 10</td>
<td>· 15</td>
</tr>
<tr>
<td>reference data</td>
<td>· 5</td>
<td>· 7</td>
<td>· 10</td>
</tr>
</tbody>
</table>

Unadjusted function points (UFP)

Value adjustment factor (VAF)

Adjusted function points (AFP)

\[ AFP = UFP \cdot VAF \]

\[ VAF = 0.65 + \frac{1}{100} \sum_{i=1}^{GSC_i} \]

\[ 0 \leq GSC_i \leq 5 \]

Discussion

Cost Estimation is Everywhere

• For example:
  - Bachelor's Thesis: Which results can I promise to deliver in 3 months time?
  - Suggestion: start to quantify your experience now.
  - Take notes on your projects (e.g., Softwarepraktikum, Bachelor Projekt, Bachelor's Thesis, Master Projekt, Master's Thesis, . . .)
    - timestamps,
    - size of program created,
    - number of errors found,
    - number of pages written,
    - etc.
  - Try to identify factors: what hindered productivity, what boosted productivity, . . .
  - Which detours and mistakes were avoidable in hindsight? How?

Content

- Cost Estimation
- Software Cost Estimation
- Expert's Estimation (Delphi Method)
- Algorithmic Estimation (COCOMO, Function Points)

(Software) Project

- Project Management
  - Goals, Common Activities
  - Excursion: Risk
  - Software Development Processes
    - Roles, Artefacts, Activities
    - Costs and Deadlines
      - phase, milestone, deadline
      - cycle, life cycle, software life cycle

Development Process Modelling

- process vs. process model

Procedure and Process Models

- "Code and Fix"
- The (infamous) Waterfall Model
A temporary activity that is characterized by
• a start date,
• specific objectives and constraints,
• established responsibilities,
• a budget and schedule, and
• a completion date.
If the objective of the project is to develop a software system, then it is sometimes called a software development project or software engineering project.
R. H. Thayer (1997)
We could refine our earlier definition as follows: a project is successful if and only if
• started at start date,
• achieved objectives,
• respected constraints,
• adheres to budget and schedule,
• stops at completion date.
Whether, e.g., objectives have been achieved can still be subjective (→ customer/user happy).

Goals of Project Management
Main and general goal:
Developer Customer
software delivery
Have a successful project, i.e. the project delivers
• defined results
• in demanded quality
• within scheduled time
• using the assigned resources.
There may be secondary goals, e.g.,
• build or strengthen good reputation on market,
• acquire knowledge which is useful for later projects,
• develop re-usable components (to save resources later),
• be attractive to employees.

Common Activities of Project Management
• Planning
• Assessment and Control
• Recognising and Fighting Difficulties as Early as Possible
• Communication
• Leading and Motivation of Employees
• Creation and Preservation of Beneficial Conditions

Without plans, a project cannot be managed. Note: mistakes in planning can be hard to resolve.

Work results and project progress have to be assessed and compared to the plans; it has to be observed whether participants stick to agreements.

Unforeseen difficulties and problems in projects are not exceptional but usual. Therefore, project management needs to constantly “screen the horizon for icebergs”, and, when spotting one, react timely and effectively. In other words: systematic risk management.

Distribute information between project participants (project owner, customer, developers, administration).

Leading means: going ahead, showing the way, “pulling” the group. Most developers want to achieve good results, yet need orientation and feedback (negative and positive).

Provide necessary infrastructure and working conditions for developers (against: demanding customers, imprecise stated goals, organisational restructuring, economic measures, tight office space, other projects, etc.).
A process has: each software development project

- explicitly prescribed (by a procedure)
- informally agreed on, or
- implicitly.

Delimits phases, may create decision artefacts, may be milestones — any processing of artefacts, manually or automatic; solves tasks.

The process by which user needs are translated into a software product. The process involves activities — all documents, evaluation protocols, software artefacts (or modules, etc.; all products emerging during a development process. Is processed by artefacts, creates/modifies artefacts, artefacts participates in — any processing of artefacts, manually or automatic; solves tasks.

Over time, the following notions proved useful to describe:

- roles of and relations between people
- the software requirements into translating
- the design in implementing
- the code, and testing,
- costs and deadlines, i.e. pursues a bunch of goals.

This recipient is the project owner. Other important goals are extension of know-how, preparation of building blocks for later projects, or utilisation of employees. The most important goal is usually to create or modify software; this software is thus the result of the project, the successful product. The project is called if the goals are reached to a high degree. This recipient is the originator or its representative. The project owner reports to the project owner. The originator is limited. The project is successful if the goals are reached to a high degree.

- Customer
- Developer
- Other important goals are extension of know-how, preparation of building blocks for later projects, or utilisation of employees.

This recipient is the project owner. Other important goals are extension of know-how, preparation of building blocks for later projects, or utilisation of employees. The most important goal is usually to create or modify software; this software is thus the result of the project, the successful product. The project is called if the goals are reached to a high degree. This recipient is the originator or its representative. The project owner reports to the project owner. The originator is limited. The project is successful if the goals are reached to a high degree.
Useful and common roles in software projects:

- Tester (tst), responsible for testing the software and raising issues.
- Developer (prg), responsible for writing and changing the code, and reporting unforeseen problems.
- Project manager (mgr), responsible for addressing issue reports.
- Invisible clients, which often represent many legal persons.

Recall: Forum Work of the Course

Example: Describe Processes

Describing Software Development Processes

An aspect of project management is to assign (a set of) people to each role:

- For each role, there is at most one person, i.e., \(|\text{people} \rightarrow \text{role}| = 1\).

Given a set of (active) roles, e.g., \(R = \{\text{ana}, \text{tst}, \text{prg}, \text{mgr}\}\), each with rights and responsibilities, e.g.,

\[
\begin{align*}
\text{ana} & \rightarrow \{\text{create modules}, \text{create fixes}\} \\
\text{tst} & \rightarrow \{\text{find faults}, \text{write test cases}\} \\
\text{prg} & \rightarrow \{\text{write code}, \text{change code}\} \\
\text{mgr} & \rightarrow \{\text{assign roles}, \text{handle issues}\}
\end{align*}
\]

Note: For each role, there is at most one person, i.e., \(|\text{people} \rightarrow \text{role}| = 1\).

The Concept of Roles

The Concept of Roles Cont'd

If there are multiple people assigned to a role, then there are no obligations and no responsibilities associated with these people. For example, if a person is assigned to the role of tester, there is no obligation to update the test cases or to detect faults, which is usually the responsibility of the tester.

The Concept of Roles

An aspect of project management is to assign (a set of) people to each role:

- For each role, there is at most one person, i.e., \(|\text{people} \rightarrow \text{role}| = 1\).

Given a set of (active) roles, e.g., \(R = \{\text{ana}, \text{tst}, \text{prg}, \text{mgr}\}\), each with rights and responsibilities, e.g.,

\[
\begin{align*}
\text{ana} & \rightarrow \{\text{create modules}, \text{create fixes}\} \\
\text{tst} & \rightarrow \{\text{find faults}, \text{write test cases}\} \\
\text{prg} & \rightarrow \{\text{write code}, \text{change code}\} \\
\text{mgr} & \rightarrow \{\text{assign roles}, \text{handle issues}\}
\end{align*}
\]

Note: For each role, there is at most one person, i.e., \(|\text{people} \rightarrow \text{role}| = 1\).

The Concept of Roles

An aspect of project management is to assign (a set of) people to each role:

- For each role, there is at most one person, i.e., \(|\text{people} \rightarrow \text{role}| = 1\).

Given a set of (active) roles, e.g., \(R = \{\text{ana}, \text{tst}, \text{prg}, \text{mgr}\}\), each with rights and responsibilities, e.g.,

\[
\begin{align*}
\text{ana} & \rightarrow \{\text{create modules}, \text{create fixes}\} \\
\text{tst} & \rightarrow \{\text{find faults}, \text{write test cases}\} \\
\text{prg} & \rightarrow \{\text{write code}, \text{change code}\} \\
\text{mgr} & \rightarrow \{\text{assign roles}, \text{handle issues}\}
\end{align*}
\]

Note: For each role, there is at most one person, i.e., \(|\text{people} \rightarrow \text{role}| = 1\).
From Concrete Process to Process Model
References


