Softwaretechnik / Software-Engineering

Lecture 3: Software Project Management

2019-05-02

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Topic Area Project Management: Content

<table>
<thead>
<tr>
<th>VL 2</th>
<th>Software Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Metrics, Properties of Metrics</td>
</tr>
<tr>
<td></td>
<td>Software Metrics</td>
</tr>
<tr>
<td></td>
<td>Software Metrics Issues</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VL 3</th>
<th>Cost Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Software) Economics in a Nutshell</td>
</tr>
<tr>
<td></td>
<td>Software Cost Estimation</td>
</tr>
<tr>
<td></td>
<td>Expert’s / Algorithmic Estimation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VL 4</th>
<th>Project Management</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Project</td>
</tr>
<tr>
<td></td>
<td>Process and Process Modelling</td>
</tr>
<tr>
<td></td>
<td>Procedure Models</td>
</tr>
<tr>
<td></td>
<td>Process Models</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Process Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CMMI, Spice</td>
</tr>
</tbody>
</table>

2/2
Software Cost Estimation Cont’d
“Estimate, document, estimate better.” (Ludewig and Lichter, 2013)

Example:

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

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

• For a better estimate: analyse what costs are composed of.

Maybe, Project 4 could re-use parts of Project 3, maybe Project 2 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"

Uncertainty with estimations (following (Boehm et al., 2000), p. 10).
Visualisation: Ludewig and Lichter (2013)
Expert’s Estimation

For example,
- Delphi Method

Algorithmic Estimation

For example,
- COCOMO
- Function Points
**Expert’s Estimation**

**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
• Constructive Cost Model:
  Formulae which fit a huge set of archived project data (from the late 70’s).
  Flavours:
  • COCOMO 81 (Boehm, 1981): variants basic, intermediate, detailed
  • COCOMO II (Boehm et al., 2000)

  All flavours are based on estimated program size \( S \) measured in DSI (Delivered Source Instructions) or kDSI (1000 DSI).
  Factors like security requirements or experience of the project team are mapped to values for parameters of the formulae.

• COCOMO examples:
  • textbooks like Ludewig and Lichter (2013) (most probably made up)
  • an exceptionally large example: COCOMO 81 for the Linux kernel (Wheeler, 2006) (and follow-ups)

---

**COCOMO 81**

<table>
<thead>
<tr>
<th>Size</th>
<th>Characteristics of the Type</th>
<th>a</th>
<th>b</th>
<th>Software Project Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small (&lt;50 KLOC)</td>
<td>Little, Not tight, Stable</td>
<td>3.2</td>
<td>1.05</td>
<td>Organic</td>
</tr>
<tr>
<td>Medium (50-300 KLOC)</td>
<td>Medium, Medium, Medium</td>
<td>3.0</td>
<td>1.12</td>
<td>Semi-detached</td>
</tr>
<tr>
<td>Large</td>
<td>Greater, Tight, Complex HW/Interfaces</td>
<td>2.8</td>
<td>1.20</td>
<td>Embedded</td>
</tr>
</tbody>
</table>

**Basic COCOMO:**
- effort required: \( E = a \cdot (S/kDSI)^b \) [PM (person-months)]
- time to develop: \( T = c \cdot E^d \) [months]
- headcount: \( H = E/T \) [FTE (full time employee)]
- productivity: \( P = S/E \) [DSI per PM] (← use to check for plausibility)

**Intermediate COCOMO:**
\[
E = M \cdot a \cdot (S/kDSI)^b \quad \text{[person-months]}
\]
\[
M = \text{RELY} \cdot \text{CPLX} \cdot \text{TIME} \cdot \text{ACAP} \cdot \text{PCAP} \cdot \text{LEXP} \cdot \text{TOOL} \cdot \text{SCED}
\]
COCOMO 81: Some Cost Drivers

\[ M = \text{RELY} \cdot \text{CPLX} \cdot \text{TIME} \cdot \text{ACAP} \cdot \text{PCAP} \cdot \text{LEXP} \cdot \text{TOOL} \cdot \text{SCED} \]

<table>
<thead>
<tr>
<th>factor</th>
<th>very low</th>
<th>low</th>
<th>normal</th>
<th>high</th>
<th>very high</th>
<th>extra high</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELY required software reliability</td>
<td>0.75</td>
<td>0.88</td>
<td>1</td>
<td>1.15</td>
<td>1.40</td>
<td></td>
</tr>
<tr>
<td>CPLX product complexity</td>
<td>0.70</td>
<td>0.85</td>
<td>1</td>
<td>1.15</td>
<td>1.30</td>
<td>1.65</td>
</tr>
<tr>
<td>TIME execution time constraint</td>
<td>1</td>
<td>1.11</td>
<td>1.30</td>
<td>1.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACAP analyst capability</td>
<td>1.46</td>
<td>1.19</td>
<td>1</td>
<td>0.86</td>
<td>0.71</td>
<td></td>
</tr>
<tr>
<td>PCAP programmer capability</td>
<td>1.42</td>
<td>1.17</td>
<td>1</td>
<td>0.86</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>LEXP programming language experience</td>
<td>1.14</td>
<td>1.07</td>
<td>1</td>
<td>0.95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOOL use of software tools</td>
<td>1.24</td>
<td>1.10</td>
<td>1</td>
<td>0.91</td>
<td>0.83</td>
<td></td>
</tr>
<tr>
<td>SCED required development schedule</td>
<td>1.23</td>
<td>1.08</td>
<td>1</td>
<td>1.04</td>
<td>1.10</td>
<td></td>
</tr>
</tbody>
</table>

• Note: what, e.g., "extra high" TIME means, may depend on project context. (Consider data from previous projects.)

COCOMO II (Boehm et al., 2000)

Consists of

• **Application Composition Model** – project work is configuring components, rather than programming
• **Early Design Model** – adaption of Function Point approach (in a minute); does not need completed architecture design
• **Post-Architecture Model** – improvement of COCOMO 81; needs completed architecture design, and size of components estimatable
Program size: 
\[ S = (1 + \text{REVL}) \cdot (S_{\text{new}} + S_{\text{equiv}}) \]

- **requirements volatility** \( \text{REVL} \):  e.g., if new requirements make 10% of code unusable, then \( \text{REVL} = 0.1 \)
- \( S_{\text{new}} \): estimated size minus size \( w \) of re-used code.
- \( S_{\text{equiv}} = w/q \), if writing new code takes \( q \)-times the effort of re-use.

**Scaling factors:**
\[ X = \delta + \omega, \quad \omega = 0.91, \quad \delta = \frac{1}{100} \cdot (\text{PREC} + \text{FLEX} + \text{RESL} + \text{TEAM} + \text{PMAT}) \]

<table>
<thead>
<tr>
<th>factor</th>
<th>very low</th>
<th>low</th>
<th>normal</th>
<th>high</th>
<th>very high</th>
<th>extra high</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREC</td>
<td>6.20</td>
<td>4.96</td>
<td>3.72</td>
<td>2.48</td>
<td>1.24</td>
<td>0.00</td>
</tr>
<tr>
<td>FLEX</td>
<td>5.07</td>
<td>4.05</td>
<td>3.04</td>
<td>2.03</td>
<td>1.01</td>
<td>0.00</td>
</tr>
<tr>
<td>RESL</td>
<td>7.07</td>
<td>5.65</td>
<td>4.24</td>
<td>2.83</td>
<td>1.41</td>
<td>0.00</td>
</tr>
<tr>
<td>TEAM</td>
<td>5.48</td>
<td>4.38</td>
<td>3.29</td>
<td>2.19</td>
<td>1.10</td>
<td>0.00</td>
</tr>
<tr>
<td>PMAT</td>
<td>7.80</td>
<td>6.24</td>
<td>4.69</td>
<td>3.12</td>
<td>1.56</td>
<td>0.00</td>
</tr>
</tbody>
</table>

\[ M = \text{RELY} \cdot \text{DATA} \cdot \ldots \cdot \text{SCED} \]

- **Product factors**
  - RELY: required software reliability
  - DATA: size of database
  - CPLX: complexity of system
  - RUSE: degree of development of reusable components
  - DOCU: amount of required documentation
- **Platform factors**
  - TIME: execution time constraint
  - STOR: memory consumption constraint
  - PVOL: stability of development environment
- **Team factors**
  - ACAP: analyst capability
  - PCAP: programmer capability
  - PCON: continuity of involved personnel
  - APEX: experience with application domain
  - PLEX: experience with development environment
  - LTEX: experience with programming language(s) and tools
- **Project factors**
  - TOOL: use of software tools
  - SITE: degree of distributedness
  - SCED: required development schedule

(Also in COCOMO 81, new in COCOMO II)
### Algorithmic Estimation: Function Points

<table>
<thead>
<tr>
<th>Type</th>
<th>Complexity</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>low</td>
<td>medium</td>
</tr>
<tr>
<td>input</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>output</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>query</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>user data</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>reference data</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

Unadjusted function points (UFP)

Value adjustment factor (VAF)

Adjusted function points (AFP) = UFP \cdot VAF

\[
VAF = 0.65 \times \frac{1}{100} \sum_{i=1}^{14} GSC_i, \\
0 \leq GSC_i \leq 5.
\]
### Algorithmic Estimation: Function Points

#### Complexity

<table>
<thead>
<tr>
<th>Type</th>
<th>low</th>
<th>medium</th>
<th>high</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>query</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>user data</td>
<td>0.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reference data</td>
<td>0.5</td>
<td>0.7</td>
<td>1.0</td>
<td></td>
</tr>
</tbody>
</table>

#### Unadjusted function points (UFP)

#### Value adjustment factor (VAF)

\[
VAF = 0.65 + \frac{1}{100} \sum_{i=1}^{14} GSC_i, \\
0 \leq GSC_i \leq 5.
\]

#### Adjusted function points (AFP)

\[
AFP = UFP \cdot VAF
\]

---

Discussion

IBM and VW curve for the conversion from AFPS to PM according to (Noth and Kretzschmar, 1984) and (Knöll and Busse, 1991).
Cost Estimation is Everywhere

• For example: Bachelor’s Thesis

Estimation Task: 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
**Project**

**Vocabulary: Project**

- **project** - A temporary activity that is characterized by having
  - 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:**
  - 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.

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).

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.

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

In other words: systematic risk management.

Quick Excursion: Risk and Riskvalue

Risk — a problem, which did not occur yet, but on occurrence threatens important project goals or results. Whether it will occur, cannot be surely predicted.

Ludewig & Lichter (2013)

\[ \text{riskvalue} = p \cdot K \]

- \( p \): probability of problem occurrence,
- \( K \): cost in case of problem occurrence.

Avionics requires: “Catastrophic Failure Conditions have Average Probability per Flight Hour of \( 10^{-9} \) (or ‘Extremely Improbable’) (AC 25.1309-1).

“problems with \( p = 0.5 \) are not risks, but environment conditions to be dealt with”
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

---

Software Development Process
Vocabulary: Software Project

(Software) Project – Characteristics:

- **Duration** is limited.
- Has an **originator** (person or institution which initiated the project).
  - The **project owner** is the originator or its representative.
  - The **project leader** reports to the project owner.
- Has a **purpose**, i.e. pursues a bunch of goals.
  - The most important goal is usually to create or modify software; this software is thus the result of the project, the **product**.
  - Other important goals are extension of know-how, preparation of building blocks for later projects, or utilisation of employees.
- The project is called **successful** if the goals are reached to a high degree.
- Has a **recipient** (or will have one).
  - This recipient is the **customer**.
  - Later **users** (conceptionally) belong to the customer.
- **Connects** people, results (intermediate/final products), and resources.
  - The **organisation** determines roles of and relations between peoples/results/resources, and the **external interfaces** of the project.

Ludewig & Lichter (2013)

---

Process

**Process** –

1. A sequence of steps performed for a given purpose; for example, the software development process.
2. See also: task; job.
3. To perform operations on data.

IEEE 610.12 (1990)

**Software Development Process** –

The process by which user needs are translated into a software product. The process involves translating user needs into **software requirements**, transforming the software requirements into **design**, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use.

IEEE 610.12 (1990)

- The **process** of a software development **project** may be implicit.
- informally agreed on, or explicitly prescribed (by a **procedure** or **process model**).
- **Note**: each software development project has a process!
Describing Software Development Processes

Over time, the following notions proved useful to describe and model (→ in a minute) software development processes:

- **role** – has responsibilities and rights, needs skills and capabilities. In particular: has responsibility for artefacts, participates in activities.

- **artefact** (or **product**) – all documents, evaluation protocols, software modules, etc.; all products emerging during a development process. Is processed by activities, may have state.

- **activity** – any processing of artefacts, manually or automatic; solves tasks. Depends on artefacts, creates/modifies artefacts.

- **decision point** – special case of activity: a decision is made based on artefacts (in a certain state), creates a decision artefact. Delimits phases, may correspond to milestone.
The Concept of Roles

In a software project, at each point in time, there is a set $R$ of (active) roles, e.g. $R = \{ \text{mgr}, \text{prg}, \text{tst}, \text{ana} \}$.

A role has responsibilities and rights, and necessary skills and capabilities.

For example,

- **mgr**: project manager
  - has the right to raise issue reports
  - is responsible for closing issue reports

- **prg**: programmer
  - has the right to change the code
  - is responsible for reporting unforeseen problems to the project manager
  - is responsible for respecting coding conventions
  - is responsible for addressing issue reports

- **tst**: test engineer
  - has the right to raise issue reports
  - is responsible for quality control

The Concept of Roles Cont’d

Given a set $R$ of roles, e.g. $R = \{ \text{mgr}, \text{prg}, \text{tst}, \text{ana} \}$,

and a set $P$ of people, e.g. $P = \{ \text{1}, \text{2}, \text{3}, \text{4} \}$, each with skills or capabilities.

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

$$\text{assign} : R \rightarrow 2^P$$

such that each person $p \in \text{assign}(r)$ assigned to role $r$

has (at least) the skills and capabilities required by role $r$.

**Note**: $\text{assign}$ may change over time, there may be different assignments for different phases.

**Sanity check**: ensure that $\text{assign}(r) \neq \emptyset$ for each role $r$.

**Example**:

- one person, one role
- multiple persons, one role
- one person, multiple roles

$$\text{assign} = \{ \text{mgr} \mapsto \{ \text{1} \}, \text{prg} \mapsto \{ \text{1}, \text{2}, \text{3}, \text{4} \}, \text{tst} \mapsto \{ \text{4} \}, \text{ana} \mapsto \{ \text{4} \} \}$$
Useful and Common Roles

Recall: roles “Customer” and “Developer” are assumed by legal persons, which often represent many people.

The same legal person may act as “Customer” and “Developer” in the same project.

· · ·

Useful and common roles in software projects:

- customer, user
- project manager
- (systems) analyst
- software architect, designer
- (lead) developer
  programmer, tester, ...
- maintenance engineer
- systems administrator
- invisible clients: legislator, norm/standard supervisory committee
Describing Software Development Processes

Over time, the following notions proved useful to describe and model (→ in a minute) software development processes:

- **role** – has responsibilities and rights, needs skills and capabilities. In particular: has responsibility for **artefacts**, participates in **activities**.

- **artefact** (or **product**) – all documents, evaluation protocols, software modules, etc.: all products emerging during a development process. Is processed by **activities**, may have **state**.

- **activity** – any processing of artefacts, manually or automatic: solves tasks. Depends on **artefacts**, creates/modifies **artefacts**.

- **decision point** – special case of activity: a decision is made based on **artefacts** (in a certain state), creates a **decision artefacts**. Delimits phases, may correspond to **milestone**.

**Example: Forum Work of the Course**

- A particular post is handled locally by Tutor A:
  - Friday, 2019-05-10, 19:37: a new post appears in the group forum: ‘Did you upload the notes?’
  - 20:03: Tutor A decides that the issue can be handled locally (by uploading the forgotten notes);
  - 20:21: Tutor A writes a local forum post ‘Sorry, forgot! Thanks for reminding.’

- A particular post needs to be escalated:
  - Monday, 2019-05-13, 14:01: a new post appears in the group forum: ‘Is that a typo?’
  - Tuesday, 2019-05-14, 9:59: Tutor B decides that the issues needs to be escalated.
  - Tuesday, 2019-05-14, 10:03: Tutor B writes a post to the internal forum
  - Tuesday, 2019-05-14, 12:47: Teaching Assistant contacts Lecturer
  - Tuesday, 2019-05-14, 13:59: Teaching Assistant writes a global posts ‘New version is uploaded, sorry.’
From Concrete Process to Process Model
How to Read a Process Model

- A **process model** (as discussed so far) **defines dependencies.**
  → which artefacts needs to be available **before starting** which activity.

- A **process model does not**
  - define when (date/time) an activity starts.
  - say that Activity A must be completed before (depending) Activity B.

**Example:**

- Tuesday, 2019-05-14, 10:03: Tutor B writes a post to the internal forum:
  “This is what I know so far. I’ll get back to the students and post more information later.”
  → Activity ‘**escalate issue**’ **started** (and **continues**)

- Tuesday, 2019-05-14, 12:47: Teaching Assistant contacts Lecturer
  → Activity ‘**handle issue glob.**’ **started** (and **continues**)

- Tuesday, 2019-05-14, 12:54: Tutor B posts further information
  → Activity ‘**escalate issue continues**’ (Tutor B is available for further questions)

- Tuesday, 2019-05-14, 13:03: Teaching Assistant writes to Tutor B: “Okay, thanks, we got it.”
  → Activity ‘**escalate issue**’ **completed.**
Tell Them What You’ve Told Them...

- **Cost Estimation**
  - It’s about experience (and based on data obtained with metrics), and often a well-kept business secret.
  - Algorithmic Cost Estimations “just” shift the estimation.
  - Cost estimation is everywhere (→ tutorials).

- **Project** has (among others)
  - project owner and leader; goals (Excursion: Risk)
  - process – each project has one

- A **process model** relates
  - roles, artefacts, activities, decision points
  - relations: responsibility, dependency, creation/modification.

- Use process models
  - descriptive (“we did it like that”), or
  - prescriptive (“please do it like that”)

- A process model can allow us to (→ exercises)
  - devise a schedule (‘who does what when’)
  - estimate and control phases and deadlines.

- Distinguish process and procedure model.
References


