Softwaretechnik / Software-Engineering
Lecture 02:
Project Management, Cost Estimation
2015-04-27
Prof. Dr. Andreas Podelski,
Dr. Bernd Westphal
Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:
• Introduction: Engineering, Quality, Software, Software Specification

This Lecture:
• Educational Objectives:
  • what characterises a project, life cycle, . . . ?
  • what is a role, a phase, a milestone, . . . ?
  • what are common activities and roles in software development projects?
  • what are goals and activities of project management? why project management?
  • what is COCOMO, what is function points? what is it good for? why to use it with care?
• Content:
  • the notion of 'project'
  • project management activities
  • what to manage: activities, people, cost and deadlines
  • cost estimation, project planning

(software) 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 or software engineering project.

R. H. Thayer (1997)

(software) project

• The duration of a project is limited.
• Each project 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.
• Each project has a purpose, i.e. pursue 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.
• The product has a recipient (or will have one). This recipient is the customer. Later users belong to the customer.
• The project links people, results (intermediate/final products), and resources. The organisation determines their roles and relations and the external interfaces of the project.

Ludewig & Lichter (2013)

Cycle and Life Cycle

• (1) A period of time during which a set of events is completed. See also: ...

IEEE 610.12 (1990)

software life cycle — The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and, sometimes, retirement phase. Note: These phases may overlap or be performed iteratively.

IEEE 610.12 (1990)

software development cycle — The period of time that begins with the decision to develop a software product and ends when the software is delivered. This cycle typically includes a requirements phase, design phase, implementation phase, test phase, and sometimes, installation and checkout phase. Notes: (1) the phases listed above may overlap or be performed iteratively, depending upon the software development approach used. (2) This term is sometimes used to mean a longer period of time, either the period that ends when the software is no longer being enhanced by the developer, or the entire software life cycle.

IEEE 610.12 (1990)

system life cycle — The period of time that begins when a system is conceived and ends when it is no longer available for use.
Goals of Project Management

• Main and general goal: a successful project, i.e. project delivers
  • defined results
  • in demanded quality
  • within scheduled time
  • using the assigned resources.

Secondary goals:
  • 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.

Activities of Project Management

• Planning – without plans, a project cannot be managed. Mistakes in planning can be hard to resolve.
• Assessment and Control – work results and project progress have to be assessed and compared to the plans; it has to be observed whether participants stick to agreements.
• Recognising and Fighting Difficulties as Early as Possible – 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.
• Communication – distribute information between project participants (project owner, customer, developers, administration).
• Leading and Motivation of Employees – leading means: going ahead, showing the way, “pulling” the group. Most developers want to achieve good results, yet need orientation and feedback.
• Creation and Preservation of Beneficial Conditions – provide necessary infrastructure and working conditions for developers (against: demanding customers, imprecisely stated goals, organisational restructuring, economy measures, tight office space, other projects, ...)

What to (Plan and) Manage?

Managing software projects involves
• tasks and activities,
• people and roles,
• costs and deadlines.

What to (Plan and) Manage (1/3)?

Tasks and Activities

• schedule
• work products
• deliverables
• work results
• project progress
• agreements
• agreements
• agreements

People and Roles

• project owner
• customer
• developers
• administration

Costs and Deadlines

• project costs
• time schedule
• project deadlines
• project milestones
• project rollback
• project learning
• project success
• project failure
• is responsible for addressing issue reports
• is responsible for respecting coding conventions

"Never attribute to malice that which can be adequately explained by stupidity."

- Most software systems are not monolithic but consist of modules or components which interact to realise the overall functionality. Overall structure is called architecture.

- Deployment, Operation, and Maintenance

- Specification of Requirements

- Analysis

- Coding and Module Test

- Integration, Test, Approval

- Project Plan (with milestones) and Schedule

- Work that commonly needs to be done in order to develop or adapt software:
  - Customer
  - Software architect
  - Project manager
  - Developer
  - Tester
  - Systems analyst

- Usually assigned to each team member:
  - Responsibility
  - Rights

- Roles typically come with an assumed norms/standard supervisory committee.

- Multiple persons, one role:
  - There is a (many-to-many) relation between elements of roles, each role a person (\( R \subseteq \text{prg}, \text{ana}, \text{tst} \)). Each person has a role (\( \text{prg}, \text{ana}, \text{tst} \)).

- Example: Customer (many-to-many) Project manager

- One person, multiple roles:
  - There is a set \( \{\text{prg}, \text{ana}, \text{tst}\} \). A programmer \( \text{prg} : \{\text{prg}, \text{ana}, \text{tst}\} \).

- One person, one role:
  - Each role is assigned to one person (\( \text{prg} = \{\text{prg}\}, \text{ana} = \{\text{ana}\} \)).

- Sent many people. The same legal person may act as "Customer" and "Developer" in the same project.
Deadlines Cont’d

- Whether a milestone is reached must be assessable by clear, objective, and unambiguous criteria.
- The definition of a milestone often comprises:
  - a definition of the results which need to be achieved,
  - the required quality properties of these results,
  - the desired time for reaching the milestone, and
  - the instance (person or committee) which decide whether the milestone is reached.
- Milestones can be part of the development contract; not reaching a defined milestone as planned can lead to legal claims.

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, quantifiable in terms of money or not.
- benefit ('Nutzen') (or: negative costs) — all benefits of a solution.
- Note: costs and benefits can be very subjective — and are not necessarily quantifiable...

Super-ordinate goal of many projects:

- Minimize overall costs, i.e. maximise 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 examples of cost/benefit pairs:
  - Costs
    - Labor during development
    - Labor during operation
    - New equipment? (purchase, maintenance, depreciation)
    - New software purchases
  - Benefits
    - Use of existing labor
    - Reduced operation labor
    - Replacement of equipment maintenance? (sale, maintenance)
    - (Other) use of new software
    - Conversion from old system to new
    - Improvement of system
    - Increased data gathering
    - Increased control
    - Employed discontent
    - Employee satisfaction
    - Training for employees
    - Increased productivity
    - Lost opportunities
    - Better market stance, basis for further growth

Costs: Economics in a Nutshell

- Distinguish current cost ('laufende Kosten'), e.g.
  - wages
  - 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
- 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)

Discovering Errors Late Can Be Expensive

Relative error costs over latency according to investigations at IBM, etc. by (Boehm, 1979).

Visualisation: Ludewig and Lichter (2013)

Software Project Management Bottom-Line

“Management, management... Can't we just sit down and write some software?”

- Quantity as Quality (Ludewig and Lichter, 2013) — the large is in general not just a multiple of the small; solutions for small problems don’t scale in general.

Example: reliability. Consider a software system with N modules, each module being correct with probability p. N modules are correct with probability p^N.

<table>
<thead>
<tr>
<th>N</th>
<th>p=0.9</th>
<th>p=0.95</th>
<th>p=0.99</th>
<th>p=0.999</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.0000267</td>
<td>0.006</td>
<td>0.37</td>
<td>0.90</td>
</tr>
</tbody>
</table>

Software Engineering as defensive discipline

Analogy: hygiene in hospital.

“Dear patient, we’re working hard to protect you from an infection.” — “Well, doctor, I thought you were working to get me well again.”

“Software Engineering is boring and frustrating for people who don’t value the defense of failures as a positive achievement.” (Ludewig and Lichter, 2013)

Project Planning and Cost Estimation

The “Estimation Funnel”
References


Wissenschaftsverlag.


SCED = SCED required developmentschedule

TOOL = TOOL use of software tools

LEXP = LEXP programming language experience

PCAP = PCAP programmer capability

ACAP = ACAP analyst capability

TIME = TIME execution time constraint

CPLX = CPLX product complexity

RELY = RELY required software reliability

kDSI = kDSI size of specifications... Wissenschaftsverlag.

cE = (effort required) / (time to develop) = (effort required) / (time to develop)

kDSI S/ 3.0

... where

• factors like security requirements or experience of the project team are mapped to parameters of the formulae.

• All based on estimated program size

• COCOMO examples:

• Basic COCOMO:... Prentice-Hall.

• Intermediate COCOMO:... Prentice-Hall.

• Detailed COCOMO:... Prentice-Hall.

• COCOMO II:... Prentice-Hall.

• Delivered Source Instructions).