Foundations of Software Engineering

Part 2: Quick intro to process, teamwork, risk and scheduling

Christian Kästner
Learning Goals

• Recognize the Importance of process
• Understand the difficulty of measuring progress
• Identify what why software development has project characteristics
• Use milestones for planning and progress measurement
• Ability to divide work and planning and replan it
• Model dependencies and schedule work with network plans and Gantt diagrams
• Identifying and managing risks
About You

Saw project fail, know why

Open Source

- yes
- no

Frustrating team experience

- multiple times
- no
- once
Process
How to develop software?

1. Discuss the software that needs to be written
2. Write some code
3. Test the code to identify the defects
4. Debug to find causes of defects
5. Fix the defects
6. If not done, return to step 1
What is a Software Process?
Software Process

“The set of activities and associated results that produce a software product”

Sommerville, SE, ed. 8
Example of Process Decisions

• Writing down all requirements
• Require approval for all changes to requirements
• Use version control for all changes
• Track all reported bugs
• Review requirements and code
• Break down development into smaller tasks and schedule and monitor them
• Planning and conducting quality assurance
• Have daily status meetings
• Use Docker containers to push code between developers and operation
Percent of Effort

Project beginning

Time

Trashing / Rework

Productive Coding

Process: Cost and Time estimates, Writing Requirements, Design, Change Management, Quality Assurance Plan, Development and Integration Plan

Project end
Percent of Effort

Project beginning

Time

Project end

100%

0%

Productive Coding

Trashing / Rework

Process
Example process issues

• Change Control: Mid-project informal agreement to changes suggested by customer or manager. Project scope expands 25-50%
• Quality Assurance: Late detection of requirements and design issues. Test-debug-reimplement cycle limits development of new features. Release with known defects.
• Defect Tracking: Bug reports collected informally, forgotten
• System Integration: Integration of independently developed components at the very end of the project. Interfaces out of sync.
• Source Code Control: Accidentally overwritten changes, lost work.
• Scheduling: When project is behind, developers are asked weekly for new estimates.
Survival Mode

• Missed deadlines -> "solo development mode" to meet own deadlines
• Ignore integration work
• Stop interacting with testers, technical writers, managers, ...
100%

Percent of Effort

Productive Coding

Process

0%

Project beginning

Time

Project end
Hypothesis

• Process increases flexibility and efficiency

• Upfront investment for later greater returns
Phase That a Defect Is Created

- Requirements
- Architecture
- Detailed design
- Construction

Phase That a Defect Is Corrected

Process models

- Ad-hoc
- Waterfall
- Spiral
- Agile
- ...

- More later
Estimating Effort
Task: Estimate Time

• A: Simple Java or web version of the Monopoly boardgame with Pittsburgh street names
  – (you)
• B: Bank smartphone app
  – (you with team of 4 developers, one experienced with iPhone apps, one with background in security)

• Estimate in 8h days (20 work days in a month, 220 per year)
Revise Time Estimate

• C: Remember Scrabble/Carcassonne experience? Is Monopoly similar/different/easier/more challenging/reusable? How much design did you do? Break down the task into ~5 smaller tasks and estimate them. Revise your overall estimate if necessary.
Constructive Cost Model (Cocomo)

• Regression formula based on project history
• Requires experience with similar projects
• Encourages documentation of experience
# Constructive Cost Model (Cocomo)

<table>
<thead>
<tr>
<th>Cost Drivers</th>
<th>Ratings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Very Low</td>
</tr>
<tr>
<td><strong>Product attributes</strong></td>
<td></td>
</tr>
<tr>
<td>Required software reliability</td>
<td>0.75</td>
</tr>
<tr>
<td>Size of application database</td>
<td>0.94</td>
</tr>
<tr>
<td>Complexity of the product</td>
<td>0.70</td>
</tr>
<tr>
<td><strong>Hardware attributes</strong></td>
<td></td>
</tr>
<tr>
<td>Run-time performance constraints</td>
<td></td>
</tr>
<tr>
<td>Memory constraints</td>
<td></td>
</tr>
<tr>
<td>Volatility of the virtual machine environment</td>
<td>0.87</td>
</tr>
<tr>
<td>Required turnabout time</td>
<td>0.87</td>
</tr>
<tr>
<td><strong>Personnel attributes</strong></td>
<td></td>
</tr>
<tr>
<td>Analyst capability</td>
<td>1.46</td>
</tr>
<tr>
<td>Applications experience</td>
<td>1.29</td>
</tr>
<tr>
<td>Software engineer capability</td>
<td>1.42</td>
</tr>
<tr>
<td>Virtual machine experience</td>
<td>1.21</td>
</tr>
<tr>
<td>Programming language experience</td>
<td>1.14</td>
</tr>
<tr>
<td><strong>Project attributes</strong></td>
<td></td>
</tr>
<tr>
<td>Application of software engineering methods</td>
<td>1.24</td>
</tr>
<tr>
<td>Use of software tools</td>
<td>1.24</td>
</tr>
<tr>
<td>Required development schedule</td>
<td>1.23</td>
</tr>
</tbody>
</table>

Study: Variability and Reproducibility in Software Engineering

• Study by Simula Research Lab in Norway
• Created System Requirements Specification for a web information system (11 pages)
• Received bids from 35 companies, 14 with schedule
• Contracted 4 companies to build the same system
Bids and time estimations

- No relationship between price and planned time or methods in bids

<table>
<thead>
<tr>
<th>Dimension</th>
<th>No. of companies</th>
<th>CV</th>
<th>1/CV</th>
<th>Reproducibility</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bids</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firm price</td>
<td>35</td>
<td>0.65</td>
<td>1.5</td>
<td>Low</td>
</tr>
<tr>
<td>Time schedule</td>
<td>14</td>
<td>0.49</td>
<td>2.0</td>
<td>Low</td>
</tr>
<tr>
<td>Emphasis on A&amp;D</td>
<td>27</td>
<td>0.20</td>
<td>4.9</td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Projects</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contractor-related costs</td>
<td>4</td>
<td>0.29</td>
<td>3.4</td>
<td>Medium</td>
</tr>
<tr>
<td>Actual lead time</td>
<td>4</td>
<td>0.14</td>
<td>7.1</td>
<td>High</td>
</tr>
<tr>
<td>Schedule overrun</td>
<td>4</td>
<td>0.87</td>
<td>1.1</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Products</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>4</td>
<td>0.48</td>
<td>2.1</td>
<td>Low</td>
</tr>
<tr>
<td>Usability</td>
<td>4</td>
<td>0.17</td>
<td>5.9</td>
<td>High</td>
</tr>
<tr>
<td>Maintainability</td>
<td>4</td>
<td>0.46</td>
<td>2.2</td>
<td>Low</td>
</tr>
</tbody>
</table>

Tender Prices for Six Projects of the Norwegian Public Roads Administration

<table>
<thead>
<tr>
<th>Construction type</th>
<th>N companies</th>
<th>Stddev</th>
<th>Mean</th>
<th>CV</th>
<th>1/CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bridge</td>
<td>2</td>
<td>15,000</td>
<td>45,000</td>
<td>0.3</td>
<td>3.0</td>
</tr>
<tr>
<td>Electrical installation</td>
<td>6</td>
<td>13,000</td>
<td>49,000</td>
<td>0.3</td>
<td>3.7</td>
</tr>
<tr>
<td>Electrical installation</td>
<td>7</td>
<td>15,000</td>
<td>57,000</td>
<td>0.3</td>
<td>3.9</td>
</tr>
<tr>
<td>Road maintenance</td>
<td>5</td>
<td>4,400</td>
<td>47,000</td>
<td>0.1</td>
<td>10.5</td>
</tr>
<tr>
<td>Road maintenance</td>
<td>3</td>
<td>2,900</td>
<td>16,000</td>
<td>0.2</td>
<td>5.6</td>
</tr>
<tr>
<td>Road maintenance</td>
<td>3</td>
<td>9,800</td>
<td>47,000</td>
<td>0.2</td>
<td>4.8</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td>4</td>
<td>10,000</td>
<td>44,000</td>
<td>0.2</td>
<td>5.2</td>
</tr>
</tbody>
</table>
## Development Process

<table>
<thead>
<tr>
<th></th>
<th>Company A</th>
<th>Company B</th>
<th>Company C</th>
<th>Company D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nationality</strong></td>
<td>Norwegian</td>
<td>Norwegian</td>
<td>Norwegian</td>
<td>International</td>
</tr>
<tr>
<td><strong>Ownership</strong></td>
<td>Private</td>
<td>By employees</td>
<td>By employees</td>
<td>Listed on exchanges</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>Oslo</td>
<td>Oslo</td>
<td>Bergen</td>
<td>Oslo + 20 countries</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Appr. 100</td>
<td>Appr. 25</td>
<td>Appr. 8</td>
<td>Appr. 13,000 worldwide</td>
</tr>
<tr>
<td><strong>Firm price</strong></td>
<td>€20,000</td>
<td>€45,380</td>
<td>€8,750</td>
<td>€56,000</td>
</tr>
<tr>
<td><strong>Agreed time schedule</strong></td>
<td>55 days</td>
<td>73 days</td>
<td>41 days</td>
<td>62 days</td>
</tr>
<tr>
<td><strong>Planned effort on A&amp;D</strong></td>
<td>28%</td>
<td>20%</td>
<td>7%</td>
<td>23%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Company A</th>
<th>Company B</th>
<th>Company C</th>
<th>Company D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dimensions</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Project</strong></td>
<td>Contractor-related costs</td>
<td>90 hours</td>
<td>108 hours</td>
<td>155 hours</td>
</tr>
<tr>
<td></td>
<td>Actual lead time</td>
<td>87 days</td>
<td>90 days</td>
<td>79 days</td>
</tr>
<tr>
<td></td>
<td>Schedule overrun</td>
<td>58%</td>
<td>23%</td>
<td>93%</td>
</tr>
<tr>
<td><strong>Product</strong></td>
<td>Reliability</td>
<td>Good</td>
<td>Good</td>
<td>Poor</td>
</tr>
<tr>
<td></td>
<td>Usability</td>
<td>Good</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>Maintainability</td>
<td>Good</td>
<td>Poor</td>
<td>Poor</td>
</tr>
</tbody>
</table>
Risk and Uncertainty
Risk management

• Key task of a project manager
• Identify and evaluate risks early
• If necessary, plan mitigation strategies
• Document results of risk analysis in project plan

• Project risks: scheduling and resources
  – e.g., staff illness/turnover
• Product risks: Quality and functionality of the product
  – e.g. used component too slow
• Business risks:
  – e.g., competitor introduces similar product
Sources of Uncertainty

• Unpredictable operating environment
  – Cybersecurity threats, device drivers
  – Unanticipated usage scenarios

• Limited predictive power of models
  – Halting, abstract interpretation, testing

• Bounded rationality of humans
  – Designers, developers
  – Customers, users
Innovative vs Routine Projects

• Most software projects are innovative
  – Google, Amazon, Ebay, Netflix
  – Vehicles and robotics
  – Language processing, Graphics

• Routine (now, not 10 years ago)
  – E-commerce websites?
  – Many control systems?
  – Routine gets automated -> innovation cycle
Case 1
Case 2
Discussion

• What are important risks in early and late phases of development?
• Analyze risks to rank them
• How can they be managed/mitigated?
Accepting and Coping with Risks

• Selectively innovate to increase value
• Improve capability and competitiveness

• Focus risk where it is needed
• Rely on precedent and convention (experience)
• Use iteration and feedback
  – prototypes, spiral development, sprints
Managing Risks

• Address risk early
• Prototyping, spiral development
• Identify mitigation strategies
Example: Bank Software

• Build on vendor and open-source components
  – OS, DB, app server, tooling, backups, information management, ...
  – Stick with reliable vendors and integrators
• Focus internal risk-taking on business differentiators
  – Web-site features, internal queries, decision algorithms
• Outsource to access expertise, lower costs, gain flexibility
  – Black-box testing
  – Overall system architecture
  – Internal engineering practices and tools
Risk analysis

• Estimate likelihood and consequences
• Requires experienced project lead
• Rough estimations usually sufficient
  – very low (<10%), low (<25%), ...
  – catastrophic, severe, acceptable, negligible
• Focus on top 10 risks
Risk planning

• Risk avoidance
• Risk minimization
• Emergency plans
## Strategies to help manage risk

<table>
<thead>
<tr>
<th>Risk</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organizational financial problems</td>
<td>Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost-effective.</td>
</tr>
<tr>
<td>Recruitment problems</td>
<td>Alert customer to potential difficulties and the possibility of delays; investigate buying-in components.</td>
</tr>
<tr>
<td>Staff illness</td>
<td>Reorganize team so that there is more overlap of work and people therefore understand each other’s jobs.</td>
</tr>
<tr>
<td>Defective components</td>
<td>Replace potentially defective components with bought-in components of known reliability.</td>
</tr>
<tr>
<td>Requirements changes</td>
<td>Derive traceability information to assess requirements change impact; maximize information hiding in the design.</td>
</tr>
<tr>
<td>Organizational restructuring</td>
<td>Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.</td>
</tr>
<tr>
<td>Database performance</td>
<td>Investigate the possibility of buying a higher-performance database.</td>
</tr>
<tr>
<td>Underestimated development time</td>
<td>Investigate buying-in components; investigate use of a program generator.</td>
</tr>
</tbody>
</table>
Planning
(Software) Projects

- One-time endeavour; unique wrt
  - Goals
  - time, financial, personal, and other constraints
  - differences to other endeavours
  - project-specific organization
- Defined, limited time (clear start and end)
- Clear goals
- Constraints: budget, resources, ...
- Requires management
  - high risk
  - coordination of experts
- Software development always proceeds in projects
- New/unique goals, innovative technology
- Intangible result, progress hard to measure
- Software projects tend to fail more often than industrial projects
Measuring Progress?

• “I’m almost done with the app. The frontend is almost fully implemented. The backend is fully finished except for the one stupid bug that keeps crashing the server. I only need to find the one stupid bug, but that can probably be done in an afternoon. We should be ready to release next week.”
Measuring Progress?

- Developer judgment: x% done
- Lines of code?
- Functionality?
- Quality?
Milestones and deliverables

• Making progress observable, especially for software
• Milestone: clear end point of a (sub)tasks
  – For project manager
  – Reports, prototypes, completed subprojects
  – "80% done" not a suitable milestone
• Deliverable: Result for customer
  – Similar to milestone, but for customers
  – Reports, prototypes, completed subsystems
Case 3

• Define tasks and milestones for the Monopoly implementation

• Which tasks have dependencies?
Project Planning

1. Identify constraints
2. Estimate project parameters
3. Define milestones
4. Create schedule

Budget, Personal, Deadlines

activities begin

- Check progress
  - Done?
    - yes
    - no
      - Reestimate project parameter
      - Refine schedule
      - Problem?
        - yes
        - renegotiate constraints
        - Technical review
      - no

- renegotiate constraints

- Technical review

- Abort?

- new feature requests

Every 2-3 weeks
Project Manager Tasks

- Divide project into work packages with measurable outcomes (each 1-10 weeks)
- Estimate time and resources
- Create order and determine parallelism
- Plan buffers for anticipated and unanticipated problems
- Software available, such as, Microsoft Project, GanttProject, Kplato, etc

- Requires experience for estimation
Activity Network

[Diagram of an activity network with milestones and work packages indicated.]

- **Work package**
- **Milestone**
Gantt Diagrams
Inaccurate time predictions are normal -> Update schedule
Reasons for Missed Deadlines

- Insufficient staff (illnesses, staff turnover, ...)
- Insufficient qualification
- Unanticipated difficulties
- Unrealistic time estimations
- Unanticipated dependencies
- Changing requirements, additional requirements
- Especially in student projects
  - Underestimated time for learning technologies
  - Uneven work distribution
Recognize Scheduling Issues Early

- Monitoring and formal reporting necessary
  - Establish who, when, what
  - Compare planned/actual data
- Measurable milestones
- Outdated schedules no meaningful management mechanism
Almost Done Problem

- Last 10% of work -> 40% of time (or 20/80)
- Make progress measurable
- Avoid depending entirely on developer estimations
Milestone Trend Analysis

Estimated completion time

Actual time
Milestone Trend Analysis

- Quickly rising
  - estimations too optimistic
- Changing trends
  - unreliable early estimations
- Ziz-zag pattern
  - unreliable estimations
- Falling
  - overly large buffers
Mitigation strategies

• Additional personal
  – especially experts for specific tasks
• Temporarily increasing hours of work (overtime, vacation freeze)
  – short-term solution only
• Improve tooling, methods, and processes
• Buy, contract, off shore
• Renegotiate / reduce functionality
  – Set priorities, incremental deployment
  – Move deadline
• Avoid: less testing/quality assurance
Team productivity

- Brook's law: Adding people to a late software project makes it later.
Teamwork (Student Teams)

(more on teams in real projects later in the course)
Expectations

• Meet initially and then regularly
• Review team policy
• Divide work and integrate
• Establish a process
• **Set and document clear responsibilities and expectations**
  – Possible Roles: Coordinator, Scribe, Checker, Monitor
  – Rotate roles every assignment
• Every team member should understand the entire solution
Team Policies

• see document

• Make agreements explicit and transparent

• Most teams will encounter some problem
Dealing with problems

- Openly report even minor team issues in individual part of assignments
- In-class discussions and case studies
- Additional material throughout semester
- We will attend one team meeting
Planning and In-Team Communication

• Asana, Trello, Microsoft Project, ...
• Github Wiki, Google docs, ...
• Email, Slack, Facebook groups, ...
Homework and Readings
Reading assignment

• Prepare a case discussion about Healthcare.gov
  – Read the TIME Magazine article “Code Red”
  – Explore Wikipedia, Youtube, other resources
  – Look at the webpage itself

• Think about what went wrong and why (both technical and nontechnical issues), how it could be fixed or prevented in future projects, ...
Homework 1

• Team assignment
• Build grad school application system
• Process focus
  – Estimate and track time
  – Plan process and observe deviations
  – Analyze risks
### Application Summary

The table below lists what is necessary to apply for admission to graduate study in the Department of Electrical and Computer Engineering at Carnegie Mellon.

If you have questions about the application process, please check the frequently asked questions page before submitting an email. If the FAQ does not answer your question, please contact the ECE Admissions Office. If you have technical questions or concerns regarding the online application website, contact the ECE Web Team.

#### Program Selection

<table>
<thead>
<tr>
<th>Program Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Program Selection</td>
</tr>
</tbody>
</table>

1. **M.S. In ECE**
   - Pittsburgh, PA
   - [More Information](#)

2. **M.S. SV**
   - Silicon Valley, CA
   - [More Information](#)

#### Required Items

<table>
<thead>
<tr>
<th>Item</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application for Admission and Financial Aid</td>
<td>Complete</td>
</tr>
<tr>
<td>Statement of Purpose</td>
<td>Open</td>
</tr>
<tr>
<td>Technical Areas of Interest Matrix</td>
<td>Open</td>
</tr>
</tbody>
</table>

### Application Checklist

The table contains both the items that can be completed and submitted online and a checklist of supplementary materials that need to be mailed in.

For your application to be eligible for consideration you must complete all of the required items in the list. Remember, you can save your materials to work on at a later time.

If the status reads "Complete" or "Closed" for all of the required items in the checklist, you have finished the application process.
Further Reading
