Fortgeschrittene Themen des Projektmanagement

Prof. Dr. Peter Müller

8. September 2009

Agenda for Today

9:15 – 10:00 Principles
10:00 – 10:45 Earned Value Method
10:45 – 11:00 Break
11:00 – 11:45 Software Estimation
11:45 – 12:30 Risk Management
12:30 – 13:45 Lunch
13:45 – 14:30 Agile Methods
14:30 – 15:15 Scrum
15:15 – 15:30 Break
15:30 – 16:15 Tools
16:15 – 16:45 Discussion
1. Principles

What is a Project?

- Definition:
  A project is a temporary endeavor undertaken to create a unique product or service

- In contrast: Operations are ongoing and repetitive

Every project has a definite beginning and a definite end

The product or service is different in some distinguishing way from all similar products and services
From Projects to Operations

- Applications are neither projects nor operations, but products

Core Activities and Project Management

- Project Management organizes and leads the project work to meet project requirements
- Core Activities ultimately create the product of a project
PM Knowledge Areas

PM activities fall into nine Knowledge Areas

1. Principles

Project Management

- Project Integration Management
- Project Cost Management
- Project Communications Management

- Project Scope Management
- Project Quality Management
- Project Risk Management

- Project Time Management
- Project Human Resource Management
- Project Procurement Management

Project Management Life Cycle

1. Principles

- Initiating Processes
- Planning Processes
- Controlling Processes
- Executing Processes
- Closing Processes

Peter Müller – Fortgeschrittene Themen des Projektmanagement, September 08, 2009
Example: Time Management

Schedule Development

Task List for Each Team Member

Schedule Updates

Status Reports

Project Plan Execution

Corrective Actions

1. Principles

Process Groups

- Project groups are not discrete one-time events
- They overlap and occur at varying levels of intensity within each phase of the project
Interaction between Phases

- Input and output of the processes depend on the phase in which they are carried out
- But processes are not limited to one phase (overlaps)

Project Success

- Definition:
  A project is successful if the specified results are delivered in the required quality and within the predetermined time and resource limits.
- Computer scientists tend to focus on scope and quality only
  - The development of a technically perfect application is not a success if the cost exceeds the price clients are willing to pay
  - Excellent project results often are worthless if they come too late (temporary market windows, external deadlines)
1. Principles

The Triple Constraint

- Project objectives are **equally important**
- Actions in one project area usually affect other areas

- **Tradeoffs** among objectives must be **managed**
- **Priorities** are set by customers and management
More Competing Objectives

Scope
Quality
Risk
Time
Cost
Customer Satisfaction

Assumptions

- Definition: *Assumptions are factors that, for planning purposes, are considered to be true, real, or certain*
  - Assumptions affect all aspects of project planning, and are part of the progressive elaboration of the project
  - Project teams frequently identify, document, and validate assumptions as part of their planning process
  - Assumptions generally involve a degree of risk
2. Earned Value Method

The Triple Constraint

- Project objectives are equally important
- Actions in one project area usually affect other areas
Planned Value (PV)

- The **cumulative** sum of the **approved** cost for activities **scheduled**
- Corresponds to the **cost baseline**
- **Budget at completion** is the estimated baseline total cost: $BAC = PV(\ end)\$

Actual Cost (AC)

- Total **cost incurred** for the project up to a specified date
- The **actual** or **real** cost of work performed
- Contains both direct and indirect cost
2. Earned Value Method

Earned Value (EV)

- The sum of **approved cost estimates** for activities **completed** up to a specified date
- An activity is completed if PV=EV, regardless of the actual cost

\[
\text{BAC} \quad \text{AC}(t) \quad \text{PV}(t) \quad \text{EV}(t) \quad \text{Earned Value}
\]

\begin{align*}
\text{Start} & \quad t & \quad \text{Planned End} & \quad \text{Actual End}
\end{align*}

2. Earned Value Method

- Expresses effort, cost, and time as **monetary value**
  - PV(\(t\)): Worth of the activities scheduled (planned)
  - AC(\(t\)): Cost spent
  - EV(\(t\)): Worth of the activities performed
- Compares the amount of work planned to what was actually accomplished to **determine cost and schedule performance**
Example

<table>
<thead>
<tr>
<th>Activity</th>
<th>PV( t )</th>
<th>AC( t )</th>
<th>EV( t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paint wall</td>
<td>800</td>
<td>1000</td>
<td>800</td>
</tr>
<tr>
<td>Paint ceiling</td>
<td>400</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>Total</td>
<td>1.200</td>
<td>1.300</td>
<td>1.100</td>
</tr>
</tbody>
</table>

Cost Performance Index (CPI)

- Compares **budgeted cost** of work performed to **actual cost**
- Indicates the **efficiency** of the project

\[
CPI = \frac{EV}{AC}
\]

- How much do we get out of one Franc we spend?

\[
CPI = \frac{1.100}{1.300} = 85\%
\]
Schedule Performance Index (SPI)

- Compares *work performed* to *work planned*
  \[
  \text{SPI} = \frac{EV}{PV}
  \]

- How fast does the project progress in relation to how fast it is expected to progress?

<table>
<thead>
<tr>
<th>Activity</th>
<th>PV (t)</th>
<th>AC (t)</th>
<th>EV (t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paint wall</td>
<td>800</td>
<td>1000</td>
<td>800</td>
</tr>
<tr>
<td>Paint ceiling</td>
<td>400</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>Total</td>
<td>1,200</td>
<td>1,300</td>
<td>1,100</td>
</tr>
</tbody>
</table>

\[
\text{SPI} = \frac{1.100}{1.200} = 92\%
\]

Calculated Estimate at Completion

- Budget modified by performance
  - If the current variances are *typical for the future*

- Actual to date plus remaining budget
  - If the current variances are *atypical for the future*

- Actual plus a new estimate for remaining work
  - If the original estimate was *fundamentally flawed*

\[
\text{CEAC}_1 = \frac{BAC}{CPI}
\]

\[
\text{CEAC}_2 = AC + BAC - EV
\]

\[
\text{CEAC}_3 = AC + ETC
\]
Interpreting EV-Indicators

- Typically, indicators are stable after 20% of the project duration
- CPI > 1: Project is in budget
- CPI < 1: Project is over budget
- SPI > 1: Project is ahead of schedule
- SPI < 1: Project is behind schedule

Golden Rules of Earned Value

- Rule 1: Earned value should be verified by physically examining the work product associated with the activity
- Rule 2: For unfinished activities, earned value estimates are usually just a guess. Apply one of the following rules consistently
  - 50/50 Rule: A task is considered 50% complete when it begins and 100% only when it is completed
  - 20/80 Rule: A task is considered 20% complete when it begins and 100% only when it is completed
  - 0/100 Rule: A task does not get credit for partial completion, only for full completion
3. Software Estimation

Estimations in Software Projects

- Mostly personnel cost (effort)
- Travel, training
- Hardware, software

Duration is essentially effort / resources

Costs

Effort

Schedule

Resources
Estimation Exercise

- How many passenger planes does Lufthansa have?
  - Not counting regional subsidiaries

- How can we approach this problem systematically?

Empirical Estimation: Expert Judgment

- Estimate is based on experience and historical data

- Involve experts in
  - Development techniques
  - Application domain

- Most common technique in practice
3. Software Estimation

Top-Down Estimation

- Estimation by analogy
  - Comparison with similar projects
  - Analysis of differences
  - Typical example: SAP introduction

**Pros**
- Quicker and less expensive than other methods
- Can be done early in the project

**Cons**
- Underestimation of difficult technical problems likely
- No detailed justification of estimate
- Be aware of scalability problems!

---

3. Software Estimation

Top-Down Estimation: Delphi Method

**Step 1:** Each expert submits
- Estimate
- Justification

**Step 2:** Each expert receives summary of all estimates

**Step 3:** Each expert submits
- New estimate
- Justification of deviation from average of previous estimates

**Step 4:** Iterate until consensus is achieved

- More accurate than ordinary expert judgment
- Eliminates outliers
- More expensive to produce
Bottom-Up Estimation

- Estimation by **decomposition**
  - Estimating the effort for **individual work packages**
  - Cost and accuracy depend on size of the work packages

**Pros**
- See "cons" of top-down estimation

**Cons**
- Underestimation because effort does not grow linearly (due to complexity, etc.)
- Underestimation of integration effort
- Requires initial system design

Program Evaluation and Review Technique

- **Goal**: Manage probabilities with simple statistics
- **Approach**: Ask several experts for three estimates
  - Optimistic, Likely (mode), and Pessimistic
- **Important formulas**
  - Mean \( M = \frac{O + 4 \times L + P}{6} \)
  - Deviation \( V = \frac{P - O}{6} \)

**Assumptions**
- Project effort is normally distributed (more than 20 work packages)
- Work package efforts are statistically independent (ignores single underlying cause of delay)
Algorithmic Estimation

Flights per resident per year \( \times \frac{1}{365} \times \frac{1}{180} \times \frac{1}{4} \times 0.8 = 380 \)

Residents in Germany

Days per year

Passengers per flight

Percentage of planes in operation

- Algorithmic estimation is based on
  - **Cost model**, represented by formula
  - **Measurement of size** (passengers, destinations, etc.)
  - **Parameters** (size of planes, planes in operation, etc.)

3. Software Estimation

Algorithmic Estimation of Software

- Basic cost model

\[
\text{Effort} = A \times \text{Size}^B \times \text{m(X)}
\]

- **Size**: Some measurement of the software size
- **A**: Constant factor that depends on
  - Organizational practices
  - Type of software
- **B**: Usually lies between 1 and 1.5
- **X**: Vector of cost factors
- **m**: Adjustment multiplier
Cost Models

\[
\text{Effort} = A \times \text{Size}^B \times m(X)
\]

- Cost models
  - Define a way to determine the size
  - Define cost factors X
  - Provide defaults for parameters A, B, m
    (based on hundreds of projects)

- Important examples
  - Function point analysis
  - Constructive cost model (COCOMO)

Measuring Size: Lines of Code

- Software size can be measured in lines of source code
  - Most commonly used metric

- **Difficult in early phases** of the project (before design is known)
  - Reuse, make-or-buy decisions

- **Influenced** heavily by choice of **programming language**

- Should only be **used indirectly**
Function Point Analysis

- Size is estimated based on requirements

Inputs

Inquiries

Function

Outputs

Internal files

External files

Functions

- **Inputs**
  - Forms, dialogs, messages, XML documents

- **Outputs**
  - Web pages, reports, graphs, messages, XML documents

- **Inquiries** (input/output combinations)
  - Simple web inputs, generally producing a single output

- **Logical internal files** (controlled by the program)
  - Tables, views or files in database

- **External files** (controlled by other programs)
  - Tables or files used from other systems or databases
### Complexity of Functions

- **Determine complexity** of each function
  
<table>
<thead>
<tr>
<th>Input</th>
<th>Simple</th>
<th>Average</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data elements</td>
<td>1-5</td>
<td>6-10</td>
<td>&gt;10</td>
</tr>
<tr>
<td>Checking</td>
<td>Formal</td>
<td>Formal, logical</td>
<td>Formal, logical, requires DB access</td>
</tr>
</tbody>
</table>

- **Weight** each function according to complexity
  
<table>
<thead>
<tr>
<th>Factor</th>
<th>Simple</th>
<th>Average</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Outputs</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>Inquiries</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Ext. files</td>
<td>7</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Int. files</td>
<td>5</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

### Cost Factors

- **Rate each element from 0 – 5**
  - 0: no influence
  - 1: insignificant influence
  - 2: moderate influence
  - 3: average influence
  - 4: significant influence
  - 5: strong influence

- **Technical complexity factor**
  - TCF = 0.65 + 0.01 × sum
  - Varies between 0.65 and 1.35
Function Point Computation

<table>
<thead>
<tr>
<th></th>
<th>Simple</th>
<th>Average</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>6 x 3 = 18</td>
<td>2 x 4 = 8</td>
<td>3 x 6 = 18</td>
</tr>
<tr>
<td>Outputs</td>
<td>7 x 4 = 28</td>
<td>7 x 5 = 35</td>
<td>0 x 7 = 0</td>
</tr>
<tr>
<td>Inquiries</td>
<td>0 x 3 = 0</td>
<td>2 x 4 = 8</td>
<td>4 x 6 = 24</td>
</tr>
<tr>
<td>Ext. files</td>
<td>9 x 5 = 45</td>
<td>0 x 7 = 0</td>
<td>2 x 10 = 20</td>
</tr>
<tr>
<td>Int. files</td>
<td>5 x 7 = 35</td>
<td>2 x 10 = 20</td>
<td>3 x 15 = 45</td>
</tr>
</tbody>
</table>

Unadjusted function points (UFP): 304
Technical complexity factor (TCF): 1.15
Adjusted function points: \( FP = UFP \times TCF \)

Calibration

- Flights per resident per year
- Passengers per flight
- Percentage of planes in operation

\[
80,000,000 \times \frac{1}{365} \times \frac{1}{180} \times \frac{4}{0.8} = 380
\]

- Assume that model (formula) is correct
- **Calibrate** model based on comparable airlines
- Estimate number of residents in the country
Determining Effort and Size

- Empirical value for effort
  - Or use a table
  \[ \text{Effort} = \frac{\text{FP}^{1.4}}{150} \]

- Empirical value for size

- Huge differences in productivity
  - Factor 10-20 between individual programmers
  - Factor 4 between companies

<table>
<thead>
<tr>
<th>Language</th>
<th>Level</th>
<th>Statements per UFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembler</td>
<td>1</td>
<td>320</td>
</tr>
<tr>
<td>C</td>
<td>2.5</td>
<td>125</td>
</tr>
<tr>
<td>C++</td>
<td>6.5</td>
<td>50</td>
</tr>
<tr>
<td>Perl</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>Pascal</td>
<td>3.5</td>
<td>90</td>
</tr>
<tr>
<td>Visual Basic 3</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>Excel</td>
<td>50</td>
<td>6</td>
</tr>
</tbody>
</table>

Function Point Analysis: Discussion

**Pros**
- Based on requirements (instead of code size)
- Can be applied in early project phases
- Can be calibrated (for company, project type)
- Counting standards by “International Function Points User Group”
- Technology-independent

**Cons**
- Estimation of overall effort (not per phase)
- Tailored towards functional decomposition (rather than OO)
- Tailored towards information systems
- Needs calibration to produce reliable results
Estimation Techniques: Discussion

**Empirical Estimation**
- Accurate if experts are experienced
- Experts can be strongly biased (over-optimism)

**Algorithmic Estimation**
- Very accurate if model is calibrated
- Calibration is very difficult and expensive
- Estimation is expensive

**Empirical studies**
- Do not show that uncalibrated algorithmic estimation is, in general, more accurate
- Show that algorithmic estimation is more accurate than experts who do not have important domain knowledge

3. Software Estimation

Other Estimation Strategies

**Parkinson’s Law**
- Work expands to fill the time available
  - Gold plating
- Effort is determined by available resources
- Important for team management

**Pricing to win**
- Cost is estimated to whatever the customer is willing to spend
- Common strategy to win projects
- Features are negotiated later, constrained by agreed costs
- Costs are fixed, not requirements
Estimating Process

1. Establish objectives
   - Why? Accuracy? Audience?

2. Determine project details

3. Select strategy and plan
   - Estimators, type of validation, historic data

4. Generate effort estimate

5. Determine team size and duration
   - Duration = \sqrt{Effort}
   - Duration = 3.0 \times \text{Effort}^{1/3}
   - (Effort in person months, Duration in months)
   - Effort = Duration \times \text{Team Size}

6. Validate and finalize assumptions
   - Different method, review

7. Document assumptions

4. Risk Management
Risk

- Definition: 
  *An uncertain event or condition that, if it occurs, has a positive or negative effect on a project objective*

- Risks have three components
  - A possible future event (uncertainty)
  - Probability of the occurrence of that event (likelihood)
  - Impact of that event (consequence)

"Reports that say that something hasn't happened are always interesting to me, because as we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns — the ones we don't know we don't know."

Donald Rumsfeld, 2003
Risk Classification

- Known risks
  - Unclear requirements
  - Inexperienced team

- Unknown risks: Foreseen based on experience
  - Difficult communication with customer
  - Fluctuation within team

- Unknowable risks: Cannot be foreseen
  - Half of the team gets fish poisoning at first social event
  - Earthquake wipes out production plant

Risk Management

1. Risk Management Planning
2. Risk Identification
3. Risk Analysis
4. Risk Response Planning
5. Risk Monitoring and Control
4. Risk Management

Risk Identification

- Risk identification workshop
- Periodic risk reporting
- Interviews with key team members
- Breakdown technique

Risk Events
- Tied to business objectives
- Related to WBS
- Preliminary prioritization

Business objectives
Product definition
WBS
Stakeholder experience
Risk checklists

Breakdown Technique

- Identify risks systematically
- Project objectives: Win, Budget, Satisfy
- Focus area: A breakdown of the project’s potential sources of risk
- Risk driver: A condition that increases the probability that a risk event will be present
### Breakdown Example

- **Budget**
  - **Resources**
    - Informal commitments
      - Resources unavailable
      - Customer resources lack skill
  - **Customer**
    - Requirements not baselined
      - Requirements may grow uncontrollably
      - Schedule delays
  - **Solution**
    - New Technology
      - Throughput never achieved
      - Unproven solution
  - **Terms and Conditions**
    - T&M subcontract
      - Supplier may exceed budget
      - Supplier default

### Risk Analysis

- Often called risk assessment or risk evaluation
- Determine
  - **Probability** of the risk to occur
  - **Impact** on the project objectives in case the risk occurs
  - **Severity** (Severity = Probability x Impact)
- Identify risks to be mitigated
- Qualitative analysis
- Quantitative analysis
  - Based on estimates and simulations
### Probability Criteria

<table>
<thead>
<tr>
<th>Qualitative Rating</th>
<th>Quantitative Rating</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>&gt;84%</td>
<td>Almost assured to happen</td>
</tr>
<tr>
<td>High</td>
<td>60-84%</td>
<td>Likely to happen</td>
</tr>
<tr>
<td>Medium</td>
<td>35-59%</td>
<td>Somewhat likely to happen</td>
</tr>
<tr>
<td>Low</td>
<td>10-34%</td>
<td>Not very likely to happen</td>
</tr>
</tbody>
</table>

### Impact Criteria

<table>
<thead>
<tr>
<th>Risk Rating</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>Has potential to cause cancellation of the project</td>
</tr>
<tr>
<td>High</td>
<td>Likely to cause significant disruption to schedule, increase in cost, or degradation of performance</td>
</tr>
<tr>
<td>Medium</td>
<td>Has potential to cause some disruption to schedule, increase in cost, or degradation of performance</td>
</tr>
<tr>
<td>Low</td>
<td>Has little potential to cause some disruption to schedule, increase in cost, or degradation of performance</td>
</tr>
</tbody>
</table>
4. Risk Management

Severity of Individual Risks

<table>
<thead>
<tr>
<th>Severity</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Very High</td>
</tr>
<tr>
<td>Very High</td>
<td>Unacceptable</td>
</tr>
<tr>
<td>High</td>
<td>Very High</td>
</tr>
<tr>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

Risk Ranking

- Prioritize risk according to
  - Severity
  - Timing
  - Time required to mitigate (preliminary estimate)
  - Etc.
- “Top 10” Approach
  - Develop mitigation strategies for top 10 risks
  - Use the top 10 as an agenda item for regular project meetings
### Risk Documentation

<table>
<thead>
<tr>
<th>Risk ID</th>
<th>WBS Number</th>
<th>Risk Event</th>
<th>Owner</th>
<th>Area of Impact (W/B/S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.04.05</td>
<td>Requirements will grow uncontrollably</td>
<td>PM</td>
<td>B/S</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Probable Impact Date</th>
<th>Risk Probability</th>
<th>Risk Impact</th>
<th>Severity</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.07.2004</td>
<td>High</td>
<td>Very high</td>
<td>Very high</td>
<td>1</td>
</tr>
</tbody>
</table>

### Risk Response Planning

- Risk Response Planning is often called **risk mitigation**

- Establish mitigation options
- Develop action plan
- Re-evaluate risks (secondary risks)
Mitigation Strategies

- **Avoid** the path or project to eliminate the risk
- **Ignore / Accept** the risk and its consequences if it occurs
- **Transfer** all or part of the risk to another party
- **Contain** the risk by specific actions to lower the probability and / or impact
- **Establish contingency**: Set funds aside to be used if the risk occurs or when later containment is deemed appropriate

Extended Documentation

<table>
<thead>
<tr>
<th>Risk ID</th>
<th>Mitigation Strategy</th>
<th>Mitigation Task</th>
<th>Responsible</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Contain</td>
<td>Use phased approach</td>
<td>PM</td>
<td>in progress</td>
</tr>
</tbody>
</table>
Execution and Post Mitigation

- Mitigation tasks have to be **integrated into project plan** (WBS)
- Execution has to be **closely monitored**
- Risk has to be **re-assessed** to check whether mitigation is successful

<table>
<thead>
<tr>
<th>Severity</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Very High</td>
</tr>
<tr>
<td>Very High</td>
<td>Very High</td>
</tr>
<tr>
<td>High</td>
<td>Very High</td>
</tr>
<tr>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

Risk Monitoring

- **Implement, track, and reassess** mitigation strategies
- **Communicate** risk plan status to stakeholders
- **Update** documents
4. Risk Management

Triggers for Risk Reassessment

Key events
- Phase exit, phase start
- Key milestones
- Key decisions (e.g., go / no-go)

Regular Assessment

Changes
- Resources
- Scope
- Environment

Customer:
Changes in
- Requirements
- Leadership
- Attitude

Technical
- Technology failure
- Unavailability
- Incompatibility

Environmental changes
- Political
- Social
- Legal

5. Agile Methods

Hermann Lehner
Overview

This introduction to agile planning is divided into five parts:

- Motivation for agile planning
- An agile approach to projects
- An agile approach to planning
- Estimating in agile projects
- The Scrum empirical development process

Agile Project Characteristics

- Many changes during the projects
- Wide “Cone of uncertainty” in the beginning
Impact on Planning

Planning …
- is focused more on the process of planning than on the plan
- encourages changes
- results in plans that can be easily changed
- is spread throughout the project

Why traditional planning fails
- Features are not developed by priority
- Planning by activity rather than feature
  - Activities don’t finish early
  - Activities don’t have a user value
  - Activities are not independent
- Uncertainty is ignored
- Multitasking causes further delays
An agile approach to projects

```
1. Team
2. Priorities
3. Iterations
4. Agile Project
5. Deliverables
6. User Stories
7. Inspect and Adapt
```

Agile Team – four Roles

- Makes decisions to increase return on investment
- Establishes common vision of the product
- Defines priorities

- Product owner
- Customer
- Developer
- Project Manager

- Funds or buys the product
- Sometimes combined with product owner
- Develops the product
- Focus more on leadership than management

Peter Müller – Fortgeschrittene Themen des Projektmanagement, September 08, 2009
5. Agile Methods

**Iterations**

- Typical **duration** between 2 and 4 weeks
- **Timeboxed**: Iteration finishes always on fixed date
- Number of implemented **features** is variable

**User Stories**

- Brief description of functionality as viewed by a user or customer of the system
- Free form, no mandatory syntax, lightweight
Deliverables

- Each iteration results in a deliverable
- Coded, tested, and potentially shippable
- Small addition of functionality

Release

- Consists of iterations that add related functionality
- Usually every 2 to 6 months
- User can see big improvement over last release
5. Agile Methods

Inspect and Adapt

- At start of each iteration, gained knowledge can lead to change of plan
- Changes add more value to a plan

Focusing on Business Priorities

- Features delivered in order specified by product owner
  
  Decision is influenced by
  - Priority list from customer
  - Team’s capabilities
  - Optimal return on investment

- Features need to have **minimized** technical dependencies
An agile approach to planning

Summary of characteristics of agile projects:

- Project generates **knowledge**. This should be used!
  - Product knowledge
  - Project knowledge

- An agile Project is more a **timed** race than a 10km race

- Result of Project is unknown and **unknowable**

---

Levels of Planning

- Planning is looking ahead … but you can’t look at things behind the **horizon**!

- Different horizons

- Take time to look further ahead once you reached the horizon
5. Agile Methods

Release Planning

Determine conditions of satisfaction

Estimate user stories

Select integration length

Estimate velocity

Prioritize user stories

Select stories * release date

until conditions are best met

Iteration Planning

Adjust priorities

Identify iteration goal

Select a story to add

Expand story into tasks

Estimate tasks

Ask for team commitment

can commit; not full

can commit; full

can not commit

Remove user story

Peter Müller – Fortgeschrittene Themen des Projektmanagement, September 08, 2009
Daily Planning

- Stand-up meeting
- Coordinate activities of tasks
- Synchronize daily efforts
- Horizon is end of day (don't plan any further!)

Estimating in agile projects

- Estimation techniques do not change
  - Expert Opinion
  - Analogy (Top Down)
  - Decomposition (Bottom Up)
  - Delphi method

- Estimation units
  - Story points
  - Ideal days
- Deriving duration (velocity)
- Estimation scale
Story Points

- Assign 1-10 points to all user stories
- Start with medium sized stories (assign 5 points)
- Estimating story points can be easier/faster than estimating ideal days
- Humans can only compare things within one order of magnitude

Deriving Duration (velocity)

- After first iteration, sum up story points for completed user stories
- Take that value as velocity of the team
- Correct velocity value after every iteration
Estimation scale

- Don’t overdo it! (Again: don’t ignore uncertainty)
- Reasonable scales are
  - 1, 2, 3, 5, and 8
  - 1, 2, 4, and 8
- Vague user stories are often described as ‘Epic’ or ‘Theme’ (not in one order of magnitude)
- One can extend scale for them
  - 13, 20, 40, 100
- Split them up as soon as knowledge allows it
The SCRUM Development Process

What is Scrum?

- Agile development process
- Establishes, maintains and monitors key control parameters.
- Use measurements to maximize flexibility while maintaining control.
Key Control Parameters

- Backlog
- Problems
- Changes
- Key Controls
- Issues
- Risks
- Solutions

Measurements

- Burnd Down Chart
- Meetings
The Scrum Process

- User Story
  - Prioritized list of features
  - Free form
    - User Stories
    - Scenarios
  - Measurement for progress
  - Changes over time
  - No commitment
Sprint Backlog

- Work to do in one sprint
- Team estimates effort

Sprint

- Get all sprint backlog items into a shippable piece of product
- Takes 30 days
- Team organizes itself
- Results presented at the end
- Influences prioritization of backlog
- On the edge of chaos

30 Days

On the edge of chaos

Peter Müller – Fortgeschrittene Themen des Projektmanagement, September 08, 2009
Daily Scrum

- 15 min meeting
- Standing (shorter)
- Tool for measurement!

- Three Questions
  - What did you do since the last meeting?
  - What will you do until the next meeting?
  - What problems do you have?

Roles

- Makes decisions to increase return on investment
- Defines backlog item priorities
- Develops the product
- Focuses on leadership than management
- Controls what's done
- Bad guy

Product owner
Customer
Scrum Team
Scrum Master
Transparency

- Daily Scrums
  transparency in team
  peer pressure

- Sprint presentations
  transparency towards customer/owner

“Done”

- A very important word in Scrum
- Done = Analyzed, designed, coded, tested, committed, documented
- Done = Removed from the backlog
6. SCRUM

Burndown Chart

Scrum Philosophy

Transparency
Timeboxed
On the edge of chaos
Getting things done (DONE)
Burndown chart
7. Tools

Spreadsheet

- Applications
  - Estimation
  - Risk management
  - Planning and controlling for small projects (task lists)
  - Reporting
  - Documentation of assumptions, open issues, change requests, identified risks, etc.
- Benefits: computations, sorting, filtering, visualization
- Popular tools: MS Excel, OpenOffice Calc
### Project Management Tool

- **Applications**
  - Scheduling
  - Resource allocation
  - Controlling
  - Reporting
- **Benefits**: Elaborate computations, elaborate visualizations, support for various methods (e.g., Earned Value)
- **Drawbacks**: Complexity
- **Popular tools**: MS Project

### Collaboration Tool

- **Applications**
  - Coordination of (possibly distributed) teams
  - Integration of configuration management, bug database, etc.
  - Information dissemination via web
- **Benefits**: Transparency (tasks, bugs, etc.), Support for agile methods
- **Drawbacks**: Complexity
- **Popular tools**: Rational Team Concert (Jazz), Trac
Bug Tracking

- Applications
  - Manage list of all known bugs
  - Maintain history for each bug
  - Manage collaboration
- Benefits: Transparency, statistics
- Widely-used products: Bugzilla, Mantis, GNATS, Flyspray

Life-Cycle of a Bug Report

Que???

I think I know what’s wrong.

Oops, that’s in my code...
Bug Report Attributes

- **Date:** report, updates, close
- **Status:** open, closed, deleted, assigned
- **Short summary**
- **Detailed description**
- **Severity and priority**
- **Platform and version number**
- **Category:** GUI, installation, certain module
- **Reproduction:** description or test-case

<table>
<thead>
<tr>
<th>Request ID</th>
<th>Summary</th>
<th>Open Date</th>
<th>Priority</th>
<th>Assigned To</th>
<th>Submitted By</th>
</tr>
</thead>
<tbody>
<tr>
<td>145766</td>
<td>Startup Fails</td>
<td>2006-03-23 23:49</td>
<td>5</td>
<td>dimah</td>
<td>be_brg_peh</td>
</tr>
</tbody>
</table>

Bug Report Guidelines

- **One issue** per report
- Make sure it is **not reported** yet
- How to write **useful** bug report
  - Summary: short but quickly and uniquely identifies bug
  - Be specific: product, version, component, platform, OS
    include list of third party software used
  - Reproducible: provide least amount of steps
Configuration Management

- **Applications**
  - Manage source code and other resources
  - Manage versions
  - Manage collaboration

- **Benefits:**
  - Permit concurrent development
  - Store whole history
  - Back-up
  - Good anchor for quality assurance activities

- **Popular tools:** cvs, subversion (svn)

---

Automatic Code Inspection

- **Applications**
  - Enforce coding conventions
  - Check for common errors (beyond compiler and IDE)

- **Benefits:**
  - Reduces effort for manual code reviews
  - Large set of pre-defined rules
  - Rules can be adapted and extended
  - Can be integrated in IDE

- **Popular tools:** CheckStyle, PMD, FindBugs
Profiler

- Applications
  - Gather performance data for application
  - Identify bottlenecks
- Benefits:
  - Prepare code optimization
  - Optimize only those parts that are actually problematic
- Popular tools: Integrated in IDE (NetBeans, Visual Studio)
Feedback

- What did you like about the course?
- What should be improved?
- Did the course meet your expectations?

Appendix
A. Appendix

A.1 Definitions

A.2 References

Project Management

- Definition of Project Management (PM):
  Project Management is the application of knowledge, skills, tools, and techniques to project activities to meet project requirements.
What is an IT-Project?

- **Definition:**
  
  *An IT-project is a project to create a product or service, of which the usage of information technology is the decisive characteristic*
  
- **Examples**
  
  - The development of a software application is an IT-project (IT-based product)
  
  - The development of a car is not an IT-project, although information technology is involved substantially

---

Project Success

- **Definition:**
  
  *A project is successful if the specified results are delivered in the required quality and within the predetermined time and resource limits.*
  
- **Computer scientists tend to focus on scope and quality only**
  
  - The development of a technically perfect application is not a success if the cost exceeds the price clients are willing to pay
  
  - Excellent project results often are worthless if they come too late (temporary market windows, external deadlines)
Progressive Elaboration

Characteristics of a unique product or service must be **progressively elaborated**

- Continuing steadily by increments
- Worked out with care and detail

- During the project, characteristics are defined in more detail as the project team develops a better and more complete understanding of the product

---

Constraints

- **Definition:** *Constraints are factors that limit the project team’s options*
- A single project may contain cost, time, human resource, technical, and other constraints
- **Examples**
  - External deadlines (e.g., Y2K, Euro)
  - Fixed upper limits for budget
  - Dependencies on other projects, etc.
Baseline

- **Definition:**
  
  *The originally approved plan plus or minus approved changes.*

- Baselines are used to compare the actual performance and forecasts of the project with the original plan.

---

Deliverables

- **Definition:**
  
  *Any measurable, tangible, verifiable outcome, result, or item that must be produced to complete a project or part of a project.*

- **Examples**
  
  - An object-oriented design, described by a UML diagram
  - A project schedule as MS Project file
  - A user guide for a new application
  - Software, delivered as compiled binary
Work Breakdown Structure (WBS)

- Definition:
  A deliverable-oriented, hierarchical grouping of project elements that organizes and defines the total work scope of the project. Each descending level represents an increasingly detailed definition of the project.

Stakeholders

- Definition
  Individuals and organizations that are actively involved in the project, or whose interests may be positively or negatively affected as a result of project execution or project completion; they may also exert influence over the project and its results.

- Key stakeholders
  - Project manager
  - Customer
  - Performing organization
  - Project team members
  - Sponsor
Float

- **Definition:**
  
  The amount of time that an activity may be delayed from its early start without delaying the project finish date

- **Float = LF – EF = LS – ES**

- **Interpretation**
  
  - Float > 0: Time is available
  - Float = 0: Situation is critical
  - Float < 0: Project is behind

- **Sometimes called Total Float, Slack, or Total Slack**

---

Critical Path

- **Definition:**
  
  The series of activities that determines the duration of the project (the longest path through the network)

- **Sum of float on critical path is zero (or negative)**

- **Critical path is important**
  
  - To shorten project duration
  - To focus progress control
  - To identify schedule risks

- **There can be several critical paths in a project**
Milestones

- **Definition:**
  *A significant event in the project, usually completion of a major deliverable*

- Milestones have no effort or duration
- Milestones do not have resources

- Example: Painting completed

---

Risk

- **Definition:**
  *An uncertain event or condition that, if it occurs, has a positive or negative effect on a project objective*

- Risks have three components
  - A possible future event (uncertainty)
  - Probability of the occurrence of that event (likelihood)
  - Impact of that event (consequence)
Risk Management

Definition:
Systematic process of identifying, analyzing, and responding to project risk. It includes minimizing the consequences of adverse events to project objectives.
Literature: IT-Projects

- Zehnder, Carl August: Informatik-Projektentwicklung, vdf Hochschulverlag, 2003
- Jenny, Bruno: Projektmanagement in der Wirtschaftsinformatik, vdf Hochschulverlag, 2000
- Steinweg, Carl: Projektkompass Softwareentwicklung, Vieweg, 2002
- Frühauf, Karol: Software-Projektmanagement und -Qualitätssicherung, vdf Hochschulverlag, 2002
- Gaulke, Markus: Risikomanagement in IT-Projekten, Oldenbourg Verlag 2002

Literature: General Project Management

- Fleming, Quentin, Koppelman, Joel: Earned Value Project Management, Project Management Institute, 2000
- PMI – Project Management Institute, www.pmi.org
Literature: Agile Methods and SCRUM

- Control Chaos
  http://www.controlchaos.com
- Google Tech Talk by Ken Schwaber
  http://www.youtube.com/watch?v=IyNPeTn8fpo

Contact Information

- Peter Müller
  - Email: peter.mueller@inf.ethz.ch
  - Web page: www.pm.inf.ethz.ch
  - Course page:
    www.pm.inf.ethz.ch/teaching/as2009/IPM/index.html
    - Login: project
    - Password: management