CMMI: Project Monitoring and Control

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

The WHAT: Project Monitoring and Control, by Tobias Stoll
- SG 1: Monitor the Project Against the Plan
- SG 2: Manage Corrective Action to Closure

The HOW (part 1): industrial practices, by Dominik Schreiber

The HOW (part 2): real-life examples, by Dominik Schreiber

Project Monitoring and Control

TECHNISCHE
UNIVERSITÄT
DARMSTADT

[Dev10]

Project Monitoring and Control
SG 1: Monitor the Project Against the Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Goal

- identify **actual** progress and performance against plan
- **contrast** results with project plan

SG 1: Monitor the Project Against the Plan
SP 1.1: Monitor Project Planning Parameters

TECHNISCHE UNIVERSITÄT DARMSTADT

- measure **actual** project planning parameters
- identify **significant** deviations to the estimates in the plan
- record results

SubPractices

- Monitor progress against schedule
- Monitor the projects cost and expended effort
- Monitor the attributes of the work products and tasks
- Monitor resources provided and used
- Monitor the knowledge and skills of the project personnel
- Document the significant deviations in the project planning parameters

SG 1: Monitor the Project Against the Plan
SP 1.2: Monitor Commitments

TECHNISCHE UNIVERSITÄT DARMSTADT

- look at commitments
- compare them to identified commitments in the plan

SubPractices

- Regularly review commitments
- Identify commitments that have not been satisfied
- Document the results of the commitment reviews

SG 1: Monitor the Project Against the Plan
SP 1.3: Monitor Project Risks

TECHNISCHE UNIVERSITÄT DARMSTADT

- Monitor risks against those in the plan

SubPractices

- Periodically review the documentation of the risks
- Revise the documentation of the risks
- Communicate risk status to relevant stakeholders

SG 1: Monitor the Project Against the Plan
SP 1.4: Monitor Data Management

TECHNISCHE UNIVERSITÄT DARMSTADT

- check data management activities
- ensure compliance of data management plans

SubPractices

- Periodically review data management activities
- Identify and document significant issues and their impacts
- Document the results of data management activity reviews

SG 1: Monitor the Project Against the Plan
SP 1.5: Monitor Stakeholder Involvement

TECHNISCHE UNIVERSITÄT DARMSTADT

- detect involvement of identified stakeholders
- ensure that the right interactions occure

SubPractices

- Periodically review the status of stakeholder involvement
- Identify and document significant issues and their impacts
- Document the results of the stakeholder involvement status reviews

SG 1: Monitor the Project Against the Plan
SP 1.6: Conduct Progress Reviews

TECHNISCHE UNIVERSITÄT DARMSTADT

- project reviews to inform stakeholders
- may not be specified in project plan

SubPractices

- Regularly communicate status on assigned activities and work products to relevant stakeholders
- Review the results of collecting and analyzing measures for controlling the project
- Identify and document significant issues and deviations from the plan
- Document change requests and problems identified in any of the work products and processes
- Document the results of the reviews
- Track change requests and problem reports to closure

SG 1: Monitor the Project Against the Plan
SP 1.7: Conduct Milestone Reviews

TECHNISCHE UNIVERSITÄT DARMSTADT

- review of results at selected milestones
- typically formal and planned

SubPractices

- Conduct reviews at meaningful points in the projects schedule
- Review the commitments, plan, status, and risks of the project
- Identify and document significant issues and their impacts
- Document the results of the review, action items, and decisions
- Track action items to closure

Project Monitoring and Control
SG 2: Manage Corrective Action to Closure

TECHNISCHE UNIVERSITÄT DARMSTADT

Goal

- react with significant deviations from plan
- take corrective actions

SG 2: Manage Corrective Action to Closure
SP 2.1: Analyze Issues

TECHNISCHE UNIVERSITÄT DARMSTADT

-

SubPractices

- Gather issues for analysis
- Analyze issues to determine need for corrective action

SG 2: Manage Corrective Action to Closure
SP 2.2: Take Corrective Action

TECHNISCHE UNIVERSITÄT DARMSTADT

-

SubPractices

- Determine and document the appropriate actions needed to address the identified issues
- Review and get agreement with relevant stakeholders on the actions to be taken
- Negotiate changes to internal and external commitments

SG 2: Manage Corrective Action to Closure
SP 2.3: Manage Corrective Actions

TECHNISCHE UNIVERSITÄT DARMSTADT

-

SubPractices

- Monitor corrective actions for completion
- Analyze results of corrective actions to determine the effectiveness
- Determine and document appropriate actions to correct deviations

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

The WHAT: Project Monitoring and Control, by Tobias Stoll

The HOW (part 1): industrial practices, by Dominik Schreiber

- Scrum
- Extreme Programming
- methodology-independent representations of progress
- Earned-Value Analysis

The HOW (part 2): real-life examples, by Dominik Schreiber

industrial practices

TECHNISCHE UNIVERSITÄT DARMSTADT

[AB06]

industrial practices
Scrum - what it is

TECHNISCHE UNIVERSITÄT DARMSTADT

Overview

- **agile** software-engineering process
- **iterative**: thinking in *sprints*
- **slim**: 3 *roles*, 4 *artifacts*, small set of *rules*
- **communicative**: daily meetings, planning, reviews (but less paperwork)

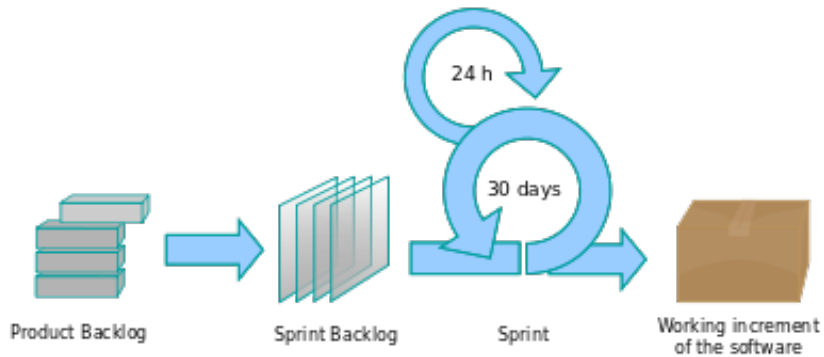


Abbildung : scrum process overview

TECHNISCHE UNIVERSITÄT DARMSTADT

Regular meetings

- **Sprint planning meeting** (part 1: whole team):
  - clean product backlog, prioritize entries
  - choose entries for next sprint
- **Sprint planning meeting** (part 2: developers):
  - convert entries to 1-day tasks ($\Rightarrow$ sprint backlog)
  - extract sprint-goal from entries
- **Sprint Review**:
  - present product to product owner, check sprint-goal
  - give feedback for last sprint, update product backlog
- **Sprint Retrospective**:
  - concrete improvements based on
  - feedback for the last sprint

industrial practices
Scrum

TECHNISCHE UNIVERSITÄT DARMSTADT

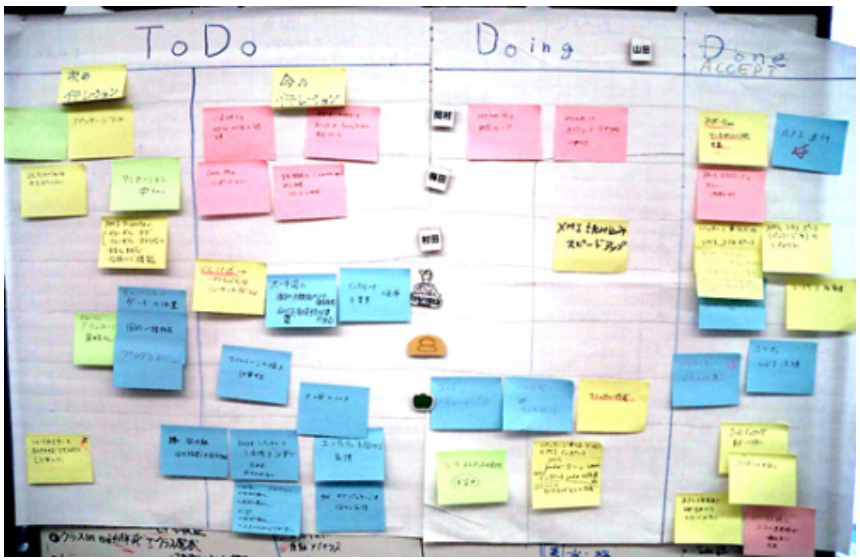


Abbildung : Scrum Taskboard

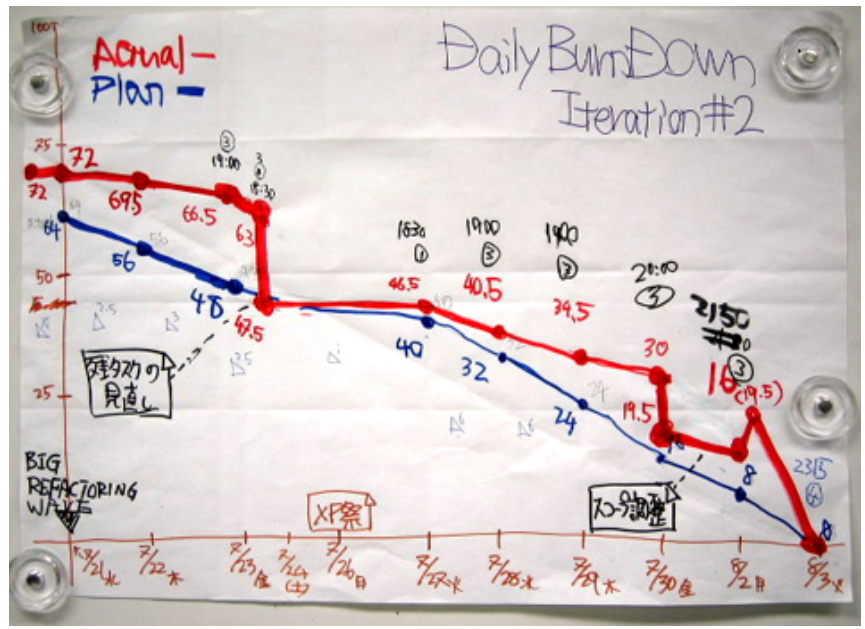


Abbildung : Scrum Burndown Chart

Extreme Programming - what it is

Overview

- **agile** software-engineering process
- strong **principles**: Pair Programming, Test-driven Development, Continuous Integration, ...

Extreme Programming - what it is

Overview

- **agile** software-engineering process
- strong **principles**: Pair Programming, Test-driven Development, Continuous Integration, ...

Differences to Scrum

- **iteration length**: week (XP) $\leftrightarrow$ month (Scrum)
- **change adaption**: always (XP) $\leftrightarrow$ not in current sprint (Scrum)
- **work order**: customer chooses (XP) $\leftrightarrow$ team chooses (Scrum)
- **engineering practices**: given (XP) $\leftrightarrow$ not given (Scrum)

industrial practices
Extreme Programming - how it supports PMC

TECHNISCHE UNIVERSITÄT DARMSTADT

through the engineering process

- **Planning Game**: release+iteration planning match results with plan constantly, split up in 3 phases:

1. *exploration phase*
   create user stories/split them into tasks
2. *commitment phase*
   commit to functionalities/assign tasks
3. *steering phase*
   adjust plan/perform tasks, match result to plan

- **Test-driven Development**: all productive code is written to make failing unit tests pass $\rightarrow$ unit tests describe the plan
- **Continuous Integration**: automated unit tests match every commit to the plan

it is the **combination** of the 12 principles that makes XP work

industrial practices
methodology-independent representations of progress

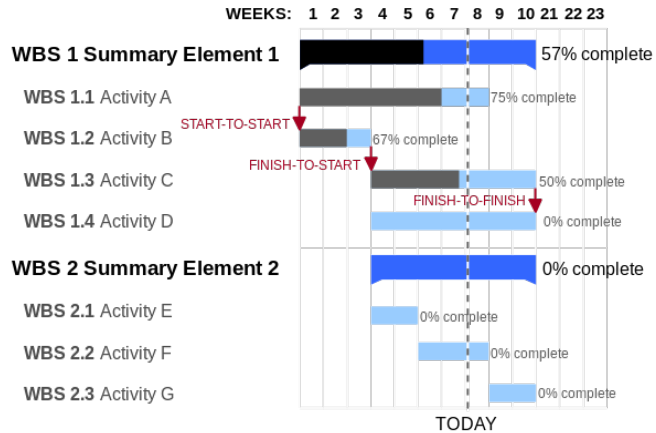
TECHNISCHE UNIVERSITÄT DARMSTADT

gantt chart

- illustrates project **schedule**
- project is broken down into elements
- each element has **start** and **end**
- progress of each element can be illustrated

network diagram

- shows **dependencies** in schedule
- same information as gantt chart, different representation



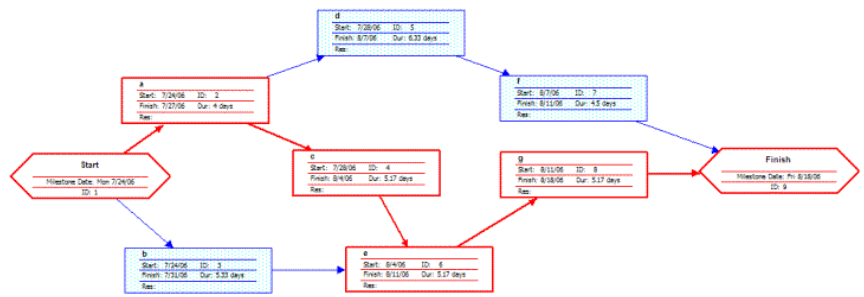


Abbildung : gantt chart, network diagram

industrial practices
Earned-Value Analysis - what it is

TECHNISCHE UNIVERSITÄT DARMSTADT

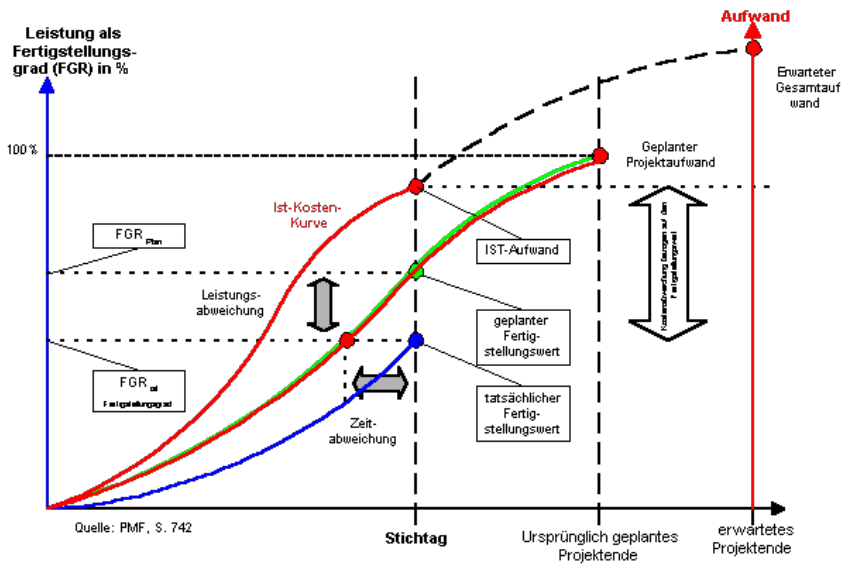


Abbildung : combined values in EVA

project controlling tool

- allows **monitoring** of *scope*, *schedule* and *cost* of a project
- allows **forecasts** for the evolving project
- not restricted to software engineering, works for **all kinds of projects**

industrial practices
Earned-Value Analysis - how it works

TECHNISCHE UNIVERSITÄT DARMSTADT

measured values

- **planned cost** (*PC*): pre-defined cost for the *whole project*
- **actual cost** (*AC)*: actual cost *until now*

metrics

- **earned value** (*EV*): actual *value* of the work until now

  simple metric: $EV = Budget \cdot \%Progress$

derived values

- **schedule variance**: $SV = EV - PC$
- **cost variance**: $CV = EV - AC$
- **schedule performance index**: $SPI = \frac{EV}{PC}$
- **cost performance index**: $CPI = \frac{EV}{AC}$

industrial practices
Earned-Value Analysis - example

TECHNISCHE UNIVERSITÄT DARMSTADT

your task

- list of **500** hand-written mail addresses
- insert this into **mailing list**
- you need **2h** for this, get **10€/h**
- **half time!** 1h left

measured values

- **PC** = $20€ \cdot \frac{1}{2} = 10€$
- **AC** = $10\frac{€}{1h} \cdot 1h = 10€$

metrics

- you were **fast**! Did 300 already!
- **EV** = $20€ \cdot \frac{300}{500} = 12€$

derived values

- **SV** = $12€ - 10€ = 2€$
- **CV** = $12€ - 10€ = 2€$
- **SPI** = $\frac{12€}{10€} = 1.2$
- **CPI** = $\frac{12€}{10€} = 1.2$

industrial practices
Earned-Value Analysis - example

TECHNISCHE UNIVERSITÄT DARMSTADT

your task

- list of **500** hand-written mail addresses
- insert this into **mailing list**
- you need **2h** for this, get **10€/h**
- **half time!** 1h left

measured values

- **PC** = $20€ \cdot \frac{1}{2} = 10€$
- **AC** = $10\frac{€}{1h} \cdot 1h = 10€$

metrics

- you were **fast**! Did 300 already!
- **EV** = $20€ \cdot \frac{300}{500} = 12€$

derived values

- **SV** = $12€ - 10€ = 2€$
- **CV** = $12€ - 10€ = 2€$
- **SPI** = $\frac{12€}{10€} = 1.2$
- **CPI** = $\frac{12€}{10€} = 1.2$

If you stay that fast, you'll finish **20min early**. Means: you'll get **€3,30** for free.

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

The WHAT: Project Monitoring and Control, by Tobias Stoll

The HOW (part 1): industrial practices, by Dominik Schreiber

The HOW (part 2): real-life examples, by Dominik Schreiber

- at openLearnWare
- at a major-client software-engineering company
- at dimetis GmbH

real-life examples
at openLearnWare - project overview

TECHNISCHE UNIVERSITÄT DARMSTADT

Project: material portal for students

- webservice for lecture material
- development started in spring 2010
- team of 2 full-time employees, 5 HiWis
- scrum-like project structure

Abbildung : tu-darmstadt.de/olw, 7.1.13

real-life examples
at openLearnWare - project structure

TECHNISCHE UNIVERSITÄT DARMSTADT

team members

- “Intellectual head” – like Scrum’s **product owner**, responsible for all “non-technical stuff”
- “Technical head” – like Scrum’s **scrum master**, responsible for all “technical stuff”
- 5 HiWis, working 8-20 hours a week – the **scrum team**

Abbildung : taskboard, december-sprint

real-life examples
at openLearnWare - project monitoring/control

TECHNISCHE UNIVERSITÄT DARMSTADT

process items

- weekly **scrum meeting** – about an hour, with all team members
- weekly **planning meeting** – about 2 hours, intellectual+technical head
- **taskboard** as a mirror of the redmine *ticket system*
- **tickets** as a *sprint backlog*
- current **QSL-Request** as *product backlog*
- **Jenkins** as *Continuous-Integration Server*

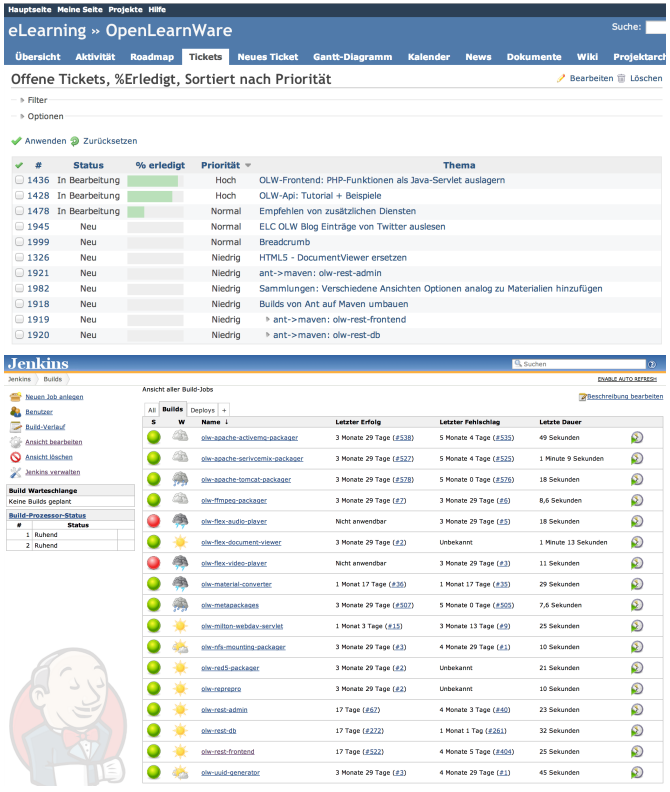


Abbildung : ticket system, ci-server

real-life examples
at openLearnWare - how the process evolved

TECHNISCHE UNIVERSITÄT DARMSTADT

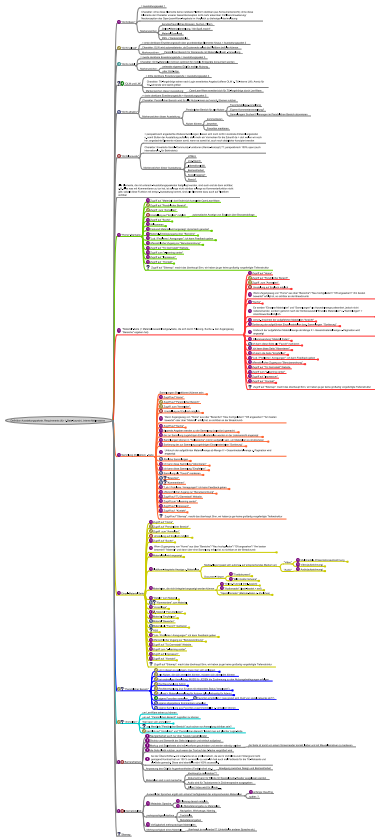


Abbildung : former product backlog

change as the only constant

- no current team member from the **founder team**
- began with **giant mind-maps** as product/sprint backlogs
- had 2-3 nearly **complete restarts**
- in the beginning: **no documentation** at all (except backlogs)

real-life examples
at dimetis GmbH

TECHNISCHE
UNIVERSITÄT
DARMSTADT

real-life examples
at a major-client software-engineering company

TECHNISCHE UNIVERSITÄT DARMSTADT

Project: Web-based software for the government

- up to 1.5 years **specification phase**
- teams of **<10 members**
- project head spends **1 day a week** with the customer

real-life examples
at a major-client software-engineering company

TECHNISCHE UNIVERSITÄT DARMSTADT

the process

- **specification phase**: before the project starts, it gets strongly *specified* (customer reviews this)
- **implementation**: according to system specification, features are implemented
- **internal reviews**: code is reviewed twice: once technical, once specialist
- **small releases**: working features are shipped to the customer early, loaded with unit tests/performance tests/specific tests
- **issue tracking**: customer show occuring bugs to developers or reports them in the issue tracker

Thank you for your attention!

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Thank you for your attention!

TECHNISCHE UNIVERSITÄT DARMSTADT

I want to hear at least **3 questions** from you.

Start now!

Bibliography

TECHNISCHE UNIVERSITÄT DARMSTADT

Julio Ariel Hurtado Alegria and M. Cecilia Bastarrica.
Implementing cmmi using a combination of agile methods.
CLEI Electron. J., 9(1), 2006.

Cmmi Development.
Cmmi® for development, version 1.3 cmmi-dev, v1.3.
Engineering, (November):482, 2010.

Extreme Programming
Principles

TECHNISCHE UNIVERSITÄT DARMSTADT

Fine scale feedback

- pair programming
- planning game
- test-driven development
- whole team

shared understanding

- coding standards
- collective code ownership
- simple design
- system metaphor

continuous process

- continuous integration
- refactoring/design improvement
- small releases

programmer welfare

- sustainable pace