Lecture 05: Examples of Process Models and Metrics

Contents & Goals

Last Lecture:
- procedure models (iterative, incremental, spiral, etc.),
- difference to process models,
- software metrics

This Lecture:
- Educational Objectives:
  - what are the constituting elements of "V-Modell XT"?
  - what does project types and tailoring mean in "V-Modell XT"?
  - how does "V-Modell XT" 'work'?
  - please explain this "V-Modell XT" building block
  - what are examples of agile process models? what are their principles?
  - describe XP, Scrum: roles, artefacts, activities?
  - is "V-Modell XT" and "agile" a contradiction?
  - what is the purpose of a process metric? What is CMMI, SPICE?
  - how are the levels of CMMI and SPICE defined?
- Content:
  - V-Modell XT
  - agile process models, XP, Scrum
  - process metrics CMMI/SPICE

From Procedure to Process Model

A process model may describe:
- organisation, responsibilities, roles;
- structure and properties of documents;
- methods to be used, e.g. to gather requirements or to check intermediate results;
- steps to be conducted during development, their sequential arrangement, their dependencies (the procedure model);
- project phases, milestones, testing criteria;
- notations and languages;
- tools to be used (in particular for project management).

Process models typically come with their own terminology (to maximise confusion?), e.g. what we call artefact is called product in V-Model terminology.

Light vs. Heavyweight Process Models

- You may hear about "light" and "heavyweight" process models.
- Sometimes, "heaviness" seems to be measured in number of rules...
- Sometimes, "heaviness" seems to be related to flexibility, adaptability during a process...
- "Light" sounds better than "heavy", so advocates of a certain process model tend to tag theirs "light" and all others "heavy".
- In the end, a process model is too "light" if it doesn't support you in doing things which are useful and necessary for your project;
- a process model is too "heavy" if it forces you to do things which are neither necessary nor useful for your project.

Thus following (Ludewig and Lichter, 2013), we will not try to assign the following process models to a "weight class".

Phase Models
The Phase Model

1. The project is planned by phases, delimited by well-defined milestones.
2. Each phase is assigned a time/cost budget.
3. Phases and milestones may be part of the development contract; partial payment when reaching milestones.
4. Roles, responsibilities, artefacts defined as needed.
5. By definition, there is no iteration of phases.
6. But activities may span multiple phases.
7. Not uncommon for small projects (few software people, small product size), small companies.

V-Modell XT

1. There are different V-shaped (in a minute) process models, we discuss the (German) "V-Modell".
2. "V-Modell": developed by company IABG in cooperation with the Federal Office for Defence Technology and Procurement ('Bundesministerium für Verteidigung'), released 1998.
3. (German) government as customer often requires usage of the V-Modell.

V-Modell XT: Project Types

1. V-Modell XT offers support for four different project types:
   - AG: project from the perspective of the customer (create call for bids, choose developer, accept product).
   - AN: project from the perspective of the developer (create offer, develop system, hand over system to customer).
   - AG/AN: customer and developer from same organisation.
   - PM: introduction and maintenance of a specific process model.

V-Modell XT: Terminology

1. Our course V-Modell XT explanation role ('Rolle') activity ('Aktivität') - step ('Arbeitsschritt') parts of activities artefact - product ('Produkt') - topic ('Thema') parts of products discipline ('Disziplin') a set of related products and activities phase project segment (?)('Projektabschnitt')
a discipline comprises one or more products,

a product may be external ('E') or initial ('I'), i.e. created always and exactly once (e.g. project plan),

a product may consist of topics,

a product may depend on other products,

an activity creates a product and belongs to a discipline,

an activity may consist of steps,

a step works on a topic,

a role may be responsible for a product or contribute,

each product has at most one responsible role.
Recall the idea of the "V shape": requirements fixed—acceptance—system specified—system delivered—architecture designed—modules designed—system realised—modules integrated—system realised—verification & validation.

V-Modell XT mainly supports three strategies to develop a system, i.e., principal sequences between decision points:

- incremental,
- component based,
- prototypical.

Advantages:

- Certain management related building blocks are part of each project, thus they may receive increased attention of management and developers.
- Publicly available, can be used free of license costs.
- Very generic, support for tailoring.
- Comprehensive, low risk of forgetting things.

Disadvantages:

- Comprehensive, tries to cover everything; tailoring is supported, but may need high effort.
- Tailoring is necessary, otherwise a huge amount of useless documents is created.
- Description/presentation leaves room for improvement.

Needs to prove in practice, in particular in small/medium sized enterprises (SME).

Rational Unified Process (RUP) exists. In contrast to "V-Modell XT", a commercial product.
"Agile denoting 'the quality of being agile; readiness for motion; nimbleness, activity, dexterity in motion' software development methods are attempting to offer an answer to the eager business community asking for lighter weight along with faster and nimbler software development processes. This is especially the case with the rapidly growing and volatile Internet software industry as well as for the emerging mobile application environment." (Abrahamsson et al., 2002)

The Agile Manifesto (2001):

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Agile Principles

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Similarities of Agiles Process Models

- iterative; cycles of a few weeks, at most three months,
- require work in small groups (6–8 people),
- dislike the idea of large, comprehensive documentation (radical or with restrictions),
- consider the customer important; recommend or request customer's presence in the project,
- dislike dogmatic rules.

Extreme Programming (XP) (Beck, 1999)

XP values:
- simplicity, feedback, communication, courage, respect.

XP practices:
- management
- integral team (including customer)
- planning game (→ Delphi method)
- short release cycles
- stand-up meetings
- assess in hindsight
- team
- joint responsibility for the code
- coding conventions
- acceptable workload
- central metaphor
- continuous integration
- programming
- test driven development
- refactoring
- simple design
- pair programming...
Scrum

• first published 1995 (Schwaber, 1995), based on ideas of Takeuchi and Nonaka

• inspired by Rugby: get the ball in a scrum, then sprint to score

• role-based; iterative and incremental; in contrast to XP no techniques proposed/required

Three roles

• product owner:
  • representative of customer,
  • maintains requirements in the product backlog,
  • plans and decides which requirement(s) to realise in next sprint,
  • (passive) participant of daily scrum,
  • assesses results of sprints

• scrum team:
  • members capable of developing autonomously,
  • decides how and how many requirements to realise in next sprint,
  • distribution of tasks self-organised, team decides who does what when,
  • environment needs to support communication and cooperation, e.g. by spatial locality

• scrum master:
  • helps to conduct scrum the right way,
  • looks for adherence to process and rules,
  • ensures that the team is not disturbed from outside,
  • moderates daily scrum,
  • responsible for keeping product backlog up-to-date,
  • should be able to assess techniques and approaches

Product Backlog

• composes all requirements to be realised,
• priority and effort estimation for requirements,
• collects tasks to be conducted,
• maintained by product owner

Release Plan

• based on initial version of product backlog,
• how many sprints, which major requirements in which sprint

Release Burndown Report

• see sprint-burndown report

Sprint Backlog

• requirements to be realised in next sprint, taken from product backlog,
• more precise estimations,
• daily update (tasks done, new tasks, newestimations)

Sprint Burndown Report

• completed/open tasks from sprint backlog,
• should decrease linearly, otherwise remove tasks from sprint backlog

Sprint Report

• which requirements have (not) been realised in last sprint,
• description of obstacles/problems during sprint

Scrum Process

• daily scrum:
  • daily meeting, 15 min.
  • discuss progress, synchronise day plan, discuss and document new obstacles
  • team members, scrum master, product owner (if possible)

• sprint: at most 30 days, usually shorter (initially longer)

• sprint review: assess amount and quality of realisations; product owner accepts results

• sprint retrospective: assess how well the scrum process was implemented; identify actions for improvement (if necessary)

Scrum: Discussion

• has been used in many projects, experience in majority positive
• team size bigger 7–10 may need scrum of scrums
• competent product owner necessary for success
• success depends on motivation, competence, and communication skills of team members
• team members responsible for planning, and for adhering to process and rules, thus intensive learning and experience necessary
• can (as other process models) be combined with techniques from XP

Process Metrics
For material goods: quality of the production process influences product quality. Idea: specify abstract criteria (metrics) to determine good production processes (e.g., to choose manufacturer). Again: a good process does not stop us from creating bad products, but (the hope is, that) it is less likely, i.e. there is a correlation:

\[
\begin{array}{c|c|c}
\text{Process quality} & \text{High} & \text{Low} \\
\hline
\text{Product quality} & \text{High} & \text{Low} \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{False Positive} & \times & \times \\
\hline
\text{True Positive} & \times & \times \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{False Negative} & \times & \times \\
\hline
\text{True Negative} & \times & \times \\
\end{array}
\]

Industry in general (production!):
ISO 9001, ISO/TS 16949 (automotive), ...

Software industry (development!):
CMM(I), SPICE

1991: Capability Maturity Model (CMM), DoD/SEI/CMU; superseded by
1997: Capability Maturity Model Integration (CMMI) (Team, 2010);
constellations: CMMI-DEV (development), CMMI-ACQ (acquisition), CMMI-SRV (service)

Goals:
• applicable to all organisations which develop software,
• make strengths and weaknesses of the real process visible, to point out ways for improvement,
• neutral wrt. technology employed in project,
• levels: higher levels have lower levels as premise,
• be consistent with ISO 15504 (SPICE)

Assumptions:
• better defined, described, and planned processes have higher maturity,
• higher maturity levels require statistical control to support continuous improvement,
• higher maturity level yields:
  • better time/cost/quality prediction;
  • lower risk to miss project goals;
  • higher quality of products.

### CMMI Levels

<table>
<thead>
<tr>
<th>Level</th>
<th>Level Name</th>
<th>Process Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initial</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Managed</td>
<td>REQM, PP, PMC, MA, PPQA, CM, SAM</td>
</tr>
<tr>
<td>3</td>
<td>Defined</td>
<td>+ RD, TS, PI, VER, VAL, OPF, OPD, OT, IPM, RSKM, DAR</td>
</tr>
<tr>
<td>4</td>
<td>Quantitatively Managed</td>
<td>+ OPP, QPM</td>
</tr>
<tr>
<td>5</td>
<td>Optimising</td>
<td>+ OID, CAR</td>
</tr>
</tbody>
</table>

Level 1 – the process is not consciously designed, just evolved (need not be bad!)
CMMI: Discussion

CMMI levels are chosen somewhat arbitrarily; "why is an area in level X, Y, Z?"
• Example: to reach CMMI level 2, an organisation has to reach GG.1, GG.2, and in that sense CMMI may hinder specific practices and specific goals.

CMMI: General/Specific Goals and Practices

CMMI: Statistics

CMMI: Processes

CMMI: Areas

CMMI: Levels

CMMI: Specific Practices

CMMI: General Goals

CMMI: Areas

CMMI: Levels

CMMI: Specific Practices

CMMI: General Goals
SPICE / ISO 15504 – 05 – 2015

• Software Process Improvement and Capability Determination

• Ideas similar to CMM(I): maturity levels, assessment, certificates

• European development, standardised in ISO/IEC 15504 (2003)

• Maturity levels: 0 (incomplete), ... , 5 (optimizing); SPICE 0 corresponds to CMMI 1

• Provides “process reference models” (in particular specific ones for automotive, aerospace, etc.)

References


