Softwaretechnik / Software-Engineering

Lecture 05: Examples of & Metrics for Process Models

2015-05-11

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Contents & Goals

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

This Lecture:
- **Educational Objectives:** Capabilities for following tasks/questions.
  - 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
Process Models
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.

Process models are legion; we will take a closer look onto:

- **Phases**, **V-Model XT**, (Rational) **Unified Process**, **Agile (XP, Scrum)**
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 project is planned by **phases**, delimited by well-defined **milestones**.

Each phase is assigned a time/cost budget.

Phases and milestones may be part of the development contract; partial payment when reaching milestones.

Roles, responsibilities, artefacts defined **as needed**.

By definition, there is **no iteration of phases**.

But **activities may span multiple phases**.

Not uncommon for small projects (few software people, small product size), small companies.
V-Modell XT
Teil 1: Grundlagen des V-Modells

V-Modell® XT
There are different **V-shaped** (in a minute) **process models**, we discuss the (German) "V-Modell".

**"V-Modell"**: developed by company IABG in cooperation with the Federal Office for Defence Technology and Procurement (‘Bundesministerium für Verteidigung’), released 1998

(German) government as customer often **requires** usage of the V-Modell

2012: **"V-Modell XT"** Version 1.4 (Extreme Tailoring) (V-Modell XT, 2006)
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 or improvement of a process model
- **project type variants**: one/more customer; development/improvement/migration; maintenance
<table>
<thead>
<tr>
<th>our course</th>
<th>V-Modell XT</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>role (‘Rolle’)</td>
<td>✓</td>
</tr>
<tr>
<td>activity</td>
<td>activity (‘Aktivität’)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>step (‘Arbeitsschritt’)</td>
<td>parts of activities</td>
</tr>
<tr>
<td>artefact</td>
<td>product (‘Produkt’)</td>
<td>!</td>
</tr>
<tr>
<td></td>
<td>topic (‘Thema’)</td>
<td>parts of products</td>
</tr>
<tr>
<td></td>
<td>discipline (‘Disziplin’)</td>
<td>a set of related products and activities</td>
</tr>
<tr>
<td>phase</td>
<td>project segment (?) (‘Projektabschnitt’)</td>
<td>'-'</td>
</tr>
</tbody>
</table>
V-Modell XT: Decision Points

Projekt genehmigt → Projekt definiert → Anforderungen festgelegt → Projekt ausgeschrieben → Angebot abgegeben → Projekt beauftragt → Projekt ausgeschrieben → Anforderungen festgelegt → Angebot abgegeben → Projekt beauftragt → Projekt abgeschlossen

Legende:
- Alle V-Modell-Projekte
- Organisationsspezifisches Vorgehensmodell
- AG/AN-Schnittstelle
- Systementwicklung

Vorgehensmodell analysiert → Verbesserung Vorgehensmodell konzipiert → Verbesserung Vorgehensmodell realisiert

Projektdurchführungsstrategie

1..* legt Reihenfolge fest → 1..* Entscheidungspunkt

Genehmigung / Order / seq. 

decided / planned

needs one or more

Product

[im Zustand „fertig gestellt“]
V-Modell XT: The V-World (naja...)
Project Roles:

Anwender

Projektleiter

SW-Entwickler

Prüfer

Organisation Roles:
V-Modell XT: Roles (a lot!)

Project Roles:

Änderungssteuerungsgruppe (Change Control Board), Änderungsverantwortlicher, Anforderungsanalytiker (AG), Anforderungsanalytiker (AN), **Anwender**, Assessor, Ausschreibungsverantwortlicher, Datenschutzverantwortlicher, Ergonomieverantwortlicher, Funktionssicherheitsverantwortlicher, HW-Architekt, HW-Entwickler, Informationssicherheitsverantwortlicher, KM-Administrator, KM-Verantwortlicher, Lenkungsausschuss, Logistikentwickler, Logistikverantwortlicher, Projektkaufmann, **Projektleiter**, Projektmanager, Prozessingenieur, **Prüfer**, QS-Verantwortlicher, SW-Architekt, **SW-Entwickler**, Systemarchitekt, Systemintegrator, Technischer Autor, Trainer

Organisation Roles:

Ak quisiteur, Datenschutzbeauftragter (Organisation), Einkäufer, IT-Sicherheitsbeauftragter (Organisation), Qualitätsmanager
V-Modell XT: Disciplines and Products (even more!)
V-Modell XT: Disciplines and Products (even more!)
V-Modell XT: Activities (as many?!)

**Projekt**

- Planung und Steuerung
  - Projektaufbauentscheidung herbeiführen
  - Projektinhalt definiert
  - Projektmanagement-Infrastruktur einrichten
  - Schutzung durchführen
  - Risiken managen
  - Projekt planen
  - Arbeitsauftrag vergeben
  - Kaufmännische Projektkalibrierung durchführen

- Berichtswesen
  - Besprechung durchführen
  - Projektbericht erstellen
  - Materialien erstellen
  - Meeting bereiten und auswerten
  - Projektstatusbericht erstellen
  - QM-bericht erstellen
  - Projekt abschließen

- Konfigurations- und Änderungsmanagement
  - Problemebeobachtung/Anderungsantrag erstellen
  - Problemebeobachtung/Anderungsantrag bewerten
  - Änderungen entscheiden
  - Änderungsautokritik führen
  - Produktbibliothek verwaltbar
  - Produktconfiguration verwaltbar

**Entwicklung**

- Anforderungen und Analysen
  - Anforderungen analysieren
  - Anforderungen festlegen
  - Sicherheitsanalyse durchführen und bewerten
  - Informationssicherheitskonzept erstellen
  - Datenschutzkonzept erstellen
  - Anforderungen festlegen
  - Anforderungsbewertung erstellen
  - Systemanalyse erstellen
  - Marktanalyse für Fertigprodukte durchführen
  - Make-or-Buy-Entscheidung durchführen
  - Lizenziertes Gesamtpaket erstellen
  - Lizenziertes Gesamtpaket bewerten

- Systemelemente
  - Zum System integrieren
  - Zum Untersegment integrieren
  - Externe Entität übernehmen
  - Externe Entität integrieren
  - SWI-Modul integrieren
  - SW-Modul integrieren
  - Externe SW-Modul übernehmen
  - Externe SW-Modul integrieren

- Systementwurf
  - Systemarchitektur erstellen
  - Unterstützungssystemarchitektur erstellen
  - SW-Systemarchitektur erstellen
  - Datenbankkonzept erstellen
  - Implementierungs- und Pflichtenrichtlinien erstellen
  - SW-Systemarchitektur erstellen
  - SW-Modul integrieren
  - SW-Modul übernehmen
  - Externe SW-Modul übernehmen
  - Externe SW-Modul integrieren

- Logistik Konzepte
  - Logistik-Konzepte erstellen
  - Logistik-Konzepte übernehmen
  - Logistik-Konzepte integrieren

- Logistik Systeme
  - Nutzungsallotierung erstellen
  - Logistik-Konzepte erstellen
  - Logistik-Konzepte übernehmen

- System Spezifikationen
  - System Spezifikationen erstellen
  - SWI-Modul Spezifikationen erstellen
  - Externe SWI-Modul-Spezifikationen erstellen

**Organisation**

- Prozessverbesserung
  - Vorgangsmuster bewerten
  - Verbesserung eines Vorgangsmusters konzipieren
  - Organisationsspezifisches Vorgangsmuster erstellen, definieren und plazieren
V-Modell XT: Activities (as many?!)
- a discipline comprises one or more product
- 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
SW-Development (‘SW-Entwicklung’)
Product States

- in Bearbeitung
- vorgelegt
- fertig gestellt

[[Erste Version des Produkts wird erstellt]]
[[Keine Prüfung durch eigenständige Qualitätssicherung notwendig UND Eigenprüfung erfolgreich]]
[[Prüfung durch eigenständige Qualitätssicherung notwendig UND Eigenprüfung erfolgreich]]
[[Prüfung durch eigenständige Qualitätssicherung nicht erfolgreich]]
[[Produkt wird erneut bearbeitet]]
Recall the idea of the "V shape":

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

- incremental,
- component based,
- prototypical.
V-Modell XT: Development Strategies

- **incremental**
- **component based**
- **prototypical**
Advantages:

- certain management related building block 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
Rational Unified Process (RUP)

**Exists.**

- in contrast to “V-Modell XT”, a commercial product
Agile Process Models
“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 customers 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.

(Ludewig and Lichter, 2013)
Extreme Programming (XP)
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

![Diagram showing relationships between XP values and practices]
Scrum
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
Scrum Documents

• **product backlog**
  • comprises 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 spring, taken from product backlog,
  • more precise estimations,
  • daily update (tasks done, new tasks, new estimations)

• **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
Assessment and Improvement of the Process

- 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:

  ![Process Quality Matrix](image)

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

- Software industry (**development!**): **CMM(I), SPICE**
CMMI® for Development, Version 1.3

CMMI-DEV, V1.3

CMMI Product Team

Improving processes for developing better products and services

November 2010

TECHNICAL REPORT

CMUSEI-2010-TR-033
ESC-TR-2010-033

Software Engineering Process Management Program

http://www.sei.cmu.edu

Carnegie Mellon
• 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>
### 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>

- **initial** – the process is not consciously designed, just evolved (need not be bad!)
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>

- **managed** (formerly: repeatable) – important areas of software development organised and prescribed to responsible people; each project may have own process
- **Areas**: requirements management (REQM), project planning (PP), project monitoring and control (PMC), measurement and analysis (MA), Process and Product Quality Assurance (PPQA), configuration management (CM), supplier agreement management (SAM)
## 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>

- **defined** — all projects of an organisation follow a unified scheme; standard process is defined, documented, and used; tailoring for projects.
- **Areas**: requirements development (RD), technical solution (TS), product integration (PI), verification (VER), validation (VAL), organisational process focus (OPF), organisational process definition (OPD), organisational training (OT), integrated project management (IPM), risk management (RSKM), decision analysis and resolution (DAR)
### 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>

- **quantitatively managed** – unified metrics enable people to detect problems early and take countermeasures.
- **Areas**: organisational process performance (OPP), quantitative project management (QPM)
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>

- **optimising** – errors and problems are analysed systematically, to avoid them in the future; process organisation/techniques change accordingly
- **Areas**: organisational innovation and deployment (OID), causal analysis and resolution (CAR)
CMMI General/Specific Goals and Practices

- CMMI certificates can be obtained via a so-called **appraisal**
- there are three levels of review methods A, B, C; A most thorough (and expensive)
- a certificate authority checks, to what amount **generic goals** GG.1, . . . , GG.3 with their **generic practices** are reached.
  **Example:** GG.2 (for level 2) includes
  - GG 2.1: create strategy for planning and installation of process
  - GG 2.2: plan the process
  - GG 2.3: allocate resources
  - . . .

- each area, like RD, has **specific goals** and **specific practices**, sometimes per level
  **Example:** RD (requirements development) includes
  - SG 1: develop customer requirements
  - SG 2: develop product requirements
  - SG 3: analyse and validate requirements

- **that is**, to reach CMMI level 2, an organisation has to reach GG.1, GG.2, and in particular for area RD SG 1 and SG 2.
CMMI Statistics

Statistics on achieved CMMI maturity levels

- **Note**: appearance in the statistics is voluntary.
CMMI: Discussion

- in CMMI, e.g. area RD requires **that** requirements are analysed, but does not state **how** — there are examples, but no particular techniques or approaches
- CMMI as such **is not** a process model in the sense of the course

- CMMI certificate is **required** by certain (U.S) government customers; may guide selection of sub-contractors (a certificate at least proves that they think about their process)
- CMMI can serve as an **inspiration** for important aspects of process models wrt. product quality

**Criticism:**
- CMM(I) assumptions are based on experience in specific projects; may not be present for all kinds of software,
- CMMI certification applies to one particular state of process management; changed processes may require new (expensive) appraisal, in this sense CMMI may hinder innovation,
- CMMI levels are chosen somewhat arbitrarily; “why is an area in level $N$ and not already in level $N - 1$?”
**SPICE / ISO 15504**

- **Software P**rocess **I**mprovement and **C**apability **D**etermination
- ideas similar to CMM(I): maturity levels, assessment, certificates
- maturity levels: 0 (incomplete), . . . , 5 (optimizing); SPICE 0 corresponds to CMMI 1
- provides “process reference models” (in particular specific ones for automotive, aerospace, etc.)
- Literature: (Hörmann et al., 2006)
References


