Topic Area Project Management: Content

VL 2
- **Software Metrics**
  - Properties of Metrics
  - Scales
  - Examples

VL 3
- **Cost Estimation**
  - "(Software) Economics in a Nutshell"
  - Expert’s Estimation
  - Algorithmic Estimation

VL 4
- **Project Management**
  - Project
  - Process and Process Modelling

VL 5
- **Process Metrics**
  - CMMI, Spice
Describing Software Development Processes

Over time, the following notions proved useful to describe and model (→ in a minute) software development processes:

- **role** – has responsibilities and rights, needs skills and capabilities.
  In particular: has responsibility for **artefacts**, participates in **activities**.

- **artefact** – all documents, evaluation protocols, software modules, etc., all products emerging during a development process.
  Is processed by **activities**, may have **state**.

- **activity** – any processing of artefacts, manually or automatic; solves tasks.
  Depends on **artefacts**, creates/modifies **artefacts**.

- **decision point** – special case of activity: a decision is made based on **artefacts** (in a certain state), creates a **decision artefacts**.
  Delimits phases, may correspond to milestone.
From Building Blocks to Process (And Back)
Content

• Procedure and Process Models
  • Procedure Model Examples
    • The (in)famous Waterfall model
    • The famous Spiral model
  • Procedure classification
    • linear / non-linear
    • prototyping
    • evolutionary, iterative, incremental
  • From Procedure to Process Models
  • Process Model Examples
    • Phase Model
    • V-Modell XT
    • Agile
      • Extreme Programming
      • Scrum
  • Process Metrics
    • CMMI, Spice
Process vs. Procedure Models
(Ludewig and Lichter, 2013) propose to distinguish: **process model** and **procedure model**.

- **A Process model** (‘Prozessmodell’) comprises
  
  (i) **Procedure model** (‘Vorgehensmodell’)
      e.g., “waterfall model” (70s/80s).
  
  (ii) **Organisational structure** – comprising requirements on
       - project management and responsibilities,
       - quality assurance,
       - documentation, document structure,
       - revision control.

      e.g., V-Modell, RUP, XP (90s/00s).

- In the literature, **process model** and **procedure model** are often used as synonyms; there is not universally agreed distinction.
Procedure Models
— Waterfall —
The (In)famous Waterfall Model (Rosove, 1967)

Waterfall or Document-Model—Software development is seen as a sequence of activities coupled by (partial) results (documents). These activities can be conducted concurrently or iteratively.

Apart from that, the sequence of activities is fixed as (basically) analyse, specify, design, code, test, install, maintain. 

Ludewig & Lichter (2013)
The (In)famous Waterfall Model (Rosove, 1967)

Waterfall or Document-Model—Software development is seen as a sequence of activities coupled by (partial) results (documents). These activities can be conducted concurrently or iteratively.

Apart from that, the sequence of activities is fixed as (basically) analyse, specify, design, code, test, install, maintain. — Ludewig & Lichter (2013)
Waterfall or Document-Model— Software development is seen as a sequence of activities coupled by (partial) results (documents). These activities can be conducted concurrently or iteratively. Apart from that, the sequence of activities is fixed as (basically) analyse, specify, design, code, test, install, maintain. 

Ludewig & Lichter (2013)
Procedure Models

— Spiral —
Recall:

Quick Excursion: Risk and Riskvalue

**Risk** – a problem, which did not occur yet, but on occurrence threatens important project goals or results. Whether it will occur, cannot be surely predicted.

Ludewig & Lichter (2013)

\[ \text{riskvalue} = p \cdot K \]

- \( p \): probability of problem occurrence.
- \( K \): cost in case of problem occurrence.

- **Avionics** requires: “Average Probability per Flight Hour for Catastrophic Failure Conditions of \( 10^{-9} \) or ‘Extremely Improbable’” (AC 25.1309-1).
- “problems with \( p = 0.5 \) are not risks, but environment conditions to be dealt with”
The Spiral Model (Boehm, 1988)

**Note:** risks can have various forms and counter-measures, e.g.,
- open technical questions (→ prototype?),
- lead developer about to leave the company (→ invest in documentation?),
- changed market situation (→ adapt appropriate features?),
- ...


Repeat until end of project (successful completion or failure):

(i) **determine** the set $R$ of risks which are **threatening** the project;
    if $R = \emptyset$, the project is successfully completed

(ii) **assign** each risk $r \in R$ a **risk value** $v(r)$

(iii) for the risk $r_0$ with the **highest risk value**, $r_0 = \max\{v(r) \mid r \in R\}$,
    find a way to **eliminate** this risk, and go this way:
    if there is no way to eliminate the risk, stop with project failure

**Advantages:**
- We know early if the project goal is unreachable.
- Knowing that the biggest risks are eliminated gives a good feeling.
Wait, Where’s the Spiral?

A concrete process using the Spiral Model could look as follows:

- Investigate goals, alternatives, side conditions
- Conduct risk analysis,
- Develop and test the next product part,
- Plan the next phase,
Procedure and Process Models

Procedure Model Examples
- The (in)famous Waterfall model
- The famous Spiral model

Procedure classification
- linear / non-linear
- prototyping
- evolutionary, iterative, incremental

From Procedure to Process Models
Process Model Examples
- Phase Model
- V-Modell XT
- Agile
  - Extreme Programming
  - Scrum

Process Metrics
- CMMI, Spice
Procedure Model Classification
Procedure Model Classification
— Linear vs. Non-Linear —
Linear vs. Non-Linear Procedure Models

- **linear**: the strict Waterfall Model (no feedback)
- **non-linear**: basically everything else (with feedback between activities)
Procedure Model Classification
— By Treatment of Artefacts —
Classification By Treatment of (Software) Artefacts

- **Prototyping:**
**Prototype** – A preliminary type, form, or instance of a system that serves as a model for later stages or for the final, complete version of the system.  
IEEE 610.12 (1990)

**Prototyping** – A hardware and software development technique in which a preliminary version of part or all of the hardware or software is developed to permit user feedback, determine feasibility, or investigate timing or other issues in support of the development process.  
IEEE 610.12 (1990)

**Rapid Prototyping** – A type of prototyping in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analysis in support of the development process.  
IEEE 610.12 (1990)

Kinds of prototypes, distinguished by…

- **usage**: demonstration prototype, functional prototype, lab sample, pilot system, etc.
- **supported activity**: explorative prot.: support analysis; experimental prot.: support design; evolutionary prot.: → evolutionary procedure
Classification By Treatment of (Software) Artefacts

- **Prototyping:**

- **Evolutionary Development:**

- **Iterative Development:**
**Evolutionary and Iterative Development**

**Evolutionary software development** – an approach which includes evolutions of the developed software under the influence of practical/field testing.

New and changed requirements are considered by developing the software in **sequential steps of evolution**.

Ludewig & Lichter (2013), flw. (Züllighoven, 2005)

---

**Iterative software development** – software is developed in **multiple iterative steps**, all of them planned and controlled.

Goal: each iterative step, beginning with the second, corrects and improves the existing system based on defects detected during usage.

Each iterative steps includes the characteristic activities **analyse, design, code, test**.

Ludewig & Lichter (2013)
Classification By Treatment of (Software) Artefacts

- **Prototyping:**

- **Evolutionary Development:**

- **Iterative Development:**

- **Incremental Development:**
Incremental Development

**incremental software development** – The total extension of a system under development remains open; it is realised in **stages of expansion**. The first stage is the **core system**.

Each stage of expansion extends the existing system and is subject to a separate project. Providing a new stage of expansion typically includes (as with iterative development) an improvement of the old components.

*Ludewig & Lichter (2013)*

**Note**: (to maximise confusion) IEEE calls our “iterative” incremental:

**incremental development** – A software development technique in which requirements definition, design, implementation, and testing occur in an overlapping, iterative (rather than sequential) manner, resulting in incremental completion of the overall software product. *IEEE 610.12 (1990)*

**One difference (in our definitions):**

- **iterative**: steps towards fixed goal,
- **incremental**: goal extended for each step; next step goals may already be planned.

**Examples**: operating system releases, short time-to-market (→ continuous integration).
Classification By Treatment of (Software) Artefacts

- **Prototyping:**

- **Evolutionary Development:**

- **Iterative Development:**

- **Incremental Development:**

- **Staircase:** pipelined incremental
**Content**

- **Procedure and Process Models**
  - **Procedure** Model Examples
    - The (in)famous Waterfall model
    - The famous Spiral model
    - Procedure classification
      - linear / non-linear
      - prototyping
      - evolutionary, iterative, incremental
  - From Procedure to Process Models
  - **Process** Model Examples
    - Phase Model
    - V-Modell XT
    - Agile
      - Extreme Programming
      - Scrum
  - **Process Metrics**
    - CMMI, Spice
Process Models
Process vs. Procedure Model

(Ludewig and Lichter, 2013) propose to distinguish: process model and procedure model.

- A Process model (‘Prozessmodell’) comprises
  
  (i) Procedure model (‘Vorgehensmodell’)
      e.g., “waterfall model” (70s/80s).
  
  (ii) Organisational structure – comprising requirements on
       - project management and responsibilities,
       - quality assurance,
       - documentation, document structure,
       - revision control.

      e.g., V-Modell, RUP, XP (90s/00s).

- In the literature, process model and procedure model are often used as synonyms; there is not universally agreed distinction.
A **process model** may describe:

- **steps** to be conducted during development, their sequential arrangement, their dependencies (the **procedure model**)
- **organisation**, responsibilities, roles
- structure and properties of **documents**
- **methods** to be used, e.g., for gathering requirements or checking intermediate results
- 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.
Content

- Procedure and Process Models
  - Procedure Model Examples
    - The (in)famous Waterfall model
    - The famous Spiral model
    - Procedure classification
      - linear / non-linear
      - prototyping
      - evolutionary, iterative, incremental
  - From Procedure to Process Models
  - Process Model Examples
    - Phase Model
    - V-Modell XT
    - Agile
      - Extreme Programming
      - Scrum
  - Process Metrics
    - CMMI, Spice
Process Models
— Phase Models —
The Phase Model

- 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** (be active during) **multiple phases**.

- Not uncommon for small projects (few software people, small product size), and small companies.
Process Models
— V-Model XT —
Teil 1: Grundlagen des V-Modells
There are different “V-shaped” 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

a role may be responsible for a product or contribute

each product has at most one responsible role

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

a product may depend on other products

an activity creates a product and belongs to a discipline

a step works on a topic

a product may consist of topics

an activity may consist of steps

<table>
<thead>
<tr>
<th>our course</th>
<th>V-Modell XT</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>role</td>
<td>('Rolle')</td>
</tr>
<tr>
<td>activity</td>
<td>activity</td>
<td>('Aktivität')</td>
</tr>
<tr>
<td></td>
<td>step</td>
<td>('Arbeitsschritt')</td>
</tr>
<tr>
<td>artefact</td>
<td>product</td>
<td>('Produkt')</td>
</tr>
<tr>
<td></td>
<td>topic</td>
<td>('Thema')</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>our course</th>
<th>V-Modell XT</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>discipline</td>
<td>('Disziplin')</td>
</tr>
<tr>
<td>phase</td>
<td>project segment (?)</td>
<td>('Projektabschnitt')</td>
</tr>
</tbody>
</table>
V-Modell XT: Decision Points

Projektdurchführungsstrategie

1.* legt Reihenfolge fest

1.* Entscheidungspunkt

benötigt

Produkt

[im Zustand „fertig gestellt“]

Legende:

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

Vorgehensmodell analysiert
Verbesserung Vorgehensmodell konzipiert
Verbesserung Vorgehensmodell realisiert
3.10.4.6 Hardware Elements to be Specified

The preparation of a specification for a hardware element is expensive and not always required. In order to adapt the specification effort to the requirements of individual projects, the »Hardware Architect« can - based on the specifications in the Project Manual and the requirements - determine which hardware elements need a »Hardware Specification«.

Criteria for the necessity of a specification may include the following: criticality of the hardware element, complexity of the requirements posed on the hardware element, test requirements specified in the »Hardware Implementation, Integration and Evaluation Concept«. In any case, a hardware specification shall be prepared for hardware elements to be tested, since this specification will be the basis for the »Evaluation Specification System Elements«. If hardware elements are classified as not to be specified, a rationale shall be included.

3.10.5 Software Architecture

Process module: Software Development

Responsible: Software Architect (when using process module Software Development)

Activity: Preparing Software Architecture

Participating: Software Developer, System Architect, System Integrator

Purpose

For every software unit identified in the system architecture, a »Software Architecture« will be developed. Based on the functional and non-functional requirements posed on a »Software Unit«, the »Software Architect« is tasked with designing a suitable »Software Architecture«. The Product Software Architecture will be used as design guide and for documenting the design decisions.

As in the system architecture development, significant architectural principles will be specified, and possible design alternatives will be examined. In accordance with the selected design alternative, the software unit will be decomposed into »Software Component «, »Software Modules « and products of the type External Software Module. Relations and interfaces between the elements and to the environment will be identified and summarized. A »Data Catalog « of the data structures exchanged at the interfaces will be prepared.

The suitability of the selected architecture for the system to be developed will be assessed. Open questions may be answered, e.g., within the scope of a prototype development.

The software architecture design may lead to changes in the system architecture. Depending on the specifications in the Project Manual, the »System Architect« will examine the change and integrate it immediately, if required. In individual cases, an explicit change request may be necessary.

The main responsibility for the design of the software architecture will be vested in the Software Architect who will be supported by the »Software Developer« and various specialists for individual subjects, e.g., logistics, safety and security, and ergonomics.

The software architecture is the central document for the preparation of additional products. It specifies all software components and software modules of the software unit. The individual elements and their specifications will be developed in accordance with these architectural requirements.


V-Modell XT: Example Building Block & Product State

SW-Development (‘SW-Entwicklung’)
V-Modell XT: (Lots of) Disciplines and Products
V-Modell XT: Activities (as many?!)
V-Modell XT: Activities (as many?!)

Entwicklung

Systemelemente

Zum System integrieren
Zum Unterstützungssystem integrieren
Zum Segment integrieren
Externe Einheit übernehmen
Zur HW-Einheit integrieren
Zur SW-Einheit integrieren
Zur HW-Komponente integrieren
Zur SW-Komponente integrieren
HW-Modul realisieren
SW-Modul realisieren
Externes HW-Modul übernehmen
Externes SW-Modul übernehmen

Logistikelemente

Systementwurf

Systemarchitektur erstellen
Unterstützungssystemarchitektur erstellen
Styleguide für die Mensch-Maschine-Schnittstelle erstellen
HW-Architektur erstellen
SW-Architektur erstellen
Datenbankentwurf erstellen
Implementierungs-, Integrations- und Prüfkonzept System erstellen
Implementierungs-, Integrations- und Prüfkonzept Unterstützungssystem erstellen
Implementierungs-, Integrations- und Prüfkonzept HW erstellen
Implementierungs-, Integrations- und Prüfkonzept SW erstellen
Migrationskonzept erstellen

Systemspezifikation

Gesamtsystemspezifikation (Pflichtenheft) erstellen
Systemspezifikation erstellen
Externe Einheit-Spezifikation erstellen
HW-Spezifikation erstellen
SW-Spezifikation erstellen
Externe-HW-Modul-Spezifikation erstellen
Externe-SW-Modul-Spezifikation erstellen
V-Modell XT: Roles (even more?!)

Project Roles:

Änderungssteuerungsgruppe (Change Control Board), Änderungsverantwortlicher, Anforderungsanalytiker (AG), Anforderungsanalytiker (AN), Anwender, 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:

Akquisiteur, Datenschutzbeauftragter (Organisation), Einkäufer, IT-Sicherheitsbeauftragter (Organisation), Qualitätsmanager
What About the Colours?
V-Modell XT considers 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/many customer(s); development/improvement/migration; maintenance
V-Modell XT: Tailoring Instance

Building Blocks

Plan

Projekt genehmigt → Projekt definiert

Anforderungen festgelegt

Projekt ausgeschrieben → Projekt beauftragt

Iteration geplant

Projektfortschritt überprüft

Abnahme erfolgt

Projekt abgeschlossen

Gesamuprojekt aufgeteilt

System spezifiziert

Feinentwurf abgeschlossen

System integriert

Verbesserung Vorgehensmodell realisiert

Gesamuprojektfortschritt überprüft

System entworfen

System realisiert

Verbesserung Vorgehensmodell konzipiert

Verbesserung Vorgehensmodell analysiert

Alle V-Modell-Projekte

Organisationsspezifisches Vorgehensmodell

ASiAN-Schnittstelle

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

- 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).
**Content**

- **Procedure and Process Models**
  - **Procedure** Model Examples
    - The (in)famous Waterfall model
    - The famous Spiral model
  - Procedure classification
    - linear / non-linear
    - prototyping
    - evolutionary, iterative, incremental
  - From Procedure to Process Models
  - **Process** Model Examples
    - Phase Model
    - V-Modell XT
    - Agile
      - Extreme Programming
      - Scrum

- **Process Metrics**
  - CMMI, Spice
Agile
“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.” (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

“continuous / sustainable delivery”
- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

“simplicity”
- Simplicity – the art of maximizing the amount of work not done – is essential.
- Working software is the primary measure of progress.

“changes”
- Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

“people”
- The best architectures, requirements, and designs emerge from self-organizing teams.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- Business people and developers must work together daily throughout the project.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

“retrospective”
- Continuous attention to technical excellence and good design enhances agility.
- 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.
- Work in small groups (6–8 people) proposed.
- 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)
Agile

— 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 XP practices]

- [Test Drive Development](spec_of_tests_for_coding)
- [Pair Programming](programmer_programmer)
Agile

— Scrum —
Scrum

- First published 1995 (Schwaber, 1995), based on ideas of Takeuchi and Nonaka.
- Inspired by Rugby (yes, the “hooligan’s game played by gentlemen”): 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 Process

- **Product Backlog**
  - (maintained by product owner)
  - comprises all requirements to be realised,
  - priority and effort estimation for requirements,
  - collects tasks to be conducted,

- **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, new estimations)

- **Sprint-burndown report**
  - completed/open tasks from sprint backlog,
  - should decrease linearly, otherwise remove tasks from sprint backlog,

- **Sprint Report**
  - which requirements (not) realised in last sprint,
  - description of obstacles/problems during sprint
**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 are 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.
Content

- Procedure and Process Models
  - Procedure Model Examples
    - The (in)famous Waterfall model
    - The famous Spiral model
  - Procedure classification
    - linear / non-linear
    - prototyping
    - evolutionary, iterative, incremental
  - From Procedure to Process Models
  - Process Model Examples
    - Phase Model
    - V-Modell XT
    - Agile
      - Extreme Programming
      - Scrum
  - Process Metrics
    - CMMI, Spice
Process Metrics
Idea (for material goods): The quality of the (production) process influences product quality.

Plan: Specify abstract criteria (metrics) to determine good production processes (e.g., to choose manufacturer).

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

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

Note: a good process does not stop us from creating bad products; (the hope is, that) bad products are less likely when using a good process, i.e. that there is a correlation:
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

Unauthorized distribution subject to be copyright.

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)
• 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.
### 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 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 is 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 SG 1 and SG 2 for area RD.
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 certification may hinder innovation,
- CMMI levels are chosen somewhat arbitrarily: “why is an area in level $N$ and not already in level $N - 1$?”
**Software Process Improvement and Capability Determination**

- 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)
Tell Them What You’ve Told Them...

- **Waterfall Model**
  - very well-known, very abstract, of limited practical use.

- **Spiral Model**
  - iterated risk assessment, e.g., for very innovative projects.

- **Classification** of processes
  - prototyping: needs purposes and questions
  - evolutionary, iterative, incremental

- **V-Model XT**
  - slightly different vocabulary,
  - quite comprehensive,
  - may serve as inspiration for, e.g., definition of roles,
  - can be tailored in various ways

- **Agile** approaches
  - XP: proposes methods and approaches
  - Scrum: focuses on management aspects

- Measure **process quality**: CMMI, Spice
References
References


