Softwaretechnik / Software-Engineering

Lecture 5: Procedure & Process Models

2017-05-15

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

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	Procedure Models
	Process Models
	Process Metrics
⋮	CMMI, Spice

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

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')

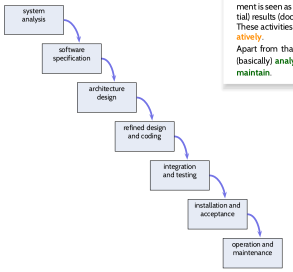
    e.g., "waterfall model" (70s/80s).
  - (ii) **Organisational structure** – comprising requirements on
    - project management and responsibilities,
    - quality assurance,
    - documentation, document structure,
    - revision control.

  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.

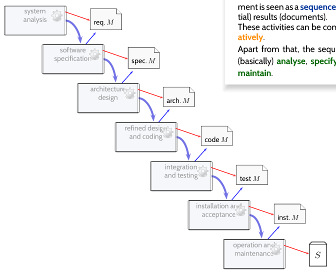
Procedure Models

The (In)famous Waterfall Model (Rosove, 1967)



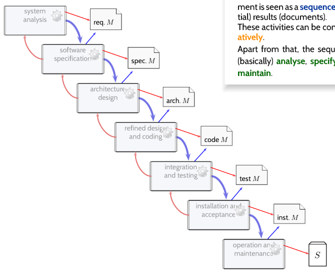
Waterfall or Document-Model– Software development is seen as a **sequence of activities** coupled by (partial) results (documents).
These activities can be conducted **concurrently** or **iteratively**.
Apart from that, the sequence of activities is fixed as (basically) **analyse, specify, design, code, 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)

The (In)famous Waterfall Model (Rosove, 1967)

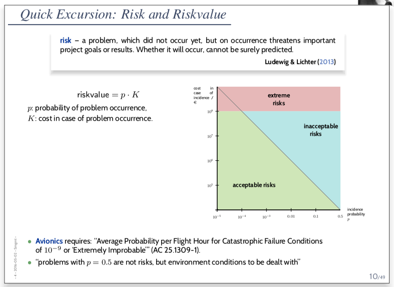


Waterfall or Document-Model– Software development is seen as a **sequence of activities** coupled by (partial) results (documents).
These activities can be conducted **concurrently** or **iteratively**.
Apart from that, the sequence of activities is fixed as (basically) **analyse, specify, design, code,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

The Spiral Model (Boehm, 1988)

ɔoehm

Recall: risk and risk value.



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)

Barry W. Boehm

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?),
- ...

Idea of **Spiral Model**: do not plan ahead everything, but go step-by-step.

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** $v(r)$

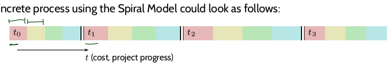
(iii) for the risk r_0 with the **highest risk value**, $r_0 = \max\{v(r) \mid r \in R\}$,
find a way to eliminate this risk, and go this way;
if there is no way to eliminate the risk, stop with project failure

Advantages:

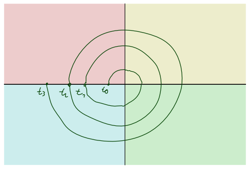
- We know early if the project goal is unreachable.
- Knowing that the biggest risks are eliminated gives a good feeling.

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,



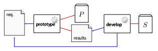
Procedure Model Classification

Linear vs. Non-Linear Procedure Models

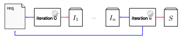
- **linear**: the strict Waterfall Model (no feedback)
- **non-linear**: basically everything else (with feedback between activities)

Classification By Treatment of (Software) Artefacts

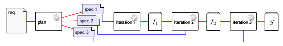
- **Prototyping**:



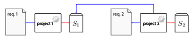
- **Evolutionary**:



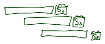
- **Iterative**:



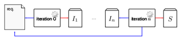
- **Incremental**:



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

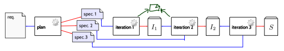


Evolutionary and Iterative Development



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)**



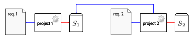
iterative software development – software is developed in **multiple iterative steps**, all of them planned and controlled.

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)

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)**

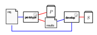
- One difference (in our definitions):
  - **iterative**: steps towards fixed goal,
  - **incremental**: goal extended for each step; next step goals may already be planned.

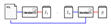
    Examples: operating system releases, short time-to-market (→ continuous integration).

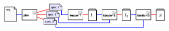
Another Characterisation of Approaches

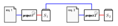
Approach	Used for Analysis of Requirements	Results Used on Target System	Has Defined Steps	Preliminary Results Used	Has Complete Plan
Rapid Prototyping	█				
Evolutionary Development	▓	█			
Iterative Development	░	█	▓	░	
Incremental Development	░	█	█	▓	░
Staircase Development		█	█	█	█

█ yes
▓ to some amount
░ to a low amount









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

From Procedure to Process Model

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.

Light vs. Heavyweight Process Models

- You may hear about **"light"** and **"heavyweight"** process models.
  - Sometimes: heavier means higher number of rules…
  - Sometimes: heavier means less flexible, adaptable process…
  - Clear: "lightweight" sounds better than "heavyweight".
- 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

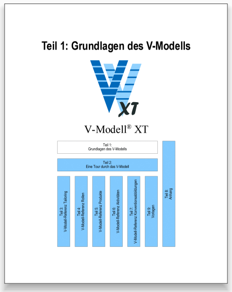
The Phase Model

- 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),
  small companies.

V-Model XT



Teil 1: Grundlagen des V-Modells

V-Modell® XT

Teil 1:
Grundlagen des V-Modells

Teil 2:
Eine Tour durch das V-Modell

Teil 3:
V-Modell-Referenz Tailoring

Teil 4:
V-Modell-Referenz Rollen

Teil 5:
V-Modell-Referenz Produkte

Teil 6:
V-Modell-Referenz Aktivitäten

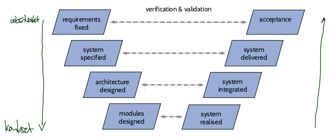
Teil 7:
V-Modell-Referenz Konventionsabbildungen

Teil 9:
Vorlagen

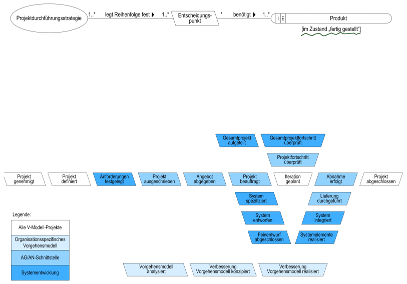
Teil 8:
Anhang

V-Modell XT

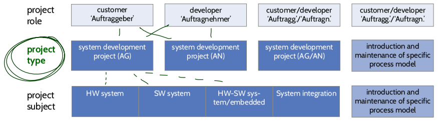
- There are different "**V-shaped**" **process models**, we discuss the (German) "V-Modell".
- **"V-Modell"**:
  - developed by company IABG in cooperation with the Federal Office for Defence Technology and Procurement ('Bundesministerium für Verteidigung'), released 1998
  - (German) government as customer often **requires** usage of the V-Modell
- 2012: "**V-Modell XT**" Version 1.4 (Extreme Tailoring) (V-Modell XT, 2006)



V-Modell XT: Decision Points



V-Modell XT: Project Types

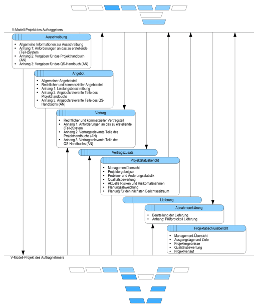

project role
customer 'Auftraggeber'
developer 'Auftragnehmer'
customer/developer 'Auftragg.'/'Auftragn.'
customer/developer 'Auftragg.'/'Auftragn.'
project type
system development project (AG)
system development project (AN)
system development project (AG/AN)
introduction and maintenance of specific process model
project subject
HW system
SW system
HW-SW system/embedded
System integration
introduction and maintenance of specific process model

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: Customer/Developer Interface


V-Modell-Projekt des Auftraggebers
Ausschreibung
Allgemeine Informationen zur Ausschreibung
Anhang 1: Anforderungen an das zu erstellende (Teil-)System
Anhang 2: Vorgaben für das Projekthandbuch (AN)
Anhang 3: Vorgaben für das QS-Handbuch (AN)
Angebot
Allgemeiner Angebotsteil
Rechtlicher und kommerzieller Angebotsteil
Anhang 1: Leistungsbeschreibung
Anhang 2: Angebotsrelevante Teile des Projekthandbuchs
Anhang 3: Angebotsrelevante Teile des QS-Handbuchs (AN)
Vertrag
Rechtlicher und kommerzieller Vertragsteil
Anhang 1: Anforderungen an das zu erstellende (Teil-)System
Anhang 2: Vertragsrelevante Teile des Projekthandbuchs (AN)
Anhang 3: Vertragsrelevante Teile des QS-Handbuchs (AN)
Vertragszusatz
Projektstatusbericht
Managementübersicht
Projektergebnisse
Problem- und Änderungsstatistik
Qualitätsbewertung
Aktuelle Risiken und Risikomaßnahmen
Planungsabweichung
Planung für den nächsten Berichtszeitraum
Lieferung
Abnahmeerklärung
Beurteilung der Lieferung
Anhang: Prüfprotokoll Lieferung
Projektabschlussbericht
Management-Übersicht
Ausgangslage und Ziele
Projektergebnisse
Qualitätsbewertung
Projektverlauf
V-Modell-Projekt des Auftragnehmers

V-Modell XT: The V-World (naja. . .)

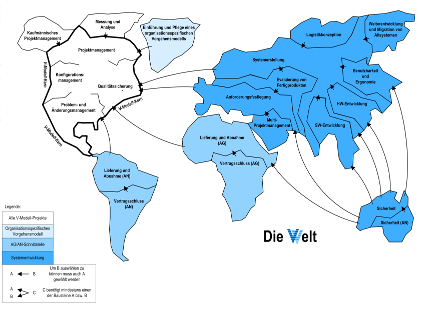
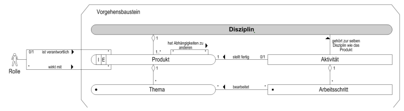
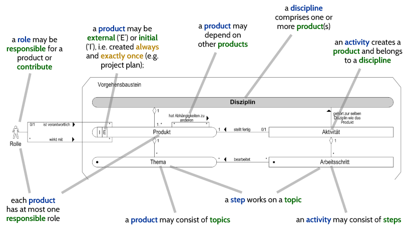

Kaufmännisches Projektmanagement
Messung und Analyse
Einführung und Pflege eines organisationsspezifischen Vorgehensmodells
Projektmanagement
Konfigurations-management
Qualitätssicherung
Problem- und Änderungsmanagement
V-Modell-Kern
V-Modell-Kern
V-Modell-Kern
Logistikkonzeption
Weiterentwicklung und Migration von Altsystemen
Systemerstellung
Evaluierung von Fertigprodukten
Benutzbarkeit und Ergonomie
Anforderungsfestlegung
HW-Entwicklung
Multi-Projektmanagement
SW-Entwicklung
Lieferung und Abnahme (AG)
Vertragsschluss (AG)
Lieferung und Abnahme (AN)
Vertragsschluss (AN)
Sicherheit
Sicherheit (AN)
Die Welt
Legende:
Alle V-Modell-Projekte
Organisationsspezifisches Vorgehensmodell
AG/AN-Schnittstelle
Systementwicklung
A B Um B auswählen zu können muss auch A gewählt werden
A B C C benötigt mindestens einen der Bausteine A bzw. B

Abbildung 6: Vorgehensbausteinlandkarte

V-Modell XT: Procedure Building Blocks



V-Modell XT: Procedure Building Blocks



a **role** may be **responsible** for a product or **contribute**

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**

a **discipline** comprises one or more **product**(s)

an **activity** creates a **product** and belongs to a **discipline**

each **product** has at most one **responsible** role

a **product** may consist of **topics**

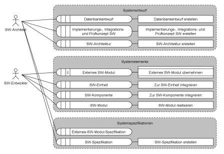
a **step** works on a **topic**

an **activity** may consist of **steps**

our course	V-Modell XT	explanation
role	role (‘Rolle’)	
activity	activity (‘Aktivität’)	
-	step (‘Arbeitsschritt’)	parts of activities
artefact	product (‘Produkt’)	
-	topic (‘Thema’)	parts of products

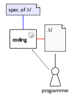
our course	V-Modell XT	explanation
-	discipline (‘Disziplin’)	set of related products / activities
phase	project segment (?) (‘Projektabschnitt’)	

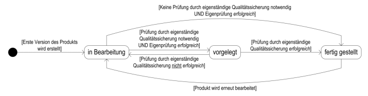
V-Modell XT: Example Building Block & Product State


SW-Architekt
SW-Entwickler
Systementwurf
Datenbankentwurf
Datenbankentwurf erstellen
Implementierungs-, Integrations- und Prüfkonzept SW
Implementierungs-, Integrations- und Prüfkonzept SW erstellen
SW-Architektur
SW-Architektur erstellen
Systemelemente
E
Externes SW-Modul
Externes SW-Modul übernehmen
SW-Einheit
Zur SW-Einheit integrieren
SW-Komponente
Zur SW-Komponente integrieren
SW-Modul
SW-Modul realisieren
Systemspezifikationen
Externes-SW-Modul-Spezifikation
SW-Spezifikation
SW-Spezifikation erstellen

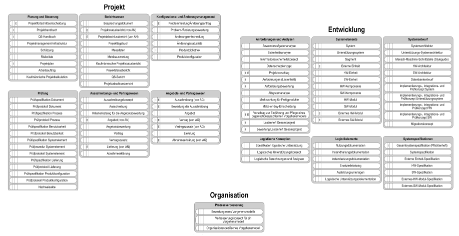
SW-Development (‘SW-Entwicklung’)

vs.

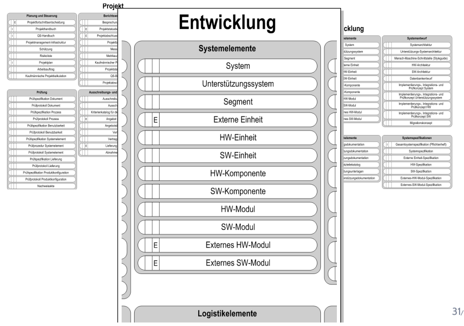



[Erste Version des Produkts wird erstellt]
in Bearbeitung
[Prüfung durch eigenständige Qualitätssicherung notwendig UND Eigenprüfung *erfolgreich*]
vorgelegt
[Prüfung durch eigenständige Qualitätssicherung nicht *erfolgreich*]
[Prüfung durch eigenständige Qualitätssicherung *erfolgreich*]
fertig gestellt
[Keine Prüfung durch eigenständige Qualitätssicherung notwendig UND Eigenprüfung *erfolgreich*]
[Produkt wird erneut bearbeitet]

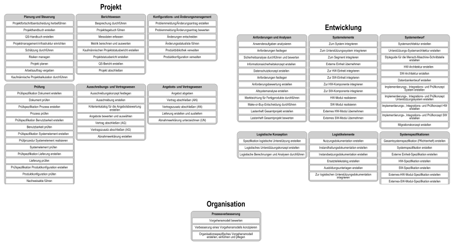
V-Modell XT: (Lots of) Disciplines and Products



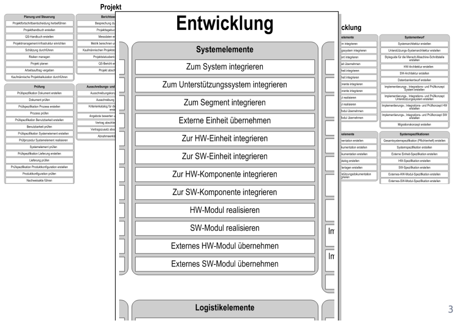
V-Modell XT: (Lots of) Disciplines and Products



V-Modell XT: Activities (as many?!)


Projekt
Planung und Steuerung
Projektfortschrittsentscheidung herbeiführen
Projekthandbuch erstellen
QS-Handbuch erstellen
Projektmanagement-Infrastruktur einrichten
Schätzung durchführen
Risiken managen
Projekt planen
Arbeitsauftrag vergeben
Kaufmännische Projektkalkulation durchführen
Berichtswesen
Besprechung durchführen
Projekttagebuch führen
Messdaten erfassen
Metrik berechnen und auswerten
Kaufmännischen Projektstatusbericht erstellen
Projektstatusbericht erstellen
QS-Bericht erstellen
Projekt abschließen
Konfigurations- und Änderungsmanagement
Problemmeldung/Änderungsantrag erstellen
Problemmeldung/Änderungsantrag bewerten
Änderungen entscheiden
Änderungsstatusliste führen
Produktbibliothek verwalten
Produktkonfiguration verwalten
Prüfung
Prüfspezifikation Dokument erstellen
Dokument prüfen
Prüfspezifikation Prozess erstellen
Prozess prüfen
Prüfspezifikation Benutzbarkeit erstellen
Benutzbarkeit prüfen
Prüfspezifikation Systemelement erstellen
Prüfprozedur Systemelement realisieren
Systemelement prüfen
Prüfspezifikation Lieferung erstellen
Lieferung prüfen
Prüfspezifikation Produktkonfiguration erstellen
Produktkonfiguration prüfen
Nachweisakte führen
Ausschreibungs- und Vertragswesen
Ausschreibungskonzept festlegen
Ausschreibung erstellen
Kriterienkatalog für die Angebotsbewertung erstellen
Angebote bewerten und auswählen
Vertrag abschließen (AG)
Vertragszusatz abschließen (AG)
Abnahmeerklärung erstellen
Angebots- und Vertragswesen
Angebot abgeben
Vertrag abschließen (AN)
Vertragszusatz abschließen (AN)
Lieferung erstellen und ausliefern
Abnahmeerklärung unterzeichnen (UN)
Entwicklung
Anforderungen und Analysen
Anwenderaufgaben analysieren
Anforderungen festlegen
Sicherheitsanalyse durchführen und bewerten
Informationssicherheitskonzept erstellen
Datenschutzkonzept erstellen
Anforderungen festlegen
Anforderungsbewertung erstellen
Altsystemanalyse erstellen
Marktsichtung für Fertigprodukte durchführen
Make-or-Buy-Entscheidung durchführen
Lastenheft Gesamtprojekt erstellen
Lastenheft Gesamtprojekt bewerten
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
Systementwurf
Systemarchitektur erstellen
Unterstützungs-Systemarchitektur 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
Logistische Konzeption
Spezifikation logistische Unterstützung erstellen
Logistisches Unterstützungskonzept erstellen
Logistische Berechnungen und Analysen durchführen
Logistikelemente
Nutzungsdokumentation erstellen
Instandhaltungsdokumentation erstellen
Instandsetzungsdokumentation erstellen
Ersatzteilkatalog erstellen
Ausbildungsunterlagen erstellen
Zur logistischen Unterstützungsdokumentation integrieren
Systemspezifikationen
Gesamtsystemspezifikation (Pflichtenheft) erstellen
Systemspezifikation erstellen
Externe Einheit-Spezifikation erstellen
HW-Spezifikation erstellen
SW-Spezifikation erstellen
Externes-HW-Modul-Spezifikation erstellen
Externes-SW-Modul-Spezifikation erstellen
Organisation
Prozessverbesserung
Vorgehensmodell bewerten
Verbesserung eines Vorgehensmodells konzipieren
Organisationsspezifisches Vorgehensmodell erstellen, einführen und pflegen

V-Modell XT: Activities (as many?!)


Projekt
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

V-Modell XT: Roles (even more?!)

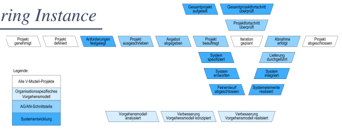
Project Roles:

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:

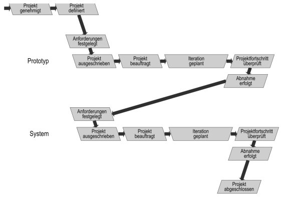
Akquisiteur, **Datenschutzbeauftragter (Organisation)**, **Einkäufer**, **IT-Sicherheitsbeauftragter (Organisation)**, **Qualitätsmanager**

V-Modell XT: Tailoring Instance



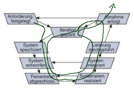
Building Blocks

Plan

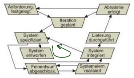


V-Modell XT: Development Strategies

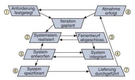
V-Modell XT mainly supports three **strategies**, i.e. principal **sequences between decision points**, to develop a system:



incremental



component based



prototypical

V-Modell XT: Discussion

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).

Agile

The Agile Manifesto

"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.*
- ***Business people and developers must work together*** *daily throughout the project.*
- *Agile processes promote* ***sustainable development****.*
  The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- ***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.*
- ***Working software is the primary measure*** *of progress.*
- ***Simplicity*** *– the art of* ***maximizing the amount of work not done*** *– is essential.*
- *Continuous* ***attention to technical excellence*** *and good design enhances agility.*
- ***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****.*
- *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.
- 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)

Extreme Programming (XP)

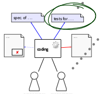
Extreme Programming (XP) (Beck, 1999)

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



Scrum

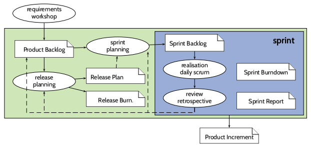
Scrum

- 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:

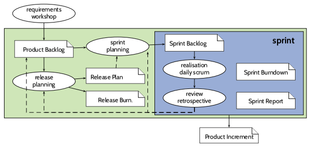
- **product owner**:
  - representative of customer,
  - maintains requirements in the **product backlog**,
  - plans and decides which requirement(s) to realise in next sprint,
  - (passive) participant of **daily scrum**,
  - assesses results of sprints
- **scrum team**:
  - members capable of developing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

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)

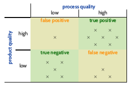
Scrum: Discussion

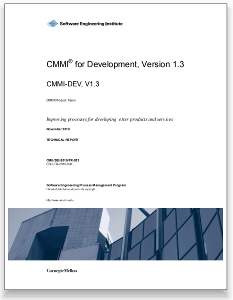
- Has been used in many projects, experience in majority positive.
- Team size bigger 7–10 may need **scrum of scrums**.
- Competent **product owner** necessary for success.
- Success depends on motivation, competence, and communication skills of team members.
- Team members are responsible for planning,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

Process Metrics

Assessment and Improvement of the Process

- **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:





Software Engineering Institute

CMMI® for Development, Version 1.3

CMMI-DEV, V1.3

CMMI Product Team

Improving processes for developing etter products and services

November 2010

TECHNICAL REPORT

CMU/SEI-2010-TR-033
ESC-TR-2010-033

Software Engineering Process Management Program
Unlimited distribution subject to the copyright.

http://www.sei.cmu.edu

Carnegie Mellon

CMMI

- 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

level	level name	process areas
1	**initial**	-
2	**managed**	REQM, PP, PMC, MA, PPQA, CM, SAM
3	**defined**	+ RD, TS, PI, VER, VAL, OPF, OPD, OT, IPM, RSKM, DAR
4	**quantitatively managed**	+ OPP, QPM
5	**optimising**	+ OID, CAR

- **initial** – the process is not consciously designed, just evolved.

CMMI Levels

level	level name	process areas
1	**initial**	-
2	**managed**	REQM, PP, PMC, MA, PPQA, CM, SAM
3	**defined**	+ RD, TS, PI, VER, VAL, OPF, OPD, OT, IPM, RSKM, DAR
4	**quantitatively managed**	+ OPP, QPM
5	**optimising**	+ OID, CAR

- **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 reources
  - …
- 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 $N-1$?"

SPICE / ISO 15504

Software **P**rocess **I**mprovement and **C**apability **D**etermination

- similar to CMM(I): maturity levels, assessment, certificates
- a 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.)
- 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

Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J. (2002). Agile software development methods. review and analysis. Technical Report 478.

Beck, K. (1999). *Extreme Programming Explained – Embrace Change*. Addison-Wesley.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21(5):61–72.

Hörmann, K., Dittmann, L., Hindel, B., and Müller, M. (2006). *SPICE in der Praxis: Interpretationshilfe für Anwender und Assessoren*. dpunkt.verlag.

IEEE (1990). *IEEE Standard Glossary of Software Engineering Terminology*. Std 610.12-1990.

Ludewig, J. and Lichter, H. (2013). *Software Engineering*. dpunkt.verlag, 3. edition.

Rosove, P. E. (1967). *Developing Computer-based Information Systems*. John Wiley and Sons.

Schwaber, K. (1995). SCRUM development process. In Sutherland, J. et al., editors, *Business Object Design and Implementation, OOPSLA'95 Workshop Proceedings*. Springer-Verlag.

Team, C. P. (2010). Cmmi for development, version 1.3. Technical Report ESC-TR-2010-033, CMU/SEI.

V-Modell XT (2006). *V-Modell XT*. Version 1.4.

Züllighoven, H. (2005). *Object-Oriented Construction Handbook - Developing Application-Oriented Software with the Tools and Materials Approach*. dpunkt.verlag/Morgan Kaufmann.