universität innsbruck
Institut für Informatik

Software Engineering

4. Vorgehensmodelle und Projektorganisation

Ruth Breu

Übersicht

- 4.1 Begriffe – Projekt und Projektmanagement
- 4.2 Vorgehensmodelle

4.1 Begriffe – Projekt und Projektmanagement

Projekt

„Vorhaben, das im Wesentlichen durch die **Einmaligkeit** aber auch Konstante der Bedingungen in ihrer **Gesamtheit** gekennzeichnet ist, wie
z. B. Zielvorgabe, zeitliche, finanzielle, personelle und andere Begrenzungen; Abgrenzung gegenüber anderen Vorhaben; projektspezifische Organisation."

– DIN 69901

„Ein Projekt ist ein **zeitlich begrenztes** Unternehmen, das unternommen wird, um ein **einmaliges** Produkt, eine Dienstleistung oder ein Ergebnis zu erzeugen."

– Project Management Body of Knowledge des amerikanischen Project Management Institute

„ein **zeit- und kostenbeschränktes** Vorhaben zur Realisierung einer Menge **definierter Ergebnisse** entsprechend vereinbarter Qualitätsstandards und Anforderungen (Erfüllung der Projektziele) …"

– IPMA Competence Baseline der International Project Management Association (IPMA)

Literatur: Projektorganisation und Management im Software Engineering, Springer

Quellen: Grechenig et al., Martin Wirsing

Begriff Projektmanagement

- "*Gesamtheit von Führungsaufgaben, -organisation, -techniken und -mitteln für die Abwicklung eines Projektes„*

(DIN 69901)

- "Project Management is the application of knowledge, skills, tools and techniques to project activities to meet project requirements."
- "Project management includes
  - Identifying requirements,
  - Establishing clear objectives,
  - Balancing the competing demands for time, quality and cost,
  - Adapting the specifications, plans and approach to the different concerns and expectations of the various stakeholders.“

(Project Management Institute, pmi.org)

Projektmanagement

- Projektmanagement umfasst alle Aufgaben bei der Durchführung von Projekten hinsichtlich
  - Vorbereitung (Struktur und Personal)
  - Planung
  - Kontrolle
  - Steuerung
- Dazu gehören auch projektübergreifende Aufgaben:
  - Projektabschluss und Ergebnisdokumentation
  - Prozessverbesserung
  - Personalführung
  - Interaktion mit dem Kunden und Koordination von Zulieferern

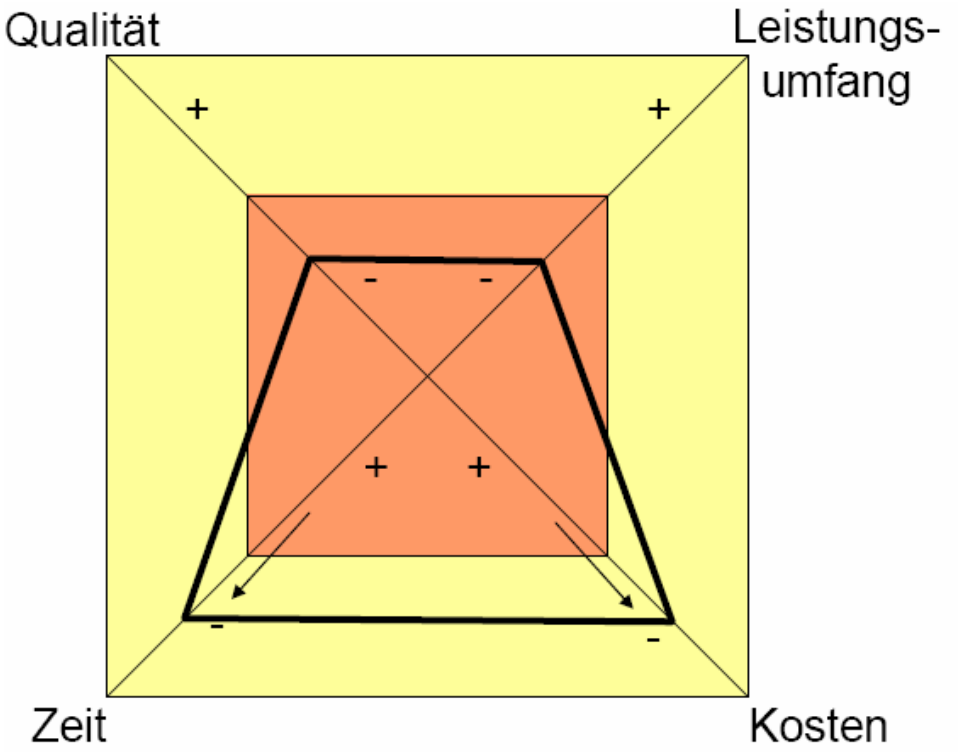
Schwierigkeiten im IT-Projektmanagement

- Effizienz vs. Flexibilität:
  - Effizienz: Einsatz bewährter Methoden zur Kostenreduktion:
    - Standardanwendungsdomäne
    - Standardentwicklungsplattform
    - Standardentwicklungsprozess
    - Standardentwicklungsumgebung
  - Flexibilität: Das Unvorhersehbare planen:
    - Änderung der Anforderungen durch den Kunden
    - Änderung der Entwicklungsplattform durch das Management
    - Mitarbeiterfluktuation
    - Änderung des Auslieferertermins durch das Management
    - Verzögerungen durch die Zulieferer

Schwierigkeiten im IT-Projektmanagement

- Immaterialität des Produkts: Software ist unsichtbar
  - für den Kunden schwer erlebbar (Kostenbewusstsein)
  - bzgl. des Fertigungsgrades schwer messbar
  - Risiko: Umfang, Kosten und Laufzeit
- Innovativität des Produkts: Software ist i.a. „Null“-Serie
  - mit dem Einsatz neuer Technologie verbunden
  - mit der Realisierung umfassender neuer Funktionen verbunden
  - Risiko: Stabilität (Qualität) und Laufzeit
- Mangelnde Prozessreife: SE ist eine junge Disziplin, damit
  - sind standardisierte und etablierte Entwicklungsprozesse großen Veränderungen unterworfen

Projektmanagement Teufelsquadrat nach Harry Sneed



- Zielgrößen auf den Diagonalen:
  - Zeit: Projektlaufzeit
  - Kosten: Budget
  - Qualität: z.B. Funktionalität, Nutzbarkeit, Wartbarkeit
  - Leistungsumfang: Anzahl ausgelieferter Funktionen
- Die von den Zielgrößen gebildete Fläche beschreibt die **Produktivität des Projekts**.
- **Die Fläche (Produktivität) eines Projekts ist invariant**
  - Wenn ein Projekt z. B. in weniger Zeit und zu geringeren Kosten abgeschlossen werden soll, verringern sich auch Leistungsumfang und Qualität.

Aufgabenfelder Projektmanagement

- Laut Project Management Institute besteht PM aus den folgenden **9 Aufgabenfeldern:**
  - Integrierende Aufgaben (integration management)
  - Umfangsmanagement (scope management)
  - Zeitmanagement (time management)
  - Kostenmanagement (cost management)
  - Qualitätsmanagement (quality management)
  - Personalmanagement (human resource management)
  - Kommunikationsmgmt. (communication management)
  - Risikomanagement (risk management)
  - Beschaffungsmanagement (procurement management)

Projekttypen

- Greenfield Projekt
  - System wird neu gebaut
  - Herausforderung: Vorstellungen der Nutzer bzw. Bedarf am Markt geeignet berücksichtigen
- Brownfield Project
  - Existierende Software (*legacy system*) wird signifikant verändert
  - (Einige) Anforderungen können vom existierenden System abgeleitet werden
  - Herausforderungen: In vielen Fällen wird das existierende System ohne Dokumentation betrieben, viele Abhängigkeiten mit anderen Systemen

Brownfield Projekte - Beispiele

- **Migrationsprojekt**
  - Austausch der Technologie-Plattform, wenig Fokus auf neue Funktionalität
  - Herausforderungen:
    - Funktionalität des alten Systems nachbilden
    - Evtl. notwendige parallele (Weiter)entwicklung des alten und neuen Systems
    - Datenmigration
- **Entwicklung neuer Releases**
  - Ein existierendes Softwaresystem wird kontinuierlich weiterentwickelt
  - Herausforderungen:
    - Kurze Release-Zyklen ermöglichen
    - Anforderungen von Usern und Markt identifizieren und priorisieren
    - Management unterschiedlicher Produkt-Varianten (z.B. branchenspezifische Varianten)

Individualsoftware - Softwareprodukte

- Individualsoftware
  - SW-Dienstleister entwickelt das System auf Basis der Anforderungen des Kunden/Nutzers
  - Kunde steuert die Anforderungen
  - Vertragsarten: Werkvertrag oder Dienstleistungsvertrag
- Softwareprodukt
  - Entwicklung durch Produktanbieter
  - Produktanbieter steuert die Anforderungen
  - Verkauf durch Lizenzmodell - Zahlung einmalig/periodisch oder nach Nutzung (Software-as-a-Service)

Systemtypen

- Informationssysteme
  - Schwerpunkt auf Daten- und Prozessunterstützung
- Eingebettete Systeme
  - Schwerpunkt auf Hardware-Steuerung
- Cyber-physical Systems
  - Integration von Informations- und eingebetteten Systemen
  - Verbindung von Geräten durch Data Analytics Services, Sensornetzwerken, mobilen Schnittstellen
  - Beispiele: smarte Medizinprodukte, autonomes Fahren, Smart Grid, Industrie 4.0

4.2 Vorgehensmodelle

Ein Vorgehensmodell (Prozessmodell) definiert

- die **Aktivitäten**, die im Laufe der Softwareentwicklung durchlaufen werden
  - z.B. Angebotserstellung oder Testen
- die **Produkte** (**Dokumente, Modelle**), die innerhalb der Aktivitäten erstellt werden
  - z.B. Pflichtenheft oder lauffähiges System
- die **Beschreibungstechniken** und **Methoden**, die in den Produkten bzw. zur Erstellung der Produkte verwendet werden
  - z.B. Klassendiagramme
- die für die Aktivitäten und Produkte **Verantwortlichen**
  - z.B. Projektmanager, Entwickler
- den **Prozess**, d.h. mögliche Abfolgen von Aktivitäten

Bekannte Vorgehensmodelle

- Allgemeine Vorgehensmodelle
  - Wasserfallmodell
  - Iterative und inkrementelle Softwareentwicklung
- Praktizierte bzw. standardisierte Vorgehensmodelle
  - V-Modell
  - Agile Vorgehensmodelle

Übersicht

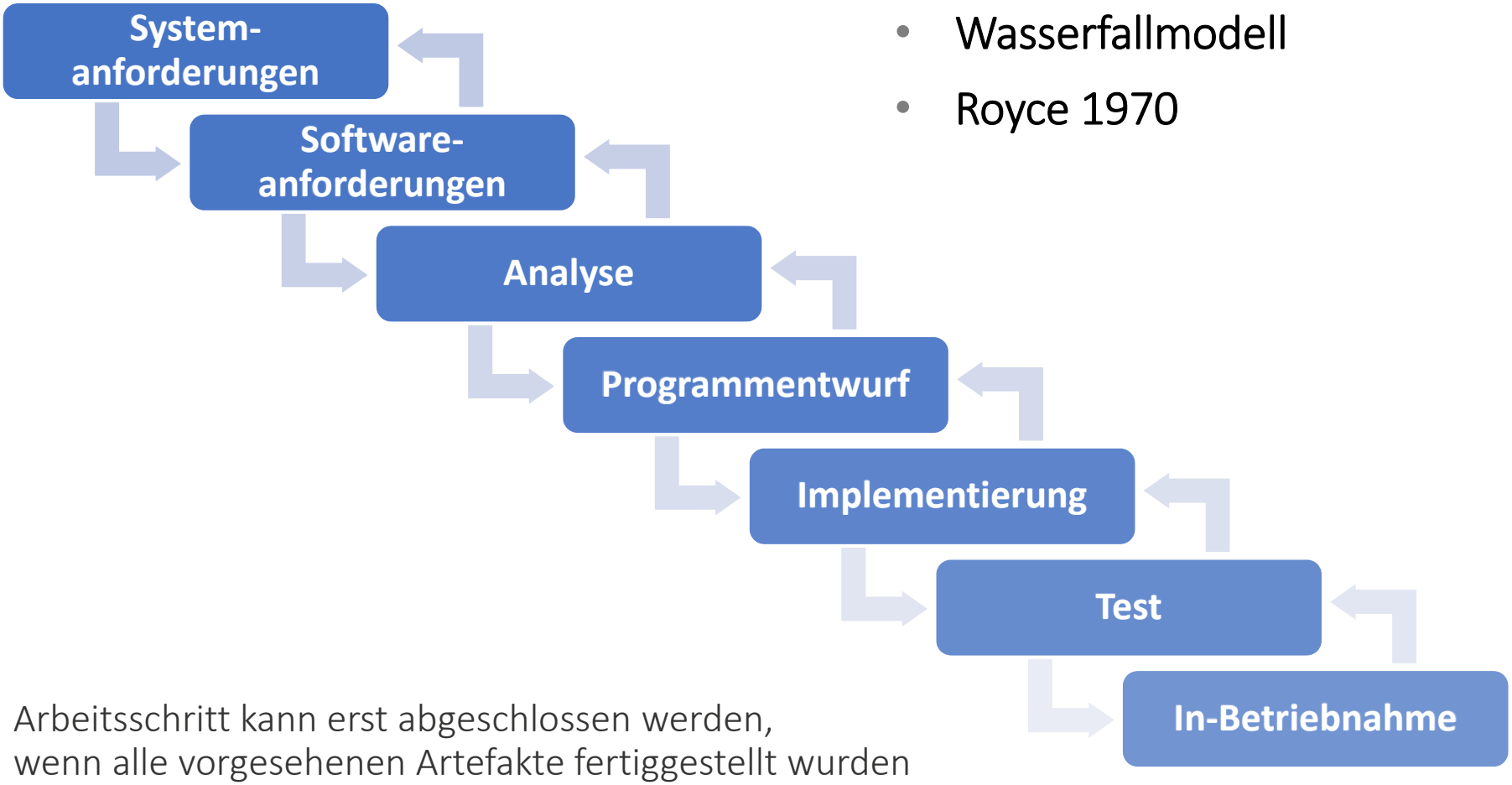
4.2.1 Allgemeine Vorgehensmodelle

4.2.2 Inkrementeller Softwareentwurf

4.2.3 V-Modell

4.2.4 Agile Softwareentwicklungmethoden

4.2.1 Allgemeine Vorgehensmodelle



- Wasserfallmodell
- Royce 1970

Arbeitsschritt kann erst abgeschlossen werden,
wenn alle vorgesehenen Artefakte fertiggestellt wurden

Probleme des Wasserfallmodells

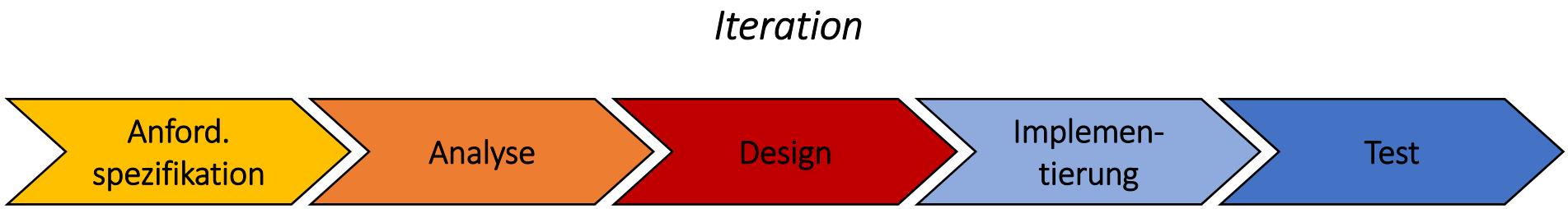
- geringe Planungssicherheit
- Kontrolle der Projektrisiken schwierig
- Implementierung zu spät verfügbar
  - „Look and feel“ der Kunden zu spät, deshalb Kundenwünsche zu spät
  - Grundsätzliche Probleme der Architektur zu spät sichtbar
  - Weitreichende Entscheidungen bei geringem Wissensstand
- keine Berücksichtigung von Wiederverwendung
  - problematisch: Integration von Altsystemen und eingekauften Komponenten

4.2.2 Inkrementeller Softwareentwurf

- Das System wird in Inkrementen (Ausbaustufen) entwickelt
  - Jedes Inkrement erweitert die Funktionalität des Systems
  - Jedes Inkrement ist ausführbar und qualitätsgesichert
  - Das erste Inkrement enthält eine Kernfunktionalität, die an den Anwendungsfällen mit höchstem Risiko orientiert ist
  - Die Inkremente können evtl. bereits an Endbenutzer ausgeliefert werden

Iteration

- „Iteration“ bezeichnet die Entwicklung eines Inkrements.
- Jede Iteration ist ein Miniprojekt, das wasserfallartig durchgeführt wird:



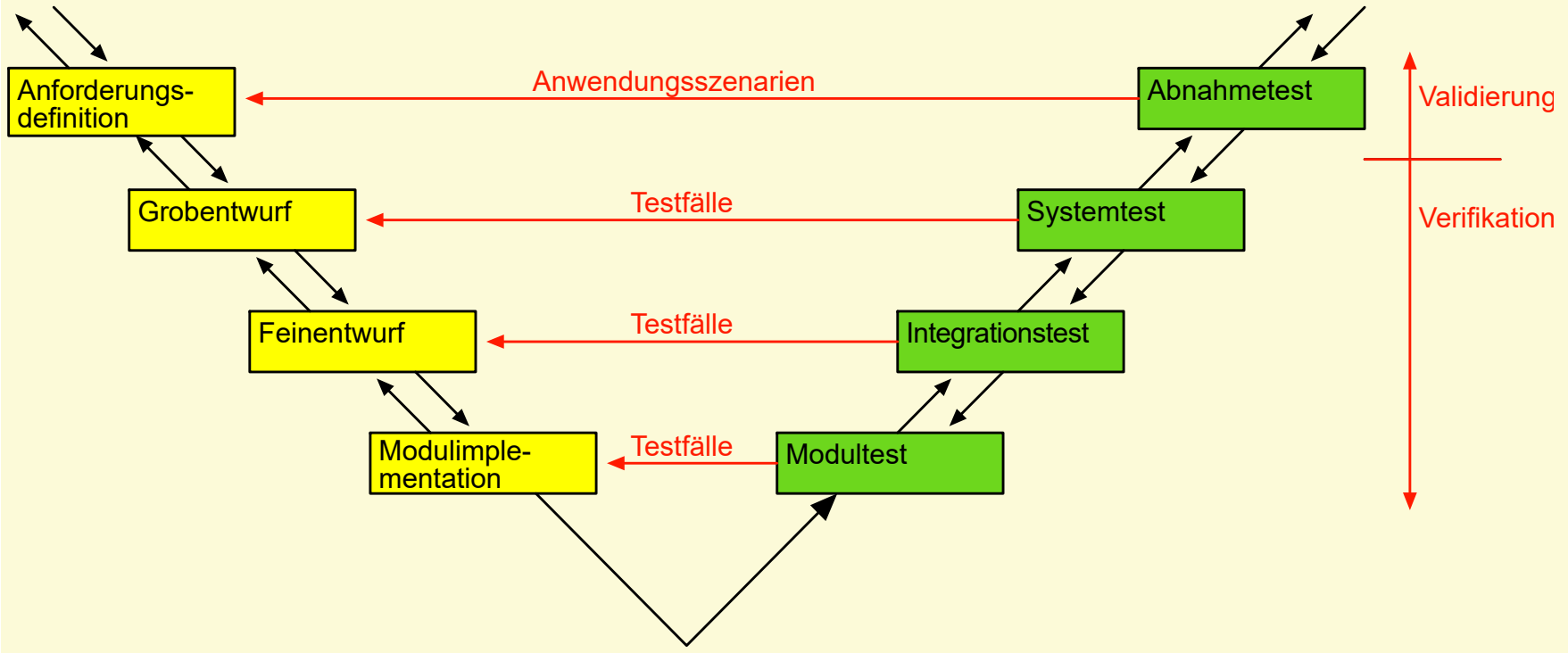
Warum Iterationen und Inkremente?

- Frühes Abschätzen und Kontrolle von Risiken
  - frühe Entwicklung einer Systemarchitektur hilft, technische Risiken früh zu erkennen (Performanz, Einsatz neuer Techniken, Einsatz von Altsystemen, ...)
- Bessere Handhabung von Änderungen
  - Kunden kommen früh mit dem System in Berührung
  - Berücksichtigung von Änderungen zu definierten Zeitpunkten
- Planungssicherheit im Entwicklungsprozess wird größer

4.2.3 V-Modell XT

- V-Modell XT - http://www.v-modell-xt.de/
- Fokus
  - Generisches Vorgehensmodell, das auf Unternehmen/Projekte angepasst werden kann (*Tailoring*)
  - Die erzeugten Produkte stehen im Mittelpunkt
  - Etabliertes Vorgehensmodell im Bereich sicherheitskritische Systeme (Embedded Systems)

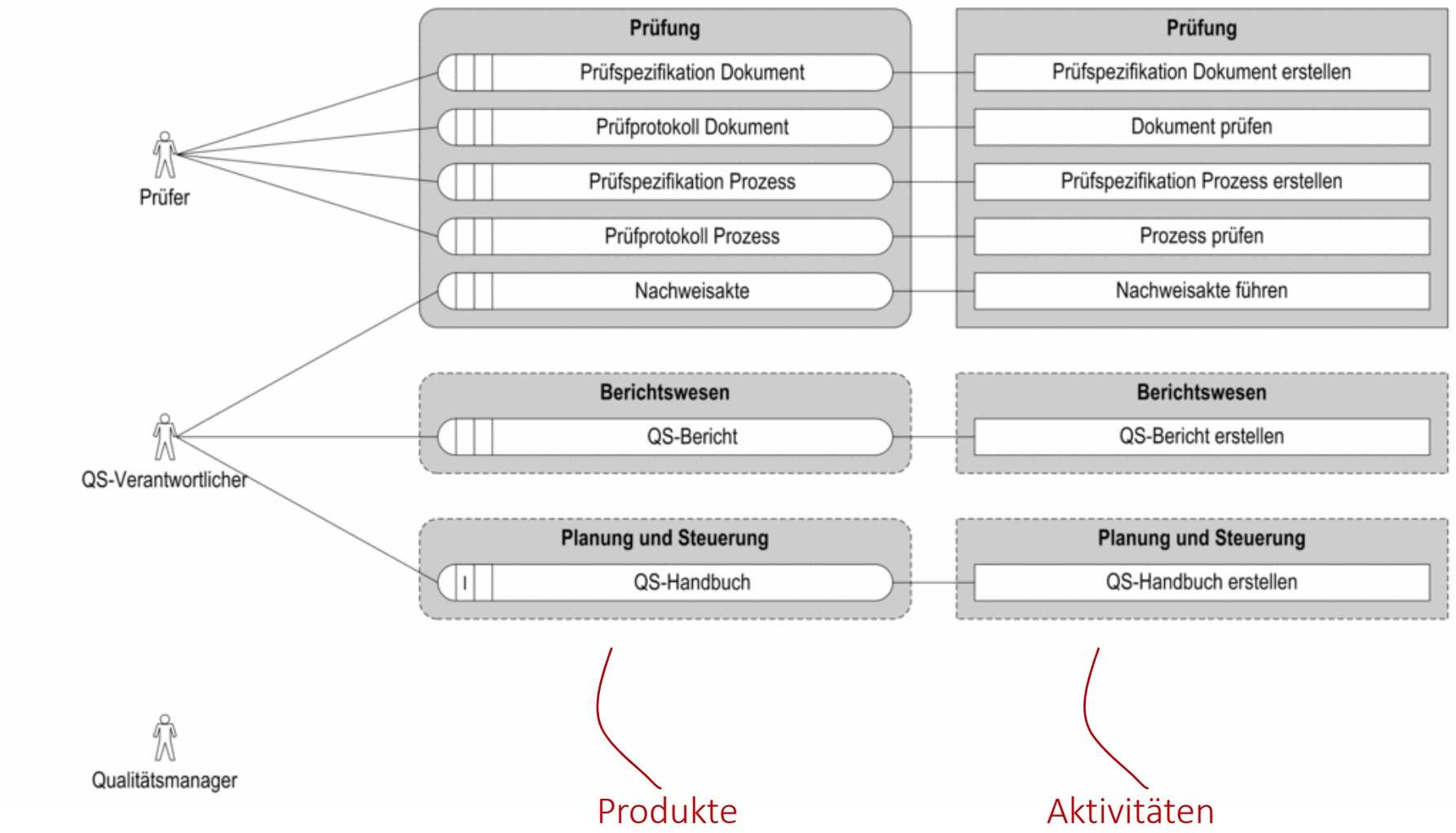
Das „V“ im V-Modell



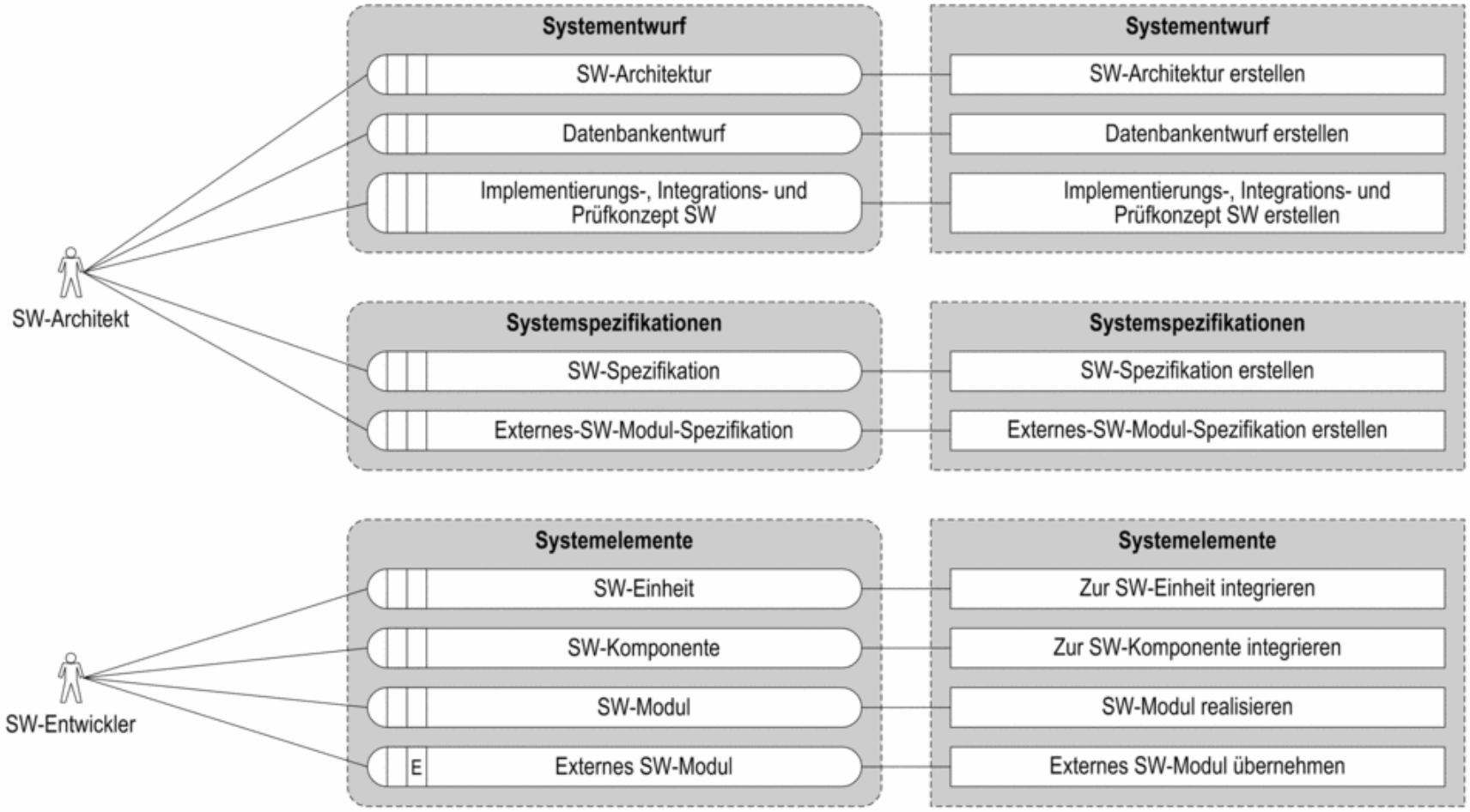
Validierung - Verifikation

- Validierung
  - Entsprechen die Anforderungen dem Bedarf des Nutzers?
  - „Are we building the right thing?“
- Verifikation
  - Erfüllt die Implementierung die Anforderungen?
  - „Are we building it right?“

Beispiel: Qualitätssicherung



SW-Entwicklung



Rollen im V-Modell

- 31 Rollen
- Beispiele
  - Akquisiteur
  - Anforderungsanalytiker (AG/AN)
  - Anwender
  - Ergonomieverantwortlicher
  - KM-Administrator
  - Lenkungsausschuss
  - Projektleiter
  - Projektmanager
  - Prüfer
  - QS-Verantwortlicher
  - SW-Architekt
  - SW-Entwickler
  - Systemarchitekt
  - Systemintegrator
  - Technischer Autor
  - Trainer

Beschreibung jeder Rolle:
- Kurzbeschreibung
- Aufgaben und Befugnisse
- Fähigkeitsprofil
- Rollenbesetzung
- Verantwortlich für Produkt …
- Mitwirkend bei Produkt …

4.2.4. Agile Softwareentwicklungsmethoden

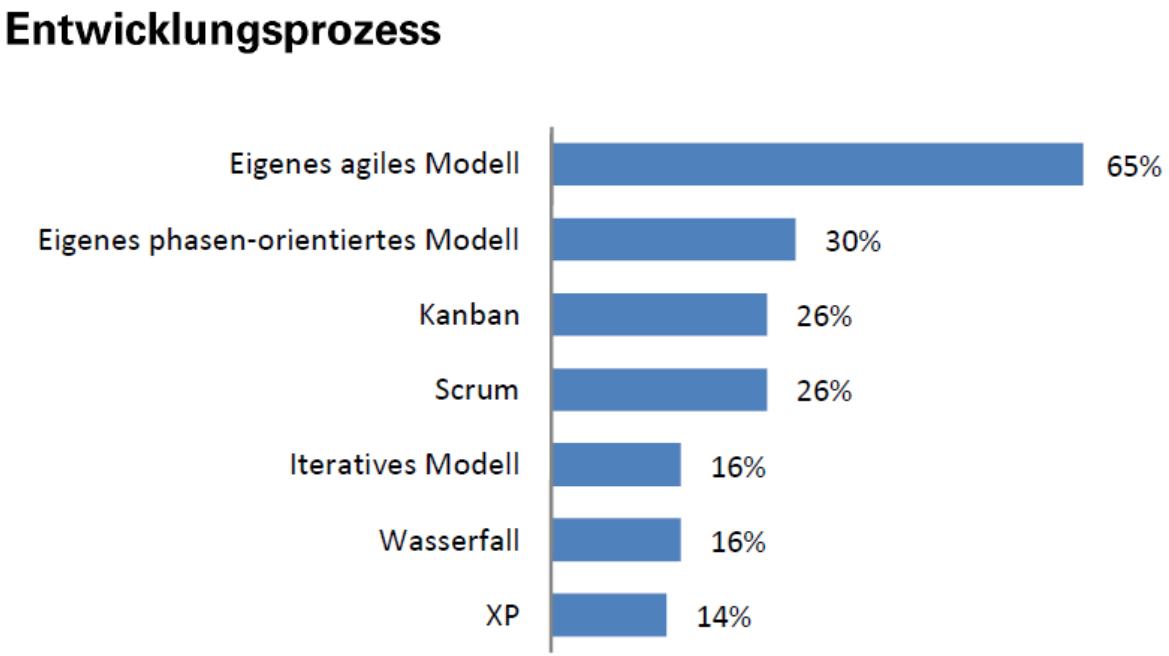
- Reaktion auf die dokumentengetriebenen Vorgehensmodelle der 1990er (z.B. V-Modell)
- Schwerpunkt auf Flexibilität, soziale Faktoren und ausführbare Artefakte

Quellen: Foliensatz Barbara Weber, Stefan Zugal, Jakob Pinggera

Übersicht

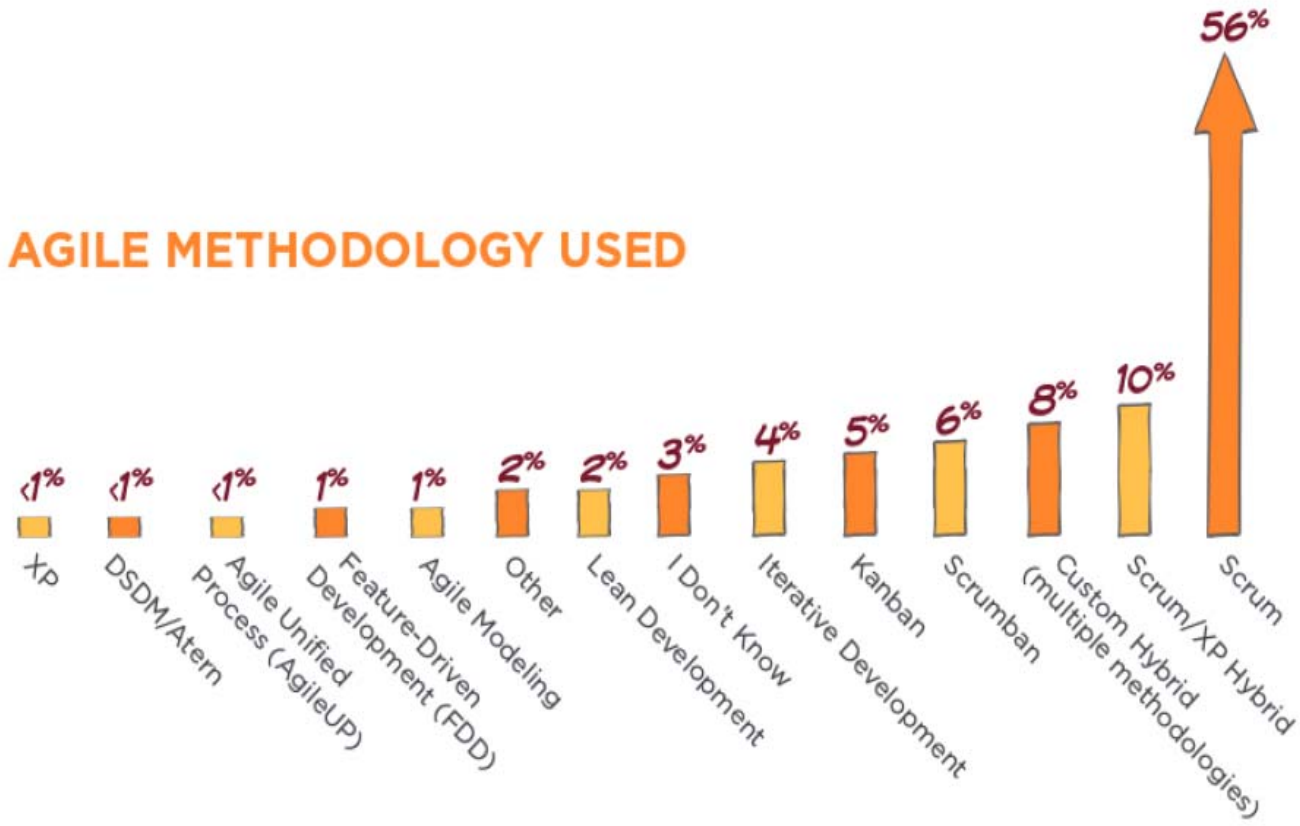
- 4.2.4.1 XP
- 4.2.4.2 Scrum
- 4.2.4.3 Allgemeine Agile Praktiken
  - Definition-of-Done
  - Planning Game
  - Testgetriebene Entwicklung und Refactoring
  - Kanban

Aktueller Stand D-A-CH



Quelle:
https://www.software-quality-lab.com/fileadmin/files/download/KnowledgeLetterPreview/SWQL-KnowledgeLetter044-Umfrage%20SW-QM%202015_prev.pdf

Etablierte Agile Methoden



Source: 9th State of Agile Survey: http://stateofagile.versionone.com/

Das Agile Manifesto

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

http://agilemanifesto.org/principles.html

(Kent Back, Alistair Cockburn, Martin Fowler et al., February 2001)

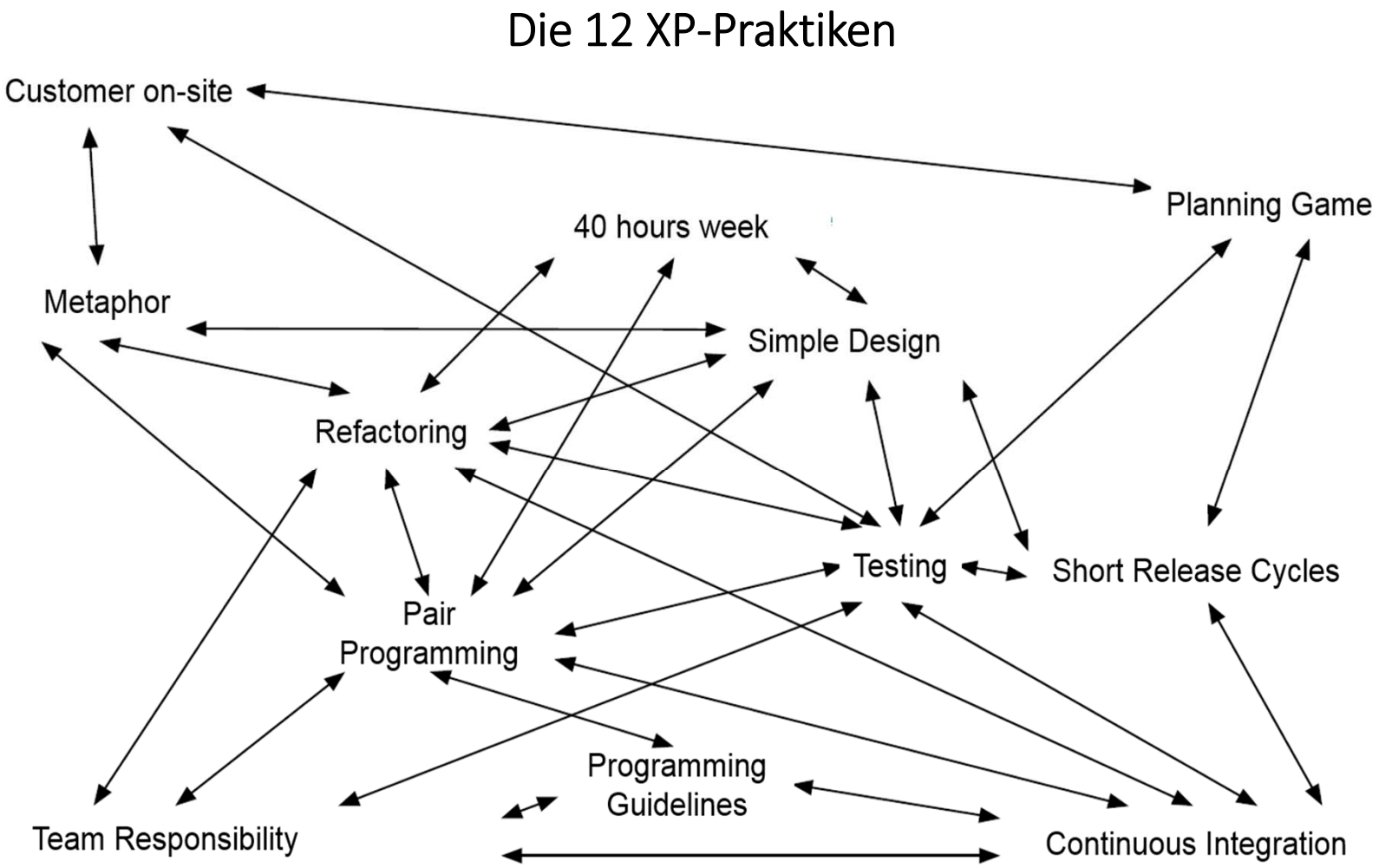
Prinzipien Agiler Softwareentwicklung (1)

- Stelle den Kunden durch frühzeitige und regelmäßige Auslieferung der Software zufrieden
- Agile Prozesse wehren sich nicht gegen Änderungen, sondern machen schnelle Änderungsprozesse zu einem Wettbewerbsvorteil.
- Lauffähige Software soll häufig, in Abständen von wenigen Wochen bis Monaten ausgeliefert werden.
- Lauffähige Software ist das wichtigste Maß für den Projektfortschritt.
- Ständige Aufmerksamkeit auf technische Qualität und gutes Design verbessert die Agilität.

Prinzipien Agiler Softwareentwicklung (2)

- Fachexperten und Entwickler müssen während des gesamten Projekts täglich zusammenarbeiten.
- Baue Projekte um motivierte Individuen herum auf. Stelle ihnen die benötigen Umgebung zur Verfügung und vertraue auf sie.
- Die effizienteste und effektivste Methode zur Verteilung von Wissen ist ein persönliches Gespräch.
- Einfachheit – die Kunst, die Arbeit, die nicht getan wird, zu maximieren – ist entscheidend.

Extreme Programming (XP)



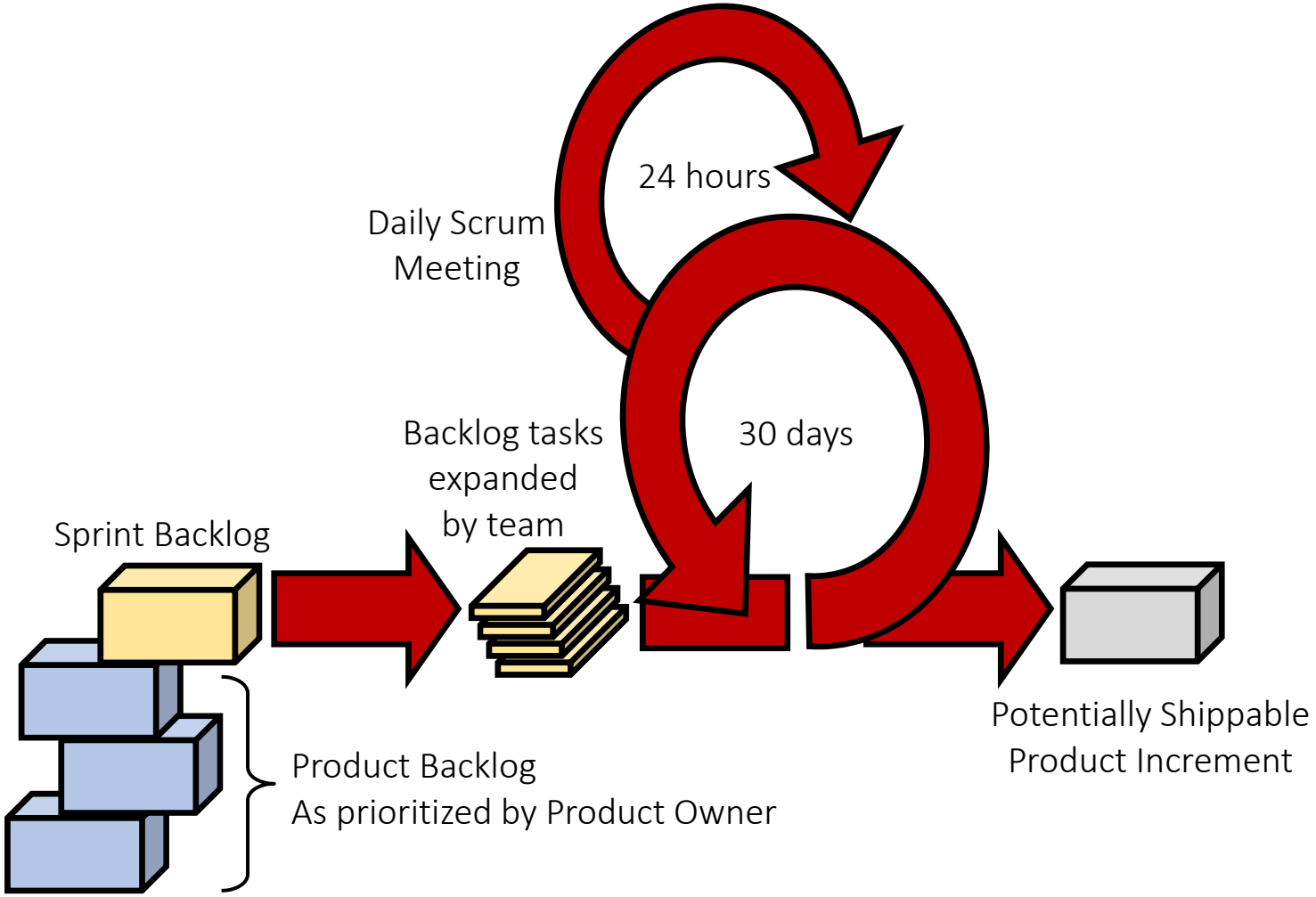
XP-Praktiken (contd.) (1)

- Small Releases – sehr kurze Iterationszeiten
- Planning Game – unkompliziertes Schätzen von Aufwänden
- Metaphor – Finde eine einfache Metapher, um das System zu beschreiben
- Simple Design – das Design soll so einfach wie möglich sein
- Testgetriebene Entwicklung und Akzeptanztests – kontinuierliches Testen des Codes, Tests werden vor dem Code geschrieben

XP-Praktiken (contd.) (2)

- Pair Programming – Zwei Entwickler sitzen vor der Maschine
- Collective Ownership – Der Code gehört dem gesamten Team, jeder darf den Code überall ändern
- Continuous Integration – Halte den Code mehrere Male am Tag lauffähig
- 40-Stunden-Woche – arbeite in tragfähigem Tempo und Dauer
- Refactoring – Richte deine Aufmerksamkeit auf qualitätsvollen Code
- Programming Guidelines – Verwende Programmierstandards zur Verbesserung der Kommunikation
- Customer-on-Site – Halte ständigen Kontakt zum Kunden/User

Scrum



Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle

Scrum: Rollen

- Die Kern-Rollen (Pigs)
  - Product owner: Stimme des Kunden
  - Scrum Master: „Facilitator“
    - Puffer zwischen Team und störenden Einflüssen von außen
    - kein Projektleiter (Entwicklungsteam hat hohes Maß an Selbstverantwortung)
    - Verantwortlich für Realisierung von Scrum-Konzepten
  - Entwickler:
    - verantwortlich für Inkremente
    - Teamgröße: 7 +/- 2
    - Cross-functional (Programmierer, QA, UI Designers etc.)
- Hilfsrollen (Chickens)
  - User, Manager, …

Chickens and Pigs

A chicken and a pig are together when the chicken says, “Let’s start a restaurant!”

The pig thinks it over and says, “What would we call this restaurant?” The chicken says, “Ham n’ Eggs!”

The pig says, “No, thanks.” I’d be committed, but you’d only be involved!”

Scrum: Artefakte und Konzepte /1

- Sprint
  - Synonym zu „Inkrement“ verwendet
  - Zielfrequenz: monatliche Sprints
    - +/- ein bis zwei Wochen
    - Ziel: konstante Dauer
    - Keine Änderung der Anforderungen während eines Sprints

Scrum: Artefakte und Konzepte /2

- Product Vision
  - Ziele des geplanten Produkts?
  - Zielgruppe und Zielsegment am Markt?
  - Konkurrenten?
  - Konkurrenzvorteil?

Scrum: Artefakte und Konzepte /3

- Product Backlog
  - Scope des Projekts
    - Meist definiert durch Liste von (bewerteten) User Stories („User soll Liste der Kunden filtern und sortieren können“) + Tasks („Exception Handling verbessern“)
    - Wird laufend erweitert
    - Sicht des Kunden
    - Bewertung in Form von „Story Points“ (vgl. Planning Game)
  - Verantwortlicher: Product Owner (Inhalt und Priorisierung)

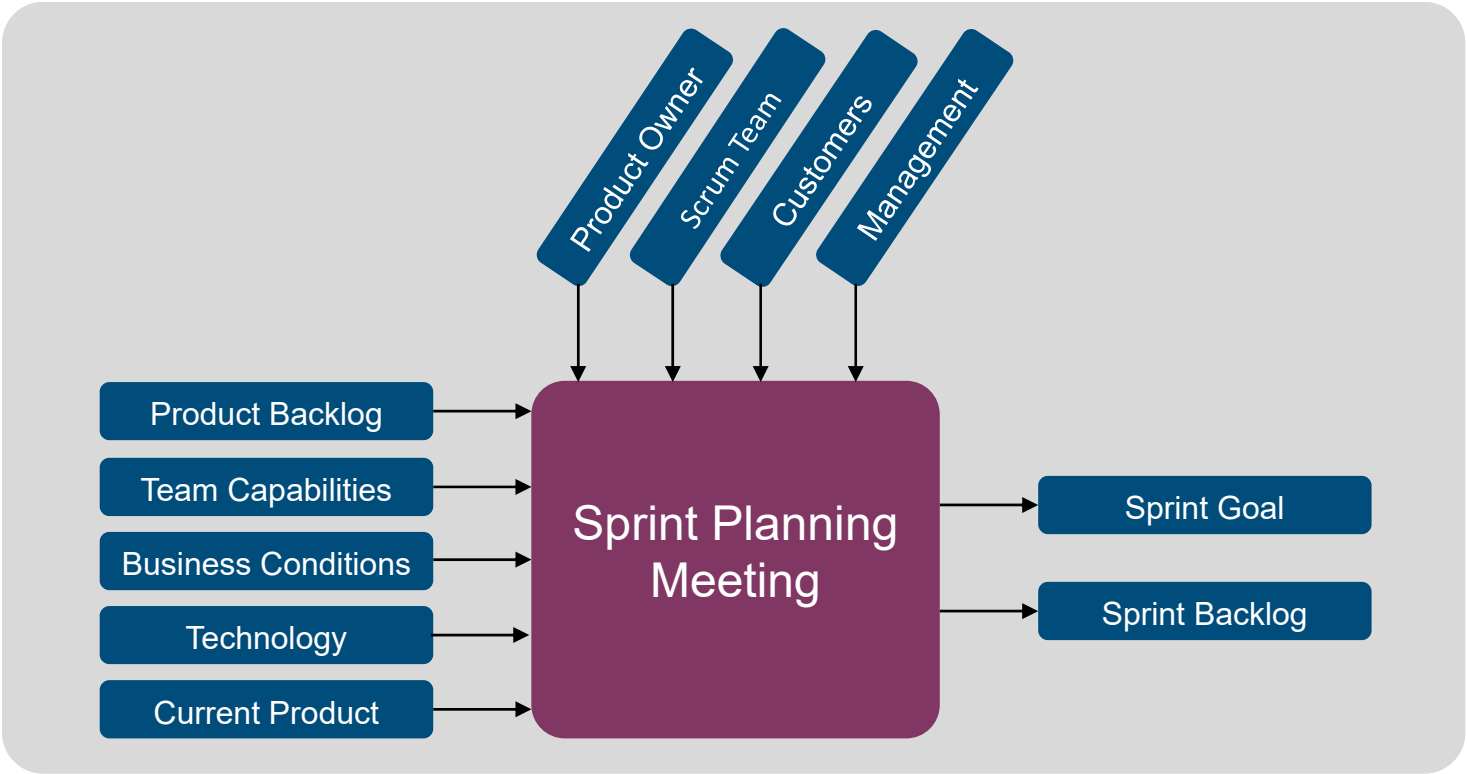
Beispiel Product Backlog

	Item #	Description	Est	By
Very High				
	1	**Finish database versioning**	20	KH
	2	**Get rid of unneeded shared Java in database**	8	KH
	-	**Add licensing**	-	-
	3	Concurrent user licensing	20	TG
	4	Demo/Eval licensing	20	TG
		Analysis Manager		
	5	File formats we support are out of date	40	TG
	6	Round-trip Analyses	100	MC
High				
	-	**Enforce unique names**	-	-
	7	In main application	20	KH
	8	In import	20	AM
	-	**Admin Program**	-	-
	9	Delete users	2	JM
	-	**Analysis Manager**	-	-
	10	When items are removed from an analysis, they should show up again in the pick list in lower 1/2 of the analysis tab	8	TG
	-	**Query**	-	-
	11	Support for wildcards when searching	20	T&A
	12	Sorting of number attributes to handle negative numbers	20	T&A
	13	Horizontal scrolling	13	T&A
	-	**Population Genetics**	-	-
	14	Frequency Manager	100	T&M
	15	Query Tool	100	T&M
	16	Additional Editors (which ones)	40	T&M
	17	Study Variable Manager	40	T&M
	18	Haplotypes	100	T&M
	19	**Add icons for v1.1 or 2.0**	-	-
	-	**Pedigree Manager**	-	-
	20	Validate Derived kindred	2	KH
Medium				
	-	**Explorer**	-	-
	21	Launch tab synchronization (only show queries/analyses for logged in users)	8	T&A
	22	Delete settings (?)	2	T&A

Quelle: http://epf.eclipse.org/wikis/scrum/Scrum/guidances/examples/example_product_backlog_B6A03674.html

Scrum: Artefakte und Konzepte /4

- Sprint Planning Meeting



Scrum: Artefakte und Konzepte /5

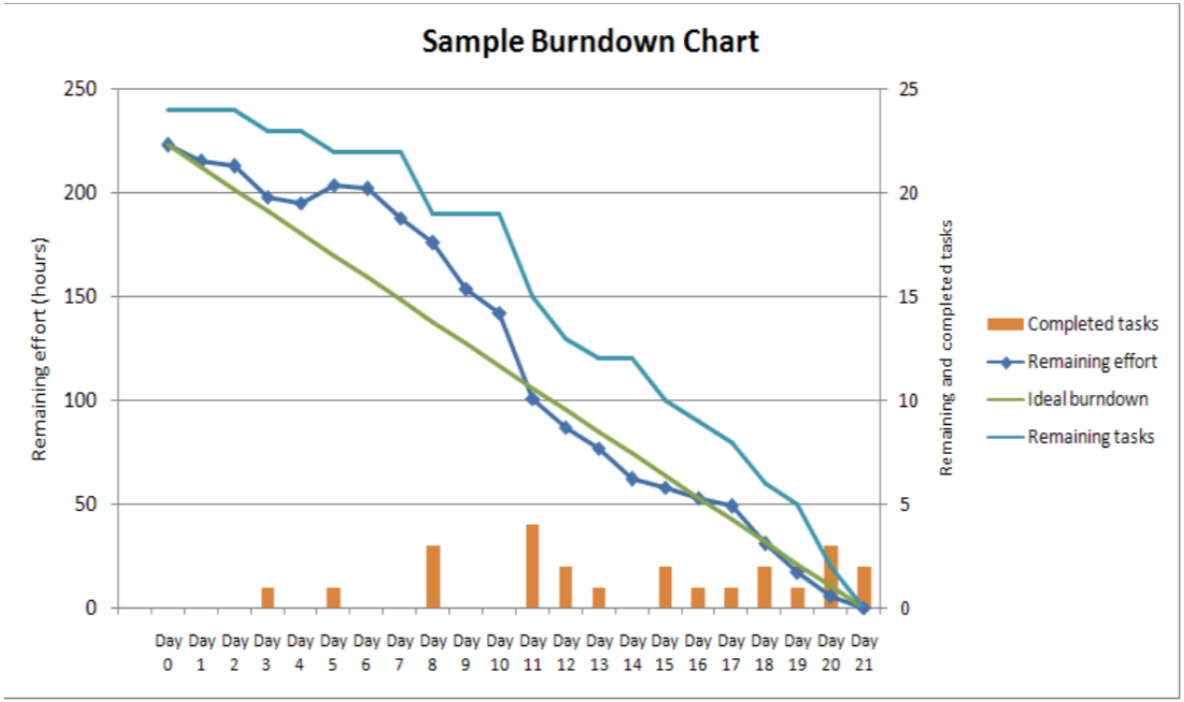
- Sprint Goal – Übergeordnetes „Thema“ des Sprints
  - „Power-User soll durch Statistiken unterstützt werden“
  - „Die Anbindung von System X soll durch real-time streaming unterstützt werden“
- Sprint Backlog
  - Herunterbrechen des Sprint Goals und der ausgewählten User Stories auf Tasks
  - Selbst-Organisation des Entwicklungsteams

Beispiel Sprint Backlog

	Days Left in Sprint	**15**	**13**	**11**	**8**	4
Who	**Description**	**24.04.20XX**	**26.04.20XX**	**28.04.20XX**	**02.05.20XX**	
	Total Estimated Hours:	**200**	**156**	**96**	**78**	
-	**User's Guide**	-	-	-	-	
SM	Start on Study Variable chapter first draft	16	16	16	16	
SM	Import chapter first draft	40	24	6	6	
SM	Export chapter first draft	24	24	24	6	
	Misc. Small Bugs					
JM	Fix connection leak	10				
JM	Delete queries	8	8			
JM	Delete analysis	8	8			
TG	Fix tear-off messaging bug	8	8			
JM	View pedigree for kindred column in a result set	2	2	2	2	
AM	Derived kindred validation	8				
	Environment					
TG	Install Jira	16	16			
TBD	Move test cases into Jira	40	40	40	40	
	Database					
KH	Killing Oracle sessions	8	8	8	8	
KH	Finish xxx database patch	8	2			
KH	Figure out why 461 indexes are created	4				

Scrum: Artefakte und Konzepte /6

- Burndown Chart
  - Verantwortlich: Entwicklungsteam



  - Team Velocity = Erledigte Story Points / Sprint

[source: wikipedia]

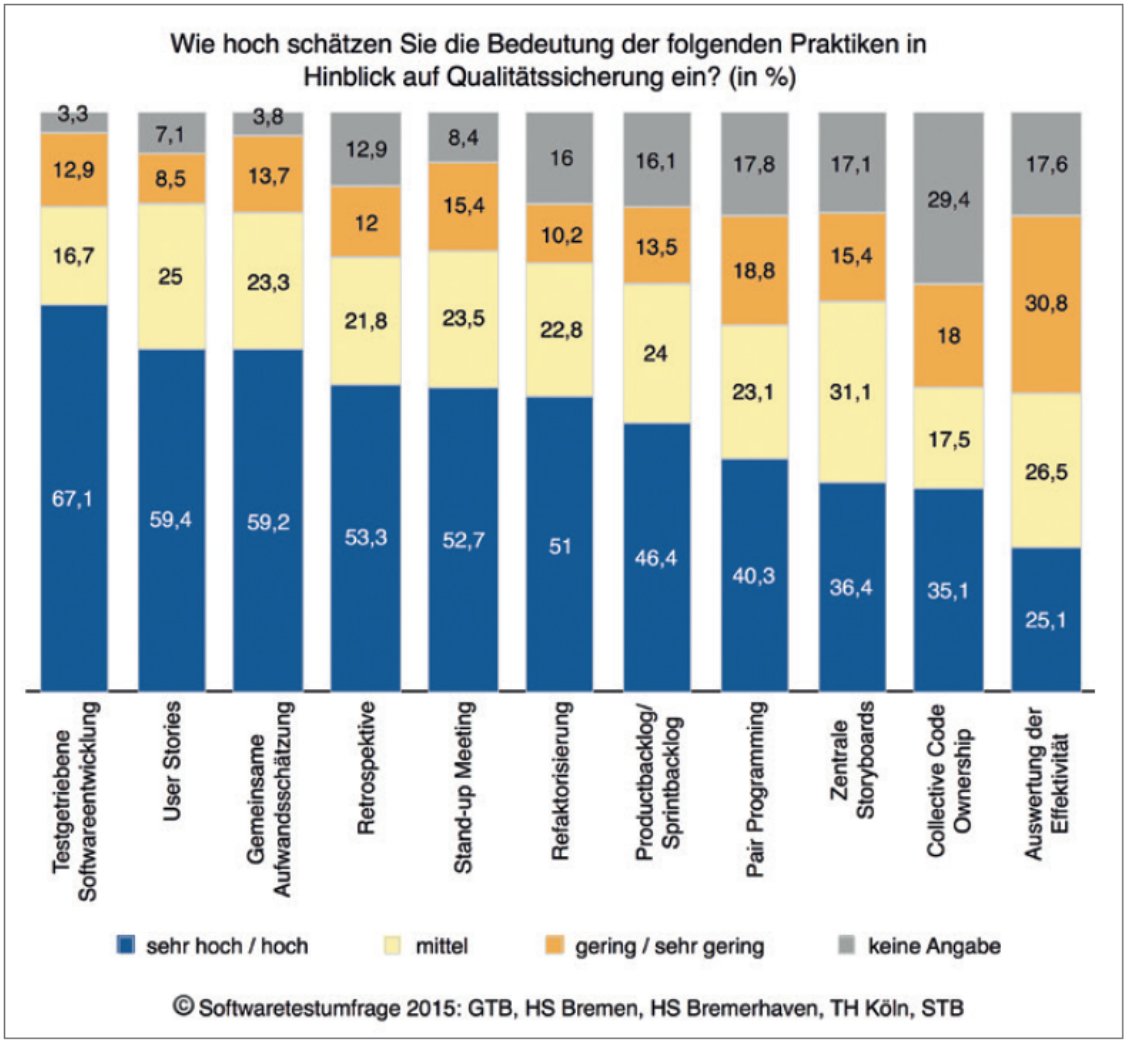
Scrum: Artefakte und Konzepte /7

- Daily Scrum Meetings
  - Täglich 15 Minuten Stand-Up Meeting
  - Keine Diskussion inhaltlicher Natur, sondern…
- Drei Fragen
  - Was hast du gestern gemacht?
  - Was machst du heute?
  - Gibt´s Probleme?
- Chicken und Pigs sind eingeladen, aber nur Pigs reden
- Nicht durch E-Mails ersetzbar
  - Das ganze Team soll jeden Tag einen Blick auf das Gesamtprojekt erhalten
  - Druck, zu halten, was man verspricht

Warum nicht agil?

- Die häufigsten Fallstricke und Herausforderungen
  - Kunden mischen sich zu sehr ein
  - Vertragliche Abwicklung erfordert viel Vertrauen zwischen Kunde und Softwaredienstleister (kein Werkvertrag möglich)
  - Unvorhersehbare Auslieferung des Endprodukts
  - Prozess ist für den Kunden zu transparent
  - Mangelnde Dokumentation
  - Keine kohärente SW-Architektur (zu sehr Fokus auf Einfachheit)
  - Sozialer Druck, Gruppenkonflikte

Aktueller Stand D-A-CH



Quelle: http://www.sq-magazin.de/de/magazin/sq-magazin-ausgabe-40.html

4.2.4.3 Allgemeine Agile Praktiken

- Definition-of-Done
- Planning Game
- Testgetriebene Entwicklung und Refactoring

Definition-of-Done (DoD) /1

- Kriterien, wann Arbeitsergebnisse auf „done“ gesetzt werden dürfen
- Zwei Sichten
  - Produktsicht (Orientierung an Qualitätskriterien des Produkts)
  - Prozess-Sicht (Definition von Aufgaben und Tätigkeiten, die durchgeführt sein müssen)
- DoD auf unterschiedlichen Ebenen
  - Pro User Story oder Backlog Item
  - Pro Sprint
  - Pro Release

SMARTe Ziele für DoD-Kriterien

- Spezifisch: DoD-Kriterien sollen eindeutig und so präzise wie möglich definiert sein
- Messbar
- Akzeptiert
- Realistisch
- Testbar

Anti-Beispiel

Quelle der nächsten Folien: https://www.software-quality-lab.com/fileadmin/files/download/KnowledgeLetterPreview/SWQL-KnowledgeLetter045-Definitions%20of%20Done_prev.pdf

DoD-Kriterien User Stories Beispiel-Checkliste

DoD-Kriterien für Story-Fertigstellung
Ist die Anforderung vollständig implementiert? (aus Sicht des PO) Sind die Bedürfnisse der Anwender abgedeckt?
Ist die Benutzbarkeit gegeben und als brauchbar bewertet?
Manuelle Abnahmetestfälle wurden mindestens einmal ausgeführt mit: ▪ Positivtests des gewünschten Verhaltens ▪ Negativtests des Verhaltens im Fehlerfall und bei Falscheingaben
Sind Unit-Tests gemäß Testkonzept erstellt und erfolgreich durchlaufen?
Wurden automatisierte Schnittstellen und GUI-Tests gemäß Testkonzept erstellt und erfolgreich durchgelaufen?
Wurden Tests mit produktivnahen Testdaten durchgeführt?

DoD-Kriterien Sprint-Ende Beispiel-Checkliste

DoD-Kriterien für Sprint-Ende
Alle für den Sprint festgelegten User Stories sind im Status „Done“
Alle für den Sprint festgelegten User Stories durch den Kunden wurden einem Abnahmetest unterzogen
Integrations- und Systemtests wurden gemäß DoD „Integrations- und Systemtestkriterien“ erfolgreich durchgeführt
In den Tests wurden keine Fehler der Klasse 1 und 2 gefunden und max. 5 Fehler der Klasse 3
Alle Tests für User Stories der Kritikalität 1 wurden automatisiert
Gesamter Quellcode ist eingecheckt
Erfolgreicher Build des Gesamtsystems wurde durchgeführt
Die neuen Teile arbeiten fehlerfrei mit bestehenden Modulen zusammen
Code-Review für die neuen Teile wurde positiv durchgeführt
Die notwendigen Dokumentationen wurden erstellt bzw. angepasst: ▪ Benutzerhandbuch ▪ Konfigurations- und Parameterdokumentation ▪ Installations- und Administrationshandbuch ▪ Architekturdokumentation ▪ Quellcodedokumentation ▪ What‘s-New-Liste ▪ Behobene Fehler-Liste ▪ Sonstige relevante Dokumentation

DoD im Prozess

- Ziel: Integration der DoD-Kriterien in den Prozess, z.B.
  - Vier-Augen-Prinzip
  - Pair-Programming
  - Projektbegleitender Qualitätsmanager
  - DoD-Checklisten in Tools einbauen
  - Workflow-Tool für DoD

Planning Game

- Ziel: Schätzen von Aufwänden (hier: User Stories)
- Technik:
  - Benutzen von Stellvertretergrößen („Story Points“ = relative Größe)
  - Konsensbildung

Planning Poker Regeln

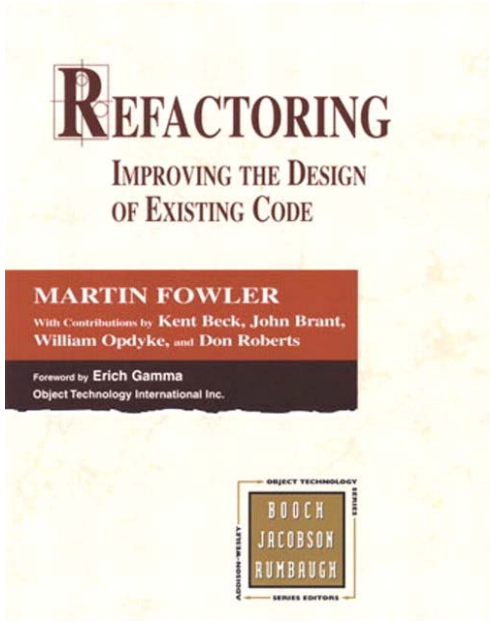
Quelle: Wikipedia

- Spielkarten mit Werten ähnlich zu Fibonacci-Zahlen
- Neutraler Moderator
- Product Owner stellt User Story vor + Diskussion
- Entwickler wählen Schätzkarte verdeckt – alle drehen die Karten gleichzeitig um
- Niedrigste und höchste Wertung werden diskutiert
- Erneute verdeckte Schätzung bis ein Konsens erreicht ist
- Diskussionen werden mit Timer beschränkt

Testgetriebene Entwicklung und Refactoring

“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

Martin Fowler



Refactoring

- ***Refactoring*** *(noun): a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior.*
- ***Refactor*** *(verb): to restructure software by applying a series of refactorings without changing its observable behavior.*

Die Zwei Hüte Metapher

Neue Funktionalität

- Erweitere das System um neue Funktionalität
- Füge neue Testfälle hinzu
- Lasse die Testfälle laufen

Refactoring

- Restrukturiere den Code
- Lasse die Testfälle laufen

Wechsle die Hüte, aber ziehe jeweils nur einen an

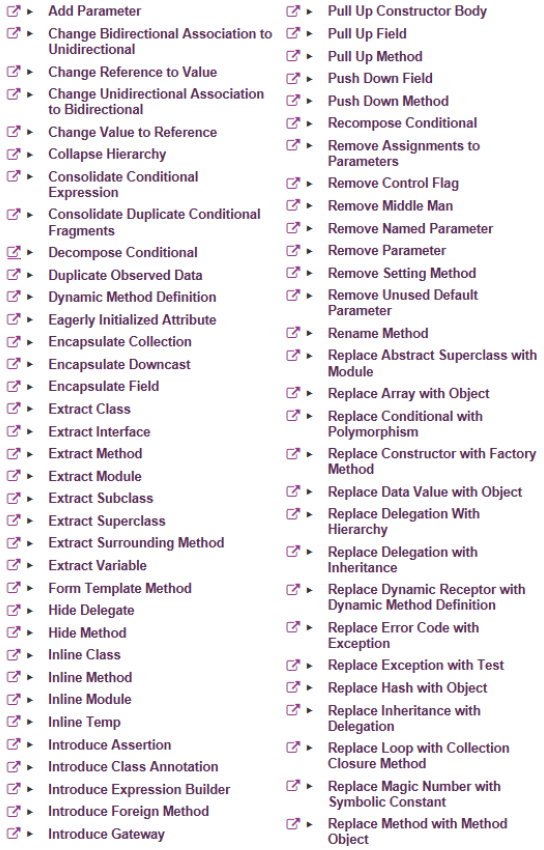
Gründe für Refactoring

- Verbesserung der Code-Struktur
- Code “aufräumen”
- Mache den Code leichter verständlich
- Erleichtere anderen, mit dem eigenen Code zu arbeiten
- Erleichtere das Finden von Fehlern
- Erleichtere die Erweiterbarkeit des Codes
- Produktivität über die Zeit erhalten
- Code Smells

Some Code Smells

- Duplicate Code
- Long Method
- Large Class
- Long Parameter List
- Shotgun Surgery – change with ripple effects
- Feature Envy – method uses parts from other class
- Switch Statements – lack of object orientation?

Fowler´s Refactoring Catalog



- Link : https://refactoring.com/catalog
- Code Smells and Refactorings: http://www.industriallogic.com/wp-content/uploads/2005/09/smellstorefactorings.pdf