Agile Softwareentwicklung mit Scrum

Gregor Liebermann M.Sc.

Aufbau der Vorlesung

Montags 15:40 – 18:40

5 CP

Aufteilung in Vorlesung (15:40 - 17:10) und Praktikum (17:10 - 18:40)

Praxisanteil größer als Vorlesungsanteil

Aufteilung in zwei Teams für das Praktikum

ca. zehn Praktikumsblöcke (acht Teilnahmen nötig für Prüfung)

Prüfung: 60 min ohne Hilfsmittel

Masterstudenten: Vortrag oder Studienarbeit

Kontaktdaten

gregor.liebermann@gmail.com

Empfohlene Literatur

Titel	Autor	Verlag	ISBN
Erfolgreich mit Scrum - Einflussfaktoren Personalmangement	Boris Gloger André Häusling	Hanser	978-3-446-42515-6
Scrum	Gloger B.	Hanser	978-3-446-41495-2
Scrum	Pichler Roman	dpunkt.verlag	978-3-89864-478-5
Agile Softwareentwicklung	Henning Wolf Wolf-Gideon Bleek	dpunkt.verlag	978-3898647014
Die Kraft von Scrum	Henning Wolf	dpunkt.verlag	978-3-89864-478-5

Empfohlene Seiten

http://www.scrumalliance.org/

http://www.it-agile.de/

http://www.scrum.org/

http://scrum-fibel.de/

Aufbau Vorlesung

Theorie im Vorlesungsstil

Praxis „Scrum“

Praxis „Teambuilding“

Gliederung Vorlesung

Einführung

Warum brauchen wir ein Vorgehensmodell?

Was bedeutet „agil“?

Was ist Scrum?

Wie funktioniert ein Team?

Andere agile Methoden

Agile skalierend

Vergleich verschiedenster Vorgehensweisen

Gastvortrag

Tools aus der Praxis

Gliederung Praktikum

Projektvorstellung und Gruppeneinteilung

Visionsworkshop

Design Thinking

Story Workshop

Estimation Meeting

Team Workshop 1 – Kennenlernen

Sprint Planning

Team Workshop 2 – Blind Colors

Review/Retrospektive

Team Workshop 3 – Kommunikation (DISG)

Kanban/Agile Games

Feedback/Wünsche des Kurses

t.b.d.

Software-Prozess-Modell

keine reine Programmiertätigkeit

Entwicklung hin zur Ingenieurtätigkeit (Software Engineering)

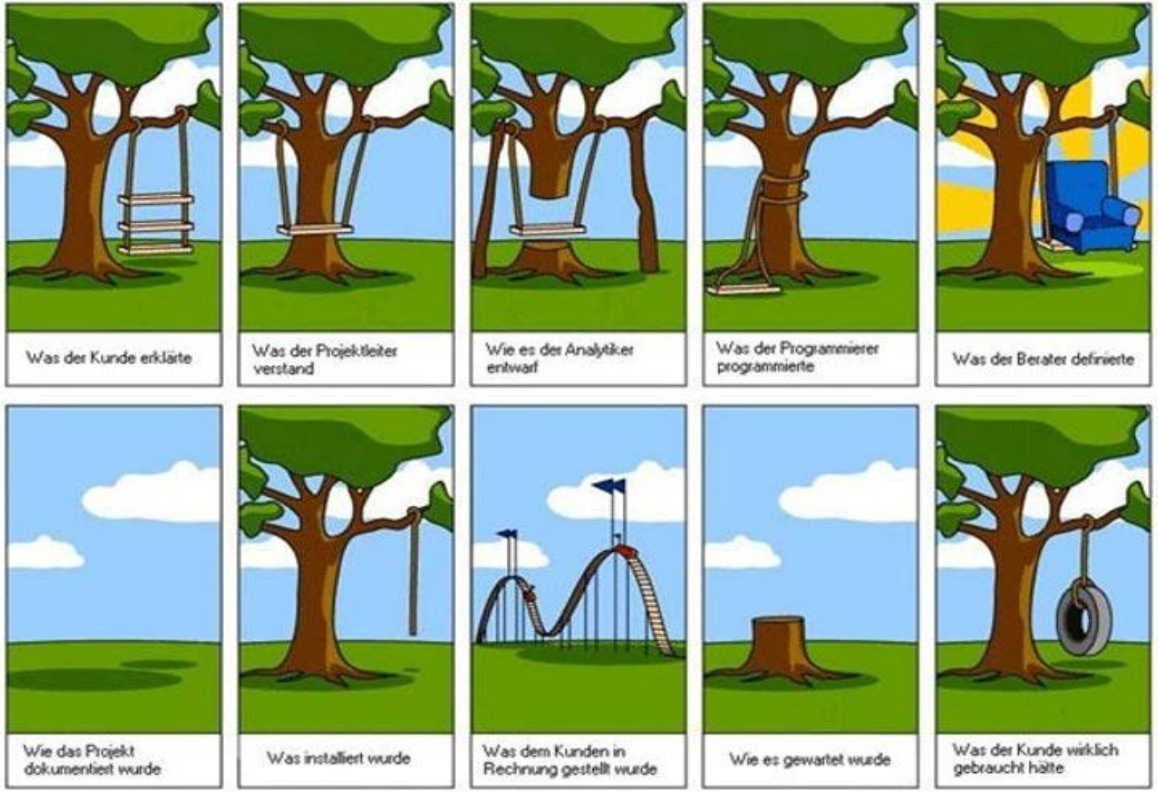
Softwareentwicklung wird zu einem Prozess

Was definiert ein Projekt?

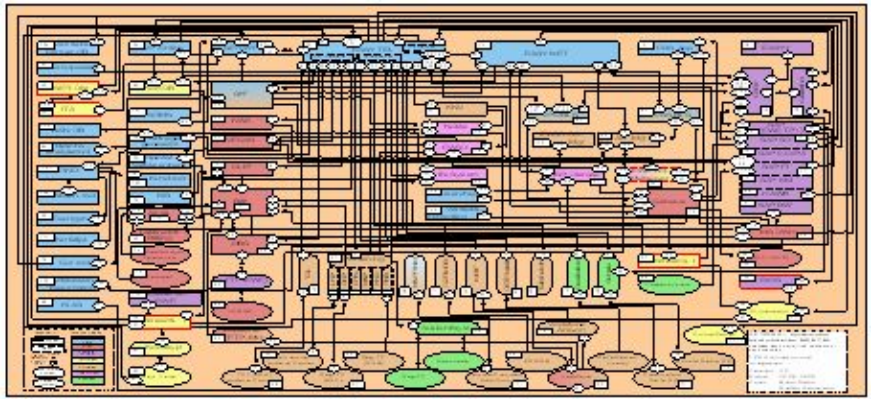
Das Ergebnis eines Projektes ist ein Produkt

- Einmaligkeit für das Unternehmen Zusammensetzung aus Teilaufgaben
- Beteiligung von Personen und/oder Stellen unterschiedlicher Fachrichtungen
- Teamarbeit
- Konkurrenz mit anderen Projekten um Personal- und Sachmittel
- Mindestdauer bzw. Mindestaufwand
- Höchstdauer bzw. Höchstaufwand
- definierter Anfang und definiertes Ende (=Ziel)

Der Klassiker



Probleme bei der professionellen Entwicklung von Softwaresystemen



Systeme werden immer komplexer

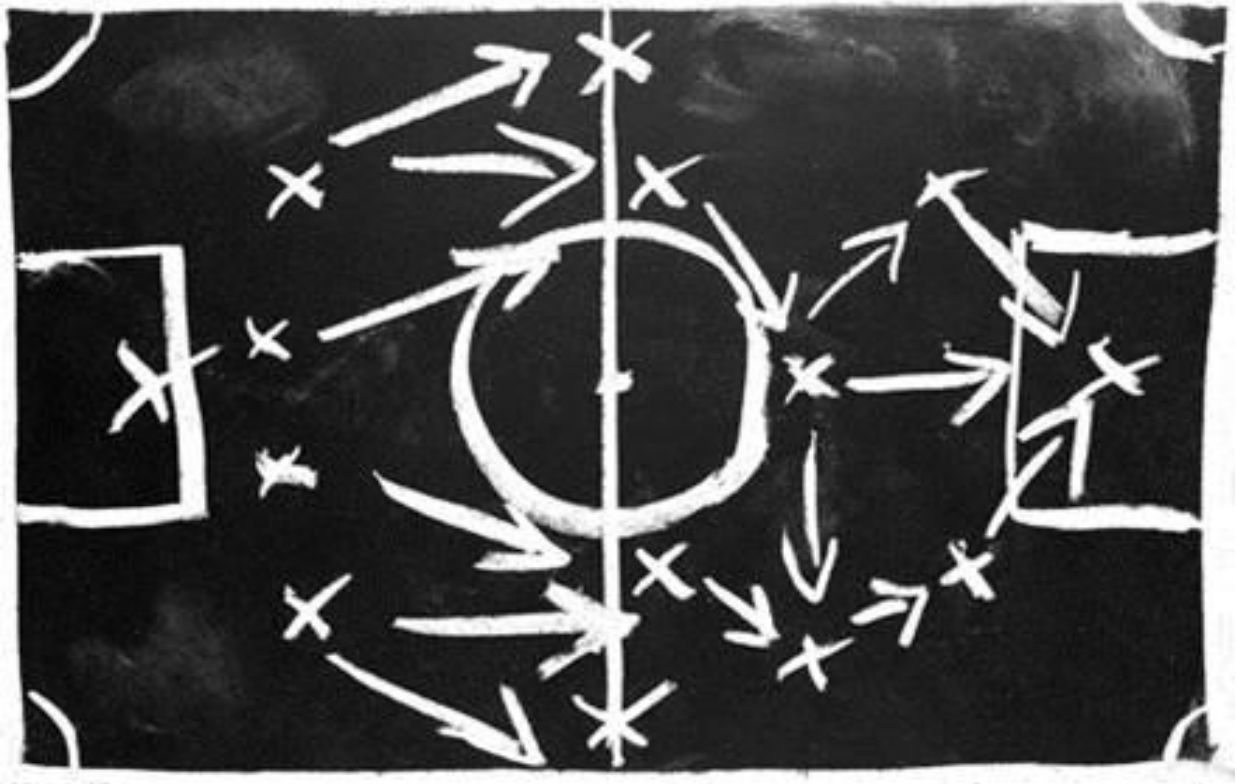
Entwicklung wird komplizierter

Wartung wird schwieriger

Schätzungen werden unübersichtlich

Überblick geht verloren

Lösung? Ein Plan muss her!



Anforderungen an die Softwareentwicklung

immer kürzere Zeiträume

immer mehr Anforderungen

Anforderungen an Qualität steigen dabei kontinuierlich

geforderte Software-Qualität

Prozessmodell beschreibt organisatorischen Rahmen

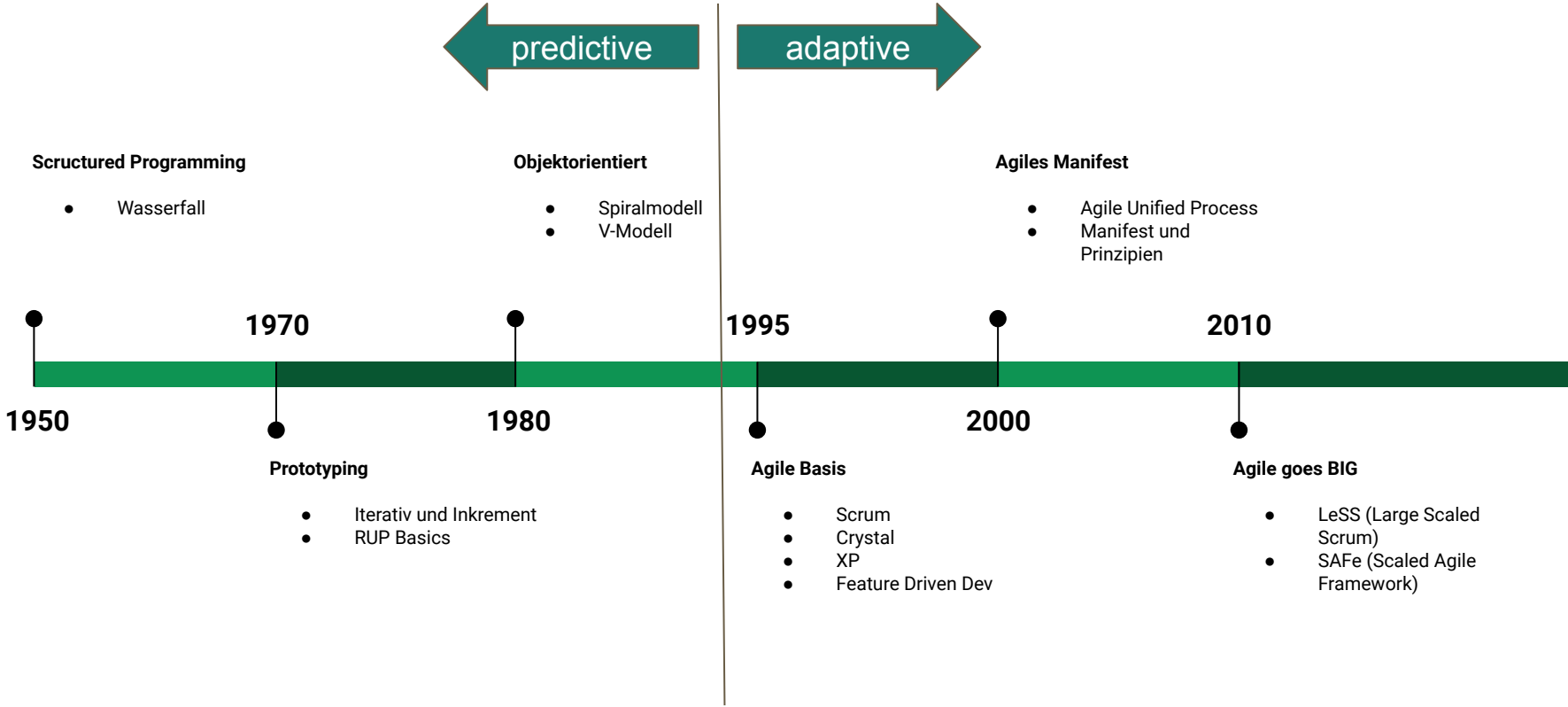
Wir brauchen ein Vorgehensmodell!

Das sagt Wikipedia:

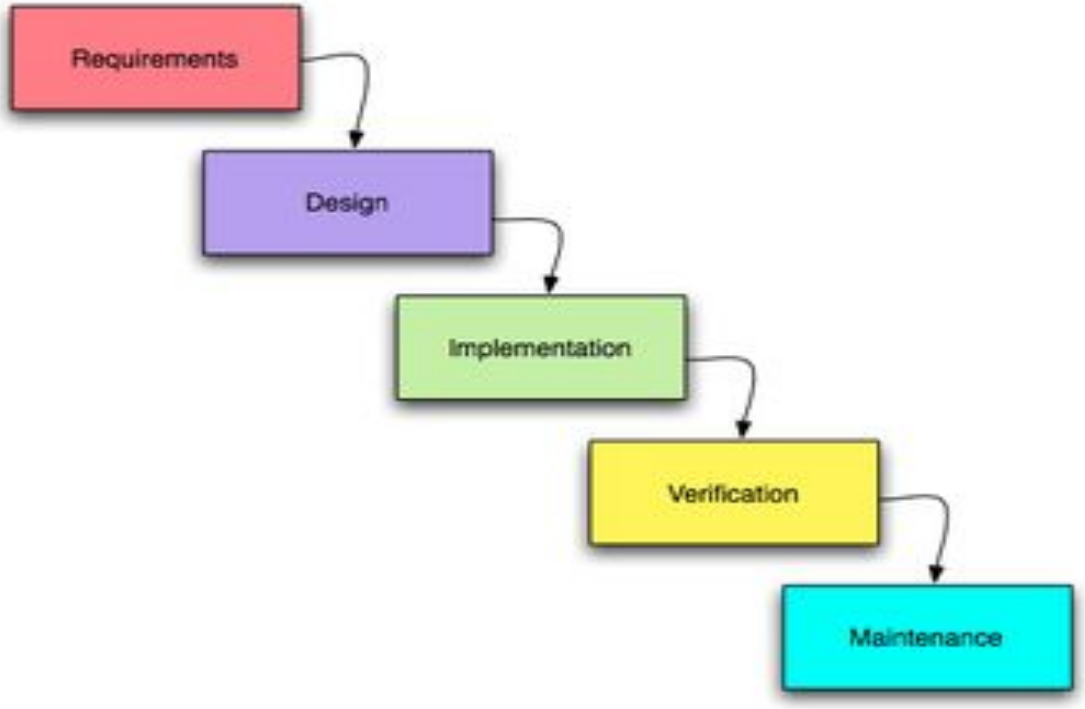
"Ein Vorgehensmodell zur Softwareentwicklung ist ein für die Softwareentwicklung angepasstes Vorgehensmodell bei der professionellen ("ingenieursmäßigen") Anwendungsentwicklung. Es dient dazu, die Softwareentwicklung übersichtlicher zu gestalten und in der Komplexität beherrschbar zu machen."

(Quelle Wikipedia)

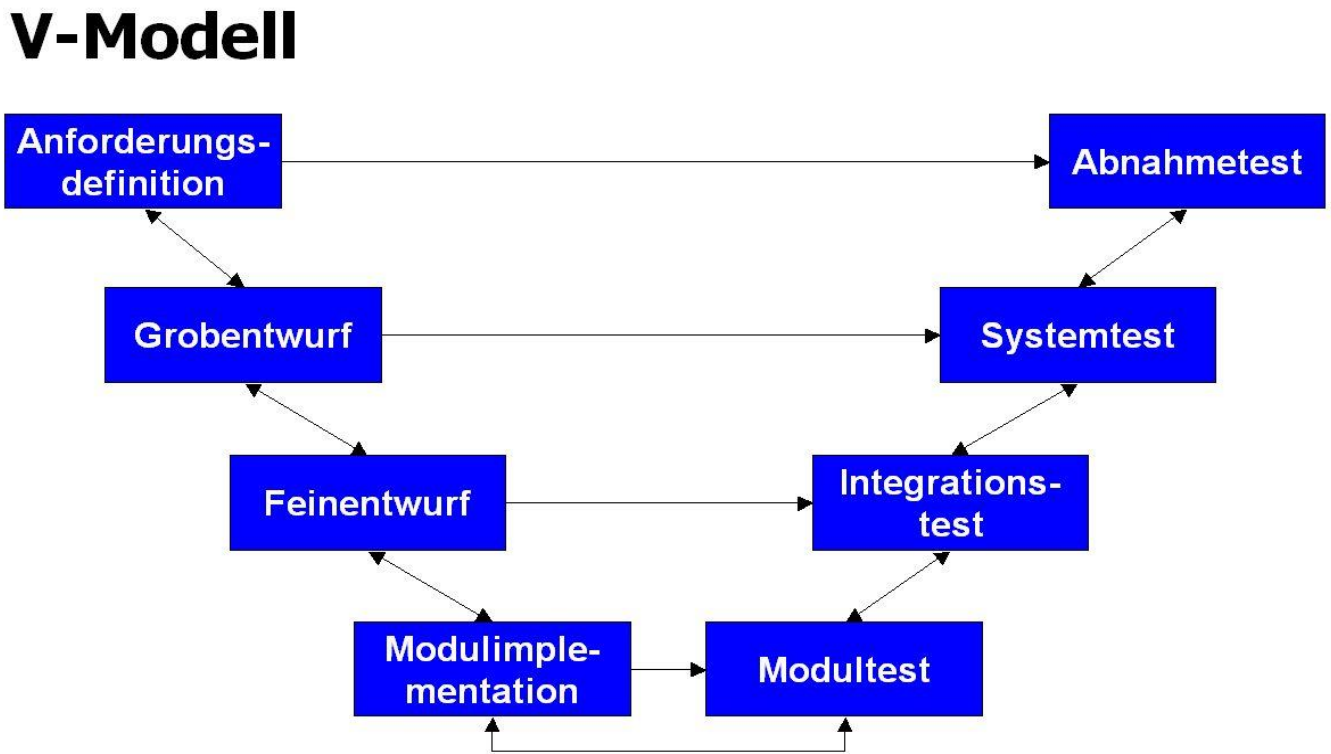
Vorgehensmodelle



Wasserfallmodell



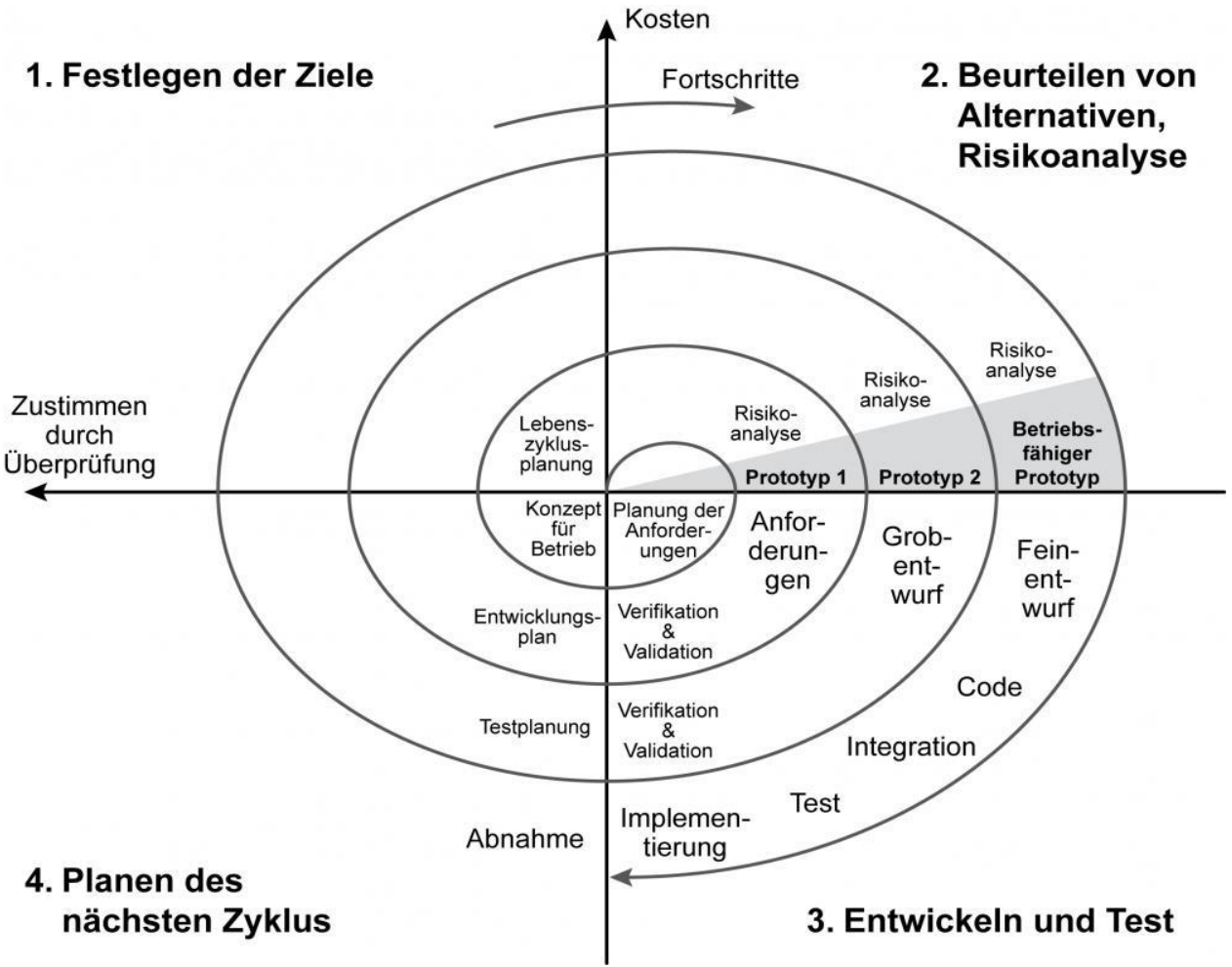
V-Modell


V-Modell
Anforderungs-definition
Abnahmetest
Grobentwurf
Systemtest
Feinentwurf
Integrations-test
Modulimple-mentation
Modultest

Rollen im V-Modell

- Projektmanager
- Projektleiter
- Rechtsverantwortlicher
- Projektadministrator
- Controller
- Q-Manager
- QS-Verantwortlicher
- Prüfer
- KM-Manager
- KM-Verantwortlicher
- KM-Administrator
- Systemanalytiker
- Systemdesigner
- SW-Entwickler
- HW-Entwickler
- Technischer Autor
- IT-Beauftragter
- SEU-Betreuer
- Datenadministrator
- Datenschutzbeauftragter
- IT-Sicherheitsbeauftragter
- Anwender
- Systembetreuer

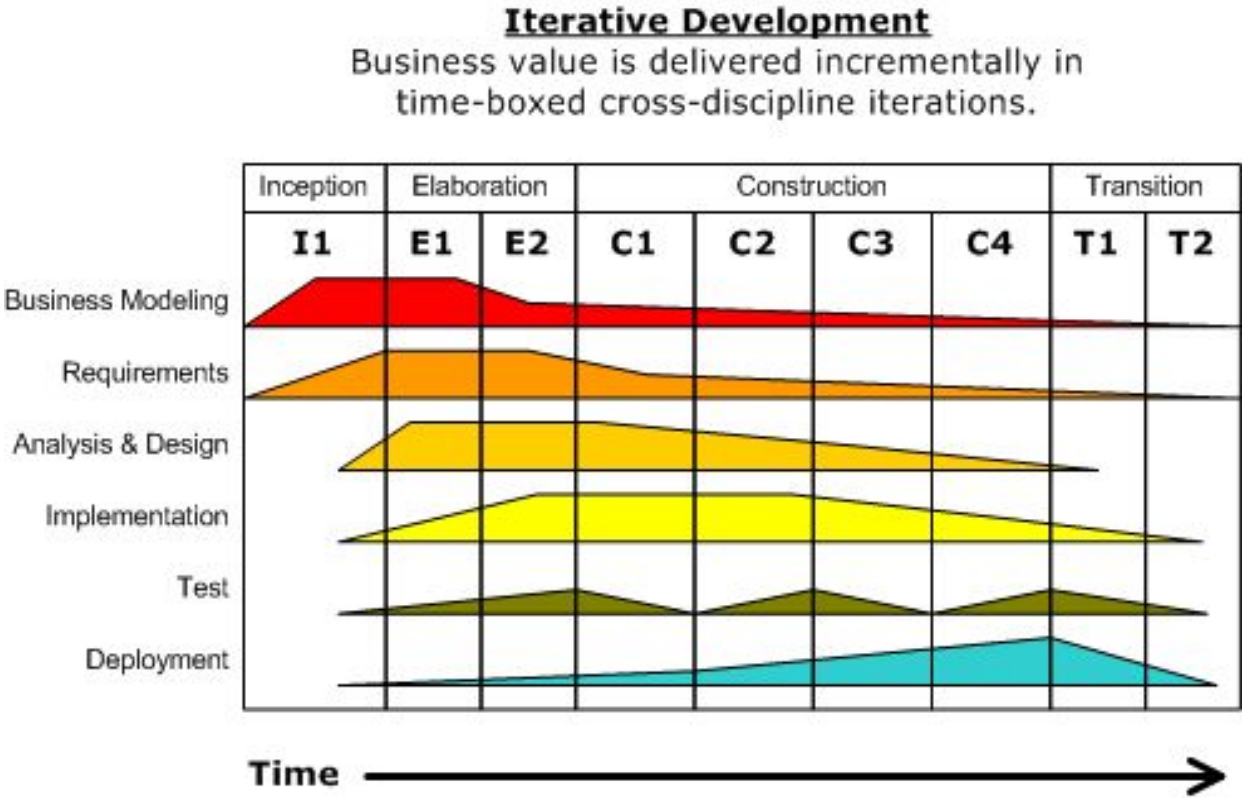
Spiralmodell



© Verlag Dr. Götz Schmidt Abb.: 2.19 - Spiralmodell (in Anlehnung am BOEHM)

Rational Unified Process (RUP)

Rational Unified Process (RUP)



Wasserfall - Pro und Contra

+	-
Übersichtlichkeit	Klare Abgrenzung in der Praxis schwierig
Einfach zu verstehen (geringer Schulungsaufwand)	Kein Zurück (im Standardmodell)
Klare Phasen	Unflexibel
Stukturierte Arbeitsaufteilung	Änderung kosten viel Zeit und Geld
Kosten/Aufwand einschätzbar	Feedback erst am Ende

V-Modell - Pro und Contra

+	-
Fest vorgegebene Rollen	Aufwendige Dokumentation
Minimiertes Risiko, Planbarkeit	Organisation und Durchführung wird nicht genau abgedeckt
Genaue Zuständigkeiten	nicht flexibel
Bessere Produktqualität	Viele Rollen
Transparente Aufarbeitung Frühere Tests, gute Testabdeckung	nicht für kleine Projekte geeignet

Spiral-Modell - Pro und Contra

+	-
Früher Prototyp	Steigender Aufwand, da teilweise doppelte Arbeit
Planbar durch mehrere Iterationen	Übersicht geht verloren
Früh testbar	Unausgeglichene Arbeitslast
Keine komplexere Strukturen	Teurer als andere Methoden
Iteratives Modell	Unflexibel, wenn man es nach Vorgaben macht

RUP - Pro und Contra

+	-

Zusammenfassung - klassische Methoden eignen sich

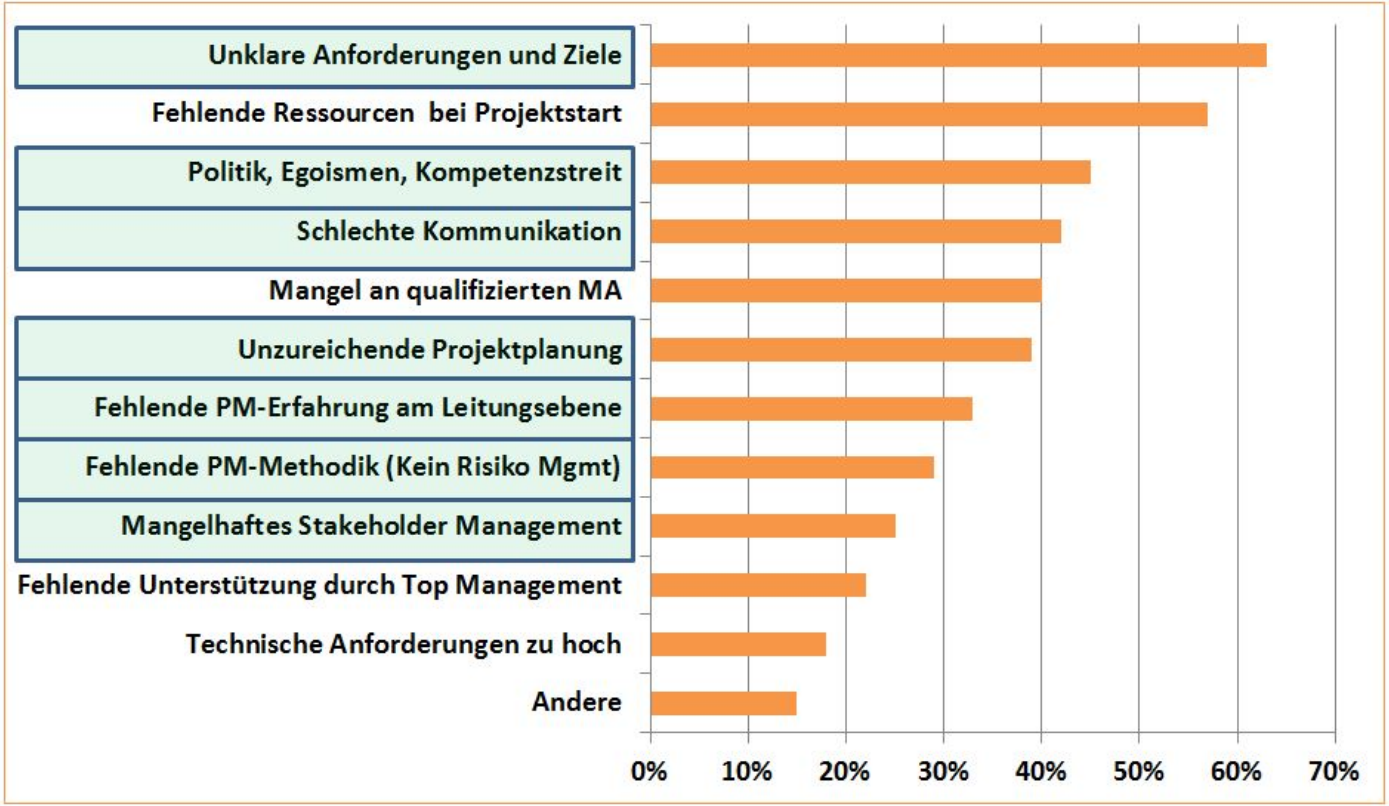
- wenn Entwickler die Anforderungen gut abschätzen können
- wenn sich Anforderungen kaum ändern
- bei großen Projekten
- wenn eine hohe Sicherheit in der Entwicklung gefragt ist ("Über"-Planung)

Problem: heutzutage ändern sich die Anforderungen relativ schnell

Warum scheitern Projekte?

(Quelle: GPM und PA Consulting: Ursachen für das Scheitern von Projekten, PAConsulting 2013)

Warum scheitern Projekte?



Realität?

Anforderungen sind **schwer abzuschätzen**

Anforderungen **ändern** sich laufend

Anforderungen ändern sich **spät** im Projekt

Distanz zu Auftraggeber während der Entwicklung

Mögliche Lösung: Agiler Ansatz

Was heißt “agil”

“agilis” (latein.) oder “agile” (engl.)

- steht für “flink, gewandt”

“Flexibler und schlanker” Entwicklungsprozess

Verschiedene Vorgehensmodelle:

- XP
- SCRUM

Ziele agiler Softwareentwicklung

Höhere **Flexibilität** als bei klassischen Modellen

Fokussierung auf die zu **erreichenden Ziele**

Angehen von technische **und** sozialen Problemen

nicht schwergewichtig und bürokratisch vorgehen

Fail fast - fail cheap - **fail early**

Das Agile Manifest

Manifesto for Agile Software Development

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

Manifesto for Agile Software Development

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

Manifesto for Agile Software Development

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

Wer hat's erfunden?

- Kent Beck (eXtreme Programming, Test-Driven Development, JUnit)
- Mike Beedle (scrum)
- Arie van Bennekum
- Alistair Cockburn (Crystal, Effective Use Cases, Wiki)
- Ward Cunningham (eXtreme Programming)
- Martin Fowler (Refactoring, Enterprise Architecture Patterns, Analysis-P.)
- James Grenning
- Jim Highsmith (Adaptive Software Development)
- Andrew Hunt (Pragmatic Programmer)
- Ron Jeffries (eXtreme Programming)
- Jon Kern
- Brian Marick
- Robert C. Martin
- Steve Mellor
- Ken Schwaber (scrum)
- Jeff Sutherland (scrum)
- Dave Thomas (Pragmatic Programmer)

Wer will kann unterschreiben :)

http://www.agilemanifesto.org/

Agile Prinzipien

Die zwölf Prinzipien hinter dem Agilen Manifest

Wir folgen diesen Prinzipien:

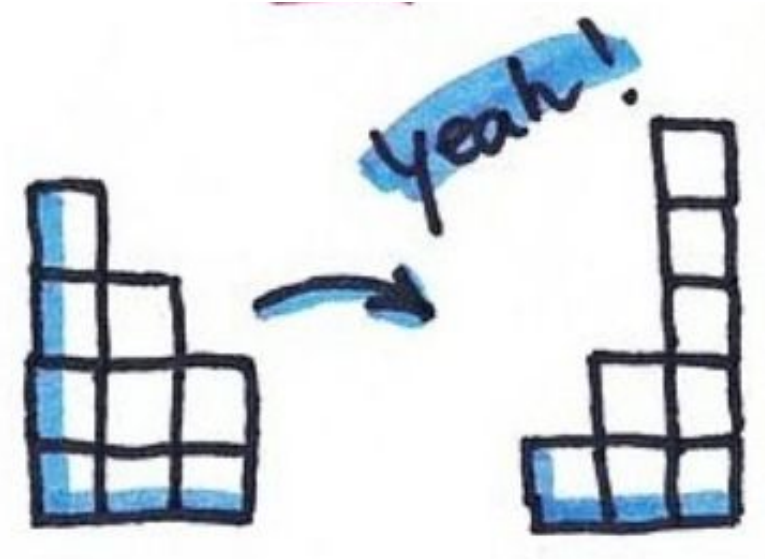
Quelle: http://agilemanifesto.org/iso/de//

#01

Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.

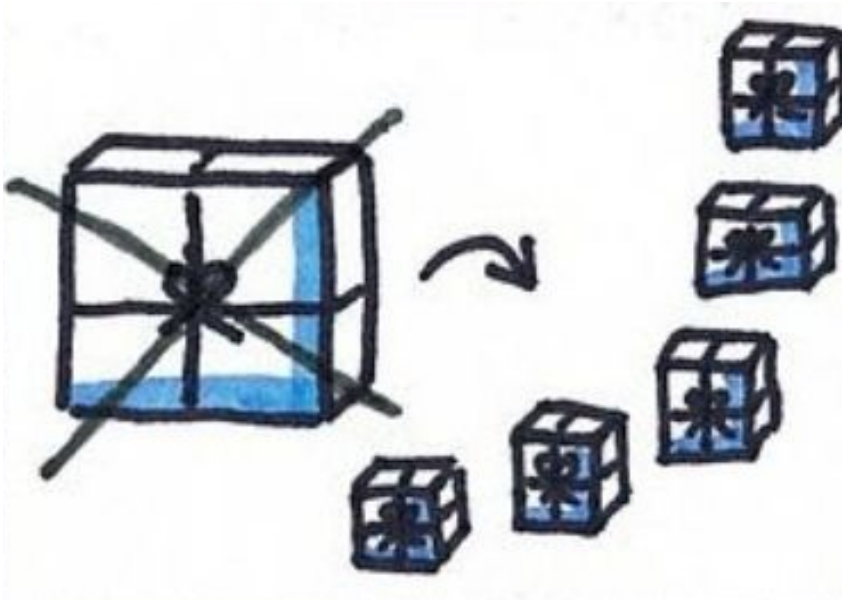
#02

Heisse Anforderungsänderungen selbst spät in der Entwicklung willkommen.
Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.



#03

Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.



#04

Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.

#05

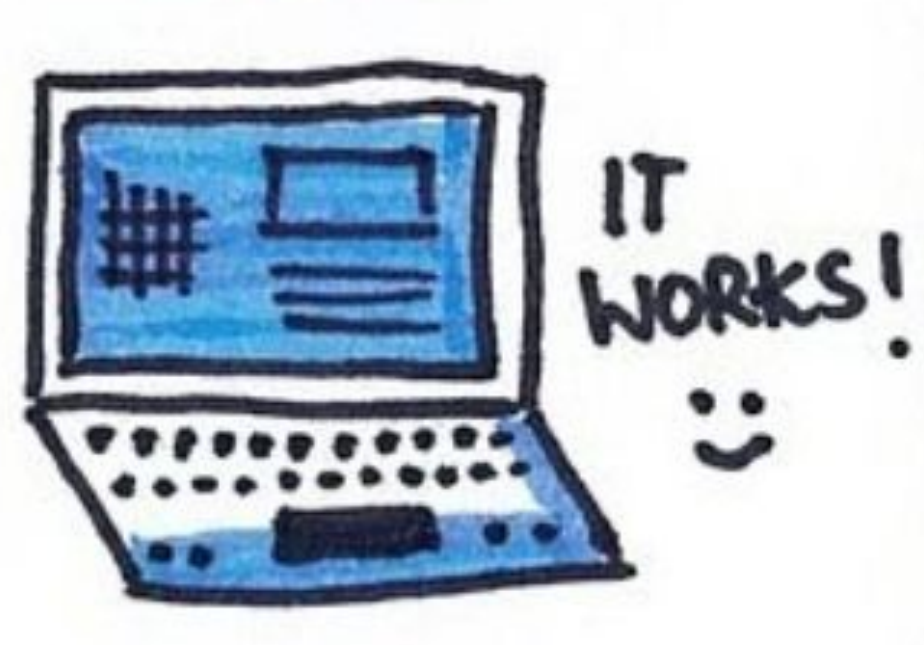
Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.

#06

Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteam zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.

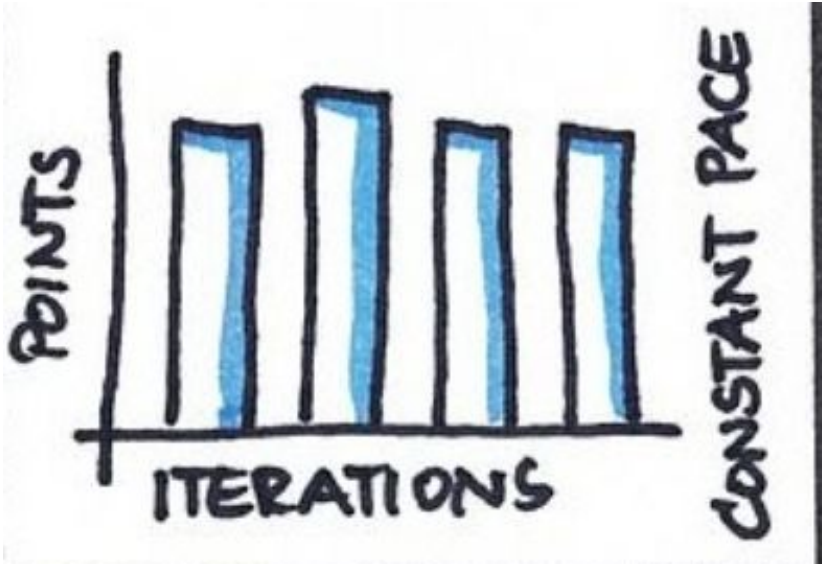
#07

Funktionierende Software ist das wichtigste Fortschrittsmaß.



#08

Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.

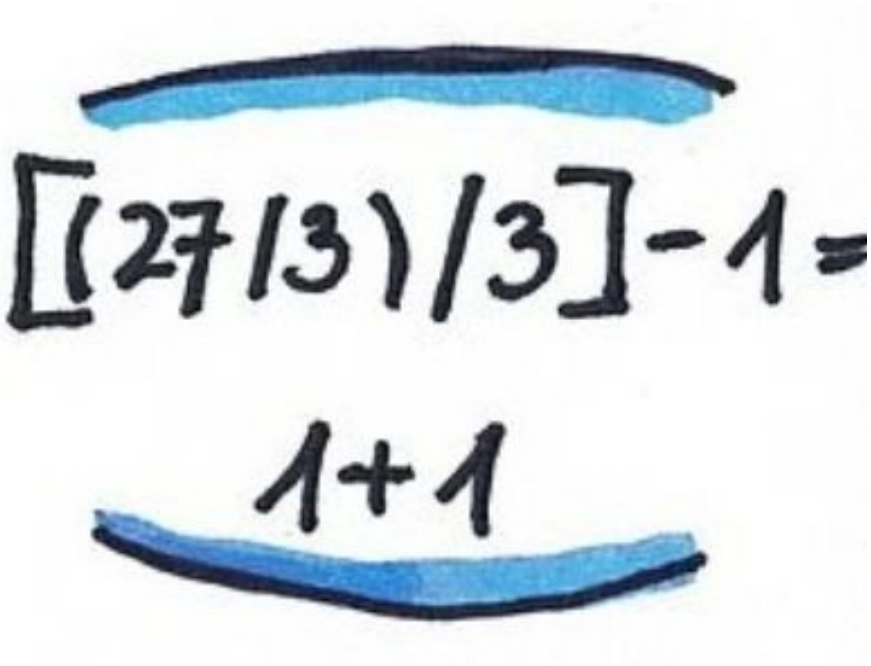


#09

Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.

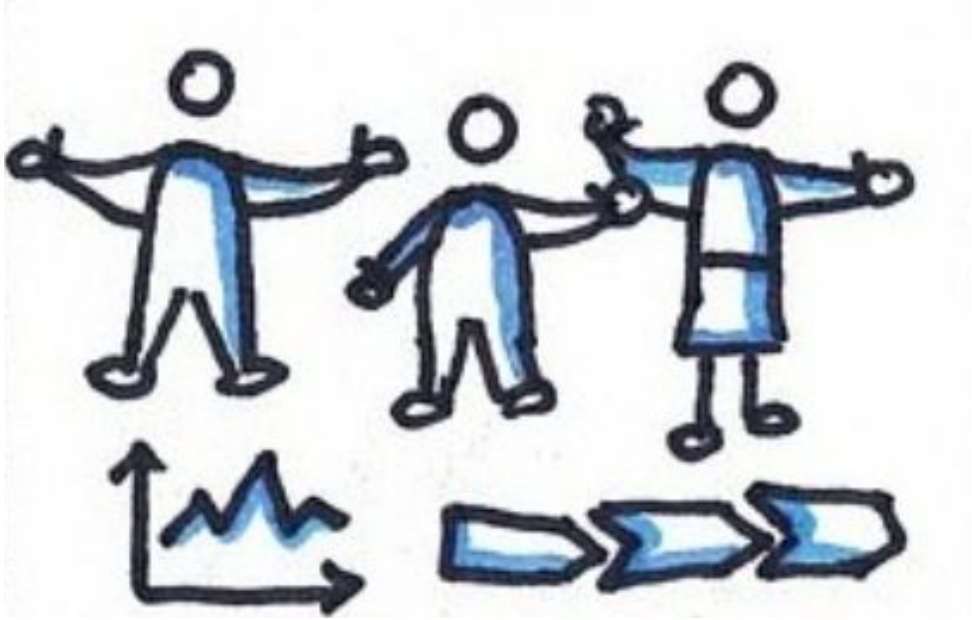
#10

Einfachheit -- die Kunst, die Menge nicht getaner Arbeit zu maximieren -- ist essenziell.



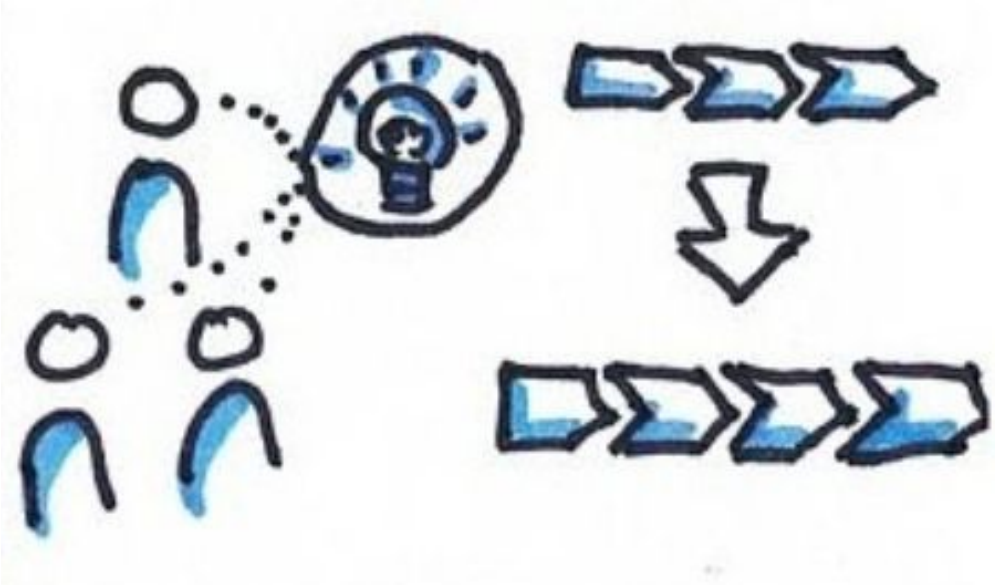
#11

Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.

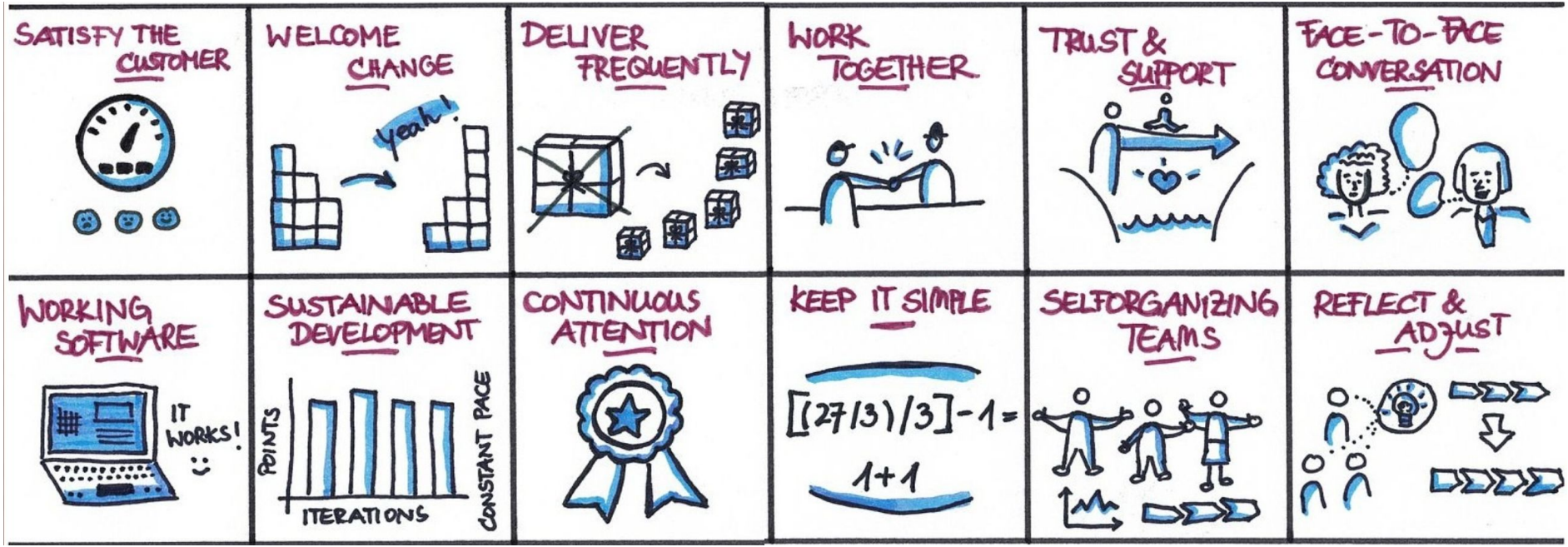


#12

In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.

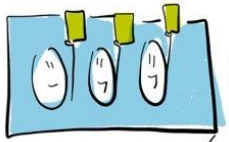


Übersicht

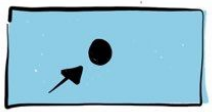

SATISFY THE CUSTOMER
WELCOME CHANGE
yeah!
DELIVER FREQUENTLY
WORK TOGETHER
TRUST & SUPPORT
FACE-TO-FACE CONVERSATION
WORKING SOFTWARE
IT WORKS!
SUSTAINABLE DEVELOPMENT
POINTS
ITERATIONS
CONSTANT PACE
CONTINUOUS ATTENTION
KEEP IT SIMPLE
[(27/3)/3]-1=
1+1
SELFORGANIZING TEAMS
REFLECT & ADJUST

Agile Werte

tanyafach.com

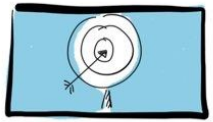


Commitment
Ja, wir wollen

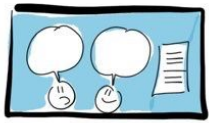


Einfachheit
einfache Texte, einfache Sprache

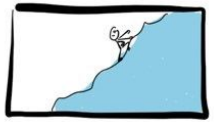
Feedback
zeitnah und wertschätzend



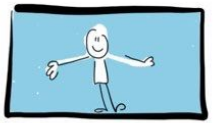
Fokus
auf das Wesentliche fokussieren



Kommunikation
Austausch fördern auf Augenhöhe



Mut
neue Ideen auszuprobieren



Offenheit
für das Fremde, Neue

Respekt
vor dem anderen

Agile Methoden

Scrum

Extreme Programming (XP)

Feature Driven Development (FDD)

Crystal

Test-driven development (TDD)

Lean Software Development

OKRs

...

Agile “Best Practices”

Story Cards

TDD

Reviews (Code)

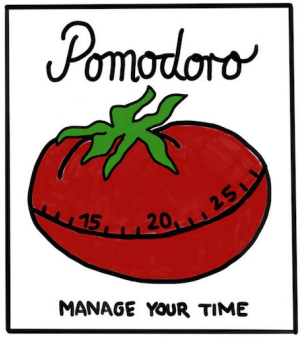
Refactoring

Iterative Entwicklung

KISS-Prinzip (keep it simple stupid) / YAGNI (you ain’t gonna need it)

...

Timeboxing



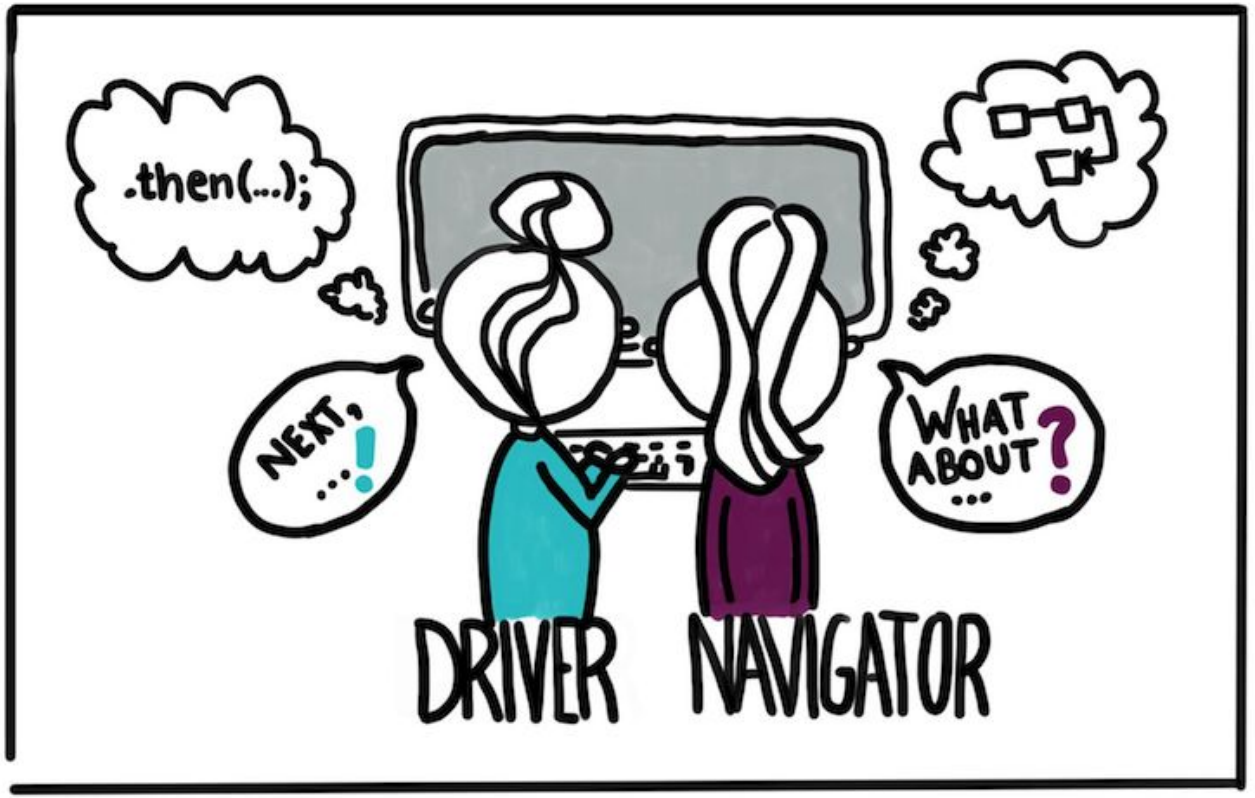
Aufgabe oder Besprechung hat feste Länge (Box)

- Ziel: Effizienz eines Projekts oder von Besprechungen erhöhen
- Erweiterung: Pomodoro Methode

Planen für feste Timeboxes (z.B. zwei Wochen)

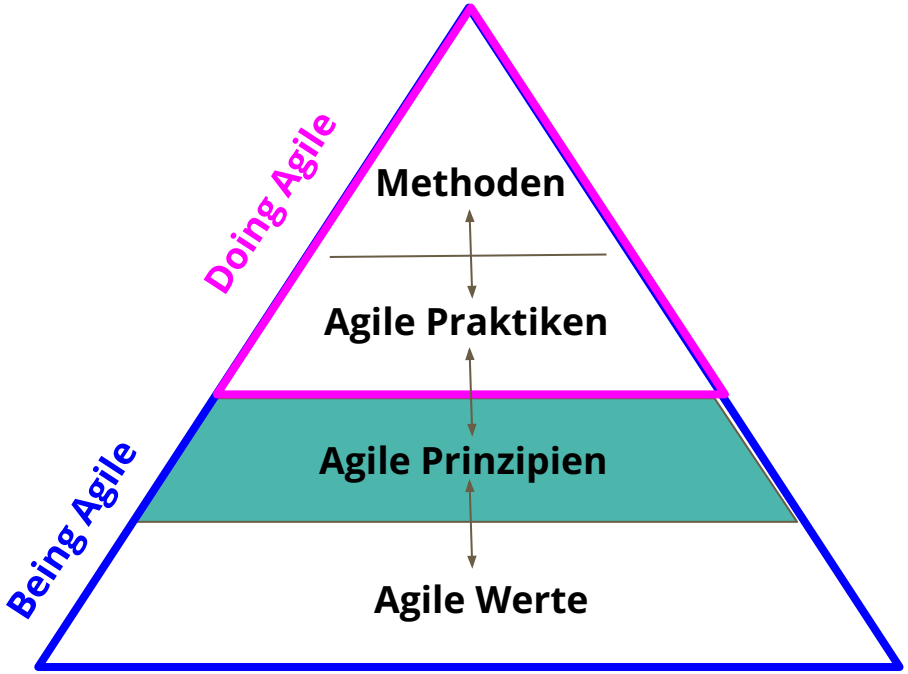
- Nicht “wie lange dauert eine Aufgabe”
- Sondern: “Welche meiner Aufgaben schaffe ich in der kommenden Timebox

Pair Programming

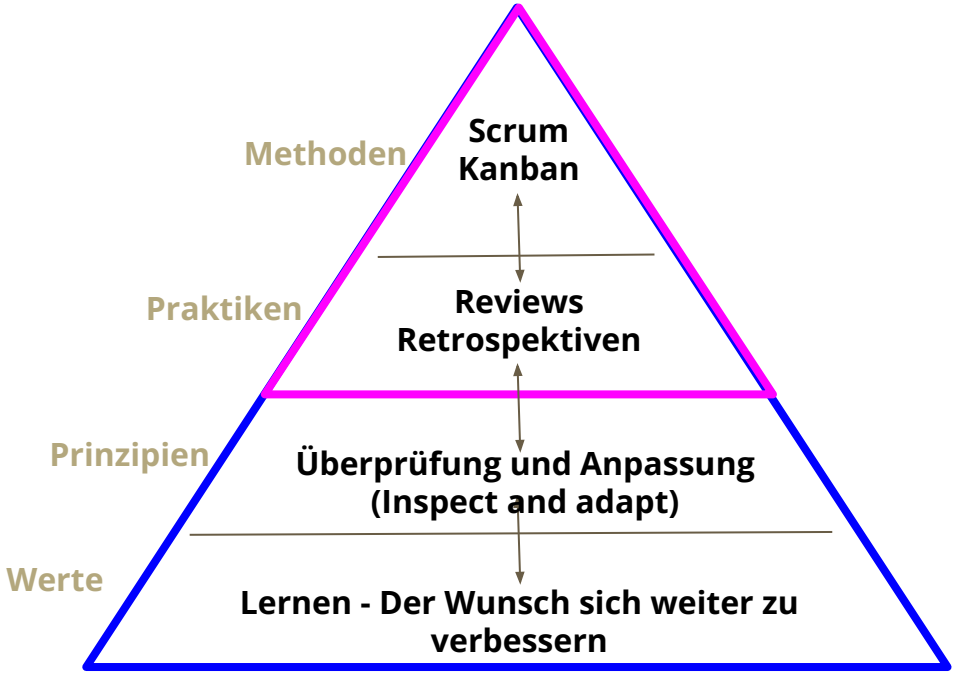


Cargo Cult

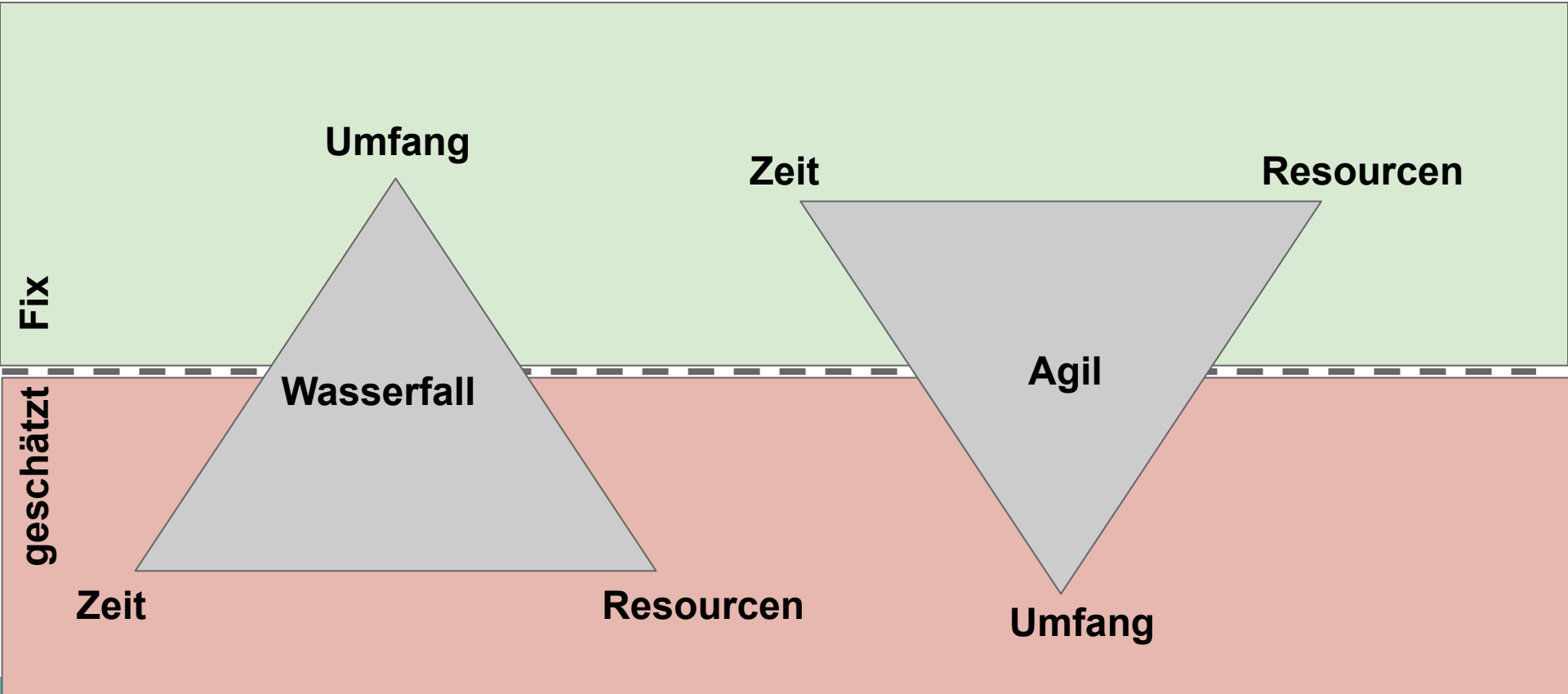
Die Agile Pyramide



Die Agile Pyramide



Vergleich: Agil vs. Wasserfall



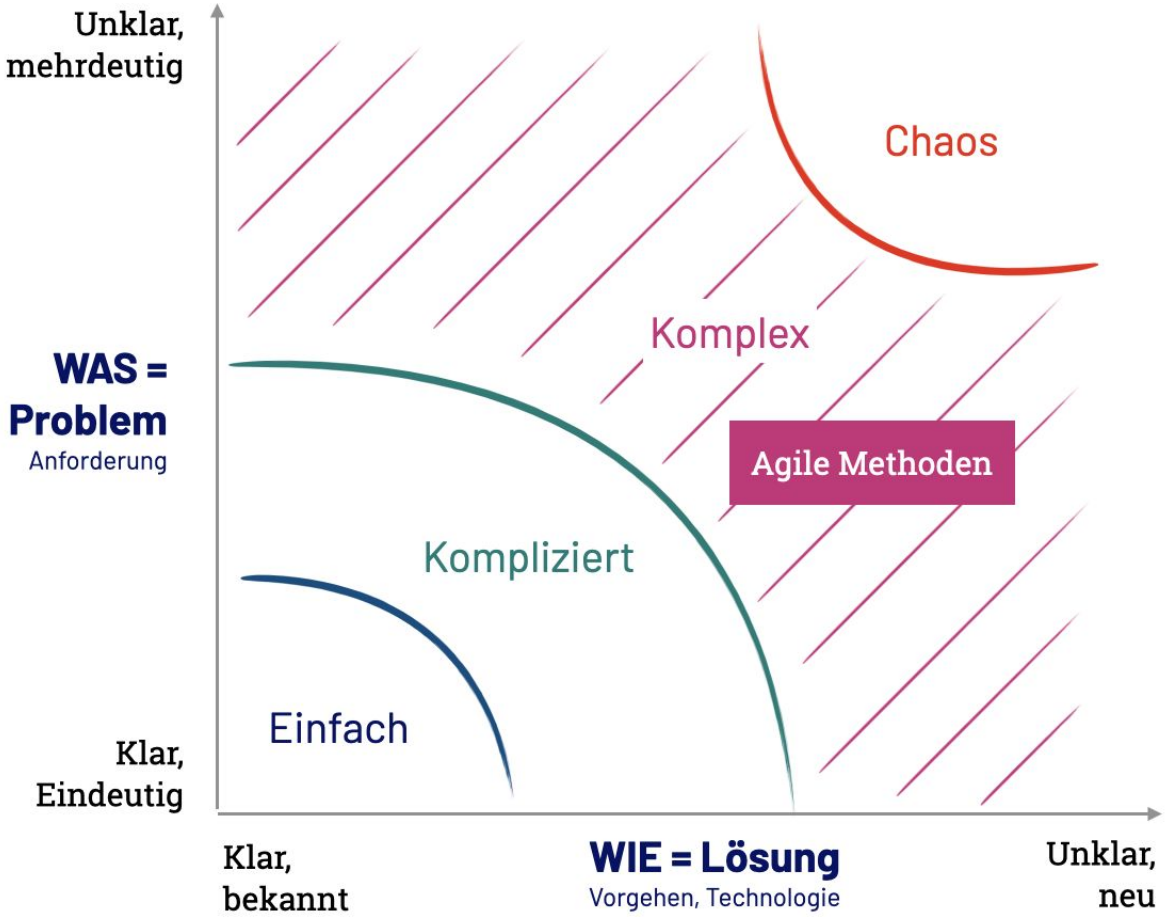
Vergleich klassisch vs. agil

Klassisch	Agil
Anforderungen zu Beginn bekannt	Anforderungen zu Beginn unscharf
Änderungen von Anforderungen während Projektverlauf schwierig	Änderungen an Anforderungen während Projektverlauf eingeplant
Hohe Kosten für späte Anforderungsänderungen	Mäßige Kosten für späte Anforderungsänderungen
Anforderungsbeschreibung aus technischer Sicht (Features)	Anforderungsbeschreibung aus Kundensicht (Anwendungsfälle)
Starrer Projektmanagementprozess	Fortlaufende Prozessverbesserungen
Kunde sieht nur Endergebnis	Kunde bewertet Zwischenergebnisse

Vergleich klassisch vs. agil

Klassisch	Agil
Wenn es eng wird, eher Meilensteine schieben	Wenn es eng wird, eher Aufwand verringern
Große Teams möglich	Relativ kleine Teams nötig
Klare Hierarchie	Selbstorganisierte Teams
Viele Spezialisten im Team	Viel gemeinsame Verantwortung
Team sitzt verteilt und ist in mehreren Projekten tätig	Team sitzt zusammen und hat Fokus auf ein Projekt
Viel Kommunikation über Dok. und lange Meetings	Viel informelle Kommunikation und Standup-Meetings
Aufwandsschätzung durch Projektleiter oder Experten	Aufwandsschätzung gemeinsam im Team

Stacey Matrix



Quelle: "https://digitaleneuordnung.de/blog/agile-methoden/" Anderas Diehl

“Agilitis” (Missverständnisse)

Jeder will “agil” sein

Anscheinend ein Allheilmittel

Jeder versteht etwas anderes unter “agil”

Alles wird besser, wenn man agil arbeitet

Agil = Daily + wildes Durcheinander + Playstation?

“Agilitis #1” Agiles Projektmanagement

Gibt es nicht! (Gibt auch kein Wasserfall-Projektmanagement)

Agile Methoden, sind Methoden, wie man Projekte abwickelt

Man kann Tools aus dem klassischen Projektmanagement in agilen Methoden anwenden - aber dadurch wir das Projektmanagement nicht agil

“Agilitis #2” Mit agilen Methoden werden Softwareprojekte besser

Nicht pauschalisierbar, manchmal besser als z.B Wasserfall, aber nicht immer

Unternehmen benötigt einen gewissen Reifegrad, um effektiv und effizient agil zu arbeiten

Allerdings:

nach dem Start der Einführung agiler Methoden werden die Projekte etwas schlechter laufen

die Teams werden sich mehr anstrengen müssen als zuvor

Trotzdem: schnelle Besserung bald sichtbar

“Agilitis #3” Wer agil arbeiten möchte, muss nach SCRUM arbeiten

SCRUM in der Regel agil

Weitere agile Methoden auch möglich

Agil arbeiten: Agiles Manifest ernst nehmen + agile Werte und Prinzipien

Methode nicht so wichtig, Mindset wichtiger

“Agilitis #4” Agil arbeiten ist die einzig richtige Methode

bei manchen Projekten und Produkten vielleicht besser als andere Methoden

Agil bei manchen Projekten und Produkten vielleicht besser als andere Methoden

Manchmal passen organisatorischen Strukturen oder die Kundenanforderungen nicht zum agilen Denken

Einführung kann auch scheitern

“Agilitis #5” Agile Projekte brauchen Projektmanagement

Agile Methoden kennen die Rolle “Projektleiter” nicht

diese Aufgaben wird von anderen Rollen wahrgenommen

Allgemein: Werkzeuge einfach anders als im klassischen Projektmanagement

“Agilitis #6” Teilagile Projektmethode

Entweder man arbeitet agil oder nicht

wenn agiles Arbeiten nicht möglich ist, kann man sich Werkzeuge aus der agilen Welt ausborgen

Zum Beispiel

- Daily Standup aus SCRUM zur täglichen Koordination des Projektteams
- Retrospektive als regelmäßige Lessons Learned

Aber nur, weil man das macht, ist man nicht agil (eher HYBRID)

sehr eingeschränkt möglich, klassische Tools in die agile Welt zu transportieren

“Agilitis #7” Das Management möchte, dass wir agil arbeiten!

Commitment des Managements ist essentiell

Management muss aber verstehen, was agil bedeutet und muss Freiräume geben

Denken ändern

Mut zum Fehler haben, sonst wird die Einführung schiefgehen

“Agilitis #8” Agile Methoden sind nicht planbar und geben keine Sicherheit

Agile Methoden strahlen eine Unsicherheit aus, die nicht real ist.
Wasserfallmethoden strahlen eine Sicherheit aus, die nicht real ist.

Kein Projekt, das klassisch abgewickelt wird, wird am Ende so geliefert, wie zu Beginn gedacht.

Projekte sind neuartig, risikoreich, dynamisch und komplex.

Agile Methoden sagen: wenn der Weg zum Ziel so ungewiss ist, dann tasten wir uns voran und prüfen immer wieder ob wir am rechten Pfad sind.

“Agilitis #9” Bei agilen Methoden weiß ich nicht was ich wann bekommen werde und was es kosten wird.

bei klassischen Methoden auch nicht

Die Annahme, dass Kosten, Inhalt und Zeit fix sind, ist nicht korrekt.

Es gibt einige Tools, mit denen man bei agilen Methoden sehr gut planen kann, was man bis wann bekommt

“Agilitis #10” Das Projekt ist zu groß um agil abgewickelt zu werden

Auch große Projekte können agil abgewickelt werden

SCRUM kann sich ab einer gewissen Größe der Organisation – egal ob Projekt oder Linie – als Bremse herausstellen

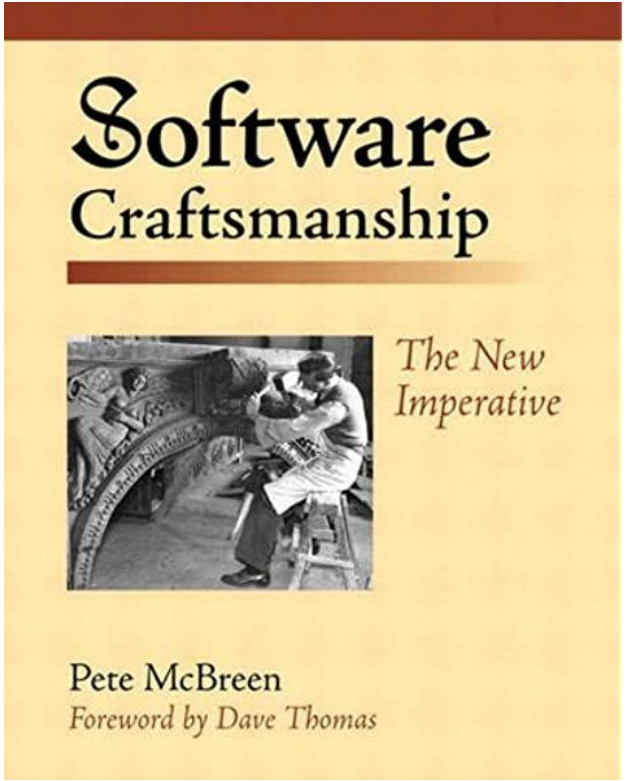
agiles Arbeiten hängt nicht von der Methode ab

Wenn sich z.B. SCRUM als nicht mehr passend herausstellt, passt man die Methode einfach an

EXKURS: Manifesto for Software Craftsmanship

Als engagierte Software-Handwerker heben wir die Messlatte für professionelle Softwareentwicklung an, indem wir üben und anderen dabei helfen, das Handwerk zu erlernen.

Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:



Manifesto for Software Craftsmanship

Nicht nur funktionierende Software, sondern auch **gut gefertigte Software**

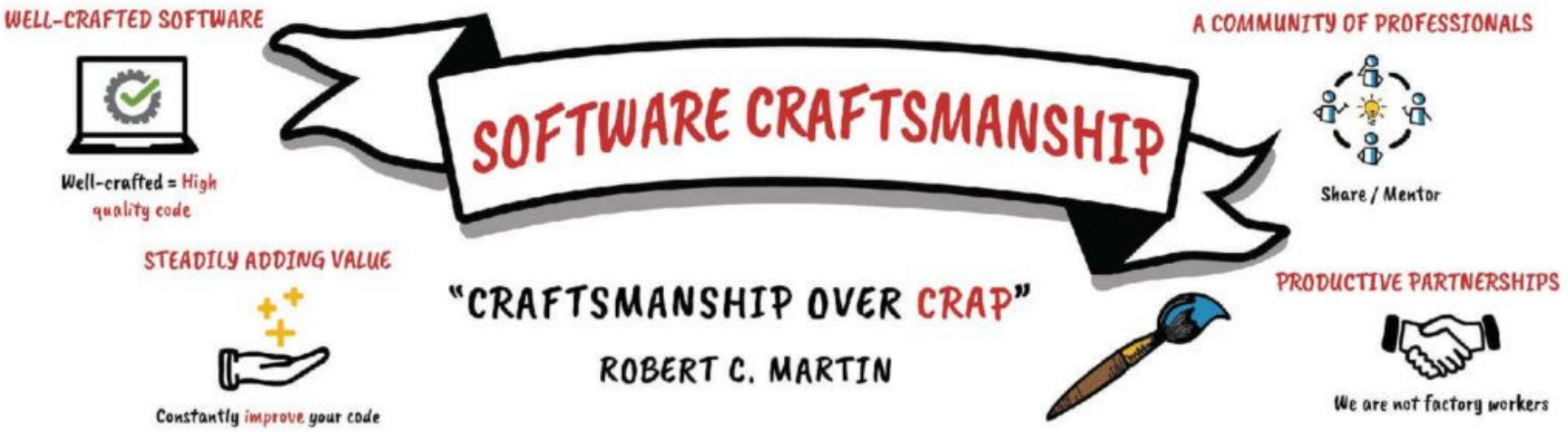
Nicht nur auf Veränderung zu reagieren, sondern stets **Mehrwert** zu schaffen

Nicht nur Individuen und Interaktionen, sondern auch eine **Gemeinschaft aus Experten**

Nicht nur Zusammenarbeit mit dem Kunden, sondern auch **produktive Partnerschaften**

Manifesto for Software Craftsmanship

Das heißt, beim Streben nach den Werten auf der linken Seite halten wir die Werte auf der rechten Seite für unverzichtbar.



Agilität und Unternehmenskultur

Unternehmenskultur anhand von zwei Dimensionen bestimmen:

Unternehmen oder der Mitarbeiter im Fokus?

aktuelle Realität oder eine zukünftige Möglichkeit im Fokus?

Daraus ergeben sich vier Kernkulturen.

CULTURE = "How we do things around here to succeed."

"The Reengineering Alternative." William Schneider

REALITY ORIENTED (Actuality)

"We succeed by working together."

Affiliation

Synergy

COLLABORATION

Partnership

Trust

People

Interaction

Teams

Diversity

Egalitarian

"We succeed by getting and keeping control."

Power

Predictability

SECURITY

CONTROL

Process

Standardization

Hierarchical

Stability

Order

PEOPLE ORIENTED

(Personal)

COMPANY ORIENTED

(Impersonal)

"We succeed by growing people who fulfil our vision."

CULTIVATION

GROW

Purpose/Faith

Dedication

Let things Evolve

Subjectivity

Creativity

"We succeed by being the best."

Efficiency

Professionalism

COMPETENCE

#1

Meritocracy

Achievement

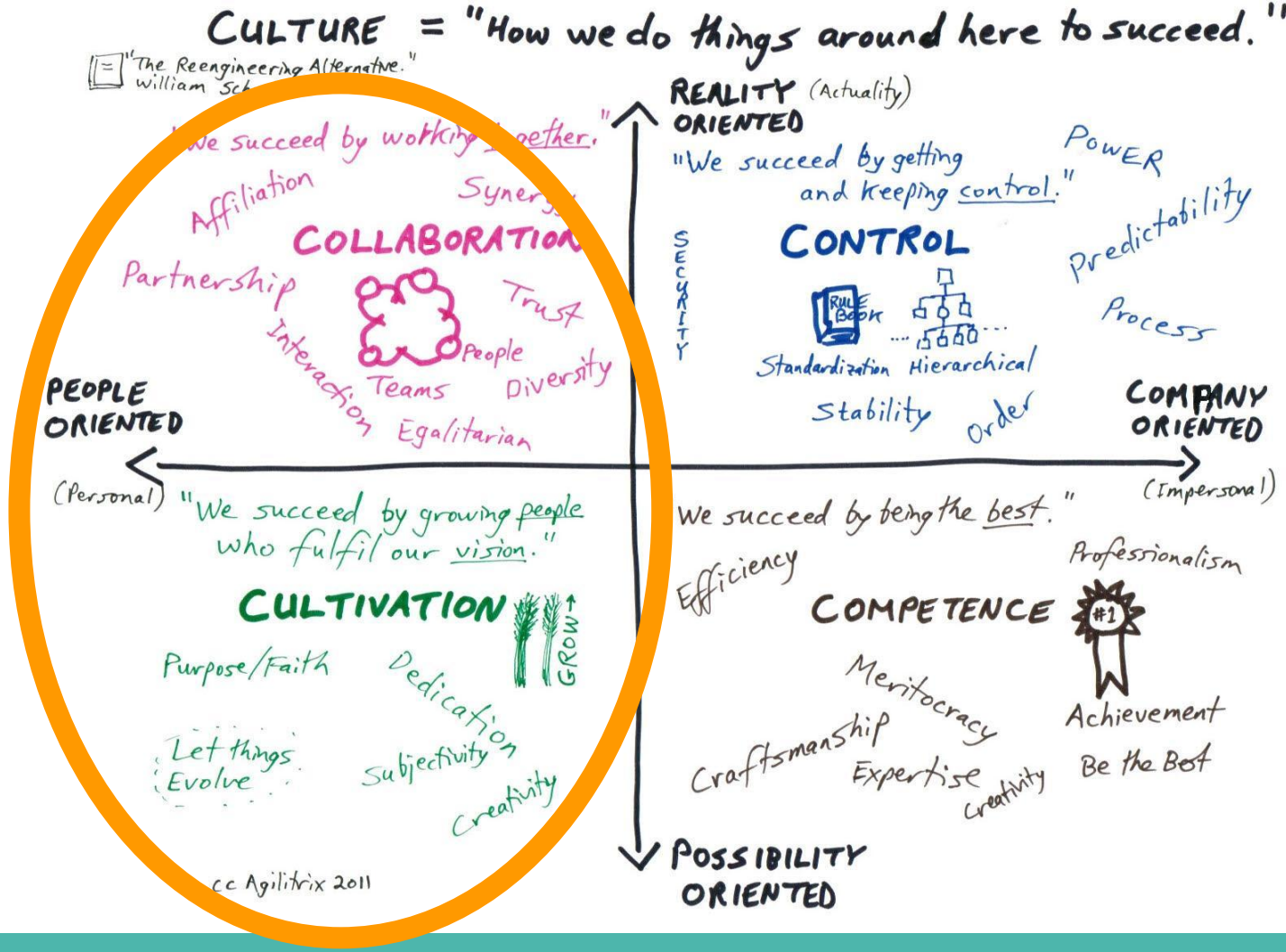
Craftsmanship

Expertise

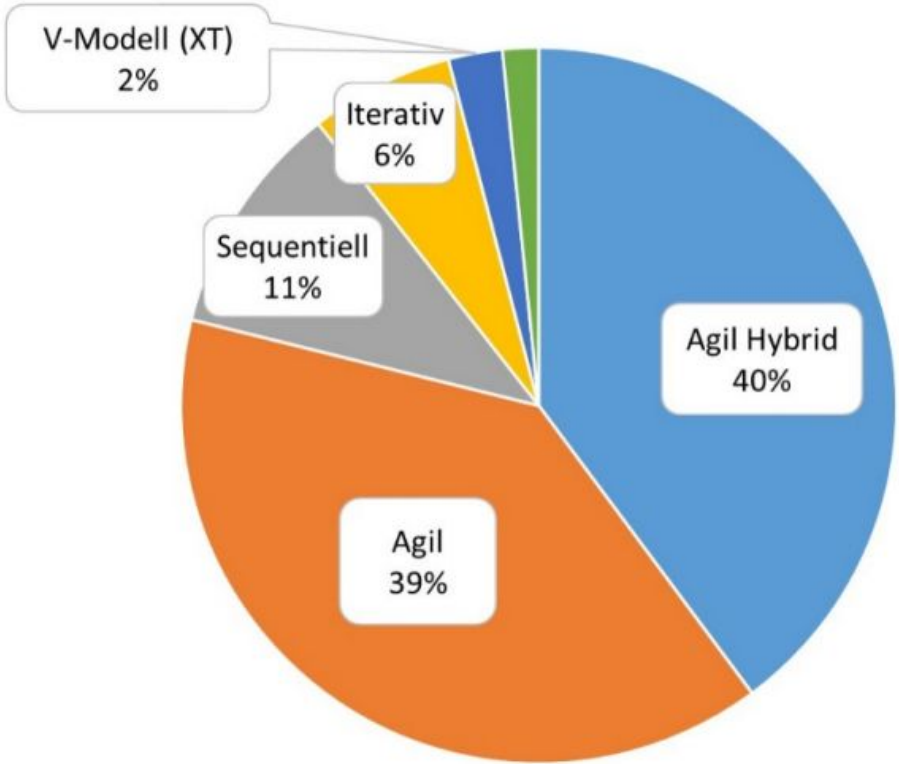
Be the Best

creativity

POSSIBILITY ORIENTED

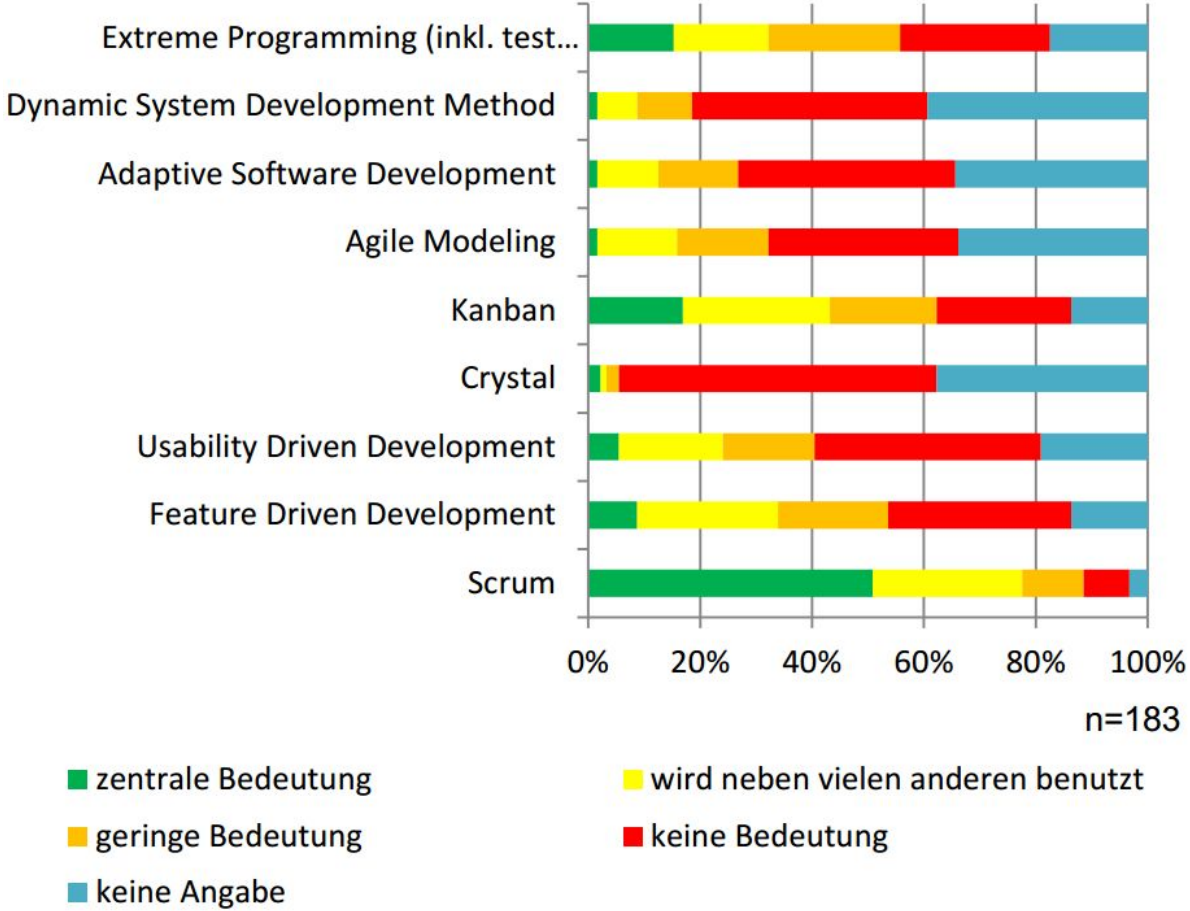

CULTURE = "How we do things around here to succeed."
"The Reengineering Alternative."
REALITY ORIENTED (Actuality)
"We succeed by getting and keeping control."
CONTROL
Power
Predictability
Process
Standardization
Hierarchical
Stability
Order
SECURITY
PEOPLE ORIENTED
(Personal)
COMPANY ORIENTED
(Impersonal)
"We succeed by working together."
Affiliation
Synergy
COLLABORATION
Partnership
Trust
People
Interaction
Teams
Diversity
Egalitarian
"We succeed by growing people who fulfil our vision."
CULTIVATION
GROW
Purpose/Faith
Dedication
Let things Evolve
Subjectivity
Creativity
"We succeed by being the best."
Efficiency
Professionalism
COMPETENCE
#1
Meritocracy
Craftsmanship
Expertise
Creativity
Achievement
Be the Best
POSSIBILITY ORIENTED
cc Agilitrix 2011

Verbreitung verschiedener Entwicklungsmethoden

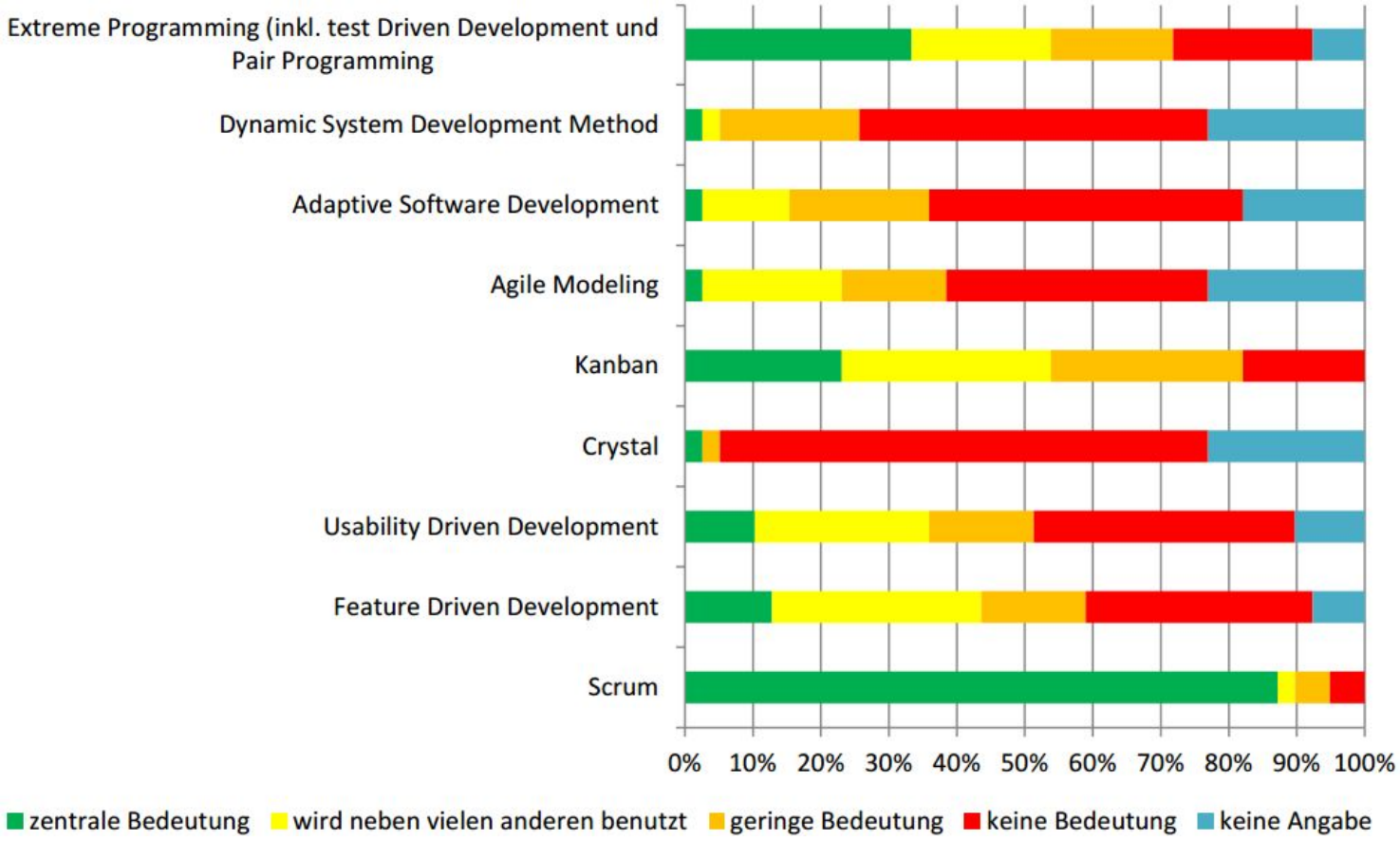


Quelle: “Wie interagieren UX-Professionals mit ihrem Umfeld und ihren Kollegen?” 2019

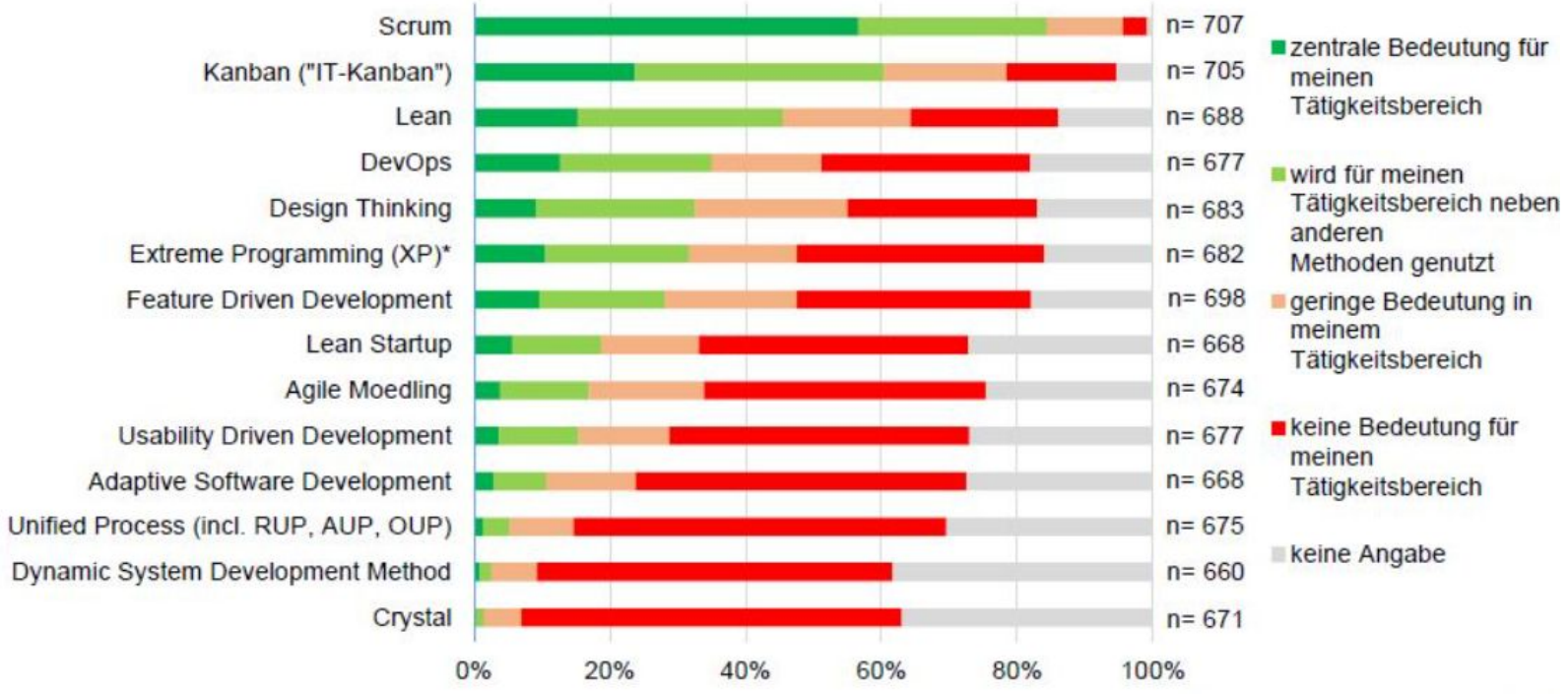
Verbreitung agiler Entwicklungsmethoden: Welche Methodik wird eingesetzt?



Verbreitung agiler Entwicklungsmethoden: Aus welcher Methodik wird sich bedient?



Welche Bedeutung haben die jeweiligen Methoden?



„Quelle: Studie Status Quo Agile – Verbreitung und Nutzen agiler Methoden, BPM-Labor HS Koblenz, Prof. Dr. Komus

Vorteile agiler Entwicklungsmethoden

Vorteile für Entwickler

Qualitätsarbeit

Direkte Kommunikation mit den Anwendern

Mehr Spaß bei der Arbeit

Vorteile für Kunden

Flexibilität

Spart Zeit und Nerven

Wettbewerbsvorteil