berlin-brandenburgische
AKADEMIE DER WISSENSCHAFTEN

Hartmut Schirmacher

„Gute“ Software entwickeln – auch im kleinen Team

Vortrag
Digital Humanities Kolloquium der Berlin-Brandenburgischen Akademie der Wissenschaften ; (Berlin) : 2018.06.01

Persistent Identifier: urn:nbn:de:kobv:b4-opus4-30756

Die vorliegende Datei wird Ihnen von der Berlin-Brandenburgischen Akademie der Wissenschaften unter einer Creative Commons Attribution- ShareAlike 4.0 Germany (cc by-sa 4.0) Licence zur Verfügung gestellt.

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

„Gute“ Software entwickeln – auch im kleinen Team

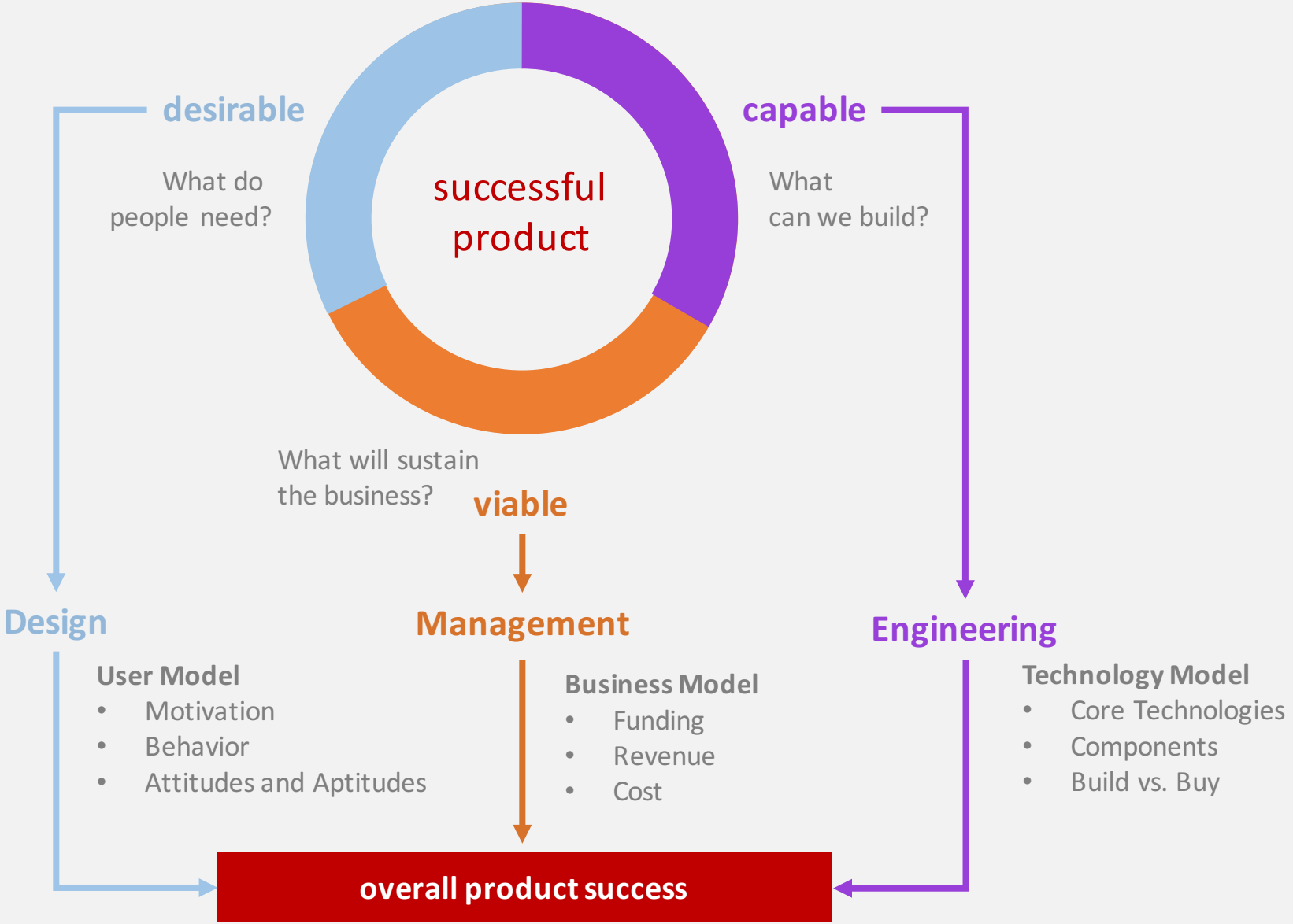
Hartmut Schirmacher

http://schirmacher.beuth-hochschule.de

Was ist ein „gutes“ Produkt?

1. Geschäftsziele erreichen (Hersteller + Kunde)
2. Benutzerziele erreichen, Benutzer zufrieden machen
3. Technisch funktionsfähig, performant

Cartoon by Satu Kyröläinen @satukyro



Frei nach Alan Cooper et al.: *About Face – The Essentials of Interaction Design*, 4th ed., Wiley 2014

Hartmut Schirmacher

- Seit 2012 nebenberuflicher Berater
  - Zur Zeit : Aufbau Software-Team für US-Startup im Bereich Neurostimulation
- Seit 2011 Professor @Beuth
  - Computergrafik, Programmierung, agiles Requirements Engineering
- 2002-2011 Visualisierungs-Startup in Berlin
  - zuletzt Leiter Produktmanagement
  - Teil eines NASDAQ-Unternehmens
  - Softwarelösungen für die Radiologie
- 2002 Dr.-Ing., Saarbrücken (Computergrafik)
- 1998 Diplom-Informatik, Erlangen

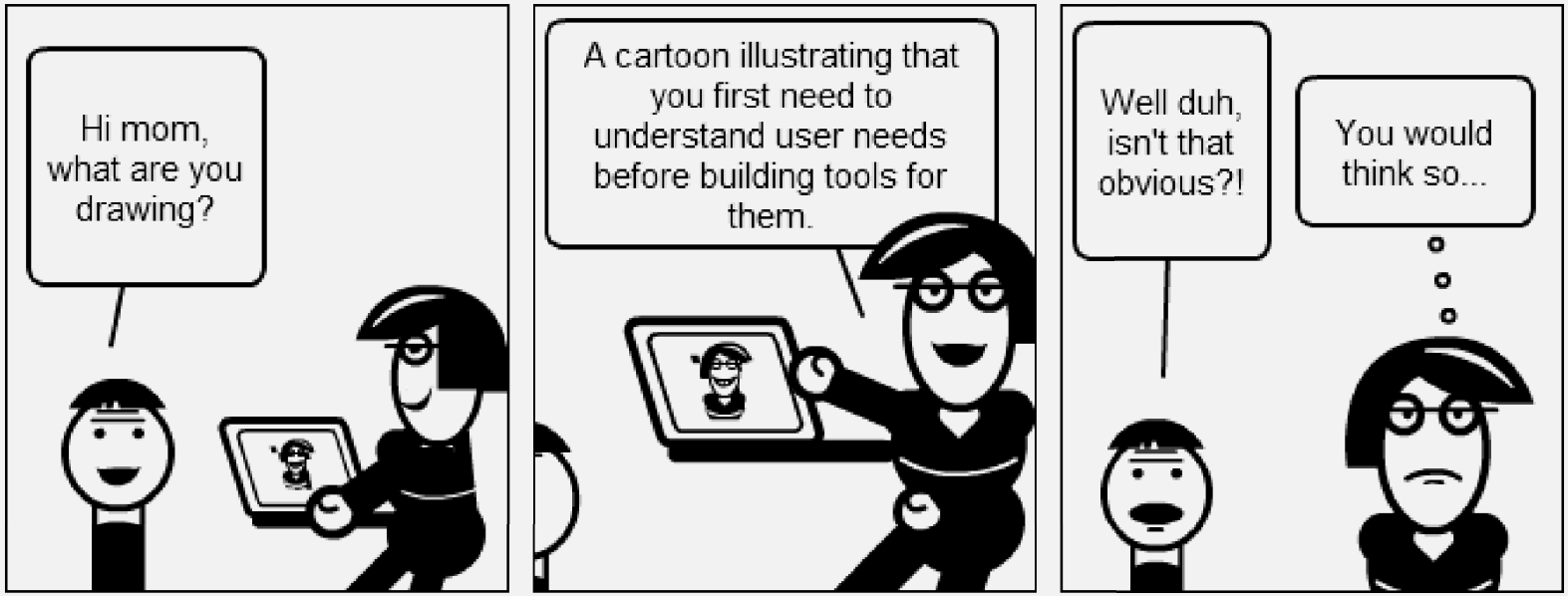
→ Kleine agile Softwareteams in streng regulierten Märkten.
→ Einzel-Software-Arbeiten in sehr begrenzten Zeitrahmen.

Agenda

- Warum ist die Beschäftigung mit Anforderungen so wichtig?
- Wie entwickelt man agil Software?
- Wie soll man dabei mit Anforderungen umgehen?
  - Klassisches Requirements Engineering
  - Goal Directed Design
  - Extreme Programming & User Stories
- Von der Idee zum Produkt
  - Aktivitäten und Artefakte
  - Wer macht was
  - Tool-Unterstützung
- Fazit / Ausblick

Warum sind Anforderungen ein wichtiges Thema?

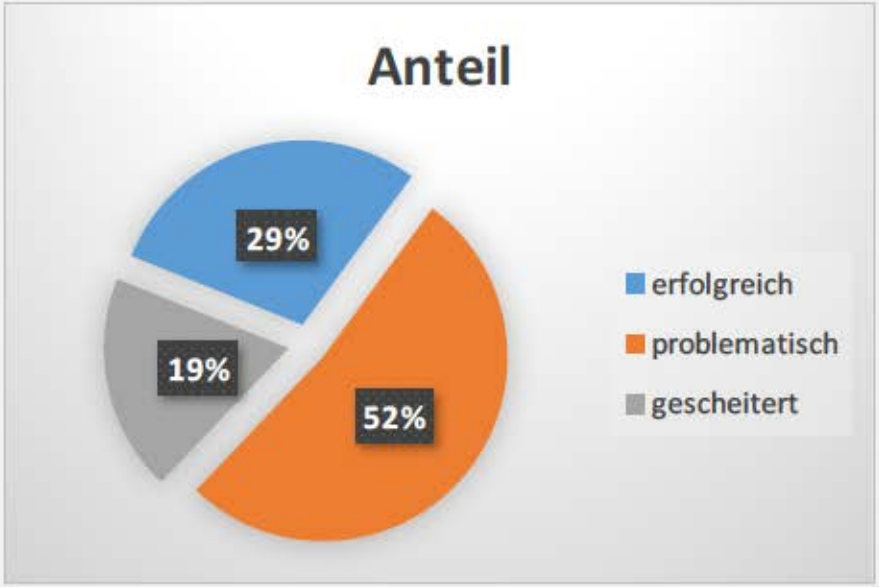
Anforderungsanalyse und –Management?



- *„Macht man heute noch Requirements Engineering? Ist doch jetzt alles agil.“*
- *„RE – das ist doch alles langweilig und versteht sich von selbst.“*

Cartoon by Satu Kyröläinen @satukyro

Chaos-Report



Untersuchung einer großen Zahl von SW-Projekten

Ergebnisse 2015:

- 29% erfolgreich
- 52% starke Zeit- und Budgetüberschreitungen, oder Kunde nicht zufrieden
- 19% gescheitert
- Je größer das Projekt, desto höheres Risiko

Hauptursachen für Scheitern:

- fehlende Zuarbeit durch Benutzer
- unvollständige/unklare Anforderungen
- häufige Anforderungsänderungen

Das alles hat mir dem Erarbeiten von Anforderungen zu tun

Standish Group: Chaos Report. Siehe https://de.wikipedia.org/wiki/Chaos-Studie

Anforderungen sind ein Kommunikationsproblem!

- Was genau möchten die Stakeholder wirklich?
- Welche Dinge wurden als Anforderungen gemeinsam erarbeitet und festgelegt?
- Große Probleme bei der Verschriftlichung

Agile SW-Entwicklung mit dem Scrum-Framework

Scrum [...]: Leute bauen Software mit gesundem Menschenverstand, indem sie immer wieder das gleiche machen und Unmengen an sexy englischen Begriffen verwenden.

Udo Wiegärtner, „War Gandalf ein Scrum Master?“,
Keynote zur Webweek Nürnberg 2016

Scrum?

Einwurf des Balls durch Luke Burgess in ein Gedränge.

Das *Angeordnete Gedränge* (englisch *scrum*) ist in verschiedenen Varianten des Rugbysports die Standardsituation, um das Spiel nach kleineren Regelverstößen [...] neu zu starten

Quelle: Wikipedia DE

Ziele von Scrum

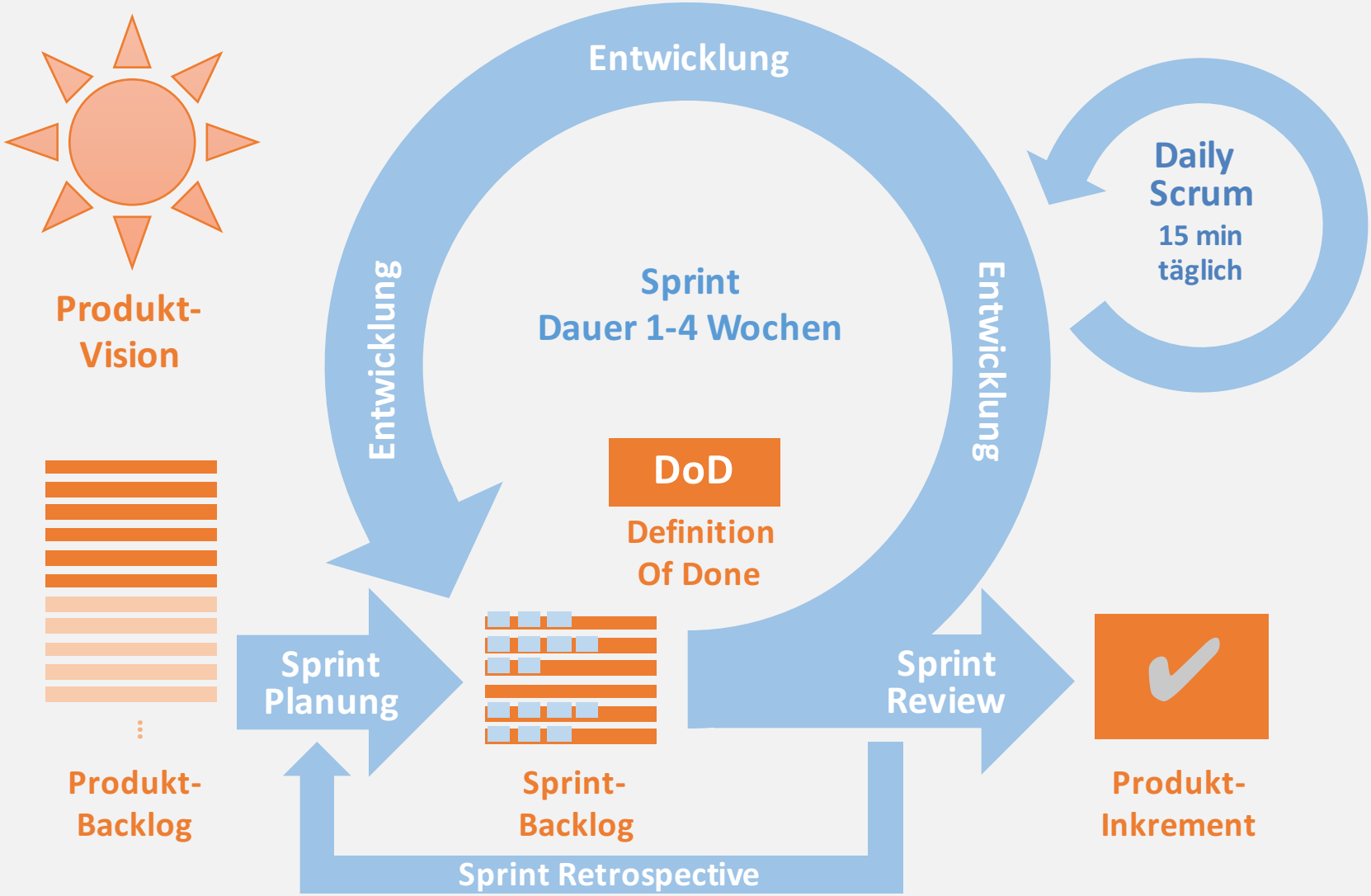
- Ideen aus der Lean Production (1980er, Japan), übertragen auf die SW-Entwicklung (Schwaber, Sutherland, 1990er)
- Bessere Wertschöpfung durch verstärkte Zusammenarbeit
- **Wissenstransparenz** → Selbstorganisation, Partizipation
- Kontinuierliche **Inspektion & Verbesserung** anstelle starrer Prozesse

Anders ausgedrückt: Ein Team ist am produktivsten, wenn

- es sich mit den Ziele und Bedürfnissen identifizieren
- die Arbeit in klare, unabhängige Einheiten eingeteilt ist
- es sich auf das Entwickeln konzentrieren kann
- Verbesserungsvorschläge wertgeschätzt und umgesetzt werden
- **schnell und kontinuierlich sichtbare Ergebnisse produziert werden**

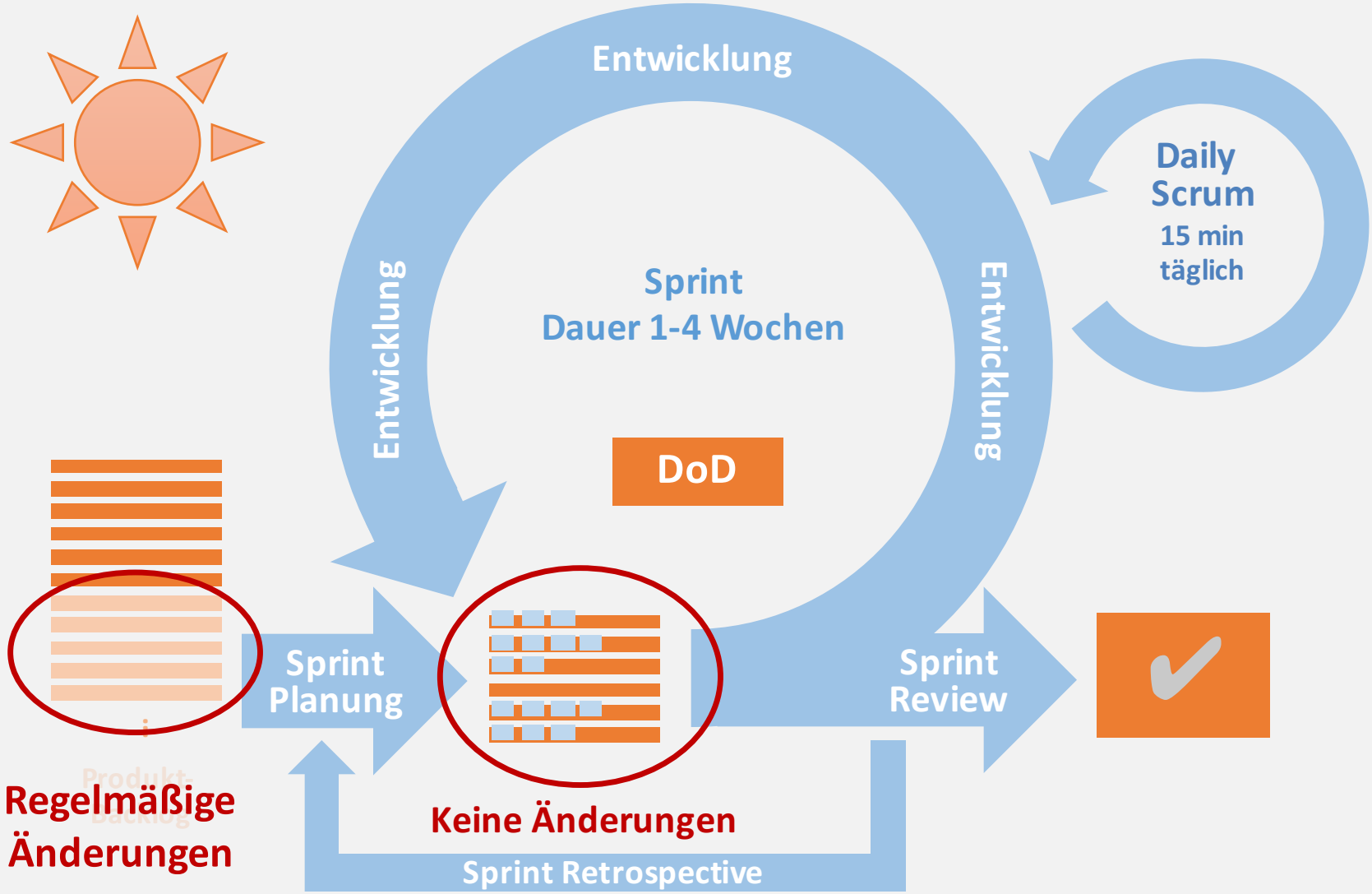
Referenz: Scrum Guide 2013 http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf

Scrum: Mechanik und Artefakte



Frei nach Drähter et at., *Scrum kurz & gut*, O'Reilly 2013

Scrum: Mechanik und Artefakte



Frei nach Drähter et at., *Scrum kurz & gut*, O'Reilly 2013

Rollen und Aufaben bei Scrum

- Nur drei Rollen:
  - **Product Owner:** muss Anforderungen priorisieren und erklären
  - **Scrum Master:** muss Treffen organisieren und moderieren und Probleme / Blockaden auflösen
  - **Entwickler:** erzeugt Produktinkremente durch Software-Entwicklung

- Ideale Größe 7 +/-2 Entwickler pro Team
  - Team soll interdisziplinär (cross functional) sein
  - Product Owner typischerweise außerhalb des Teams
  - Scrum Master oft Teil des Entwicklerteams

Scrum: Fazit

- Sprint-Zyklus: Review, Feedback, Planung, Änderungen
- Programmierer sind während des Sprints im „Flow“
- Das Produktbacklog als Kommunikations-Schnittstelle zwischen Stakeholdern, Management und Entwicklung

Ohne Backlog:
“Egal wie Ihr das macht, macht es bis Ende Juni!“

Mit Backlog:
„Du willst X schon bis Ende des nächsten Sprints? Dann sag mir, welche der Einträge oberhalb von X wir dafür opfern können.“

OK, so entwickelt man also
agil inkrementell Software.

Aber wer erzeugt wie das
Produktbacklog?

Und wie ist der Bezug zu
Anforderungen und Zielen?

Was ist Requirements Engineering?

Requirements Engineering:

Das Requirements Engineering ist ein systematischer und disziplinierter Ansatz zur Spezifikation und zum Management von Anforderungen (A.) mit den folgenden Zielen:

(1) Die relevanten A. zu kennen,
Konsens unter den Stakeholdern über die A. herzustellen,
die A. konform zu vorgegebenen Standards zu dokumentieren [...]

(2) [...] das Risiko zu minimieren, dass das System nicht den Wünschen und Bedürfnissen der Stakeholder entspricht.

*) Pohl, Rupp, *Basiswissen Requirements Engineering* , 4. Auflage 2015

Klassisches Requirements Engineering

- Stakeholder und Ziele ermitteln
- Anforderungen ermitteln
- Systemkontext ermitteln und dokumentieren
- Anforderungen dokumentieren und kategorisieren
  - Funktional vs. Qualität vs. Constraint,
  - Natürlichsprachlich (Sprachschablonen) oder modellbasiert
  - Use Cases, Diagramme, ...
- Anforderungen verfeinern und prüfen
- Anforderungen und Abhängigkeiten verwalten, Traceability

Siehe z.B. Pohl, Rupp, Basiswissen Requirements Engineering , 4. Auflage 2015

Klassisches Requirements Engineering

Stakeholder 1

Stakeholder 2

Stakeholder N

78

20

100

40

Ziel A

Ziel B

...

Ziel B.1

Ziel B.2

Anforderung NDF34

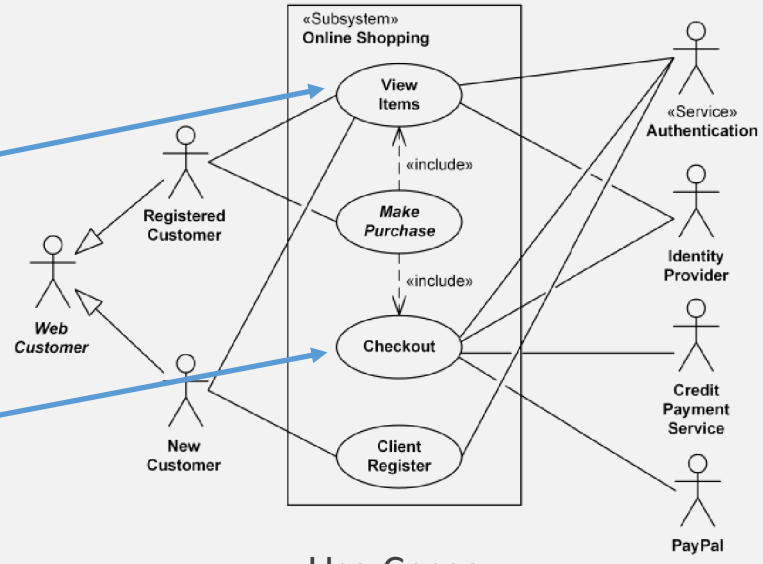
...

Anforderung NDF34.1

Anforderung NDF34.2

Anforderung NDF34.2.1

Anforderung NDF34.2.2



Use Cases

Klassisches Requirements Engineering

"This document, by its very size, ensures that it will never be read."

– Sir Winston Churchill

Stakeholder N

40

Ziel B.2

Anforderung NDF34

...

Anforderung NDF34.1

Anforderung NDF34.2

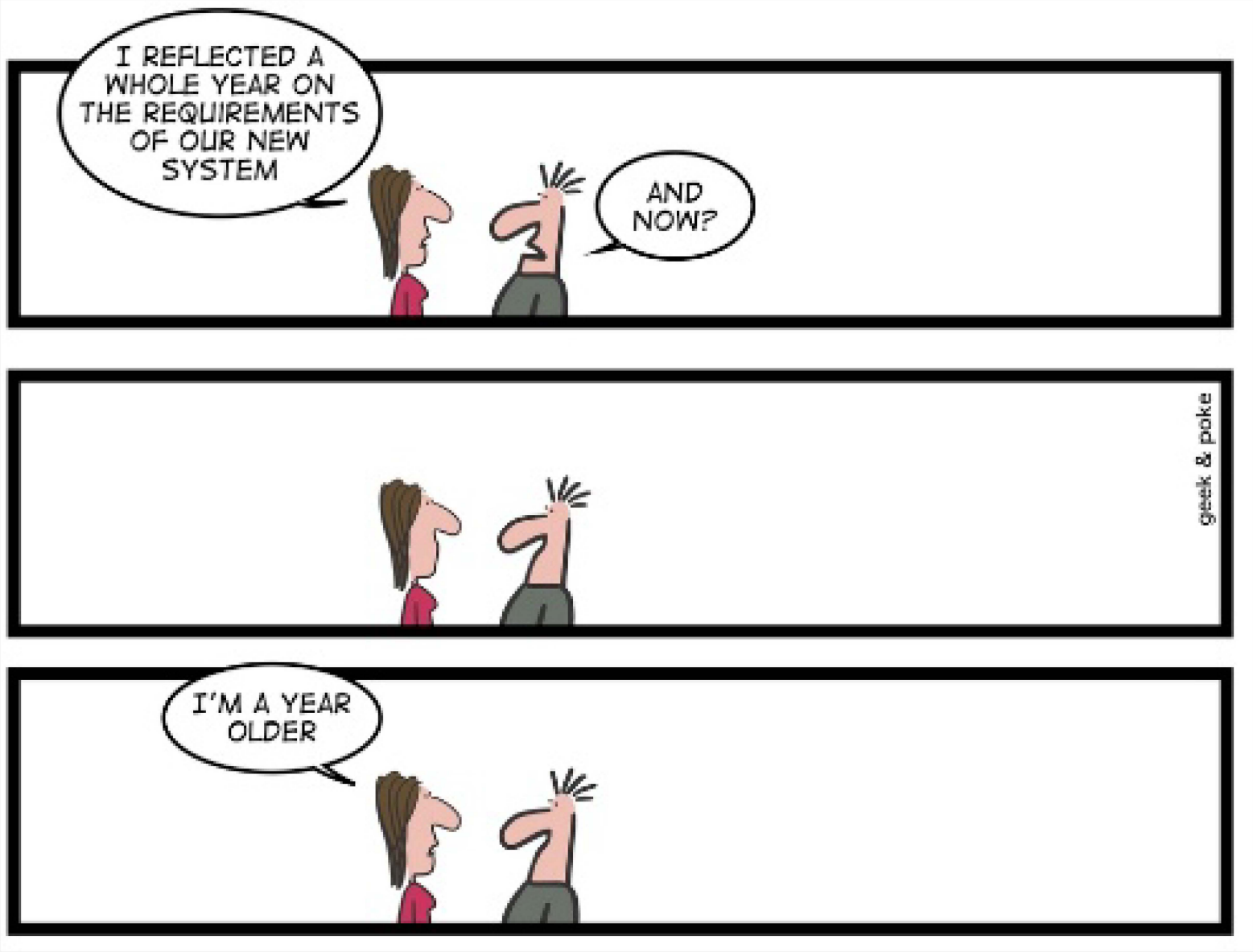
Anforderung NDF34.2.1

Anforderung NDF34.2.2

- Traceability: Verfolgung der Abhängigkeiten bei Änderung einzelner Artefakte
- Hohe Komplexität der dokumentierten Artefakte und Beziehungen
- Frust / Informationsverlust bei einfacher „Übergabe" an die Entwickler
- Angst vor Änderungen weit oben im Baum
- Modellierung oft mit stark funktionalem Fokus (eher was als warum)

PayPal

Use Cases



Cartoon: Oliver Widder, Geek and Poke Series, http://geek-and-poke.com (CC-BY-3.0)

Extreme Programming / User Stories

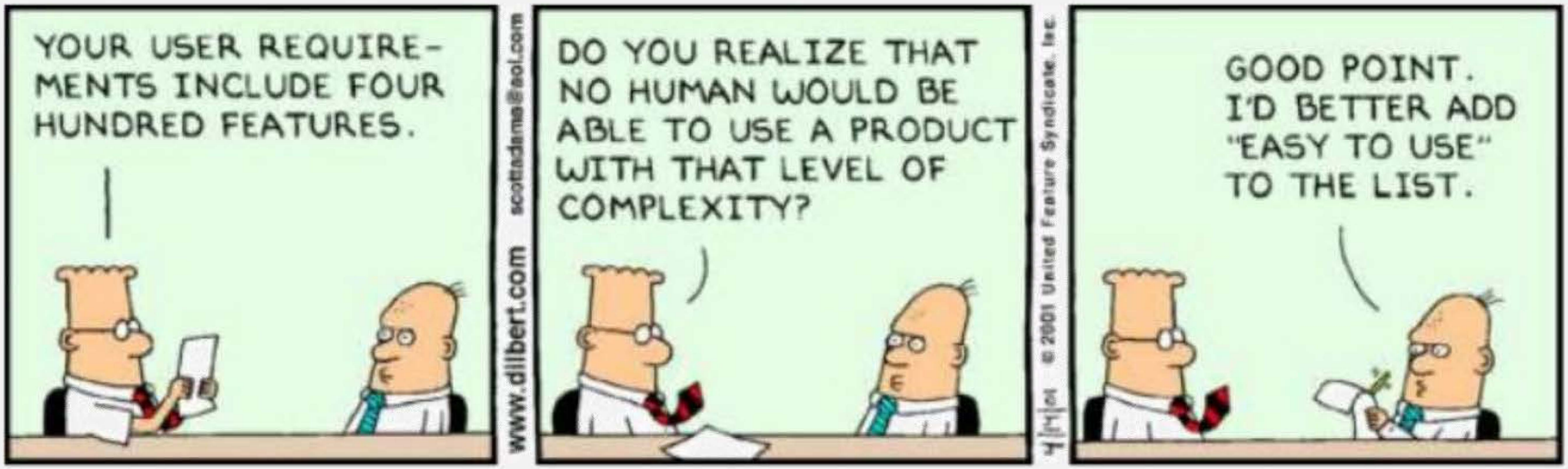
- Direkte, nichtschriftliche Kommunikation
- Story = „mentaler Anker“
- Kurze Entwicklungszyklen
- Frühe Schätzung
- Inkrementelles Design
- Pair Programming
- ...

Bildquelle: flyb.com

Literatur: Kent Beck & Cynthia Andres: *Extreme Programming Explained*, 2nd ed, Addision-Wesley 2005.

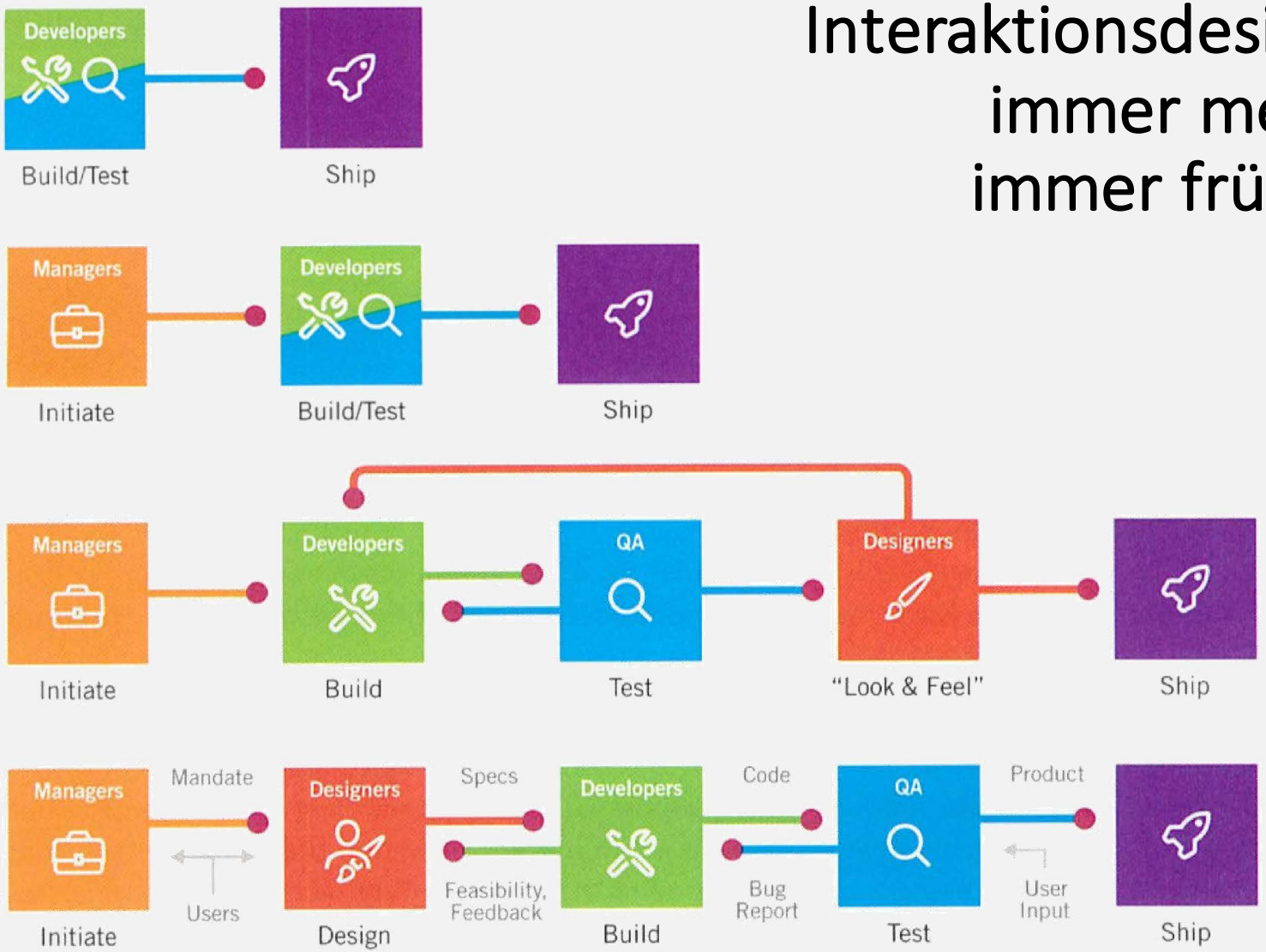
Systemübersicht

Bildmaterial mit freundlicher Genehmigung von Ursula Meseberg, MicroTOOL GmbH



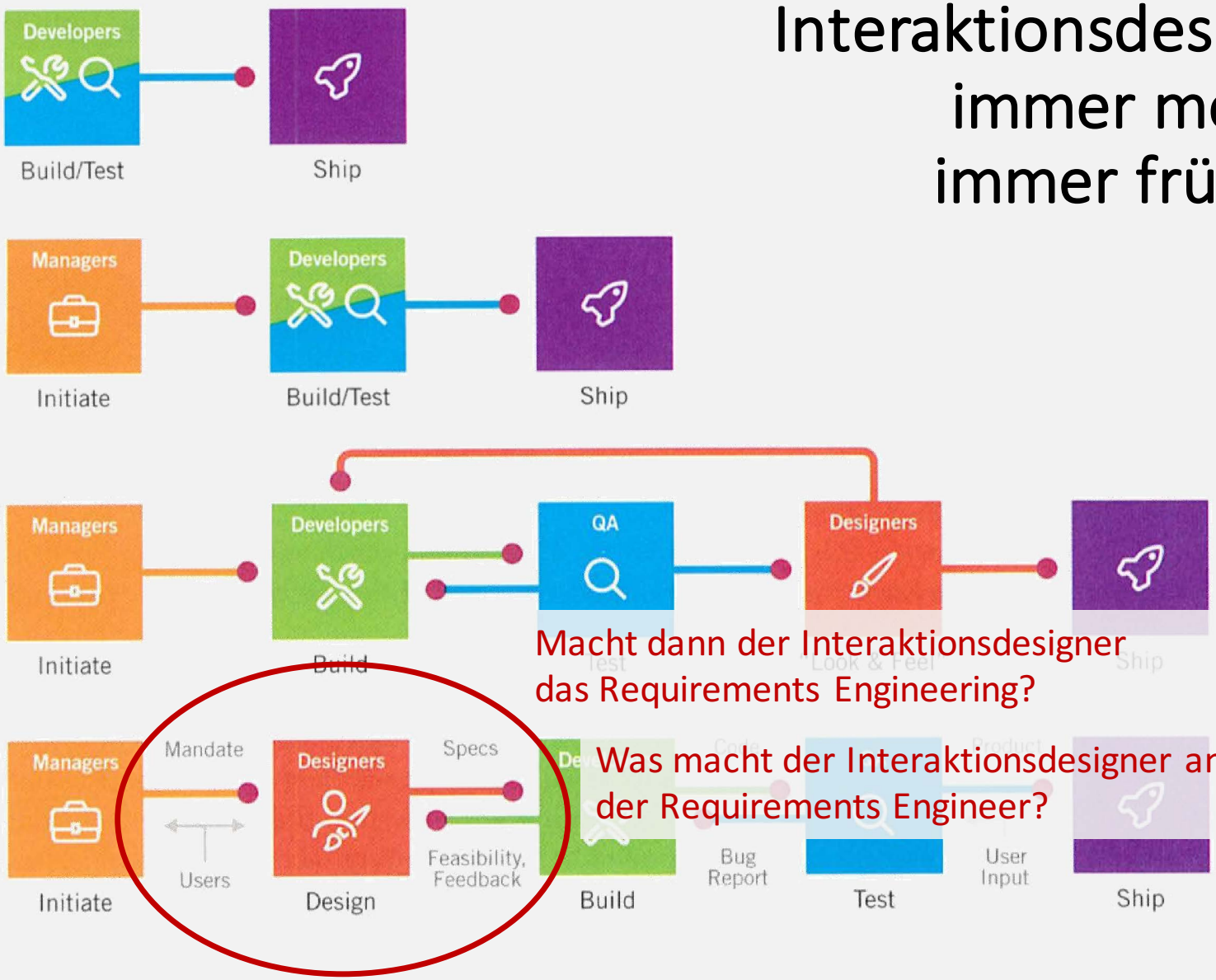
Cartoon by Scott Adams / www.dilbert.com

Interaktionsdesign: immer mehr, immer früher



Bildquelle: Alan Cooper et al.: *About Face – The Essentials of Interaction Design*, 4th ed., Wiley 2014

Interaktionsdesign: immer mehr, immer früher



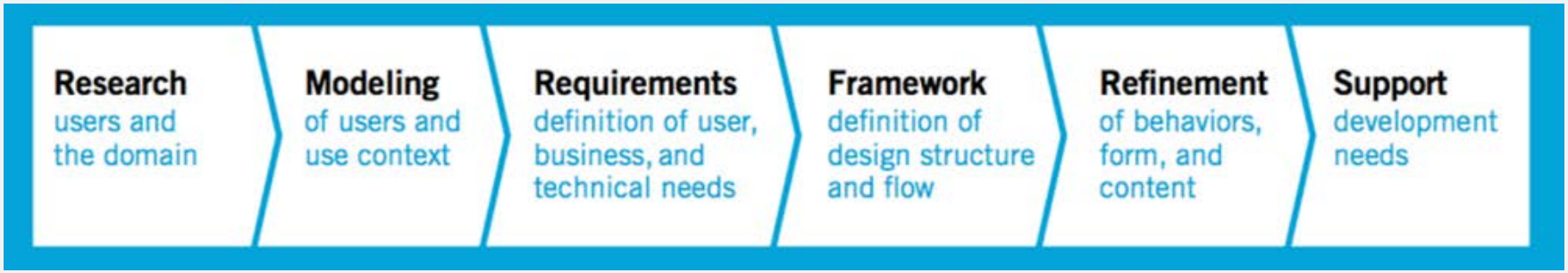
Macht dann der Interaktionsdesigner das Requirements Engineering?

Was macht der Interaktionsdesigner anders als der Requirements Engineer?

Alan Cooper et al.: *About Face – The Essentials of Interaction Design*, 4th ed., Wiley 2014

Goal Directed Design nach Alan Cooper

- Unterscheide Ziele von Aktivitäten
- **Ziele sind wichtiger als Aktivitäten**
- Verstehe das **mentale Modell der Benutzer**
- Durchlebe Schlüssel-Pfade durch das zukünftige Produkt



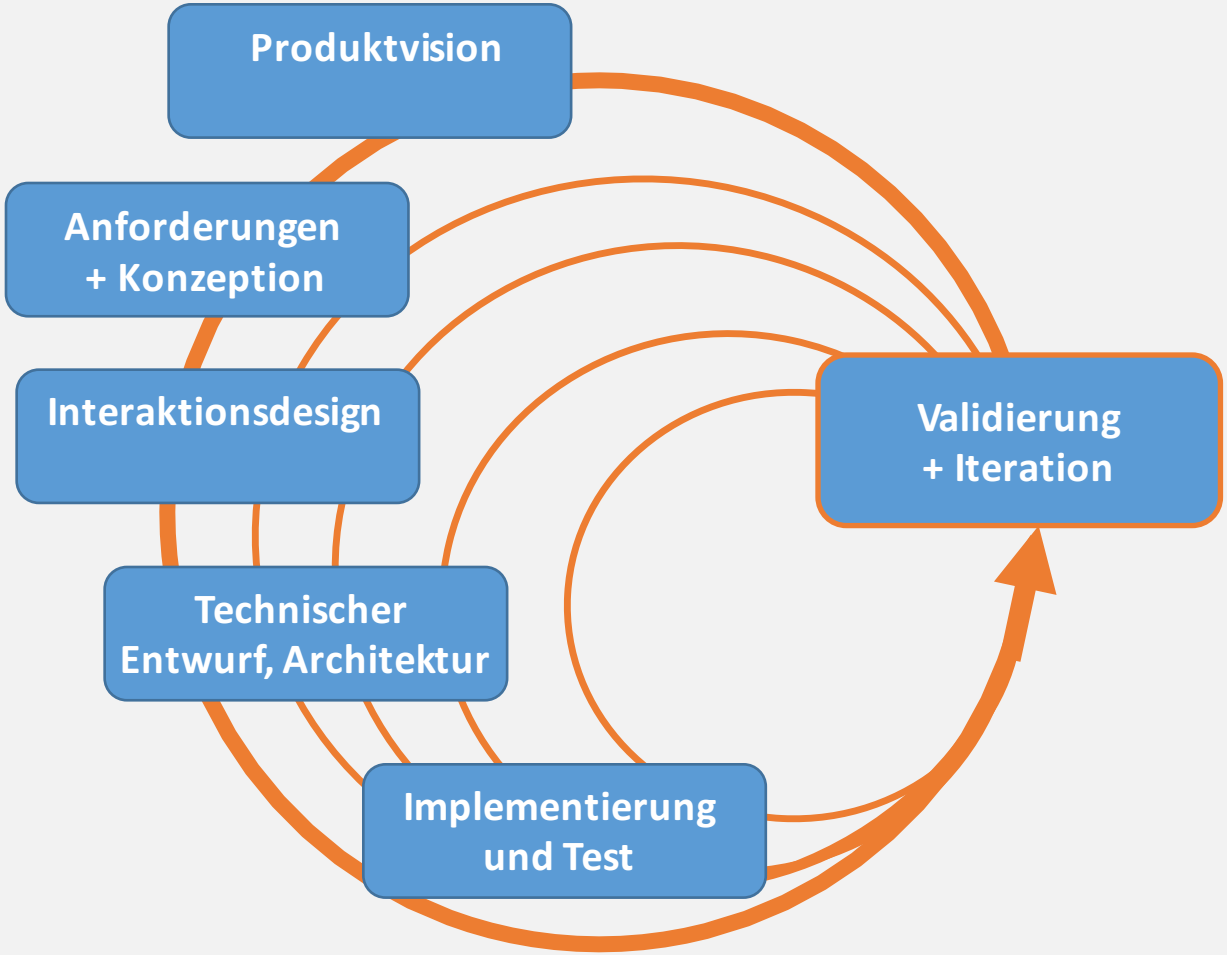
Personas **Kontext-Szenarien**
Workflow
Information Flow

Alan Cooper et al.: About Face – The Essentials of Interaction Design, 4th ed., Wiley 2014

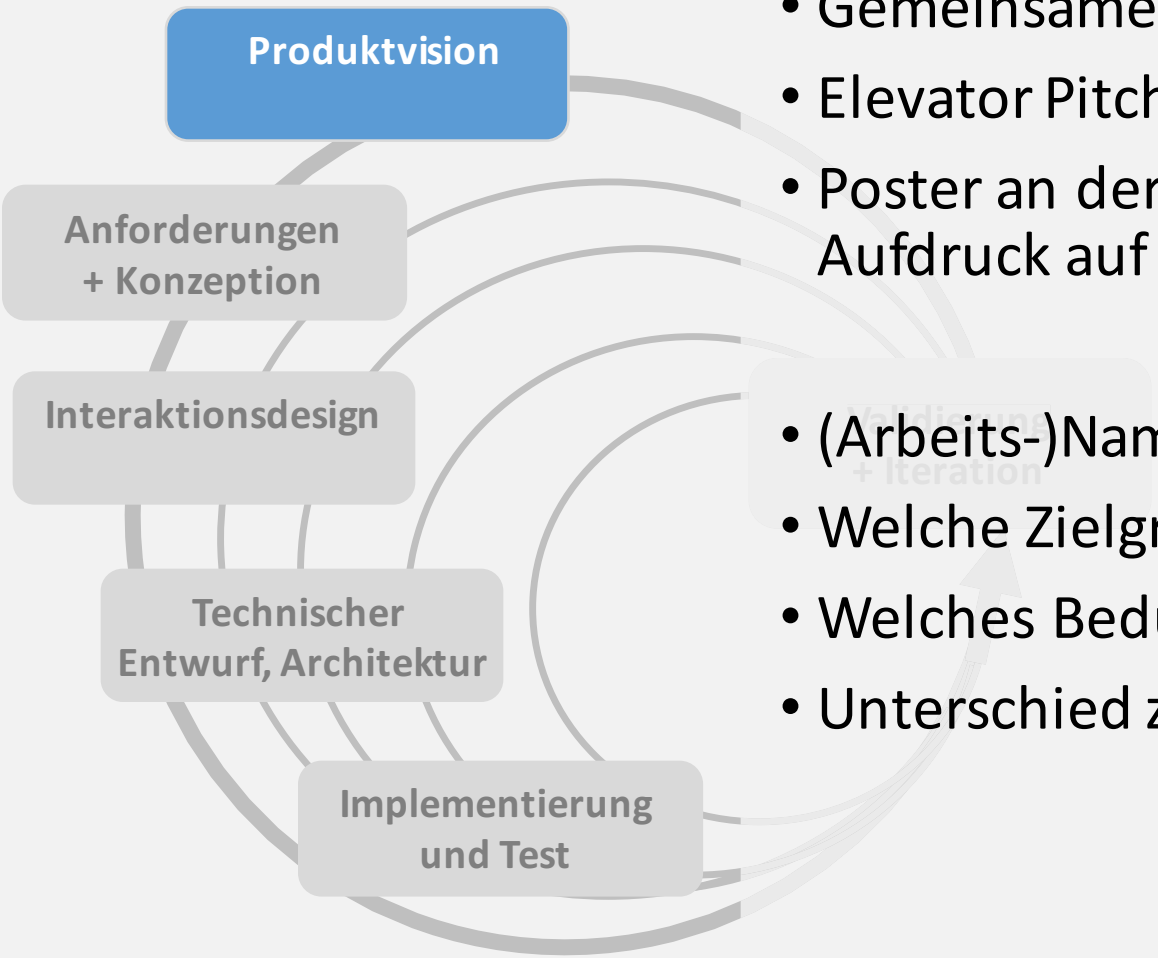
Von der Idee zum Produkt

Aktivitäten und Artefakte

Von der Idee zum Produkt



Produktvision



- Gemeinsame Vision des Teams
- Elevator Pitch (30 Sekunden)
- Poster an der Wand,
  Aufdruck auf der Kaffeetasse, ...

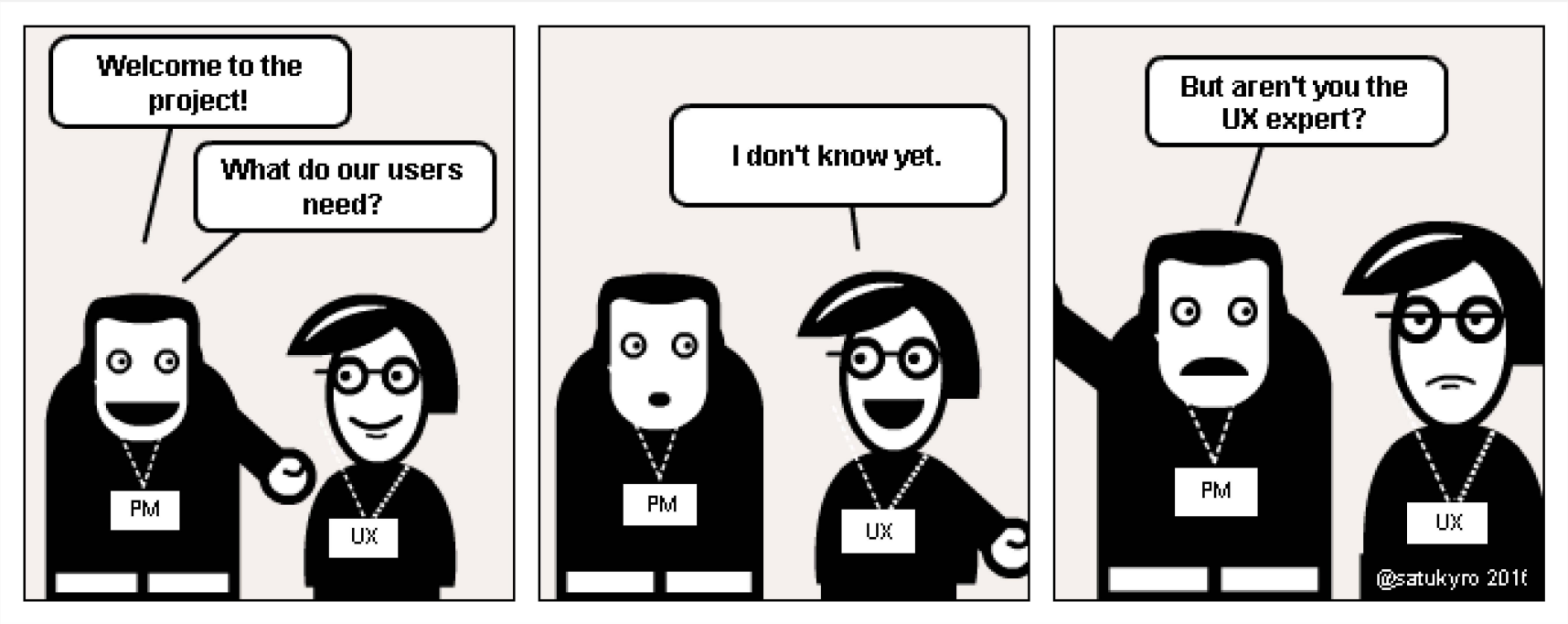
- (Arbeits-)Name?
- Welche Zielgruppe?
- Welches Bedürfnis?
- Unterschied zum Wettbewerb?

Tipps: https://www.scrum.org/resources/blog/10-tips-product-owners-product-vision

Produktvision: *Creativate*

- Zielgruppe: Hobbytreibende und Kreative
- Bedürfnisse:
  - Gleichgesinnte in der Nachbarschaft und auf Reisen finden
  - Treffen, Veranstaltungen, Kurse spontan besuchen oder buchen
- Unterscheidungsmerkmale:
  - Zum klassischen Verein: einfache Vernetzung über Ortsgrenzen hinweg, höhere Flexibilität und Vielfalt bei der Wahrnehmung von Angeboten
  - Zu unspezifischen soziale Medien: besser auf spezifische Zielgruppen abgestimmte Definition und Suche von Angeboten

Textwüste – viel besser wäre ein schön gestaltetes Poster mit ersten Mood- und Designelementen, und knackigere kurze Texte.



Cartoon by Satu Kyröläinen @satukyro

Anforderungsanalyse

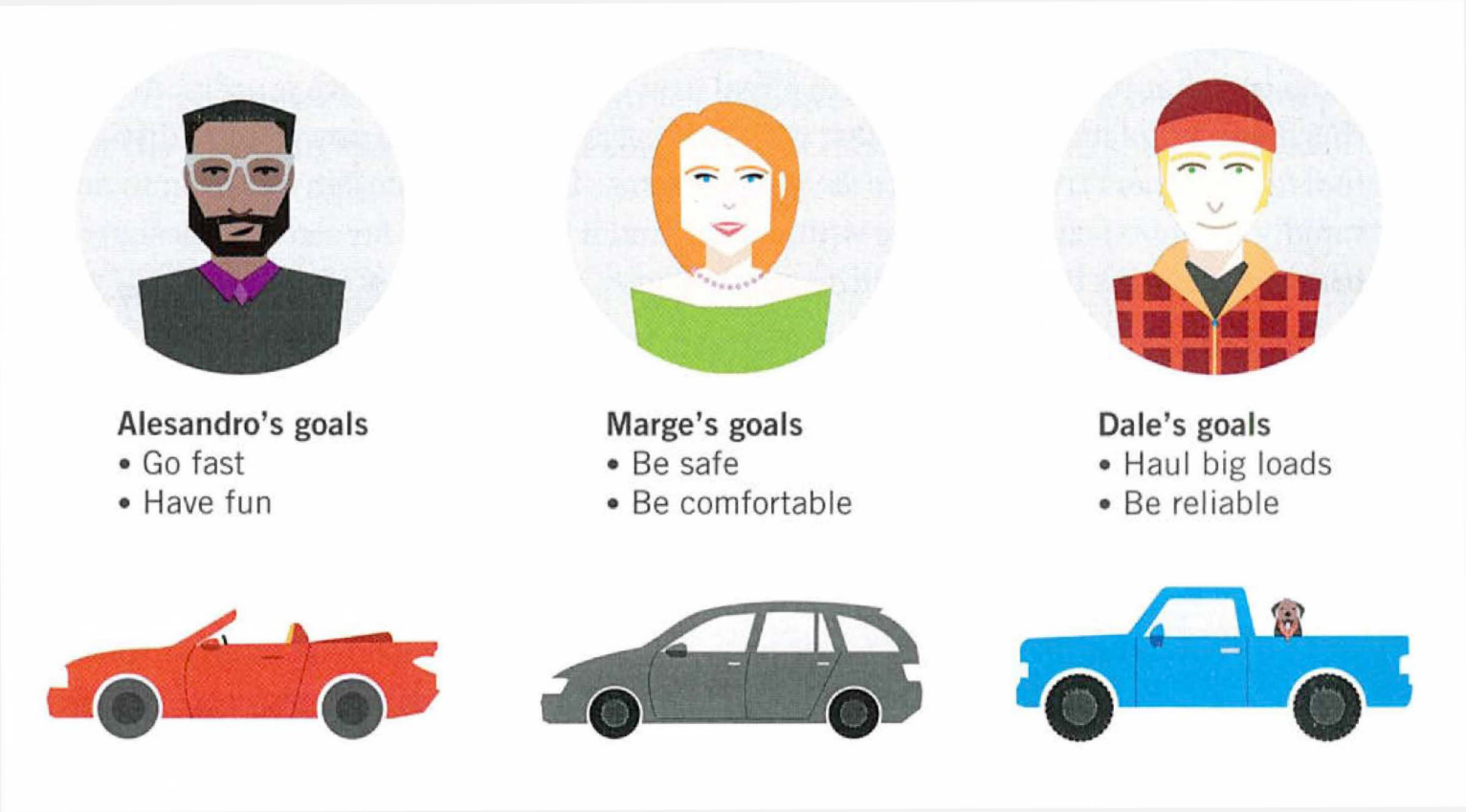
- Zielgruppe (Nutzer, Käufer)
- Weitere Stakeholder
- Motivation / Ziele
- Szenarien
- User Stories
- Produktbacklog
- Funktion vs. Qualität
- Scope / Projektumfang
- Verfeinerte Stories

Anforderungsanalyse: Stakeholder

- Stakeholder: jedeR, der/die Anforderungen beeinflusst
  - Auch Entwickler, Tester, Management
- Immer vertreten sollten sein:
  - Benutzer – wer wird das Produkt bedienen
  - Käufer – wer wird das Produkt bestellen / bezahlen?
- Idealerweise reale Personen als Vertreter der Zielgruppe(n)
  - Optimal für detaillierte Befragung, Beobachtung, Validierung
- Modellierung von Personas
  - Virtuelle Charaktere
  - Persönliche Motivation und Ziele
  - Relevante Charaktereigenschaften

Methoden:
- Brainstorming mit Stakeholdern
- Umfrage zu spezifischen Annahmen
- Nachdenken, Szenarien durchspielen

Personas



Bildquelle: Alan Cooper et al.: *About Face – The Essentials of Interaction Design*, 4th ed., Wiley 2014

Stakeholder und Ziele für Creativate

Welche Stakeholder fehlen hier?

Stakeholder	Beschreibung	Prio	Ziel
Paul	CEO von Creatix	++++	Große Plattform mit großer Userbase
		++++	Fördern von Kreativität anstelle von Medienkonsum
		++	Spielt selbst Saxophon, Angebote für Hobbymusiker schaffen
Chistopher	[Persona] Studi, 25, lernt gerade Gitarre	++++	Nach der Arbeit spontan Anregungen zu lockeren kostenfreien Musikertreffen erhalten (Jam Sessions mit Toleranz für Einsteiger)
		++	Je nach Lernfortschritt und Interesse passende Kursangebote finden und für jeweils ein paar Wochen buchen
		+	Inspirierende Konzerte finden
Ulla	[Persona] Angestellte, 44, sucht mehr Abwechslung beim Chorgesang	++++	Spontane Treffen nach Feierabend von Leuten zum mehrstimmigen Singen
		++	Chor-Hopping - Chöre finden, die für jeweils ein Projekt SängerInnen suchen.

Stakeholder und Ziele für Creativate

Wer sind die Anbieter? Was sind ihre Ziele?

Stakeholder	Beschreibung	Prio	Ziel
Paul	CEO von Creatix	++++	Große Plattform mit großer Userbase
		++++	Fördern von Kreativität anstelle von Medienkonsum
		++	Spielt selbst Saxophon, Angebote für Hobbymusiker schaffen
Chistopher	[Persona] Studi, 25, lernt gerade Gitarre	++++	Nach der Arbeit spontan Anregungen zu lockeren kostenfreien Musikertreffen erhalten (Jam Sessions mit Toleranz für Einsteiger)
		++	Je nach Lernfortschritt und Interesse passende Kursangebote finden und für jeweils ein paar Wochen buchen
		+	Inspirierende Konzerte finden
Ulla	[Persona] Angestellte, 44, sucht mehr Abwechslung beim Chorgesang	++++	Spontane Treffen nach Feierabend von Leuten zum mehrstimmigen Singen
		++	Chor-Hopping - Chöre finden, die für jeweils ein Projekt SängerInnen suchen.

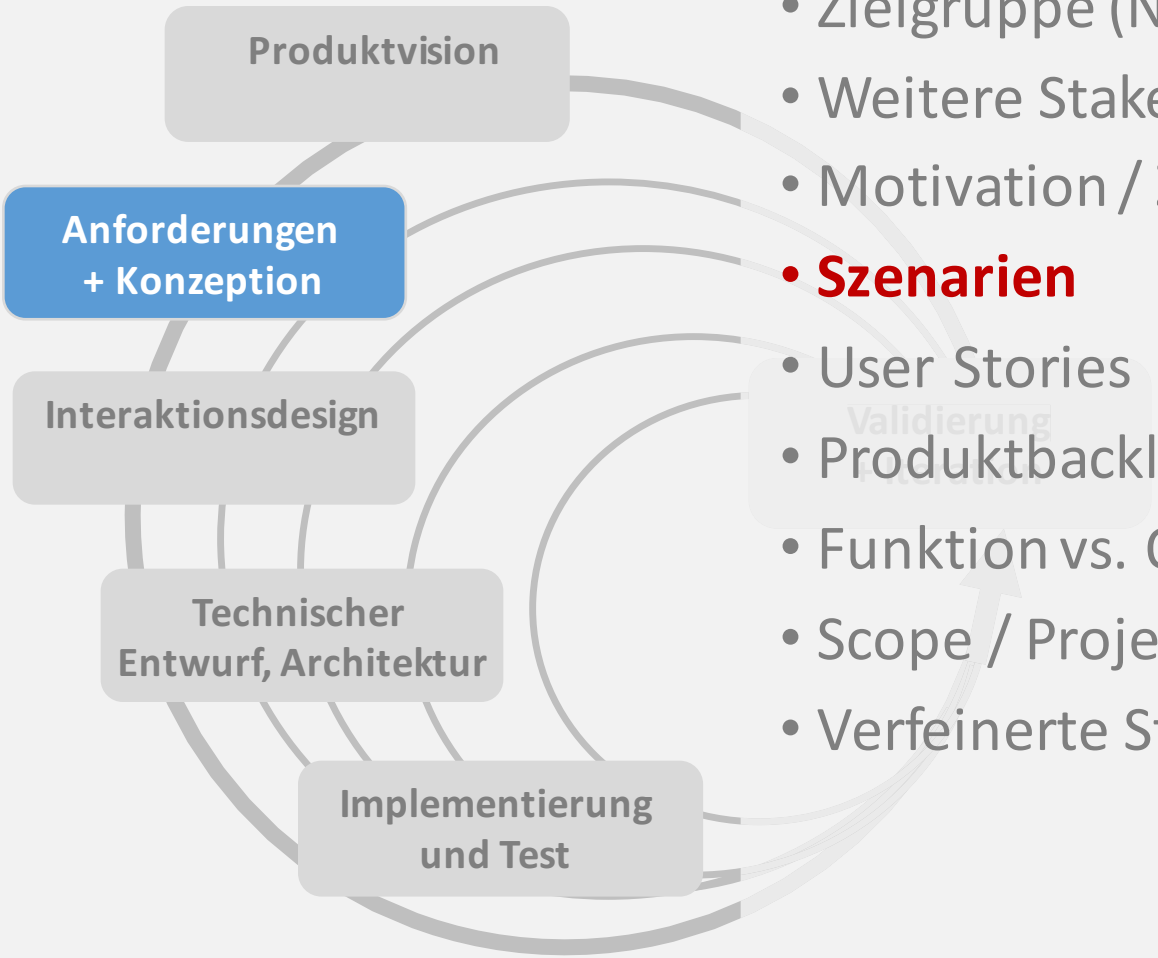
Geschäftsmodell für Creativate

Wer profitiert und möchte und kann zahlen?

Stakeholder	Beschreibung	Prio	Ziel
Paul	CEO von Creatix	++++	Große Plattform mit großer Userbase
		++++	Fördern von Kreativität anstelle von Medienkonsum
?		++	Spielt selbst Saxophon, Angebote für Hobbymusiker schaffen
Chistopher	[Persona] Studi, 25, lernt gerade Gitarre	++++	Nach der Arbeit spontan Anregungen zu lockeren kostenfreien Musikertreffen erhalten (Jam Sessions mit Toleranz für Einsteiger)
		++	Je nach Lernfortschritt und Interesse passende Kursangebote finden und für jeweils ein paar Wochen buchen ?
?		+	Inspirierende Konzerte finden
Ulla	[Persona] Angestellte, 44, sucht mehr Abwechslung beim Chorgesang	++++	Spontane Treffen nach Feierabend von Leuten zum mehrstimmigen Singen ?
		++	Chor-Hopping - Chöre finden, die für jeweils ein Projekt SängerInnen suchen.

Marktanalyse; Businessplan; ggf. Anpassung Produktvision.

Anforderungsanalyse



- Zielgruppe (Nutzer, Käufer)
- Weitere Stakeholder
- Motivation / Ziele
- **Szenarien**
- User Stories
- Produktbacklog
- Funktion vs. Qualität
- Scope / Projektumfang
- Verfeinerte Stories

Szenarien

- Ausgewählte, wichtige Beispiele
- Beschreiben die ideale User Experience
- In Form lockerer Erzählungen

Mischung von

- Abläufen
- Qualitätskriterien / Akzeptanzkriterien
  (wann/warum ist die Nutzung des Produkts erfolgreich?)
- Details des Benutzungskontextes

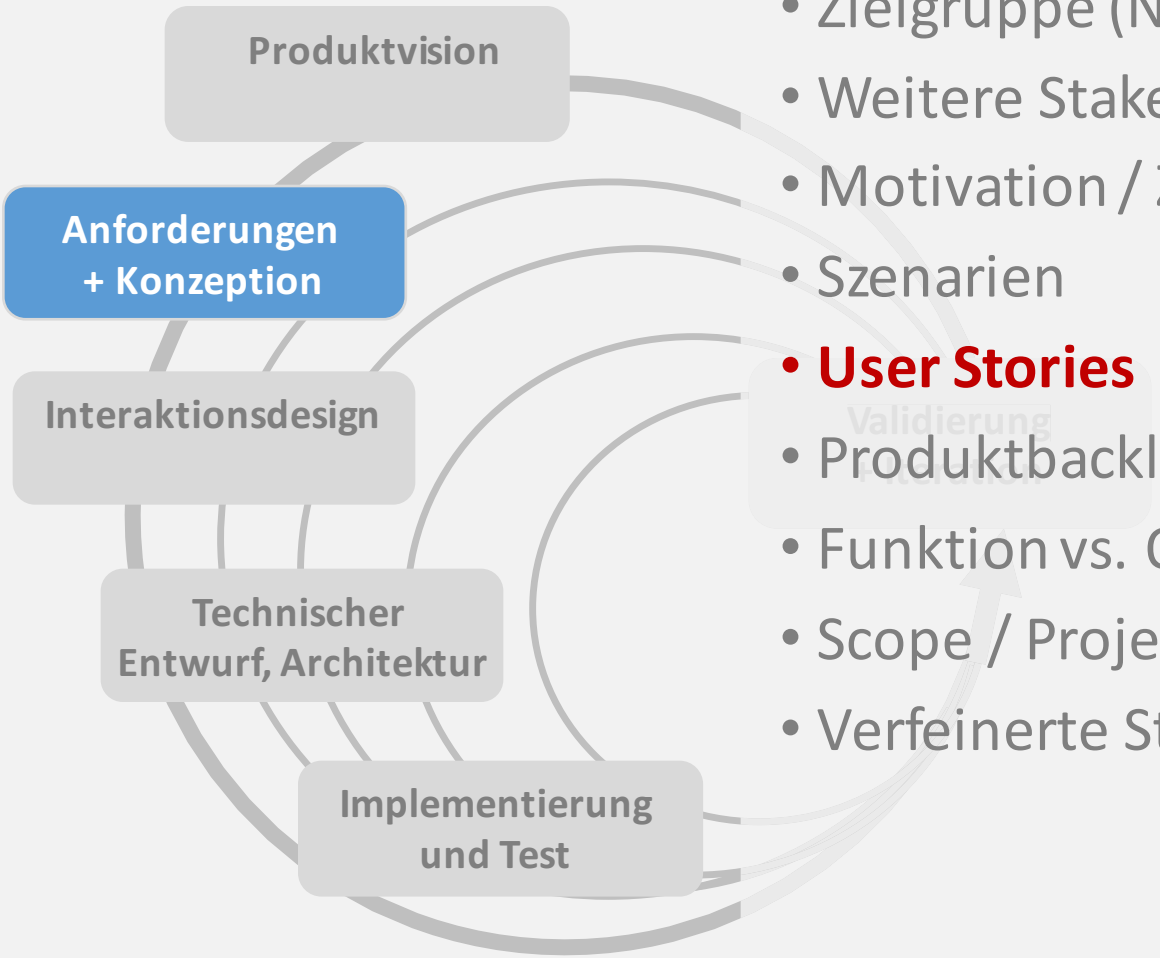
Szenario: Spontanes Musikertreffen

- Christopher hat heute keinen Unterricht mehr und die Sonne lacht. Er möchte noch ein bisschen unter Leuten sein und dabei Gitarre spielen. Er will nicht allzu weit fahren. Die Gitarre ist noch zuhause.
- Er sucht auf seinem Handy mit der Creativate-App nach Treffen mit den Eigenschaften „Musizieren“, „kostenfrei“, „außen“ in der Nähe seiner Wohnung noch am gleichen Tag und prüft anhand einer Karte und der jeweiligen Kurzbeschreibung, welche Veranstaltung ihm zusagt.
- Er findet eine Veranstaltung in einer Stunde in einem Park in seiner Nähe. Das Motto ist „Lagerfeuer-Songs der 90er“. Ein Sänge organisiert das ganze, zwei weitere Sänger und ein Gitarrist haben schon zugesagt. Notenmaterial ist teilweise vorhanden.
- Er signalisiert dem Gastgeber, dass er und seine Gitarre kommen wollen und er weiteres Notenmaterial mitbringt. Er fährt nach Hause, holt seine Sachen, und trifft sich zur angegebenen Zeit mit seinen Gleichgesinnten.
- Nach dem schönen Nachmittag im Park bewertet er die Veranstaltung. Er wird von der App auch aufgefordert, die Skill-Levels der anderen Teilnehmer zu bewerten.

Szenario: Chorprojekt

- Ulla hat Lust, in nächster Zeit mal wieder ein klassisches Requiem mit einem großen Chor und Orchester mitzusingen. Sie hat schon einige Jahre Chorerfahrung und kann einigermaßen gut vom Blatt singen.
- Sie sucht abends am Laptop auf der Creativate-Webseite nach Projekten mit dein Eigenschaften „Chor“, „Klassik“, einem Startdatum in den nächsten drei Monaten, und einem regelmäßigen Probendatum dienstags oder donnerstags.
- Sie schaut sich die Profile und Anforderungen der suchenden Chöre an und meldet sich bei zwei Projekten an, die gut zu ihren Vorstellungen passen.
- Sie macht ein paar Proben mit, bevor sie sich für ein Projekt entscheidet, dort aufgenommen wird und den Projektbeitrag einzahlt. Die Noten werden vom Chor besorgt.
- Während des Projekts bewertet Sie den Chor (aus Sicht einer Sängerin) und wird von der Chorleiterin und einigen Mitsängerinnen ebenfalls bewertet.

Anforderungsanalyse



- Zielgruppe (Nutzer, Käufer)
- Weitere Stakeholder
- Motivation / Ziele
- Szenarien
- **User Stories**
- Produktbacklog
- Funktion vs. Qualität
- Scope / Projektumfang
- Verfeinerte Stories

Story Card: Die Vorderseite

<ID>

Als <Rolle> möchte ich <Akivität/ Qualität> [, um <Ziel/Nutzen>] .

<Priorität> <Aufwand>

Optionale weitere Angaben:

- Titel, Risikobewertung, Ursprüngliche vs. aktuelle Schätzung, ...

User Stories: Beispiele

ID	Rolle	Aktivität	Ziel
1	Suchender	eigene Skills angeben	passende Veranstaltungen finden
2	Suchender	ortsbasiert nach Veranstaltungen suchen	lange Wege vom aktuellen Standort oder einem späteren Standort vermeiden
3	Veranstalter	Skills und Bewertungen der Teilnehmer sehen	passende Teilnehmer finden
4	Veranstalter	Skills und Bewerungen der Teilnehmer sehen	Teilnehmern bestätigen / absagen / abraten
5	Veranstalter	nach Teilnehmern mit bestimmten Eigenschaften suchen	eine optimale Gruppe von Leuten aktiv einladen
6	Benutzer	Suche nach mir durch Veranstalter verhindern	weil nur ich allein der Initiator meiner Aktivitäten sein möchte
7	Suchender	bei einer Veranstaltung anmelden	sicher zu sein, dass die Teilnahme klappt
8	Teilnehmer	Veranstaltung bewerten	anderen helfen, gute von schlechten Veranstaltungen zu unterscheiden

Story Card: Die Rückseite

Akzeptanzkriterien:

- Angenommen <Vorbedingung>, wenn <Aktion> dann <Ergebnis>.
- ...

Referenzen auf weiteres Material:

- ...

- Kriterien sollten so konkret sein, dass man daraus automatische Tests ableiten kann (Behaviour Driven Development, Gherkin-Syntax *)

*) Beispiel siehe https://de.wikipedia.org/wiki/Cucumber_(Software)

Das Prinzip der 3 C's

Card	Die Karte, Vorder- und Rückseite
Conversation	Kommunikation, Gespräche zum Inhalt und den Details der Story
Confirmation	Durchführung der Akzeptanztests

Ron Jeffries, Essential XP: Card, Conversation, Confirmation, 2001
http://xprogramming.com/articles/expcardconversationconfirmation/

Anforderung = Story + X

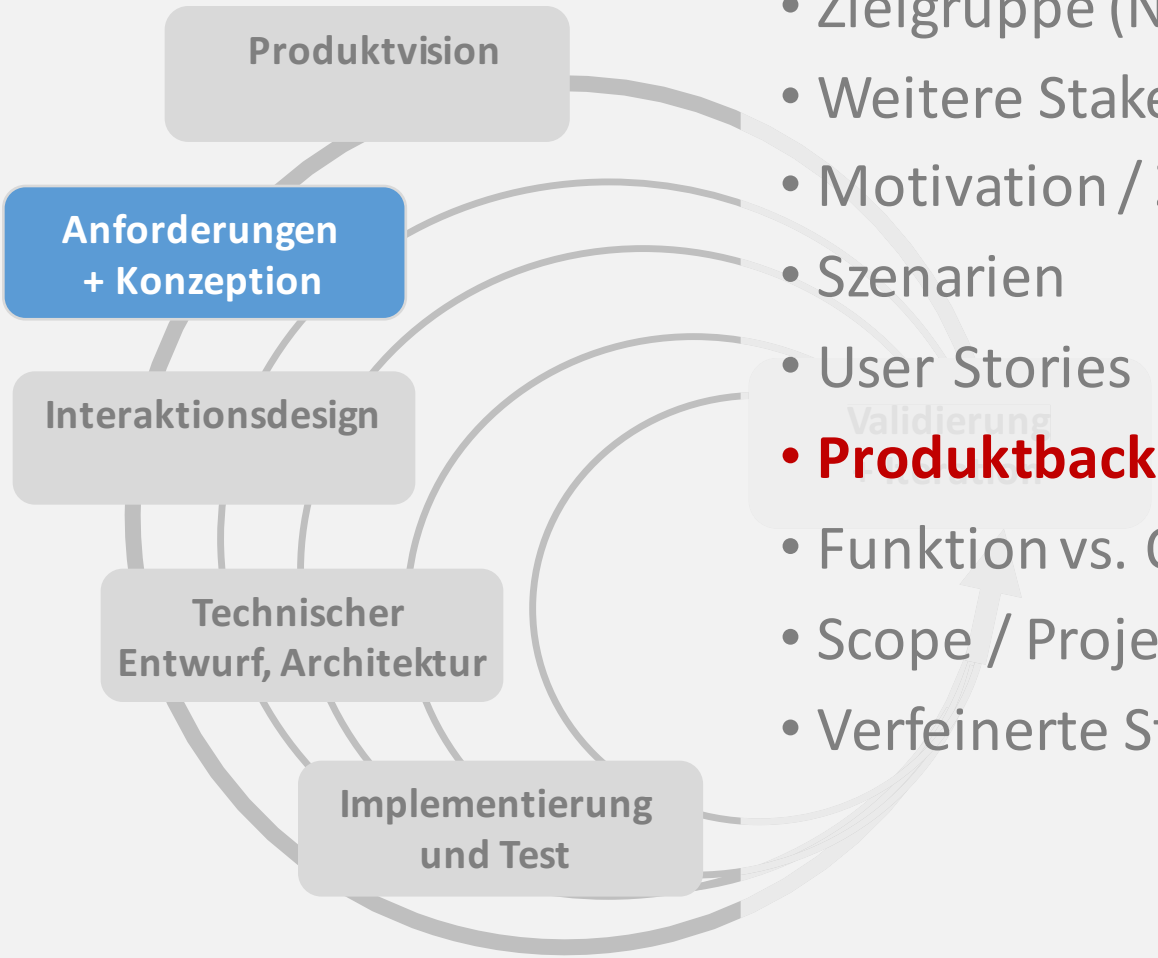
- Je nach Projektart und Umfang variiert die Menge der Informationen zu einer Anforderung, die man verschriftlichen möchte oder muss.

- Versionierung
- Review und Freigabe-Status
- Links zu Spezifikationen, Modellen, Tickets, Tests, ...

- Story = Einstiegspunkt, Kurzfassung, mentaler Anker

Bildquelle: https://www.solutionsiq.com/resource/blog-post/what-big-user-stories-could-be-telling-you/

Anforderungsanalyse



- Zielgruppe (Nutzer, Käufer)
- Weitere Stakeholder
- Motivation / Ziele
- Szenarien
- User Stories
- **Produktbacklog**
- Funktion vs. Qualität
- Scope / Projektumfang
- Verfeinerte Stories

Produktbacklog

- Bringe Stories in Reihenfolge der Priorität
- Priorität = was liefert den **größten Wert** (für die Stakeholder)
  - Abwägung gegen geschätzten Aufwand
  - Keine Zahlen/Skalen, Festlegung durch Reihenfolge im Backlog
- Attribuiere Stories mit grober Aufwandsschätzung
  - z.B. XS, S, M, L, XL, XXL – relativ zueinander zu verstehen

- Füge ggf. Stories für ein Minimal Viable Product hinzu, mit welchen das Projekt gestartet werden kann.

Produktbacklog Beispiel

ID	Aufwand	Rolle	Aktivität	Ziel
34	XXL	Veranstalter	einfach eine kostenlose Veranstaltung anbieten	Zeit mit Gleichgesinnten verbringen
12	XXL	Suchender	ortsbasiert nach Veranstaltungen suchen	lange Wege vom aktuellen Standort oder einem späteren Standort vermeiden
7	M	Suchender	bei einer Veranstaltung anmelden	sicher sein, dass die Teilnahme klappt
9	M	Veranstalter	Veranstaltung schließen	unangemessene Teilnehmerzahl vermeiden
…	…	…	…	…

- Auf dieser Basis grobe Zeiteinteilung in Sprints („im letzten Sprint haben wir 5 M-Stories geschafft“)
- Je weiter oben im Backlog, desto konkreter sollte die Story werden (XXL Stories verfeinern / aufteilen!)

Produktbacklog: Pflege

- Der Product Owner treibt das „Backlog Grooming“
- Priorisierungs–Sitzungen
  - Im Team, mit Entwicklung, Management / Vertreter der Kunden / ...
- Dokumentiere Status
  - Z.B. Entwurf / Inhalt abgestimmt / Prio festgelegt

Nach erstem Release: Change Management

- Eingehende Anfragen (Bug Reports, Enhancement Requests)
- Bewertung / Priorisierung durch Team (CCB)

Anforderungsanalyse

- Zielgruppe (Nutzer, Käufer)
- Weitere Stakeholder
- Motivation / Ziele
- Szenarien
- User Stories
- Produktbacklog
- **Funktion vs. Qualität**
- Scope / Projektumfang
- Verfeinerte Stories

Funktionalität vs. Qualität*

- Was vs. wie gut?
- Welche Funktionen muss ich implementieren vs.
  was müssen Design und Architektur sicherstellen?

Unterscheidung wichtig:

- Haben wir genügend über die Qualitätsanforderungen nachgedacht?
- Was müssen Interaktionsdesign und Architektur leisten?

Beispiele

- Als Suchender möchte ich mindestens fünf Orts-Favoriten anlegen und abrufen können, um schnell Veranstaltungen in der Nähe meiner häufigsten Aufenthaltsorte zu finden.
- Als Suchender möchte ich nie länger als eine Sekunde auf die ersten Ergebnisse einer Veranstaltungssuche warten.

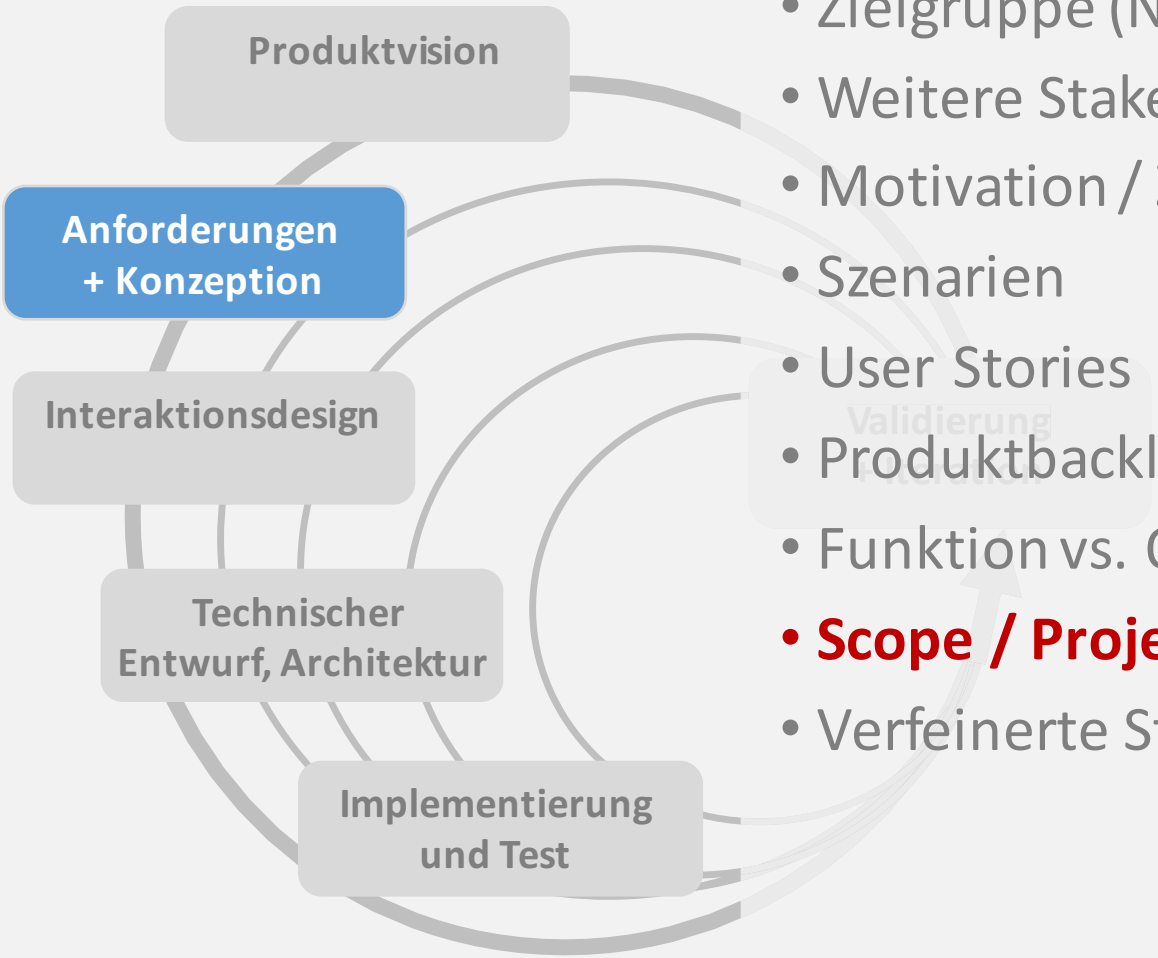
*) Qualitätsanforderungen sind auch unter dem Begriff *nichtfunktionale Anforderungen* bekannt.

Qualitätsanforderungen im Backlog?

- Macht meistens keinen Sinn!
  (Bsp. „Antwortzeit stets unter 1 Sekunde“)

- Qualitätsanforderungen gesammelt bei der SW-Architektur berücksichtigen
- Qualitätsanforderungen auf funktionale Anforderungen abbilden
  (meistens durch Vorgabe eines spezifischen Lösungsansatzes)
- Ursprüngliche Qualitätsanforderung / Akzeptanzkriterien
  aber auf jeden Fall beibehalten und referenzieren

Anforderungsanalyse



- Zielgruppe (Nutzer, Käufer)
- Weitere Stakeholder
- Motivation / Ziele
- Szenarien
- User Stories
- Produktbacklog
- Funktion vs. Qualität
- **Scope / Projektumfang**
- Verfeinerte Stories

Scope / Projektumfang

Möglichst früh Projektumfang klarstellen:

- Welche Stories gehören minimal zum erfolgreichen Projekt (erstes Release)
- Welche Stories werden definitiv nicht umgesetzt (in separate Liste, Entscheidung dokumentieren!)

Releaseplanung

- Horizonale Trennlinien im Backlog
- Je weiter unten eine Story steht, desto mehr wird sie zum „Stretch Goal“ für ein Release

Anforderungsanalyse

- Zielgruppe (Nutzer, Käufer)
- Weitere Stakeholder
- Motivation / Ziele
- Szenarien
- User Stories
- Produktbacklog
- Funktion vs. Qualität
- Scope / Projektumfang
- **Verfeinerte Stories**

Kriterien für gute User Stories

I ndependent	unabhängig von anderen Stories
N egotiable	Details sind verhandelbar: erfasse die Essenz der Anforderung
V aluable	liefert Mehrwert
E stimable	Arbeitsaufwand ist gut abschätzbar
S mall	maximal N Story Points / Personentage
T estable	testbar / konkrete Akzeptanzkriterien

Bill Wake, Invest in Good Stories and Smart Tasks, 2003: http://xp123.com/articles/invest-in-good-stories-and-smart-tasks

Verfeinerte Stories: Beispiel

ID	Aufwand	Rolle	Aktivität	Ziel
78	XL	Gast	über eine App registrieren und anmelden	um Veranstalter zu werden oder mich für eine Veranstaltung anzumelden
34	XXL	Veranstalter	einfach eine kostenlose Veranstaltung anbieten	Zeit mit Gleichgesinnten verbringen

Verfeinerte Stories: Beispiel

ID	Aufwand	Rolle	Aktivität	Ziel
78	XL	Gast	über eine App registrieren und anmelden	um Veranstalter zu werden oder mich für eine Veranstaltung anzumelden
34	XXL	Veranstalter	einfach eine kostenlose Veranstaltung anbieten	Zeit mit Gleichgesinnten verbringen
34.1	M	angemeldeter Benutzer	neue einmalige Veranstaltung anlegen	
34.2	M	Veranstalter	Liste der von mir angebotenen Veranstaltungen einsehen	
34.3	S	Veranstalter	Titel, Beschreibung, Zeit und Ort angeben	

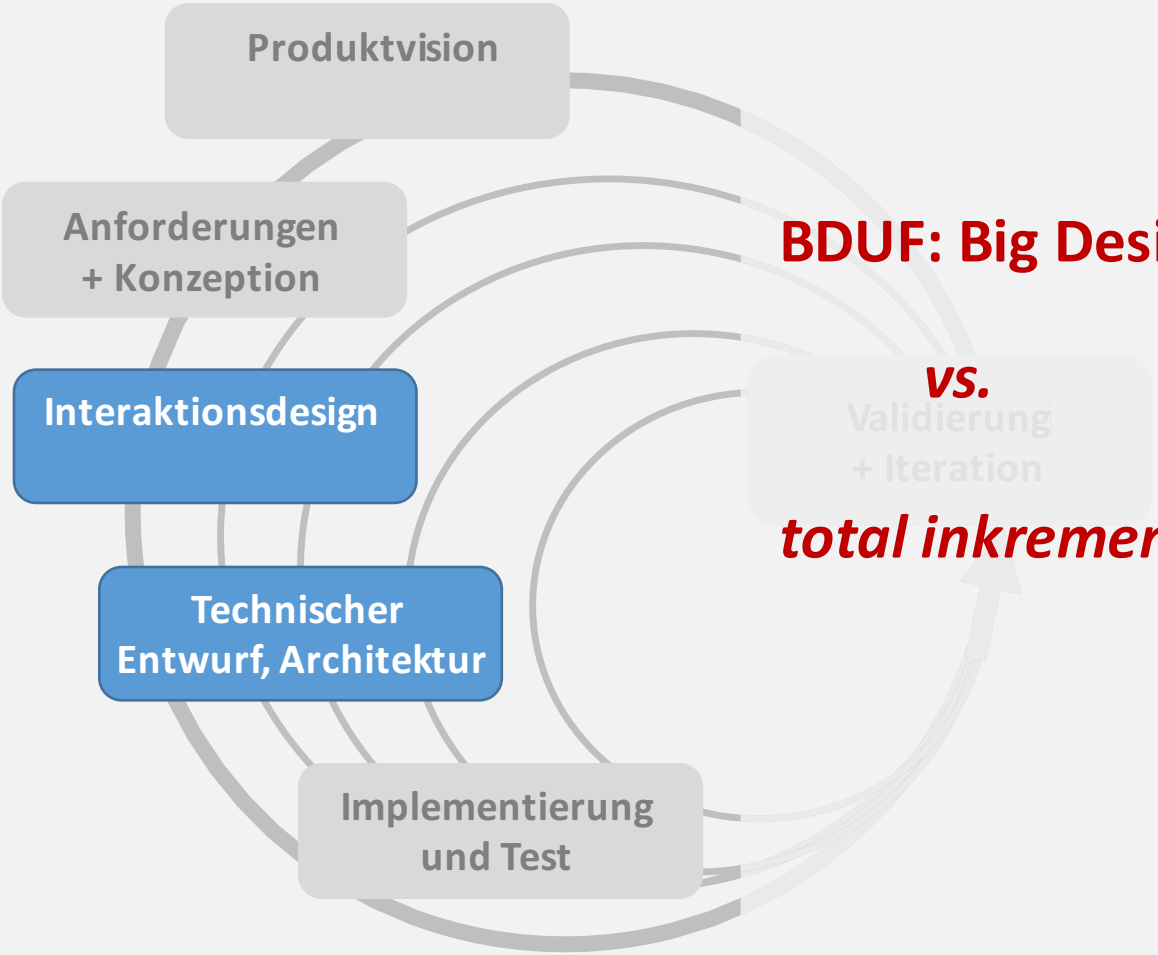
Verfeinerte Stories: Beispiel

ID	Aufwand	Rolle	Aktivität	Ziel
78	XL	Gast	über eine App registrieren und anmelden	um Veranstalter zu werden oder mich für eine Veranstaltung anzumelden
34	XXL	Veranstalter	einfach eine kostenlose Veranstaltung anbieten	Zeit mit Gleichgesinnten verbringen
34.1	M	angemeldeter Benutzer	neue einmalige Veranstaltung anlegen	
34.2	M	Veranstalter	Liste der von mir angebotenen Veranstaltungen einsehen	
34.3	S	Veranstalter	Titel, Beschreibung, Zeit und Ort angeben	
12	XXL	Suchender	ortsbasiert nach Veranstaltungen suchen	lange Wege vom aktuellen Standort oder einem späteren Standort vermeiden

Verfeinerte Stories: Beispiel

ID	Aufwand	Rolle	Aktivität	Ziel
78	XL	Gast	über eine App registrieren und anmelden	um Veranstalter zu werden oder mich für eine Veranstaltung anzumelden
34	XXL	Veranstalter	einfach eine kostenlose Veranstaltung anbieten	Zeit mit Gleichgesinnten verbringen
34.1	M	angemeldeter Benutzer	neue einmalige Veranstaltung anlegen	
34.2	M	Veranstalter	Liste der von mir angebotenen Veranstaltungen einsehen	
34.3	S	Veranstalter	Titel, Beschreibung, Zeit und Ort angeben	
12	XXL	Suchender	ortsbasiert nach Veranstaltungen suchen	lange Wege vom aktuellen Standort oder einem späteren Standort vermeiden
12.1	M	Gast	Liste angebotener Veranstaltungen in bestimmtem Zeitraum einsehen	
12.2	S	Gast	Details zu einem Angebot aufrufen	
12.3	L	Gast	Suche örtlich einschränken durch Eingabe einer Position	
12.4	S	angemeldeter Benutzer	Schnellauswahl der Zeit: sofort, in 30 Min, heute Abend, kommendes WoE	
12.5	M	angemeldeter Benutzer	Schnellauswahl des Orts: Zuhause, Favoriten (zuletzt/häufig verwendet)	

Designaktivitäten

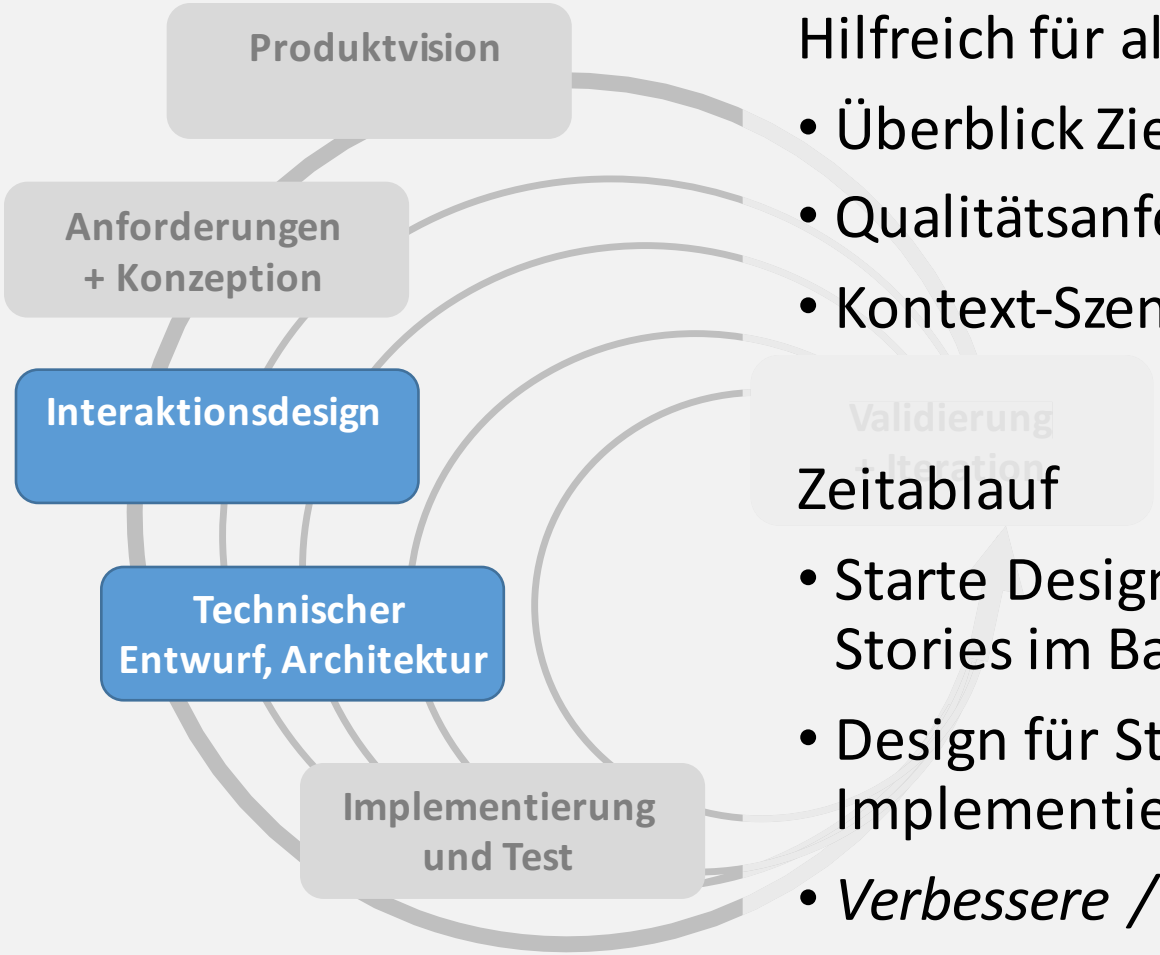


BDUF: Big Design Up Front

vs.

total inkrementell?

Designaktivitäten



Hilfreich für alle Designaktivitäten:

- Überblick Ziele + Funktionen
- Qualitätsanforderungen
- Kontext-Szenarien

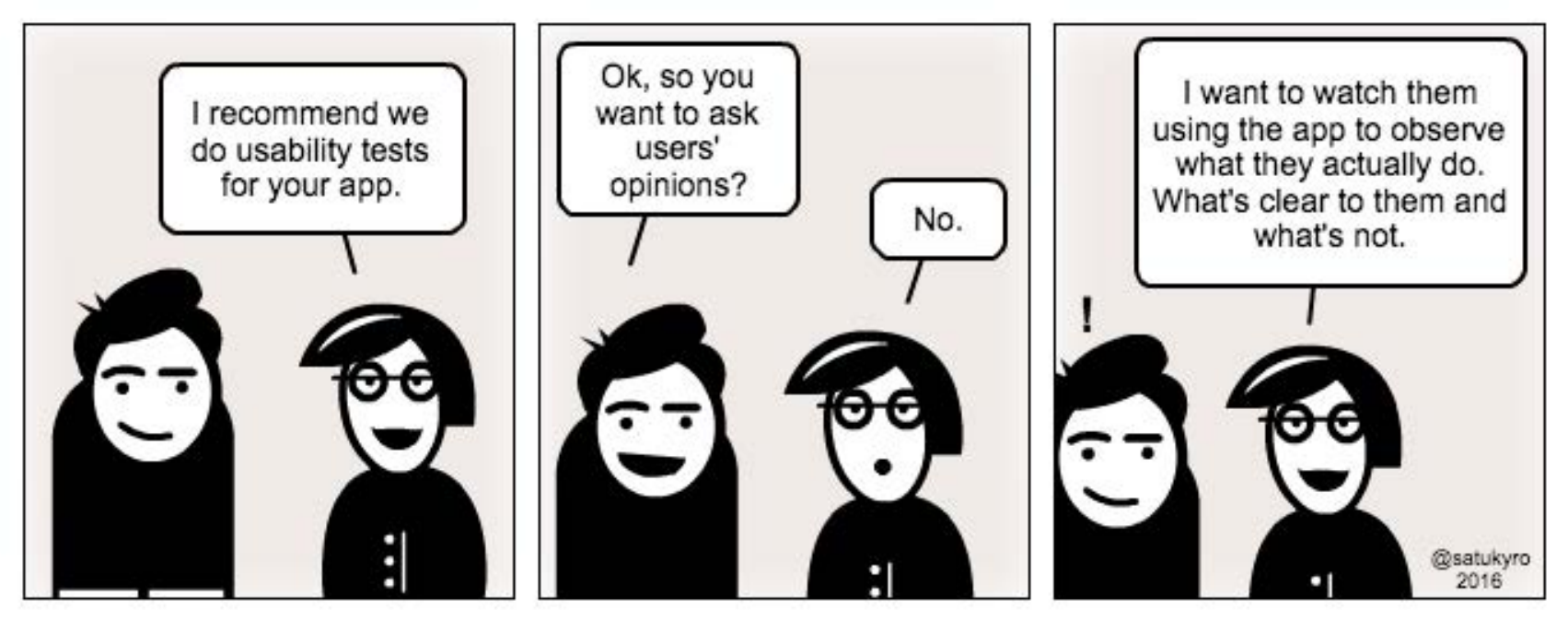
Zeitablauf

- Starte Design für die obersten Stories im Backlog
- Design für Stories soll vor deren Implementierung fertig sein
- *Verbessere / ergänze wenn nötig*

Interaktionsdesign

- Erstelle das Interaction Framework
  - Welche Screens
  - Grundsätzliches Layout
  - Wichtige Interaktionselemente
- Key Path Scenarios
  - Aus Basis der Kontextszenarien
  - Wie durchlebt der Benutzer das Produkt
- Konzeption und Feedback-Runden mittels Wireframes
- Pixel Perfect Entwürfe von Screens / Elementen
  - Oft auch Erweiterung zum Click Dummy

Usability Testing



Validierung auf Basis von

- Click Dummies oder
- Software-Prototypen

Cartoon by Satu Kyröläinen @satukyro

Software-Architektur (1)

- Constraints* ermitteln / festlegen
  - Unterstütze Zielplattformen
  - Unterstützte Sprachen
  - ...

- Schnittstellen ermitteln / festlegen
  - UI - siehe Interaktionsdesign
  - Systemkontext: wie bettet sich das System in andere existierende Systeme ein

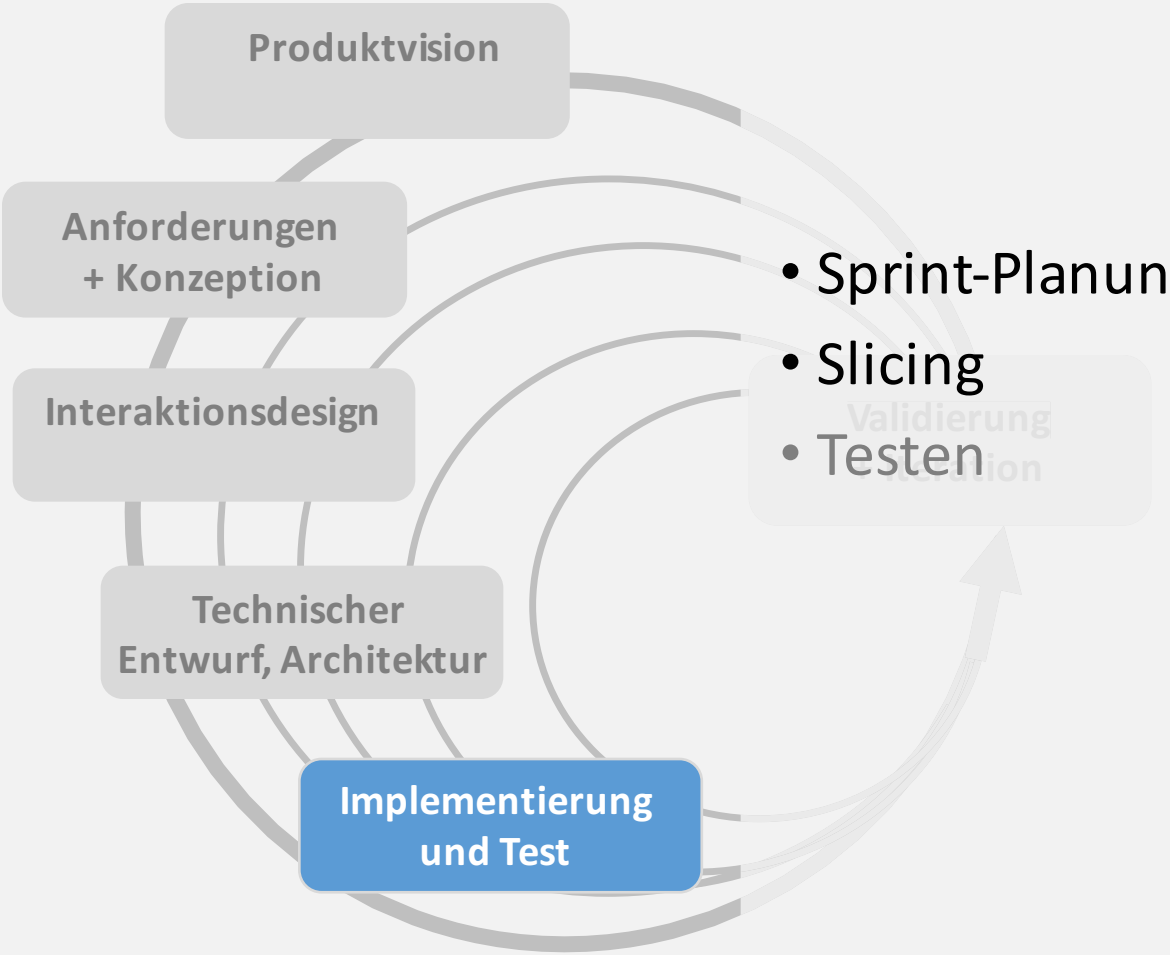
*) Constraints = Anforderungen, welche lediglich den Lösungsraum einschränken

Software-Architektur (2)

- Statische Sicht / Komponenten
  - Überblick der Funktionen, z.B. Use Case Diagramm
  - Welche Komponente soll für welche Funktionen zuständig sein
  - Welche Komponente hilft welche Qualitätsanforderung zu erfüllen
  - Einbindung externer Komponenten (3rd Party)
  - Grobstruktur Datenmodell

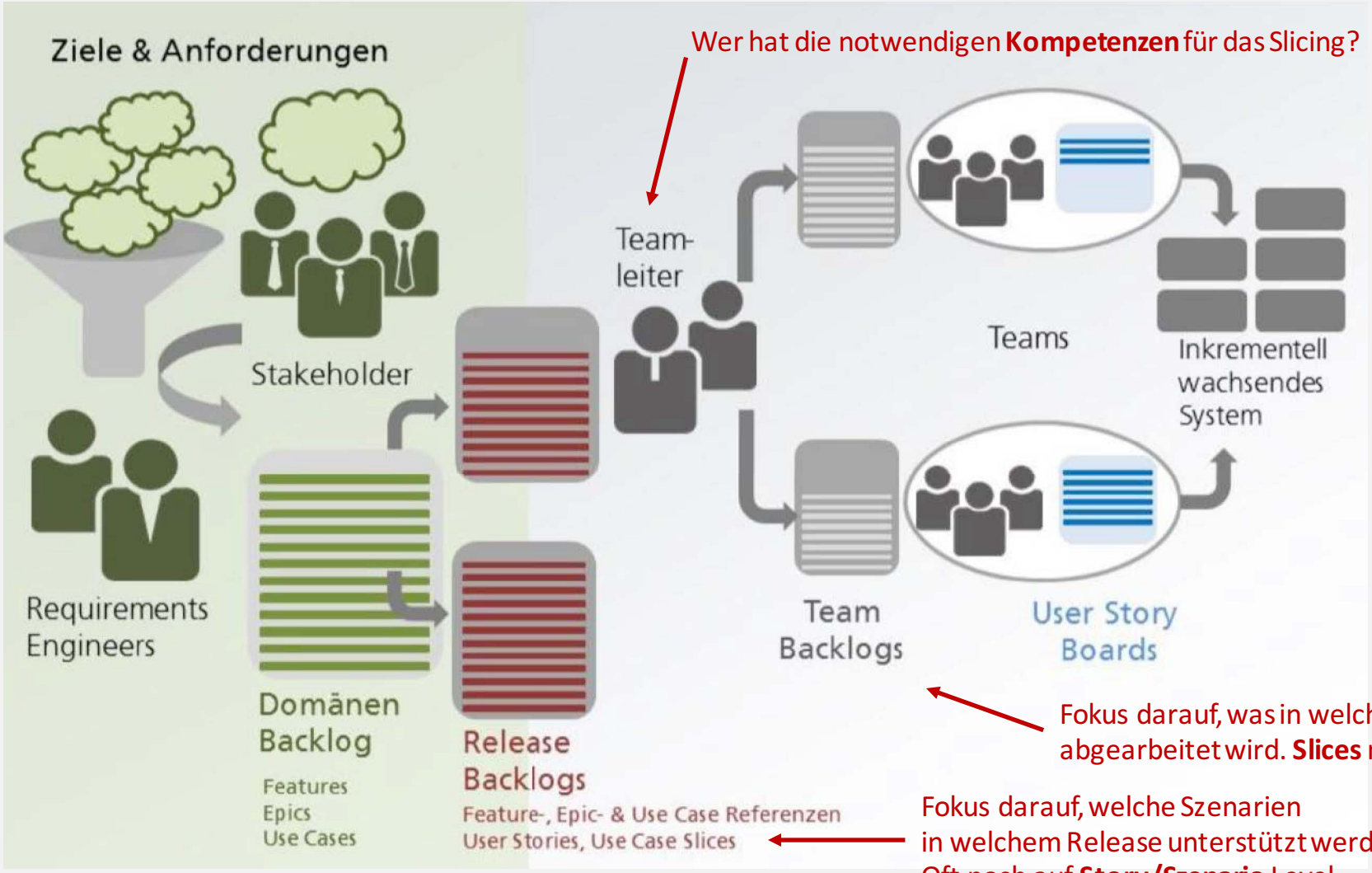
- Dynamische Sicht / Verhalten
  - Zustände und Zustandsänderungen
  - Entdecken bisher nicht im Design berücksichtigter Zustandsübergänge

Implementierung und Test



- Sprint-Planung
- Slicing
- Testen

Vom Produktbacklog zum Sprint-Backlog



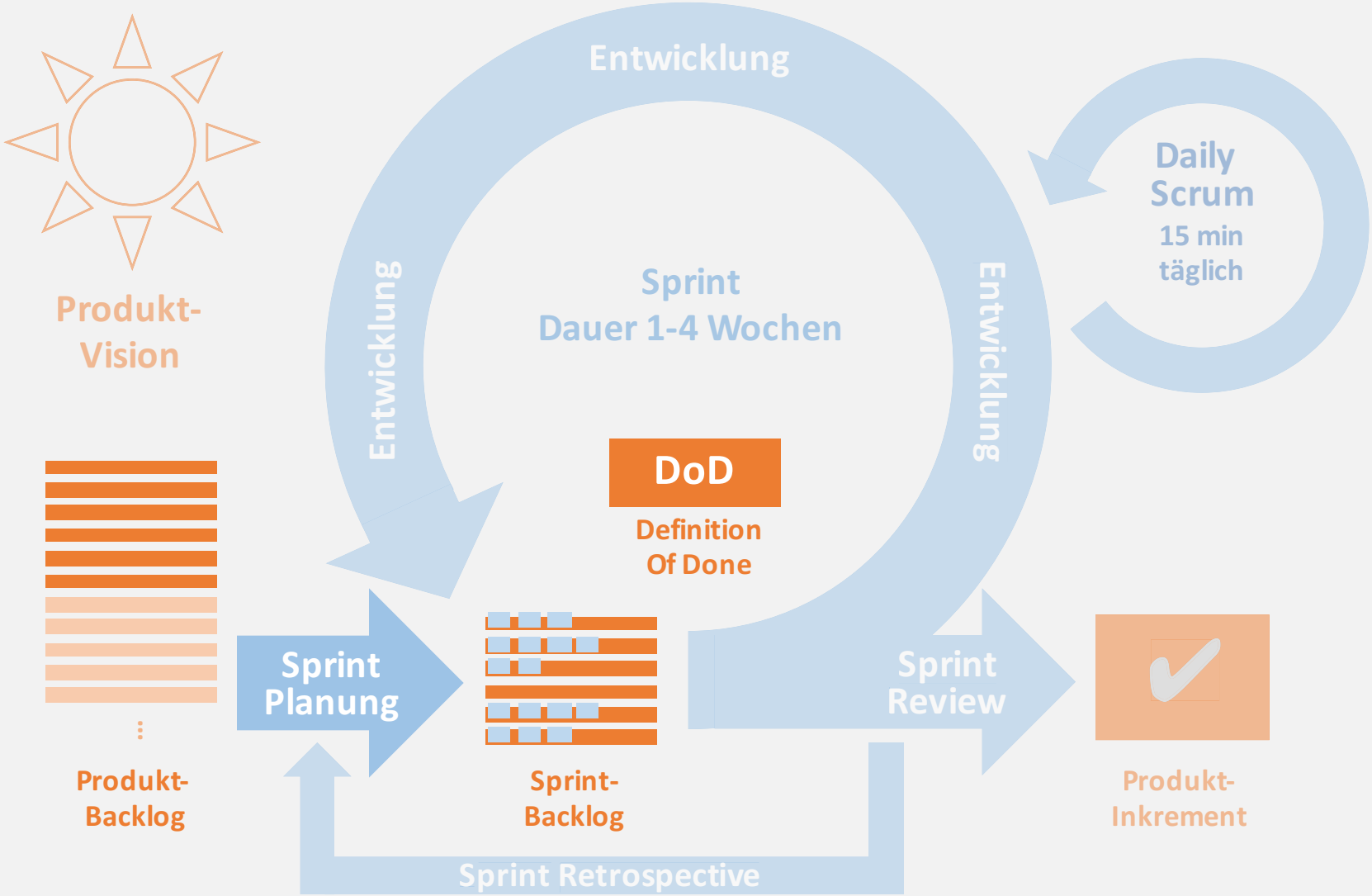
Wer hat die notwendigen **Kompetenzen** für das Slicing?

Fokus darauf, was in welcher Reihenfo
abgearbeitet wird. **Slices** notwendig.

Fokus darauf, welche Szenarien
in welchem Release unterstützt werden.
Oft noch auf **Story/Szenario** Level.

Bildquelle: U. Meseberg, *objectiF RPM Methodenhandbuch*, 2. Auflage 2017, MicroTOOL/intern

Sprintplanung



Frei nach Drähter et at., *Scrum kurz & gut*, O'Reilly 2013

Sprintplanung

Sprint-Planung Teil 1

- PO erklärt Stories / Slices
- Genauere Schätzung
- Wie viele Stories im nächsten Sprint?

Sprint Planung Teil 2

- SW-Team verfeinert Slices
- Klärung Architekturfragen

Frei nach Drähter et at., *Scrum kurz & gut*, O'Reilly 2013

Verfeinerung vs. Slicing

Verfeinerung:

Anforderungen so weit herunterbrechen, dass die Schritte der Szenarien klar werden, Akzeptanzkriterien existieren, und der Aufwand abschätzbar wird.

- Das Ergebnis sind immer noch Anforderungen, meistens in Form ausgearbeiteter Szenarien / Stories + Qualitätsanforderungen + Constraints
- Grundlage für Planung und Entwurf

Slicing:

Aufteilen von Anforderungen in Einheiten, welche unter Berücksichtigung des aktuellen Entwicklungsstandes und innerhalb eines Sprints umsetzbar sind.

- Das Ergebnis sind Slices/Tickets/Aufgaben für das Entwicklungsteam.
- Wird für agile Entwicklung (Scrum, Kanban, kurze Zyklen) benötigt

Bildquelle: Google Bildersuche / unbekannt

Anforderungen an Slices

Slice idealerweise so, dass weiterhin

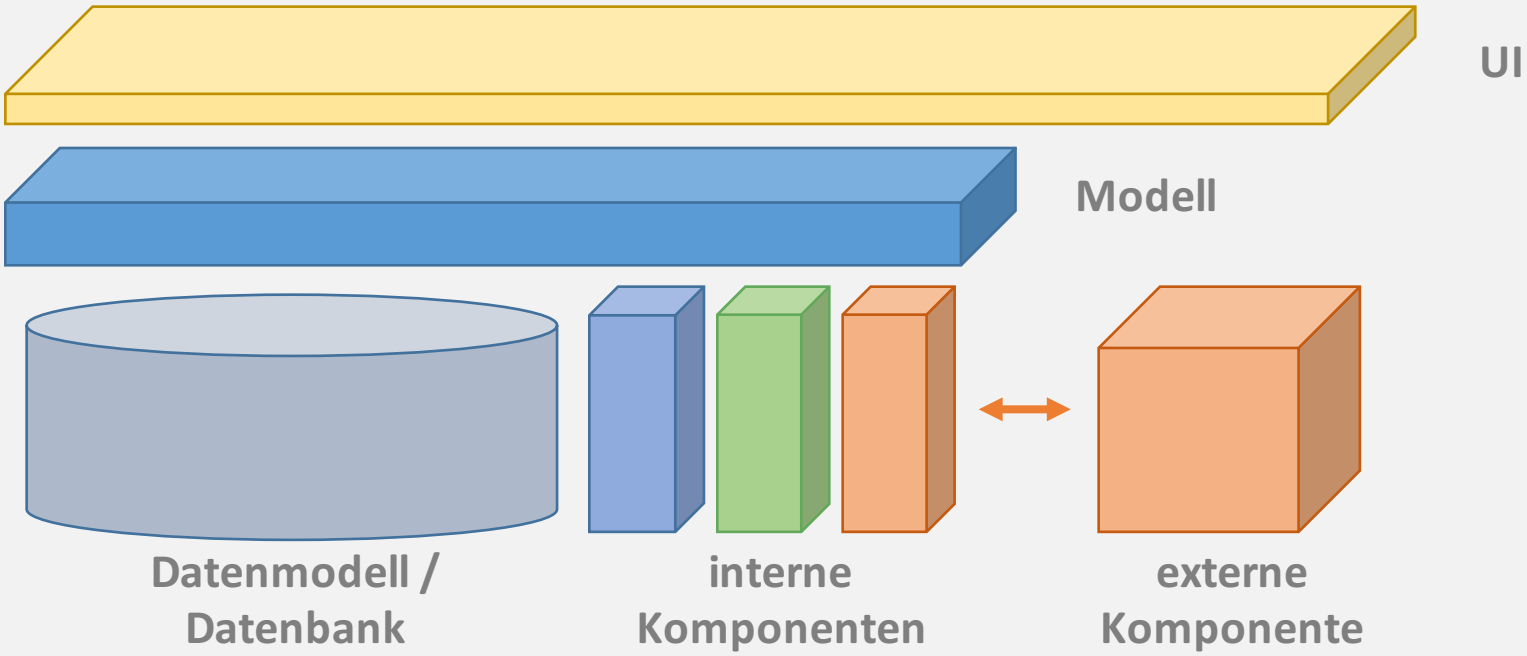
- jeder implementierte Slice <u>funktioniert</u>,
- jeder Slice einen zusätzlichen <u>Wert</u> schafft,
- jeder Slice <u>Feedback</u> vom Anwender erzeugen kann

(Prinzipien für Scrum / Backlog-Einträge*)

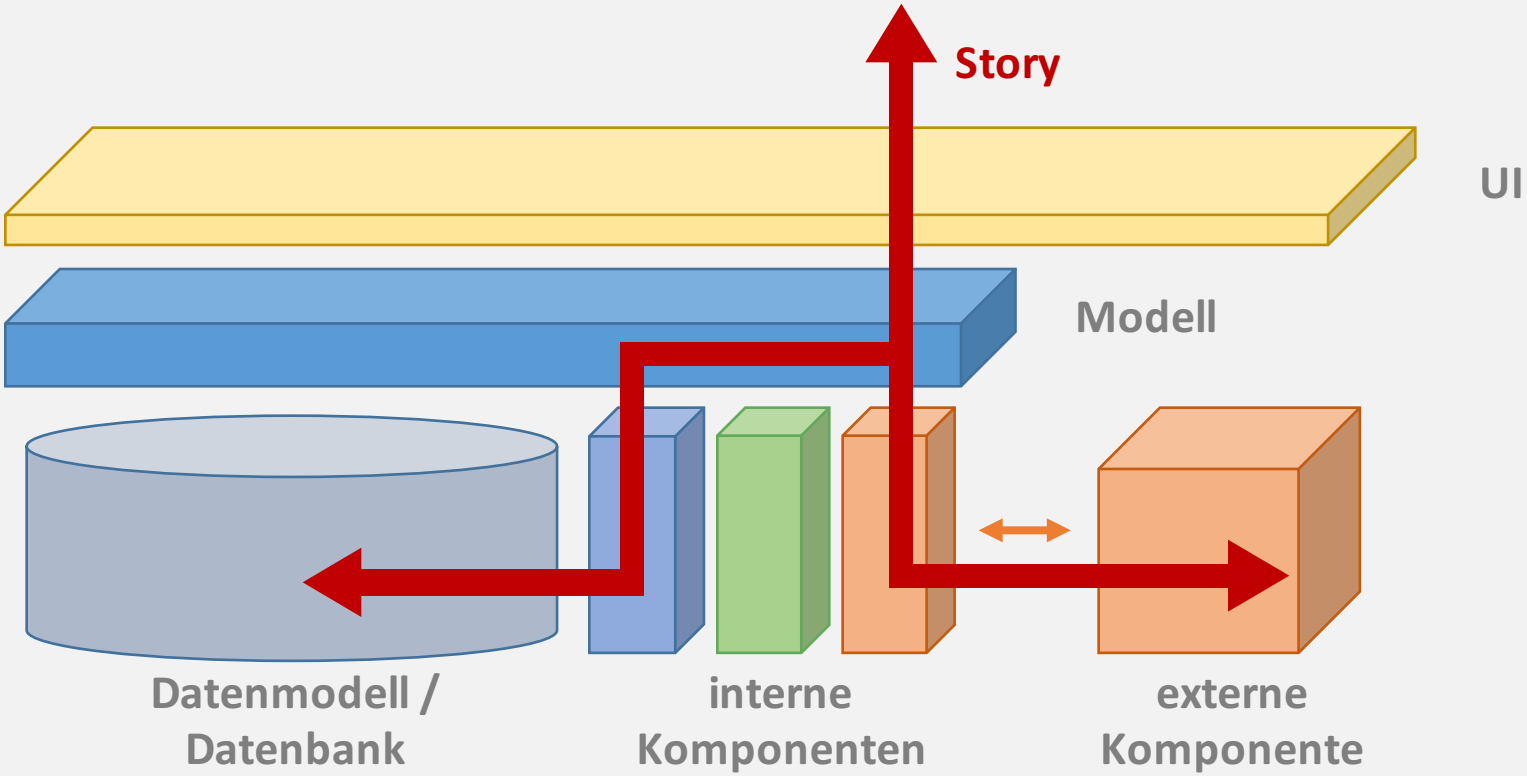
Bildquelle: Google Bildersuche / unbekannt

*) Scrum Guide 2013 http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf

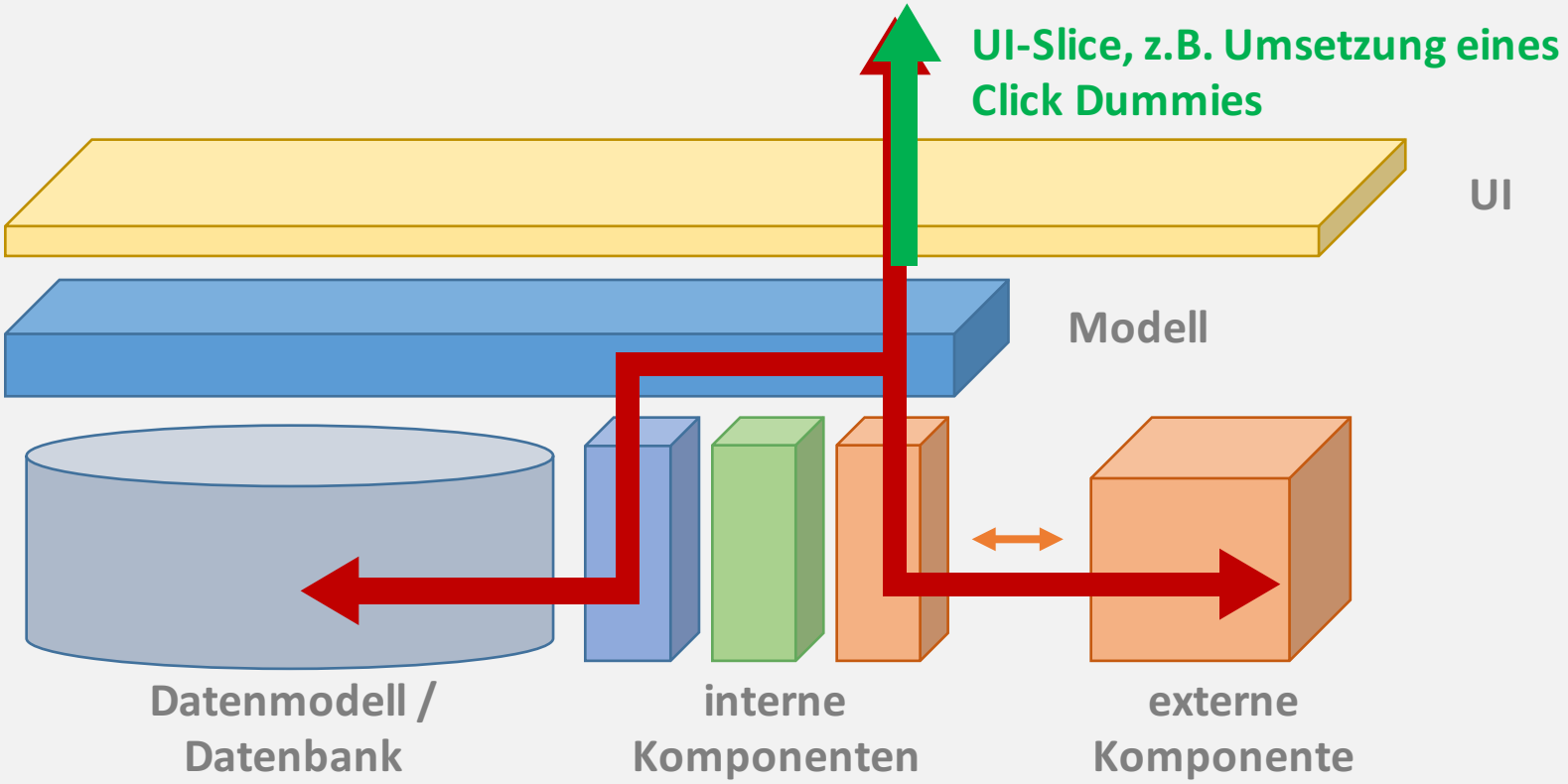
Gegeben eine typische Software-Architektur...



Eine Story erstreckt sich fast immer über mehrere Ebenen und Komponenten

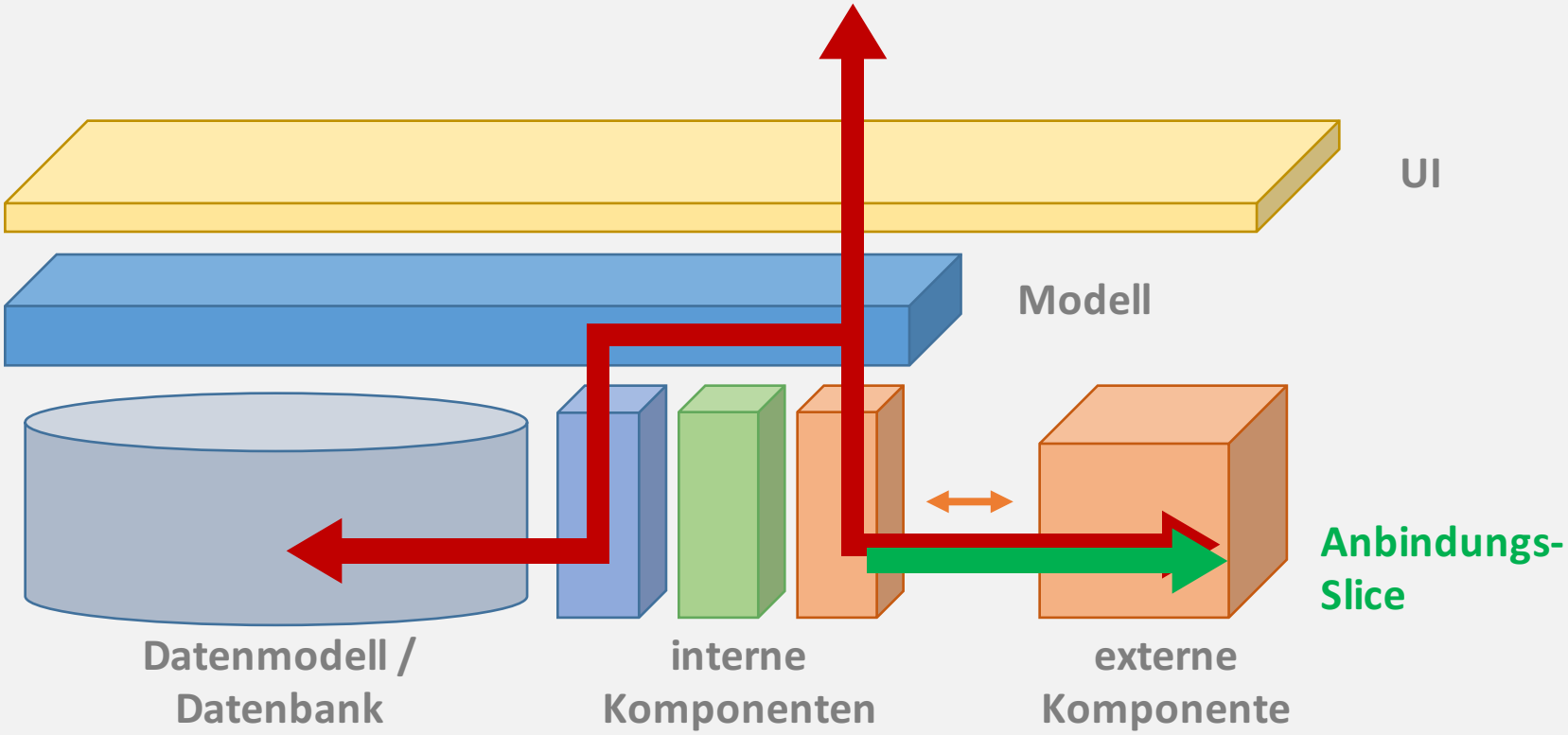


Slicing: UI vs. Rest / Ebene für Ebene?



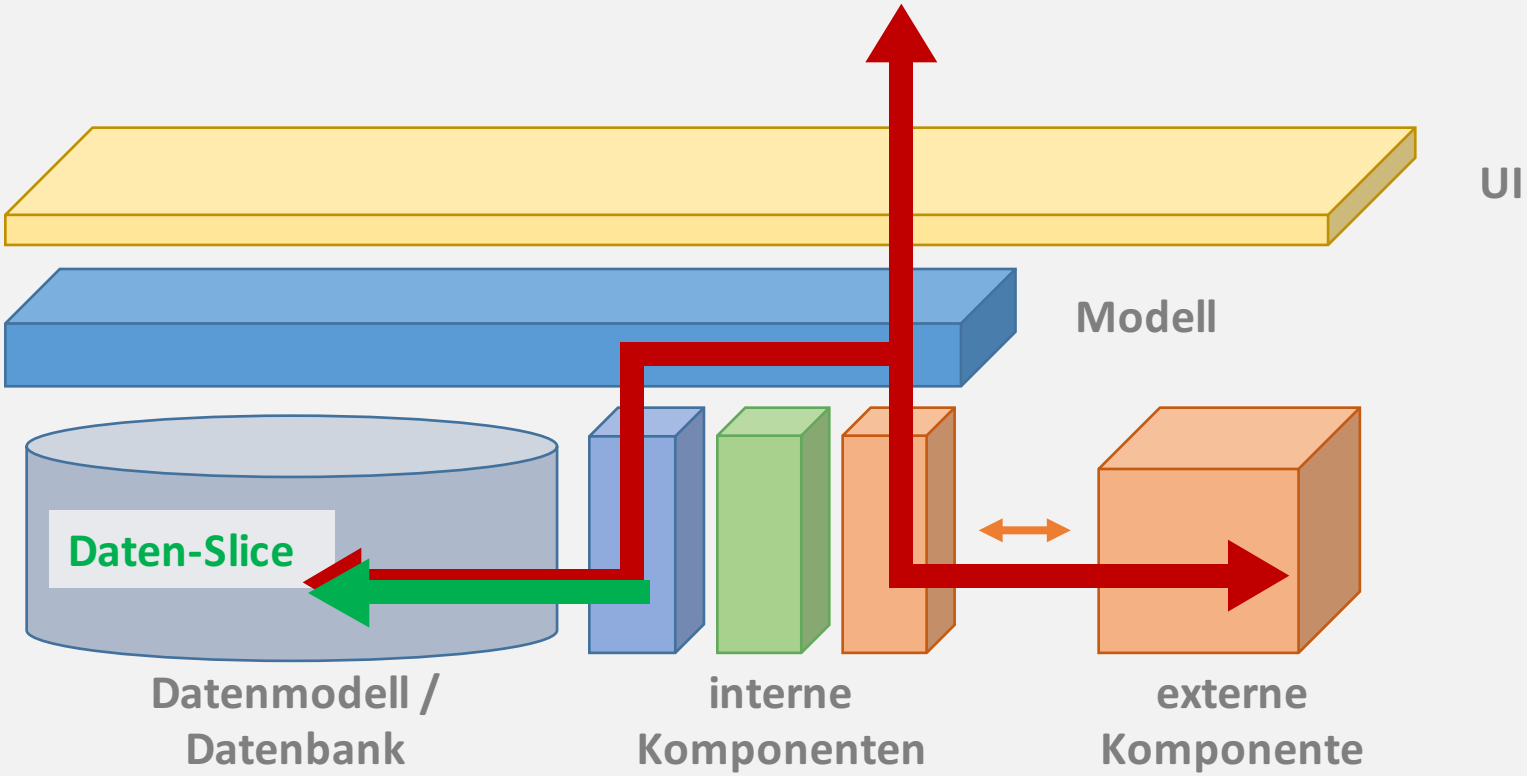
- erlaubt nicht das genaue Durchspielen eines Szenarios
- liefert keinen Wert (für User), aber erlaubt frühzeitiges Feedback (jedoch nur genau so viel wie für einen einfachen Click Dummy)

Slicing: eine Komponente zuerst?



- erlaubt nicht das Durchspielen eines Szenarios
- liefert keinen Wert (für User) und erlaubt kein Feedback
- Validierung (durch Stakeholder) schwierig – Ergebnis nicht greifbar

Slicing: Datenmodell oder -Anbindung zuerst?



- gleiche Probleme wie bei einzelnen Komponenten
- Daten müssen i.d.R. als ganzes modelliert werden, aber Implementierung + Test funktionieren besser inkrementell

HOW TO SPLIT A USER STORY

1 PREPARE THE INPUT STORY

Does the big story satisfy INVEST* (except, perhaps, small)?

NO → Combine it with another story or otherwise reformulate it to get a good, if large, starting story.

YES → Is the story size 1/10 to 1/4 of your velocity?

→ You're done.

→ Continue. You need to split it.

2 APPLY THE SPLITTING PATTERNS

start here

WORKFLOW STEPS

Does the story describe a workflow?

Can you split the story so you do the beginning and end of the work-flow first and enhance with stories from the middle of the workflow?

Can you take a thin slice through the workflow first and enhance it with more stories later?

OPERATIONS

Does the story include multiple operations? (e.g. is it about "managing" or "configuring" something?)

Can you split the operations into separate stories?

BUSINESS RULE VARIATIONS

Does the story have a variety of business rules? (e.g. is there a domain term in the story like "flexible dates" that suggests several variations?)

Can you split the story so you do a subset of the rules first and enhance with additional rules later?

DEFER PERFORMANCE

Does the story get much of its complexity from satisfying non-functional requirements like performance?

Could you split the story to just make it work first and then enhance it to satisfy the non-functional requirement?

SIMPLE/COMPLEX

Does the story have a simple core that provides most of the value and/or learning?

Could you split the story to do that simple core first and enhance it with later stories?

VARIATIONS IN DATA

Does the story do the same thing to different kinds of data?

Can you split the story to process one kind of data first and enhance with the other kinds later?

MAJOR EFFORT

When you apply the obvious split, is whichever story you do first the most difficult?

Could you group the later stories and defer the decision about which story comes first?

INTERFACE VARIATIONS

Does the story have a complex interface?

Is there a simple version you could do first?

Does the story get the same kind of data via multiple interfaces?

Can you split the story to handle data from one interface first and enhance with the others later?

last resort

BREAK OUT A SPIKE

Are you still baffled about how to split the story?

Can you find a small piece you understand well enough to start?

→ Write that story first, build it, and start again at the top of this process.

→ Can you define the 1-3 questions most holding you back?

→ Write a spike with those questions, do the minimum to answer them, and start again at the top of this process

→ Take a break and try again.

3 EVALUATE THE SPLIT

Are the new stories roughly equal in size?

NO → Try another pattern on the original story or the larger post-split stories.

YES → Is each story about 1/10 to 1/4 of your velocity?

→ Do each of the stories satisfy INVEST?

→ Try another pattern.

→ Are there stories you can deprioritize or delete?

→ Try another pattern. You probably have waste in each of your stories.

→ Is there an obvious story to start with that gets you early value, learning, risk mitigation, etc.?

→ Try another pattern to see if you can get this.

→ You're done, though you could try another pattern to see if it works better.

* INVEST - Stories should be:
Independent
Negotiable
Valuable
Estimable
Small
Testable

AGILE FOR ALL
www.agileforall.com

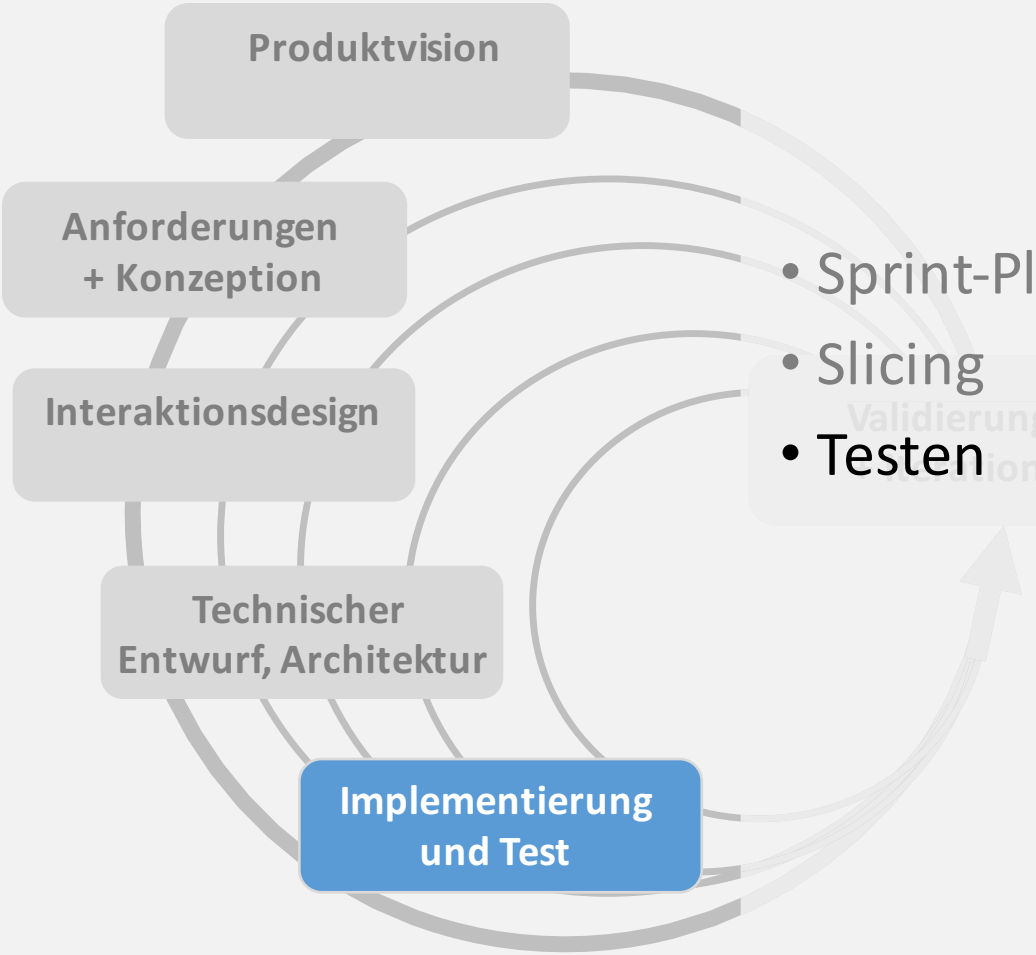
Visit http://www.richardlawrence.info/splitting-user-stories/ for more info on the story splitting patterns

Copyright © 2011-2013 Agile For All. All rights reserved.

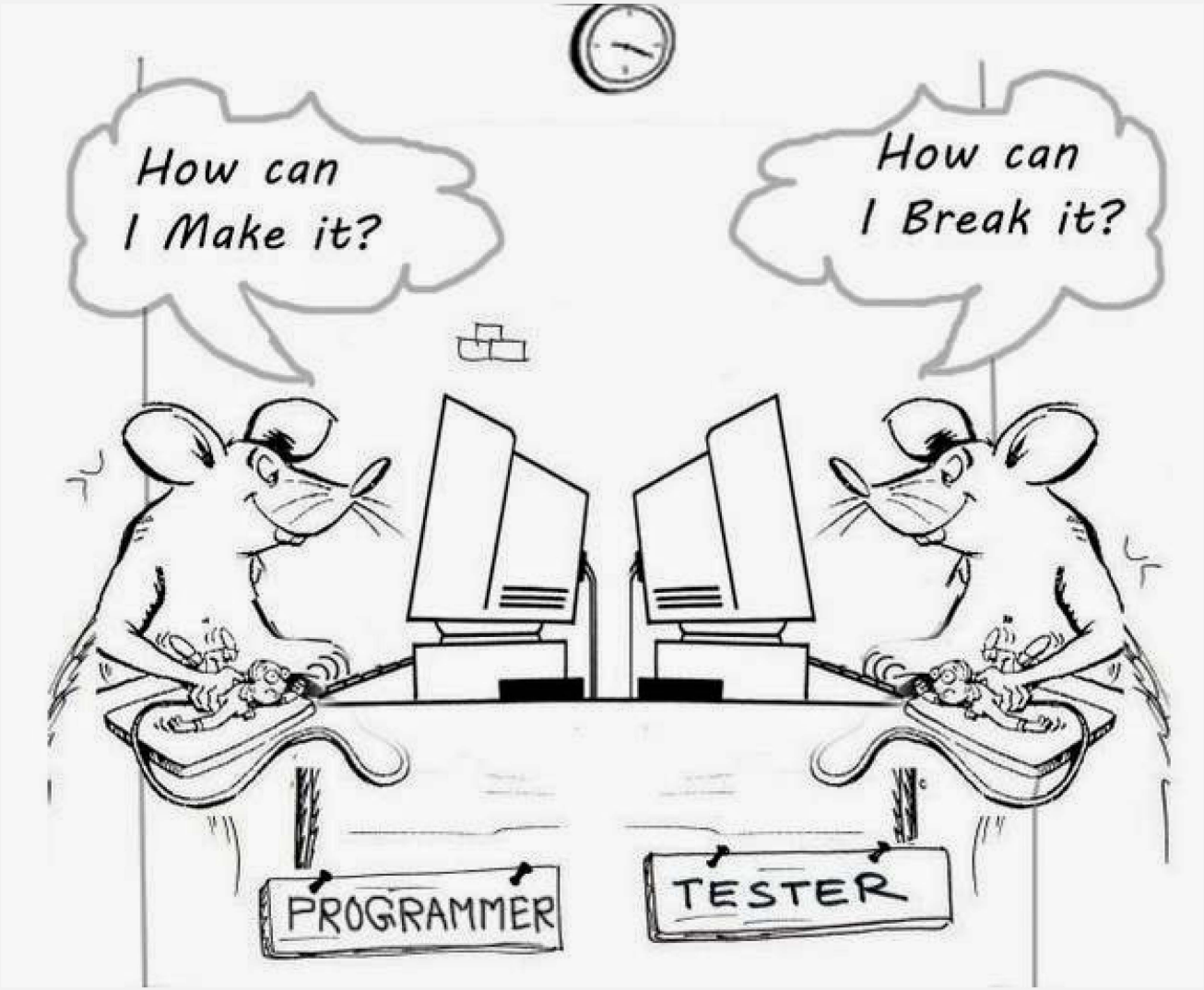
Last updated 3/26/2013

Bildquelle: Richard Lawrence 2012, http://agileforall.com/new-story-splitting-resource/

Implementierung und Test

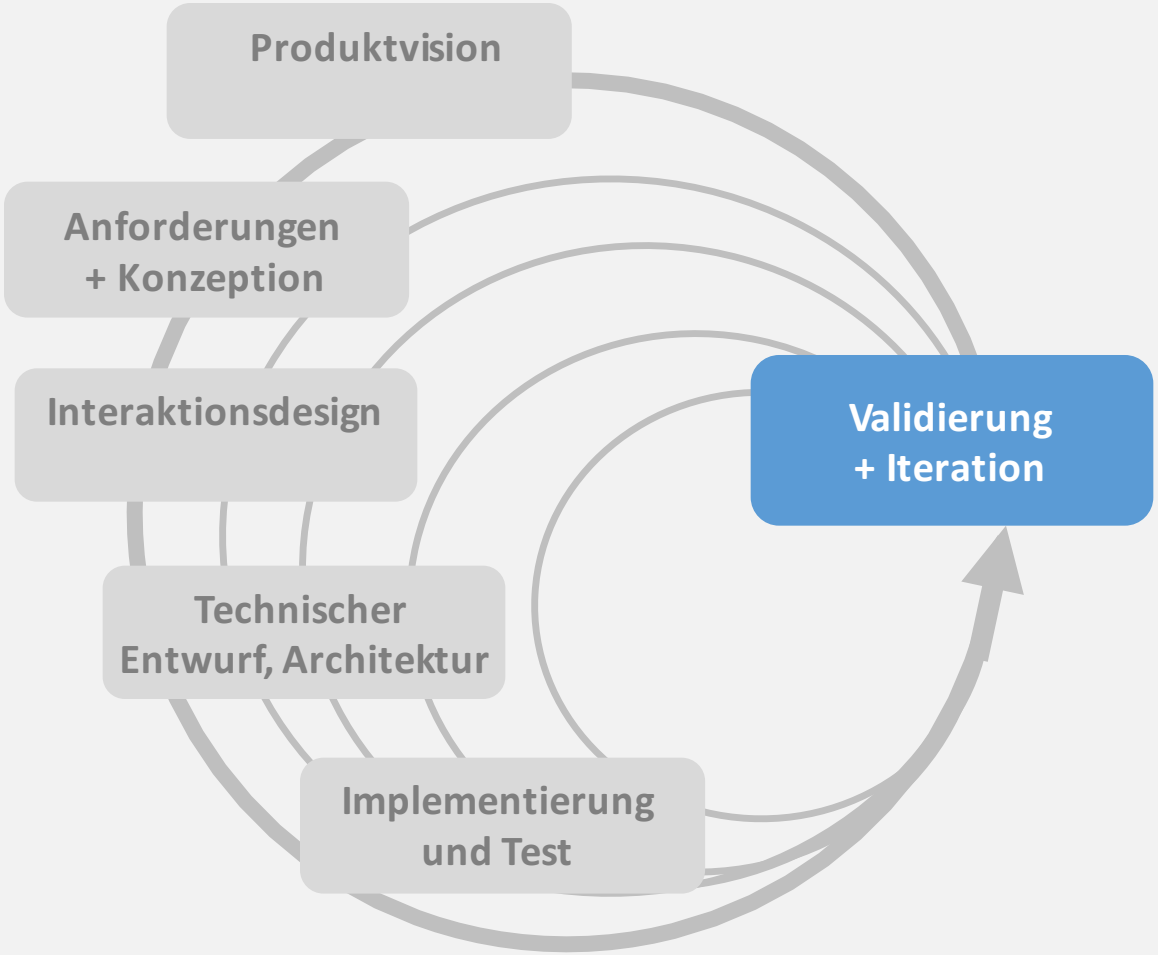


- Sprint-Planung
- Slicing
- Testen



Bildquelle: Mimul, http://www.mimul.com/pebble/default/2012/09/17.html

Validierung und Iteration



Nach jedem Sprint:

- (internes) Release
- Test durch PO
- Test durch Stakeholder
- Validierungstests gegen Stories und Ziele
- Retrospektive

- Traue Dich, alles zu hinterfragen
- Halte Vision, Stories, Spezifikation aktuell (am besten *im* Sprint)

Von der Idee zum Produkt

Wer macht was?

Wer macht was?

- Business Analyst
- Requirements Engineer
- InteraktionsdesignerIn
- Software-ArchitektIn

Je kleiner das Projekt / Unternehmen, desto unwahrscheinlicher

- **Software-EntwicklerIn**

Mindestens eineR, sonst keine Software ☺

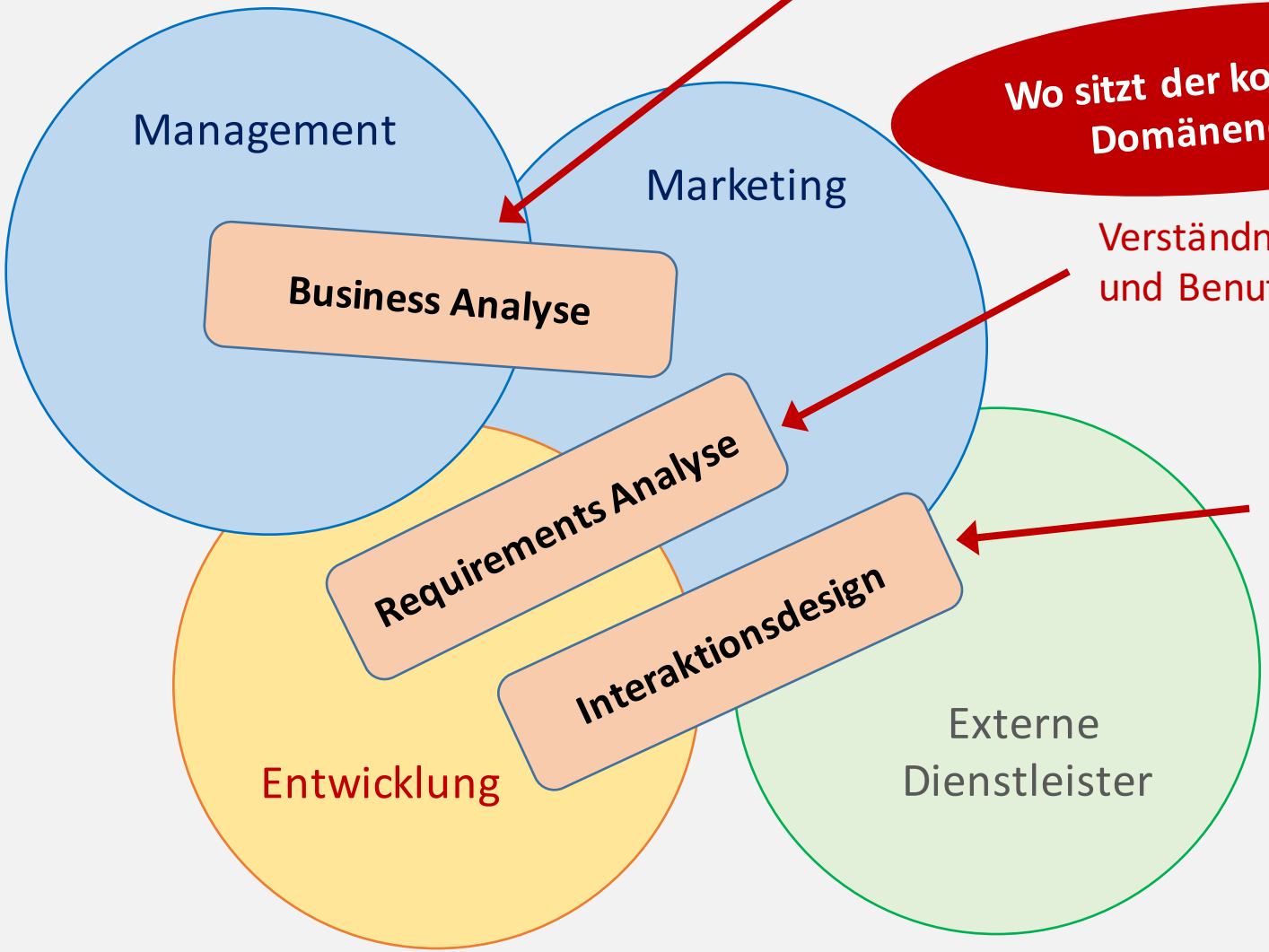
Wer macht was - KMU

Verständnis der Mechanismen
im anvisierten Marktsegment,
eigene Geschäftsinteressen

Wo sitzt der kompetenteste Domänenexperte?

Verständnis der Kunden-
und Benutzerdomäne

Fähigkeit zu Analyse
und Detailarbeit,
Software-Ergonomie,
gestalterische
Begabung



Von der Idee zum Produkt

Tool-Unterstützung

Tools für Scrum

Jira (Atlassian)

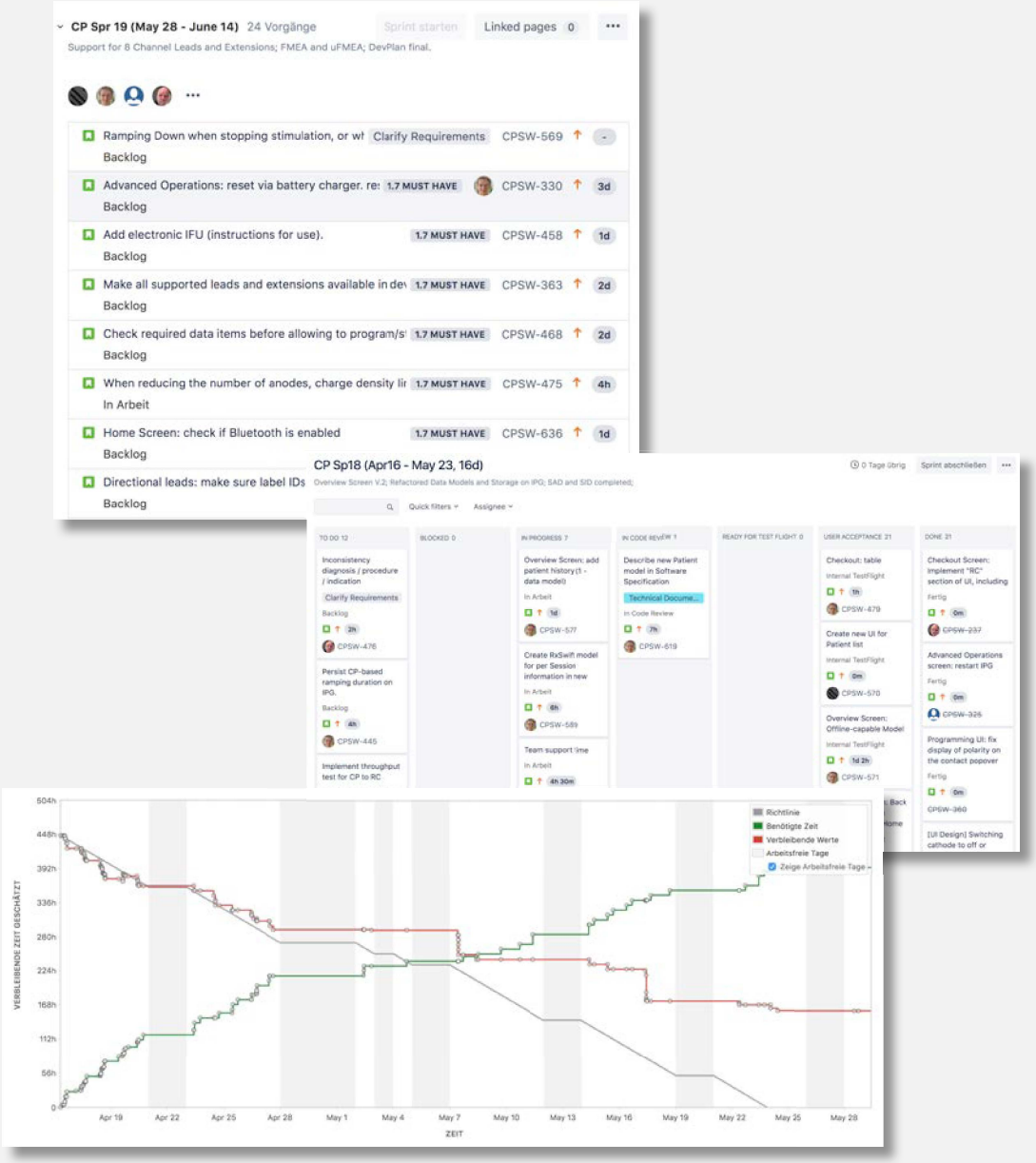
- (Sprint) Backlog
- Scrum-Board
- Burndown-Chart
- Tests (Plugins)

Was kann Jira nicht:

- Anforderungen
- Planung

Spezifikationen:

- Confluence (Wiki)



*) von Gojira (jap. für Godzilla), Nachfolger von Bugzilla, siehe https://de.wikipedia.org/wiki/Jira_(Software)

Tool-Unterstützung

Minimum:

- Tabellen für Planung
- Ticket-System für Stories / Slices / Bugs
- Wiki oder Cloud-Drive für Spezifikationsdokumente
- Grafische Tools für Interaktionsdesign + Architektur
- SW-Entwicklungs-Tools (IDE, CI-Server, ...)

Besser:

- Integrierte Tools für Ziele, Stories, Anforderungen (Listensicht, Kartensicht, Releaseplanung, ...)
- Story Mapping, ...

Tool-Unterstützung

Wichtig: Entwickler-Slice enthält direkte Links zu

- Anforderungen (Stories mit Zielen)
- Spezifikationen

→ Bei web-basierten Tools meistens einfach durch Links

Problematisch ohne integrierte Tools:

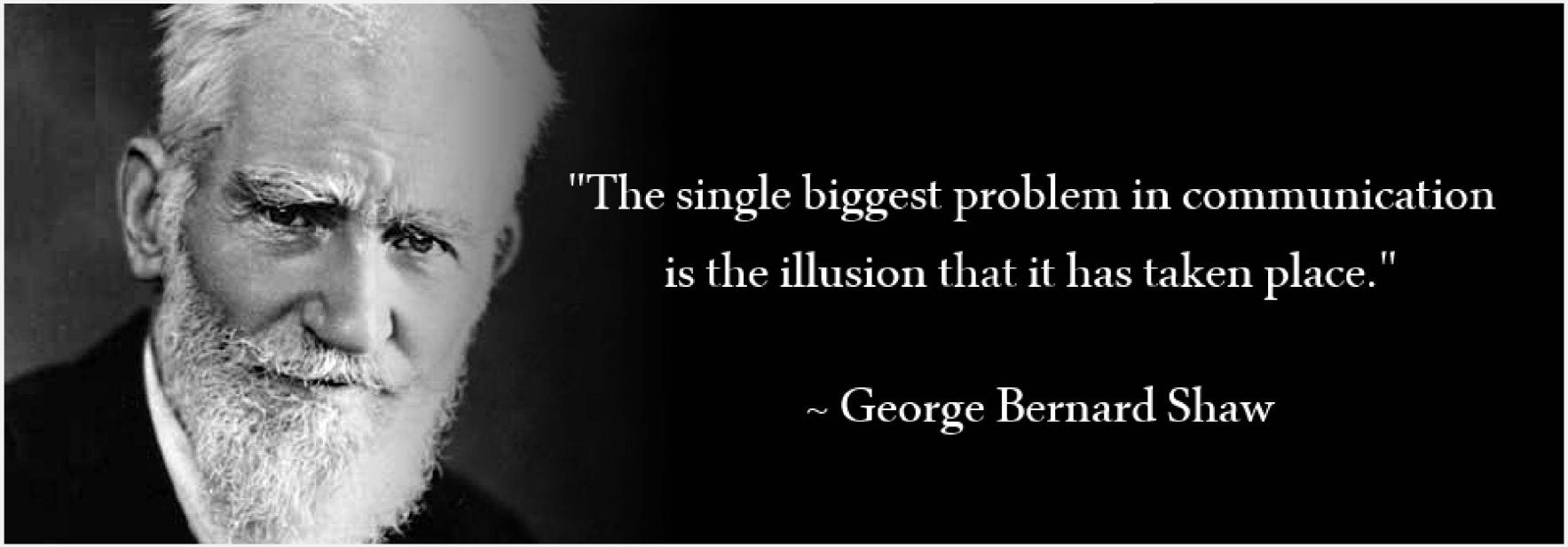
- Versionierung
- Review & Status (Entwurf vs. abgestimmt)
- Änderungsmanagement / Verfolgung von Abhängigkeiten

Fazit

Cartoon by Satu Kyröläinen @satukyro

Das gesamte Team muss verstehen:

- Zielgruppe + Stakeholder
- Mentales Modell und Ziele
- Benutzungskontext


"The single biggest problem in communication
is the illusion that it has taken place."
~ George Bernard Shaw

Rede regelmäßig mit Deinen Stakeholdern.

Product Backlog, Stories und Sprint Reviews sind Deine Freunde.

Nutze sie zur Kommunikation.