

# Vorstellung Agile Methoden

Marc Schöchlin

Amselweg 28/1  
71686 Remseck  
ms@256bit.org

Stuttgart, 10. September 2019

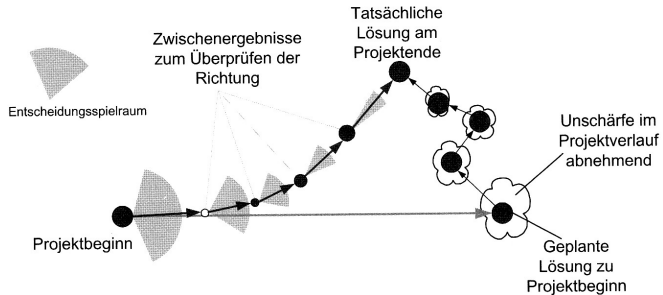
# Überblick

- 1 **Einführung**
  - Der klassische Weg
- 2 Agiles Arbeiten
- 3 Kanban
- 4 Scrum
- 5 Abschluß

# Gründe für Projektunsicherheiten

- Kunde weiss am Anfang nicht genau was er braucht
- Auftragnehmer versteht nicht genau, was der Kunde will
- Auftragnehmer unterschätzt / überschätzt Aufwände (Planung)
- Nicht alle Anforderungen und Stakeholder sind bekannt
- Kunde hat widersprüchliche Anforderungen, z.T. wg. Politik
- Politik, Management Risiken
- Änderungen in den Prioritäten, Geschäftsprozessen etc. während des Projektes
- Projekt in komplexe Projektlandschaft eingebunden
- Technische Risiken, z.B. Infrastruktur hält nicht was sie verspricht

# Projektfortschritt und Entscheidungsspielraum



# Klassische Antworten und Situationen

- Anforderungen
  - Ziele: genaues Verständnis der Anforderungen, Anforderungstabilität, (Verfolgbarkeit)
  - Am Anfang des Projekts möglichst vollständig abstimmen (Bei Risiken in Stufen vorgehen)
  - Möglichst genaue Spezifikation/Pflichtenheft erstellen
  - Änderungen nur über ein striktes Change Request Verfahren
- Architektur
  - Ziele: Architektur = stabiles Gerüst der Software
  - In früher Projektphase entwickeln (mindestens Grobarchitektur)
  - Anforderungen späterer Phasen berücksichtigen
- Dokumente
  - Ziele: Klarheit, Review-Fähigkeit bzw. Abstimmbarkeit, Stabilität und Reduktion von Kommunikation
  - Alles wird dokumentiert (Prinzip der Schriftlichkeit), wichtig sind insbesondere Spezifikationsdokumente und Architekturdokumente
  - Für Offshore-Partner, Wartungsteam, das Team

# Klassische Antworten und Situationen

- Risikomanagement
  - Projekte werden in der Regel in (wenigen) Stufen (3-12 Monate) durchgeführt
  - Aktiver (häufig unsystematischer) Umgang mit Risiken (Einzelmaßnahmen wie technischer Durchstich, Risikoliste, ...)
  - Risiken häufig nur pauschal berücksichtigt
- Pläne
  - Am Anfang des Projektes grobe Planung (auch zur Preisfindung),
  - Feinplanung bei Bedarf
  - Wenn's knapp wird: Weglassen/Verschieben unwichtiger Funktionen oder Qualitätseinbußen (z.B. Testaufwände kürzen)
  - Ziele nach Priorität : (1) Einhaltung des Termins, (1) Einhaltung des Budgets, (3) Qualität, (4) Vollständige Funktionalität
- Verträge
  - Festpreisprojekte zur Kostenkontrolle
  - Preis im Verhältnis zur Funktionalität schwer einschätzbar

# Klassische Antworten und Situationen

- Kunde

- integriert in Phasen abhängig von seiner eigenen Verfügbarkeit
- ist über Anforderungsdefinition, Dokument-Reviews und System- und Abnahmetest einbezogen
- Entwicklung: selten gemischte Teams bedingt durch Projektphasen
- Kommunikation zum Teil reglementiert durch Projektleiter („Ein-Ansprechpartner“-Modell) oder über Dokumente
- Ziele: Langfristige Kundenbindung, Vertrauen, verbindliche Absprachen

- Prozesse

- Aktivitäten / Workflows im Detail festgelegt
- Viele Rollen
- Viele, genau festgelegte Artefakte (Doku, Code und Co.)
- Ziele: Planbarkeit, Wiederholbarkeit, Verbesserbarkeit (CMM)

# Auswirkungen

- Standard Projekteskalation: Grün ... Gelb ... Dunkelrot ... Gelb
- Projektplan und Realität driftet stark auseinander
- Change Requests machen große Schmerzen (Projektverzug, Kosten)
- Große Teile der Fachlichkeit ergeben sich während des Projekts
- Kunde bekommt eigentlich nicht, was er will, sondern vielleicht noch nicht mal das was im Vertrag steht
- Definierte Prozesse sind oft schwer durchzuhalten
- Projektziel muß mit massivem Aufwand erkaufte werden
- Projektziel muß mit massiver Scopereduktion erkaufte werden
- Chef-Architekten-Denkmäler
- weniger Spaß für Projektmitarbeiter



# Überblick

## 1 Einführung

## 2 **Agiles Arbeiten**

- Allgemein
- Kernaspekte

## 3 Kanban

## 4 Scrum

## 5 Abschluß

# Manifesto for Agile Software Development

Was ist uns wichtiger?

***Individuals and interactions***

over processes and tools

***Working software***

over comprehensive documentation

***Customer collaboration***

over contract negotiation

***Responding to change***

over following a plan

► 2001 - <http://agilemanifesto.org>

# Die agile Vorgehensweise

- Das Team übernimmt Verantwortung
- Ausrichten der Arbeitsweise auf die kontinuierliche Änderung von Anforderungen
- Probleme iterativ lösen, „Minimal viable Product“
- Bewusst sich um Kommunikation bemühen
- Ein kontinuierlicher Ergebnisfluß steht im Fokus
- Produktivität und nicht Effizienz steht im Fokus
- Konzentration auf möglichst nur eine Aufgabe
- Stetiger Wissenstransfer, Wissens-Silos vermeiden
- Arbeiten in Zyklen
- Transparenz mit allen Beteiligten
- Konsequente Umsetzung
- Aktives Arbeiten an der kontinuierlichen Verbesserung
- Fehlerkultur, Respektvoller und lösungsorientierter Umgang mit Fehlern
- Ermitteln von KPIs zur Erfolgsmessung

# Auswirkungen auf die Teamkultur

- Gemeinsame Verantwortung - kooperativer Führungsstil
- Das Team organisiert sich weitgehend selbst
- Führung findet über das Vermitteln von Visionen und Strategie statt
- Teamleiter unterstützt Team die Aufgaben möglichst eigenverantwortlich zu bearbeiten
- Entscheidungen werden hauptsächlich im Team gefällt
- Feedbacks bzw. Verbesserungsprozesse finden regelmäßig und häufig statt (Retro und OneToOne)
- Kontinuierliches Lernen steht im Vordergrund
- Peering, gemeinsames Bearbeiten von Problemen
- Daily Standup Meeting: Was war, was wird, wo gibt es Schwierigkeiten?
- Persönlicher Kontakt mit allen Beteiligten wird aktiv angestrebt
- Projektteams bilden sich abteilungsübergreifend

# Überblick

- 1 Einführung
- 2 Agiles Arbeiten
- 3 Kanban**
  - Einführung
- 4 Scrum
- 5 Abschluß

# Grundsätzliches

## Definition

Kanban is an approach to change management.

It isn't a software development or project management lifecycle or process.

David Anderson / Vater des Software-Kanban (2007)

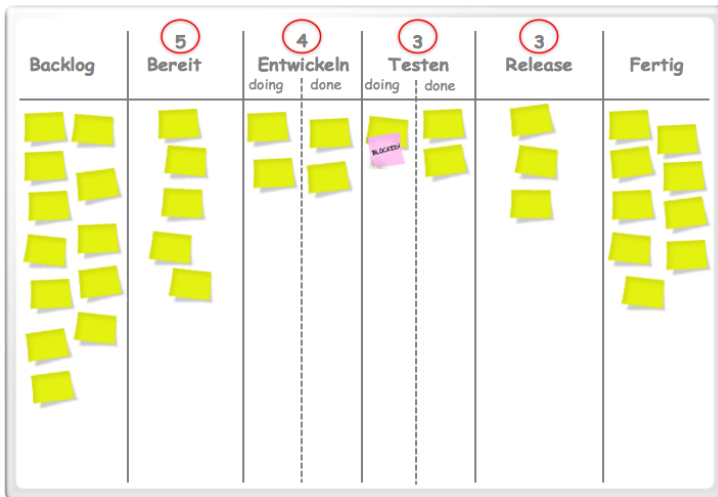
## Kanban in a nutshell

- Small, cross-functional teams
- Product split into small, roughly, estimated stories
- Iterations
- Continuous improvement

# Kanban in der Anwendung

- Gut geeignet für ungeplante Tätigkeiten: Bugs, Optimierungen, Betriebsprobleme, reaktive Themen, Servicemanagement, ...
- Gut geeignet für gut überschaubare Entwicklungsprozesse
- Begrenze die Menge begonnenner Arbeit
- Verkürze die Zeit zwischen Beginn und Erledigung einer Aufgabe
- Konzentration und Fokussierung auf die aktuelle Aufgabe
- Vermeiden von Taskwechsel
- Visualisiere den Fluss der Arbeit
- Mache die Regeln für den Prozess explizit (Publizieren wann wie gehandelt wird)
- Miss und steuere den Fluss

# Kanban Board





# Kanban Board

DevOps Kanban

Schnell-Filter: Nur meine Vorgänge Zuletzt aktualisiert

5 Aufgaben Max. 10 1 OnHold Max. 2 1 Wird Ausgeführt Max. 3 6 Abnahme PO Max. 5 1 Fertig Veröffentlicht...

~ Beschleunigen 2 Vorgänge

~ Alles andere 12 Vorgänge

SUP-349  
↑ Stage Hybris-Server @ eMail-Adressen

SUP-354  
↑ Neuer MA L...

SUP-355  
↑ Typsystem Update am 12.3.2015

SUP-357  
↑ fashion.man nicht eingespielt?

SUP-359  
↑ Checkbilder fehlen

SUP-312  
↑ Oracle Testsystem bei P...

SUP-322  
↑ Zabbix für ... erreichbar machen

SUP-328  
↑ Zugriff für Cukes auf Test 1

SUP-318  
↑ Umlaute im Log auf test und stage

SUP-325  
↑ Datenbankpools Hybris anpassen

SUP-328  
↑ Anpassung hat IP-Adressen-Bereich erweitert. Anpassungen notwendig?

SUP-347  
↑ Existiert unter ... ein Ordner /services?

SUP-358  
Release Hotfix 3.16.1

SUP-360  
Hotfix branch 3.16.1 erstellen

# Überblick

- 1 Einführung
- 2 Agiles Arbeiten
- 3 Kanban
- 4 Scrum**
  - Grundsätzliches
  - Im Detail
  - Auswirkungen
- 5 Abschluß

# Begriff



Scrum (engl. das Gedränge) ist ein Vorgehensmodell mit Meetings, Artefakten, Rollen, Werten und Grundüberzeugungen, das beim Entwickeln von Produkten im Rahmen agiler Softwareentwicklung hilfreich ist.

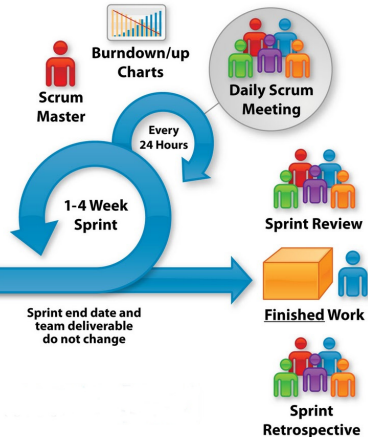
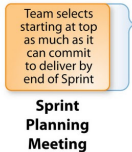
# Wikipedia

- OOPSLA 1995, erster Konferenzbeitrag
- Begriff durch Jeff Sutherland, Ken Schwaber geprägt
- Umsetzung von Lean Development für das Projektmanagement
- Kernprinzipien
  - Transparenz: Fortschritt und Hindernisse eines Projektes werden täglich und für sichtbar festgehalten
  - Überprüfung: Regelmäßige Lieferung
  - Anpassung: Anforderungen und Implementierungen entwickeln sich. Neubewertung und Anpassung nach jeder Lieferung

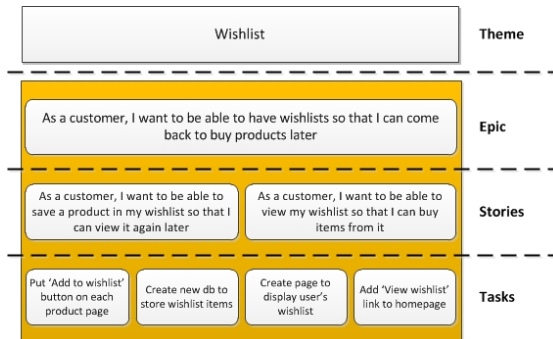
# Der Scrum Prozess

## The Agile: Scrum Framework at a glance

Inputs from Executives,  
Team, Stakeholders,  
Customers, Users



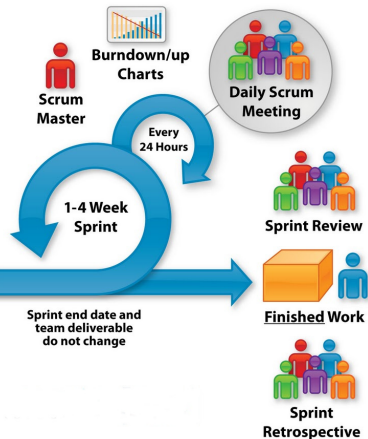
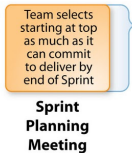
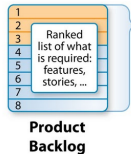
# Die Planungseinheiten



# Der Product Owner

## The Agile: Scrum Framework at a glance

Inputs from Executives,  
Team, Stakeholders,  
Customers, Users



# Der Product Owner

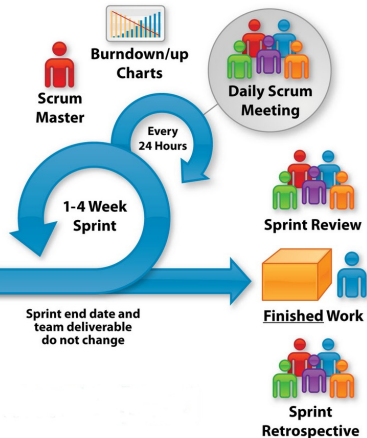
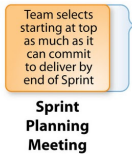
- Verwaltet das Produktbacklog
- Bringt Stories in einen bearbeitbaren Status (Erfüllung der „Definition of Ready“)
- Definiert Produkt-Features
- Priorisiert Features abhängig vom Marktwert
- Akzeptiert oder weist Arbeitsergebnisse zurück
- Bestimmt Auslieferungsdatum und Inhalt
- Ist verantwortlich für das finanzielle Ergebnis des Projekts (ROI)
- Passt Features und Prioritäten nach Bedarf für jeden Sprint an



# Das Team

## The Agile: Scrum Framework at a glance

Inputs from Executives,  
Team, Stakeholders,  
Customers, Users



# Das Team

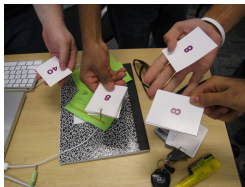
- Typischerweise 5-9 Personen
- Funktionsübergreifend / Interdisziplinär  
(QS, Programmierer, UI-Designer, System-Engineer, etc.)
- keine Titel (aber manchmal nicht vermeidbar)
- Mitglieder sollten Vollzeitmitglieder sein - wenige Ausnahmen  
(z.B. sehr selten benötigte Spezialisten)
- Mitgliedschaft kann sich nur zwischen Sprints verändern
- Teams organisieren sich selbst
- Team erfüllt die „Definition of Done“

# Scrum Planning Poker



Aufgaben werden durch das Team in einem abstraktem Storypoint-Komplexitätsmaß unabhängig von der Qualifizierung der Teammitglieder geschätzt.

# Scrum Planning Poker

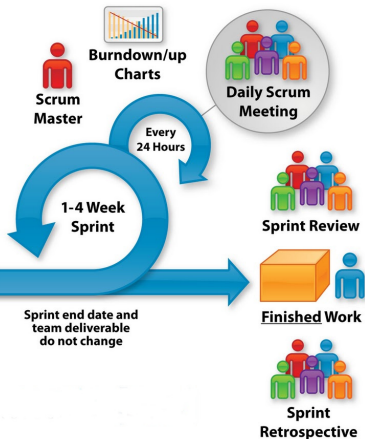
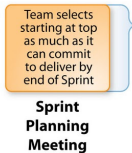


- Gemeinsame Einschätzung durch Pokern und Austausch von Argumenten
- Summe der Komplexitäten = Umfang des Sprints
- Betrachtung der Leistung des letzten Sprints und Commitment des Teams für bevorstehenden Sprint
- PO kann mit frühzeitigem Schätzen und der Betrachtung der Storypoints z.B. 1-3 Sprints vorausplanen

# Das Daily Standup

## The Agile: Scrum Framework at a glance

Inputs from Executives,  
Team, Stakeholders,  
Customers, Users



# Daily Standup



- Ziel: Die Zusammenarbeit koordinieren und den Arbeitsfluss optimieren (Reviews, Peering, Hindernisse beseitigen, ...)
- Unbedingt vermeiden: Technische Diskussionen, reines Statusreporting, Unpünktlichkeit bzw. Nichtanwesenheit
- Was gehört ins Daily?
  - Gestern habe ich ...
  - Heute will ich ...
  - Mich behindert ... ich komme nicht weiter an ... weil ...
  - Ich könnte Unterstützung bei ... gebrauchen

# Scrum Board I

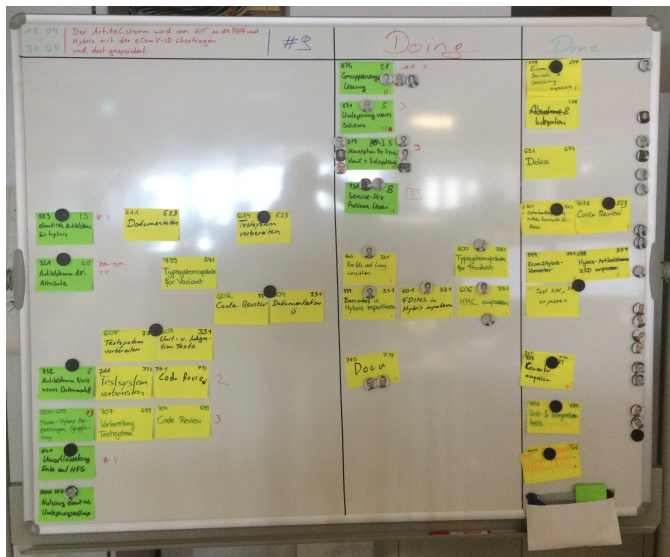
Domäne I

SPRINT: 10.02. - 02.03. | SCHNELL-FILTER: Nur meine Vorgänge | Zuletzt aktualisiert | Sticky Items | User Story | Bug | Service & Support

Plan | Arbeit | Bericht | Board | View in Tempo

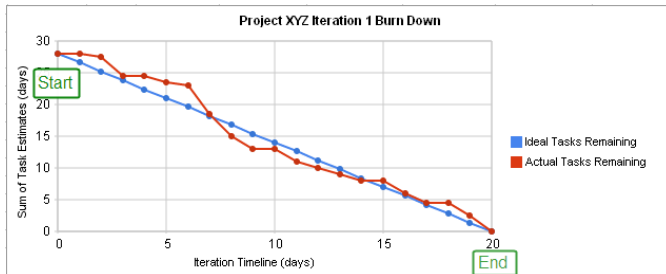
Aufgaben	Wird Ausgeführt	Abnahme PO	Fertig
<div>DOMI-344 3 Unteraufgaben - Bestandsdaten Anpassung Berechnung Shop-Bestand</div>			<div>DOMI-449 ↑ UnitTest</div>
<div>DOMI-332 6 Unteraufgaben - Bestandsdaten Adaption Genehmigungsstatus</div>	<div>DOMI-451 ↓ Testfälle mit PM erstellen</div> <div>DOMI-452 ↑ UnitTest</div>		<div>DOMI-450 ↑ Wiederverwendung ProductStatusService</div> <div>DOMI-453 ↑ Integrationstest anpassen</div> <div>DOMI-456 ↑ Analyse bestehender Import</div> <div>DOMI-459 ↑ Doku erstellen</div>
<div>DOMI-368 5 Unteraufgaben - Analyse Auswirkungen Auflösung auf Datenmodell in hybris</div>			<div>Dieses Board wurde aktualisiert: Aktualisieren - Ignorieren</div>

## Scrum Board II





# Das Burndown Chart



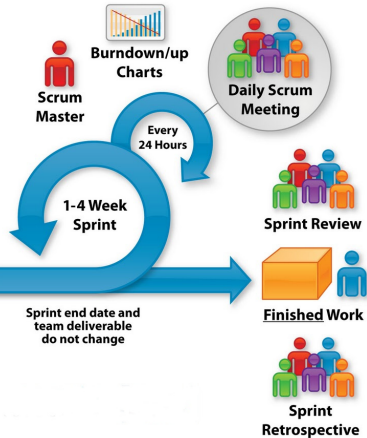
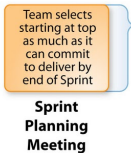
# Der Scrum Master

- Repräsentiert das Management gegenüber dem Projekt
- Verantwortlich für die Einhaltung von Scrum-Werten und -Techniken
- Beseitigt Hindernisse
- Stellt sicher, dass das Team vollständig funktional und produktiv ist
- Unterstützt die enge Zusammenarbeit zwischen allen Rollen und Funktionen
- Schützt das Team vor äußeren Störungen

# Der Scrum Zyklus

## The Agile: Scrum Framework at a glance

Inputs from Executives,  
Team, Stakeholders,  
Customers, Users



# Das Sprint Review Meeting

- Das Team präsentiert, was es während eines Sprints erreicht hat
- Der PO gibt einen Ausblick auf den nächsten Sprint
- Typischerweise in Form einer Demo der neuen Features oder der zugrunde liegenden Architektur
- Zwei Stunden zur Vorbereitung
- Keine Folien
- Das ganze Team nimmt teil
- Jeder ist eingeladen

# Die Sprint Retrospektive

- Nach jedem Sprint
- bei 3 Wochen Zyklus, ca. 2 Stunden
- Das ganze Team nimmt teil: Scrummaster, Produktowner, Team
- Regelmäßiger Rückblick auf Erfolge und Verbesserungsmöglichkeiten
- Nutzung methodischer Ansätze
  - Retormat
  - Tastycupcakes
- Identifizieren und Auswählen von Sprint-Team-Zielen
- Nachhalten Sprint-Team-Zielen

# Was bringt Scrum dem Fachbereich?

- Verlässlichkeit - alle 3 Wochen ein Release
- keine Change Request Diskussionen mehr
- Bessere Ergebnisse durch Orientierung am Kunden und am Benutzer
- Strukturierung in Epics und Stories vereinfacht die Priorisierung
- Transparenz und kurzfristigere Steuerung
- Priorisierung des Backlogs nach Return On Investment und somit Businesswert
- Kostenkontrolle: „Danke. Ich habe ausreichend Funktionalitäten erhalten.“
- Productowner als Anlaufstelle für Fachlichkeit, die Technik und den Zeitplan
- Produktentscheidungen zum Zeitpunkt der Relevanz
- keine Produktentscheidungen ohne Informationen

# Was bringt Scrum dem Entwicklungsteam?

- Das Team organisiert sich und seine Arbeit selbst
- Enge fachliche Zusammenarbeit mit dem Product Owner
- Konzentration und Fokussierung
- Besser funktionierende Software durch iterative Vorgehensweise
- Code wird wartbarer, Bewußtsein dass alles im Fluß ist
- Bessere Softwarearchitektur - passende Lösungen
- Anforderungen entwickeln sich sinnvoll und können mitgestaltet werden
- Kontinuierliche Verbesserung: keine Ownership, Wissenstransfer, Retros, ...
- Einfachheit - die Kunst, die Menge nicht getaner Arbeit zu maximieren
- Definition of Ready: Klare Kriterien für die Implementierungsfähigkeit einer Story
- Definition of Done: Klare Kriterien für den Abschluß einer Aufgabe

# Überblick

- 1 Einführung
- 2 Agiles Arbeiten
- 3 Kanban
- 4 Scrum
- 5 Abschluß**
  - Letztes Slide
  - Agiles Werkzeug
  - Referenzen



# Fragen ?

## Werkzeug für das agile Arbeiten:

- Ein Moderationskoffer
- Ein Wiki System  
(Teamorga, Dokumentation, Prozesse, ...)
  - [XWIKI](#)
  - [Atlassian Confluence](#)
- Ein Board  
(Arbeitsorganisation, Transparenz, Visualisierung, ...)
  - Ein physisches Board
  - [Trello](#)
- Ein skalierendes Chatwerkzeug  
(Themenbezogene Kommunikation statt Directchat)
  - [Rocketchat](#)
  - [Mattermost](#)
- Ein Team das Verantwortung übernehmen will
- Methodische Unterstützung bei der Etablierung
- Geduld und Konsequenz
- Die Akzeptanz der Stakeholder, des Unternehmens

## Literatur / Referenzen

- [Scrum – auf dem Bierdeckel erklärt](#)
- [The Toyota Way: Fourteen Management Principles from the World's Greatest Manufacturer](#)
- [Who is the Professional Scrum Master](#)
- [Scrumguide / Sutherland, Schwaber - praktisch „die Erfinder“ von Scrum](#)
- DevOps for Developers / Michael Hüttermann (Apress Verlag)
- The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win / Kim, Behr, Spafford (IT Revolution Press)
- Peopleware: Productive Projects and Team / Tom DeMarco
- [Der Sourcecode dieser Präsentation / Download](#)