# Requirements Engineering im agilen Umfeld

12 November 2015 | **Software Engineering** | **Martin Habicht**
**Lesezeit:** 4 Minutes

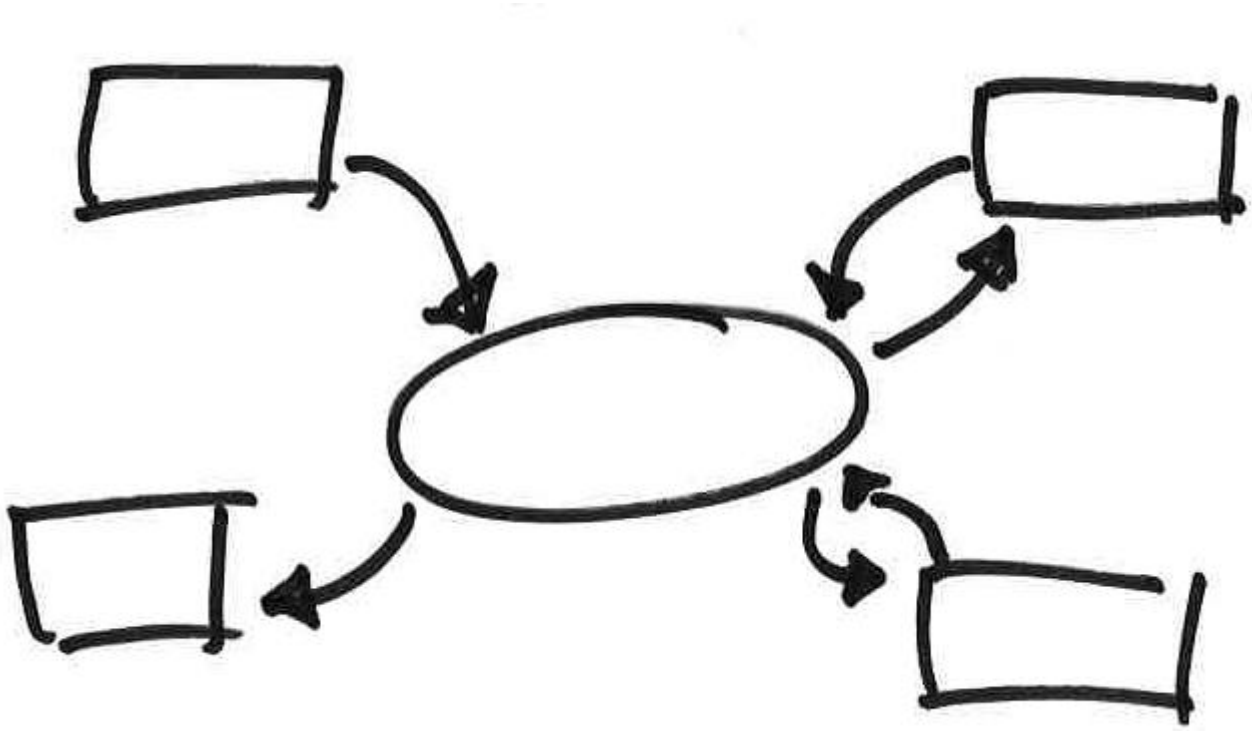
**In meinen Kursen (IREB CPRE) zeige ich den Kurs-Teilnehmern die wichtigsten Tätigkeiten und Resultate im Requirements Engineering (RE). Hier werde ich immer wieder gefragt, wie sich die RE-Tätigkeiten unterscheiden, je nachdem ob das Projekt „klassisch“ oder „agil“ geführt wird.**

Meine Antwort:

1. Die RE-Tätigkeiten an sich sind genau die gleichen!
2. Die Reihenfolge ist komplett anders und das beeinflusst das „Schneiden“.
   Was ich damit meine, illustriere ich gerne anhand von drei konkreten RE Resultaten (im Fachjargon: Artefakte). Ich habe bewusst diese drei gewählt, da sie typische RE-Resultate sind und zentrale Aspekte des zu erstellenden Systems beschreiben. Ich spreche hier lieber von RE-Resultat als von RE-Artefakt, da nicht das Dokument an sich im Zentrum steht, sondern nur als Mittel zum Zweck dient: Nämlich ein tolles Software-System zu bauen. Unter „Schneiden“ verstehe ich das Unterteilen des Systems in kleinere Einheiten, im Englischen oft auch „decomposition“ genannt.

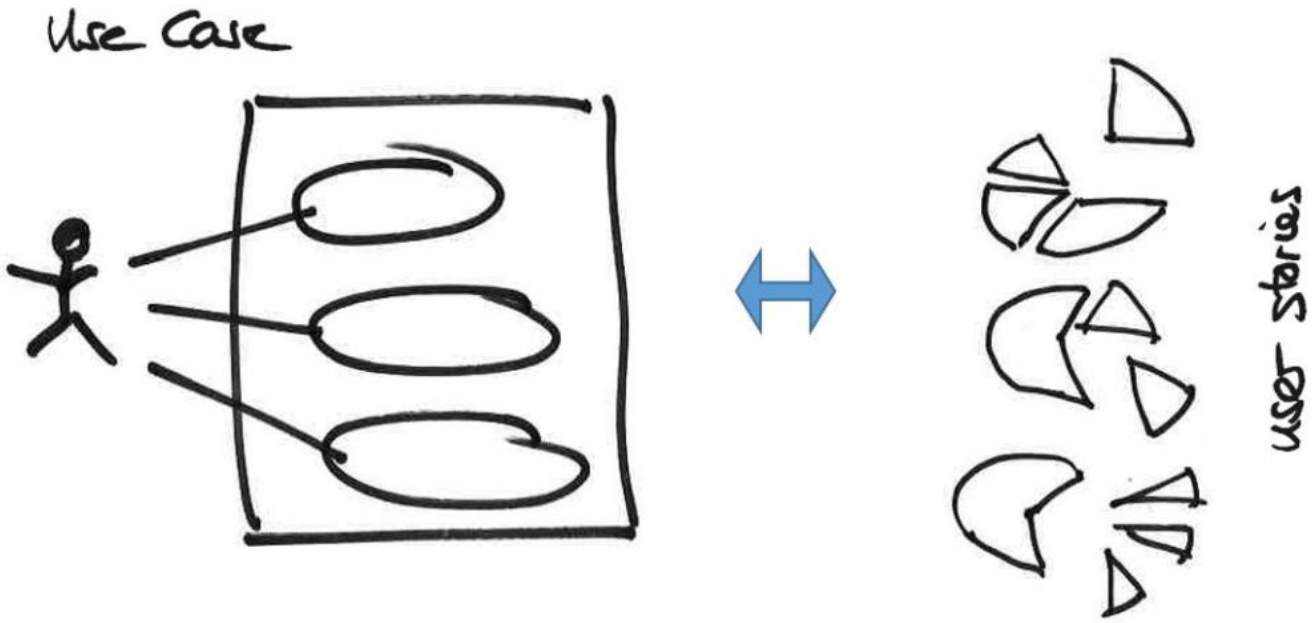
### Kontextdiagramm

Das Kontextdiagramm zeigt das Big Picture mit dem System als Blackbox und den Datenflüssen von und nach aussen.



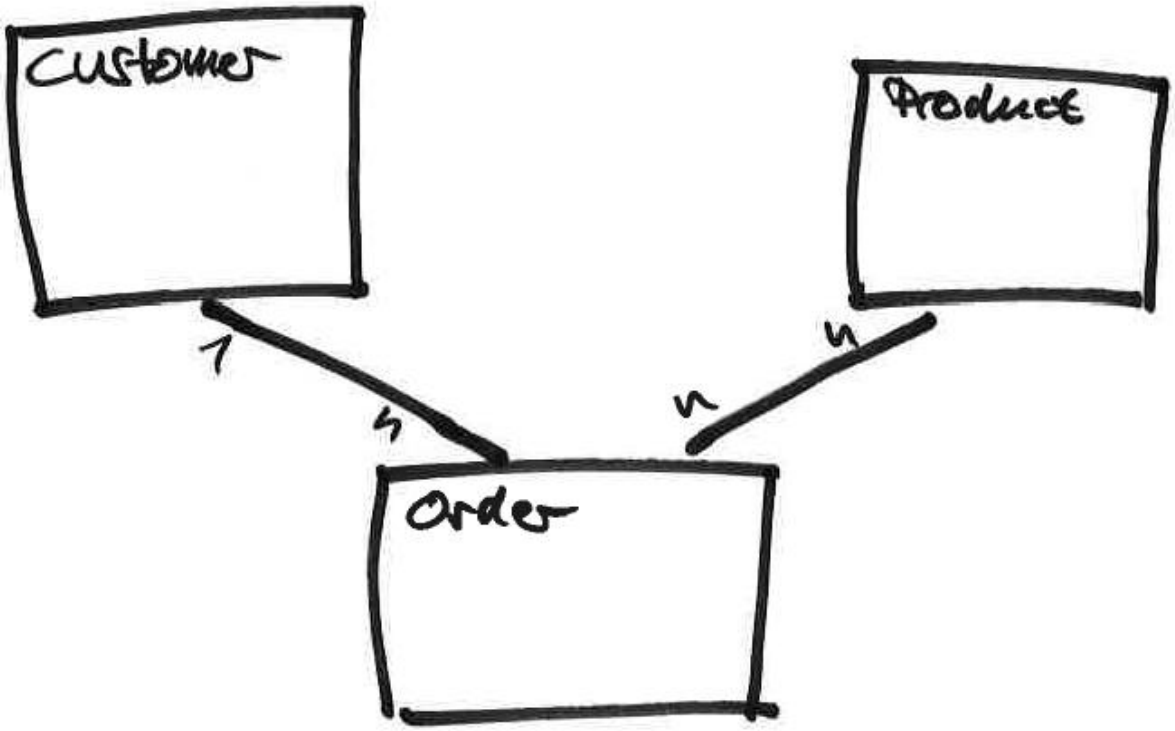
**Use Cases bzw. User Stories**

Diese zeigen die Funktionen des Systems als Menge von Abläufen, verschiedenen Alternativszenarien und sind strukturiert nach Usern (im Fachjargon: Aktoren oder Personas). Im agilen Umfeld erstellt man gewöhnlich „User Stories“, welche keinen kompletten Use Case beschreiben, sondern nur einen kleinen Ausschnitt davon. Ivar Jacobson bezeichnet dies als „Use Case Slice“, heute ist aber der Begriff „User Story“ üblich. „Use Case Slices“ bzw. „User Stories“ sind viel kleiner und feingranularer als ganze „Use Cases“.



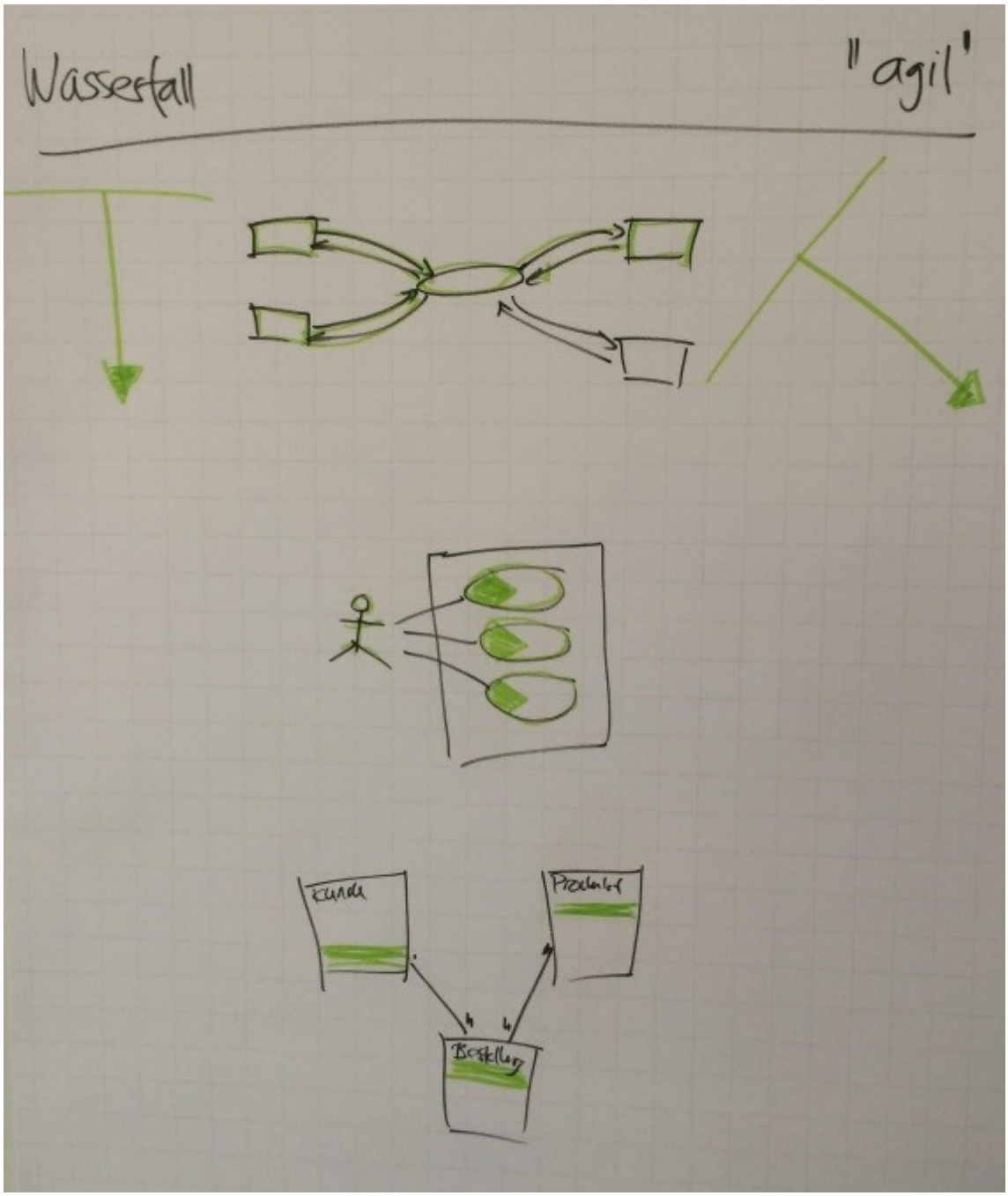

**Business Objekt Struktur**

Diese zeigt die statische Struktur der „Daten“ als Business Objekte mit deren Feldern und Beziehungen.




**Das klassische Vorgehen**

Im klassischen Vorgehen (Wasserfallmodell, bei dem alle Requirements im Voraus erhoben werden) beginnt das RE gewöhnlich „von oben nach unten“, wie das untenstehende Beispiel verdeutlicht. Hier wird das ganze Kontextdiagramm erstellt, dann alle Use Cases und schliesslich die ganze Business Objekt Struktur. Natürlich geschieht auch das in Zyklen, d.h. jedes neue Diagramm bringt neue Erkenntnisse, eine neue Schärfung von Details, so dass auch alle vorhergehenden Diagramme überarbeitet werden müssen, um konsistent zu sein. Aber die Denkweise ist *im Wesentlichen „von oben nach unten“*.




### Das agile Vorgehen

Beim agilen Vorgehen werden die gleichen RE-Resultate benötigt, aber *in der Reihenfolge „schräg von der Seite her“* erstellt. Anstelle des kompletten Kontextdiagramms reicht meist schon ein Teil für den ersten Release, nämlich jener, mit der höchsten Priorität. Anschliessend kann bereits der dazugehörige Ausschnitt aus den Use Cases erstellt werden, zusammen mit den notwendigen Attributen der Business Objekt Struktur. Die so erstellten Teile sind im obenstehenden Bild grün eingefärbt. Ich habe im Bild den grünen Pfeil bewusst

schräg und nicht horizontal gezeichnet, da es sich am Anfang lohnt, etwas mehr Zeit ins Big Picture (hier das Kontextdiagramm) zu investieren.

Im CPRE Kurs verwende ich als Beispiel einen Webshop. Anstatt nun den ersten Use Case (z.B. die Produktsuche) mit allen Facetten und Features zu beschreiben, beschränke ich mich hier auf eine ganz rudimentäre Suchfunktion für den ersten Release. Anstatt den zweiten Use Case (z.B. die Bestellung) mit allen Varianten zu beschreiben, reduziere ich diesen auf einen Minimalablauf. Wichtig ist jedoch, dass die Minimal-Features in sich eine geschlossene Menge bilden, die für einen Release Sinn machen. Beispiel: Eine rudimentäre Bestellung mit Versand ist ok, nicht aber eine variantenreiche Bestellung ohne Versand. Gleiches gilt für die Business Objekt Struktur. Hier brauche ich für den ersten Release nicht alle Datenfelder, aber alle wichtigen Objekte bereits in einer rudimentären Struktur. Im Beispiel oben (Webshop mit Kunde, Produkt und Bestellung) macht es keinen Sinn, im ersten Release nur den Kunden zu modellieren, im zweiten das Produkt und im dritten die Bestellung. Schon im ersten Release muss ich die Business Objekte Kunde, Produkt und Bestellung in einer vereinfachten Form modellieren.

**Fazit**

Die RE-Tätigkeiten und deren Resultat sind im „agilen“ Vorgehen genauso essentiell wie im „klassischen“ und beide produzieren auch praktisch die gleichen RE-Resultate. Beim „agilen“ Vorgehen muss ich mir jedoch genau überlegen, was ich in welcher Reihenfolge benötige. Beim „klassischen“ Vorgehen brauche ich diese Überlegung nicht und spezifiziere daher das System meist *von oben nach unten*. Hier ist es im Prinzip egal, in welcher Reihenfolge ich die Spezifikation erstelle, da ich diese ohnehin erst dann den Entwicklern gebe, wenn sie vollständig ist. Beim „agilen“ Vorgehen mache ich mir somit frühzeitig Gedanken, welche Funktionen ein erster rudimentärer Release beinhalten soll. Dieses frühe „Schneiden“ macht das agile Vorgehen am Anfang anspruchsvoller, doch die hierfür investierte Zeit gewinne ich in späteren Phasen mehrfach zurück, da wesentliche Themen bereits sehr früh geklärt sind.

Ich wünsche euch viel Spass und Erfolg damit!