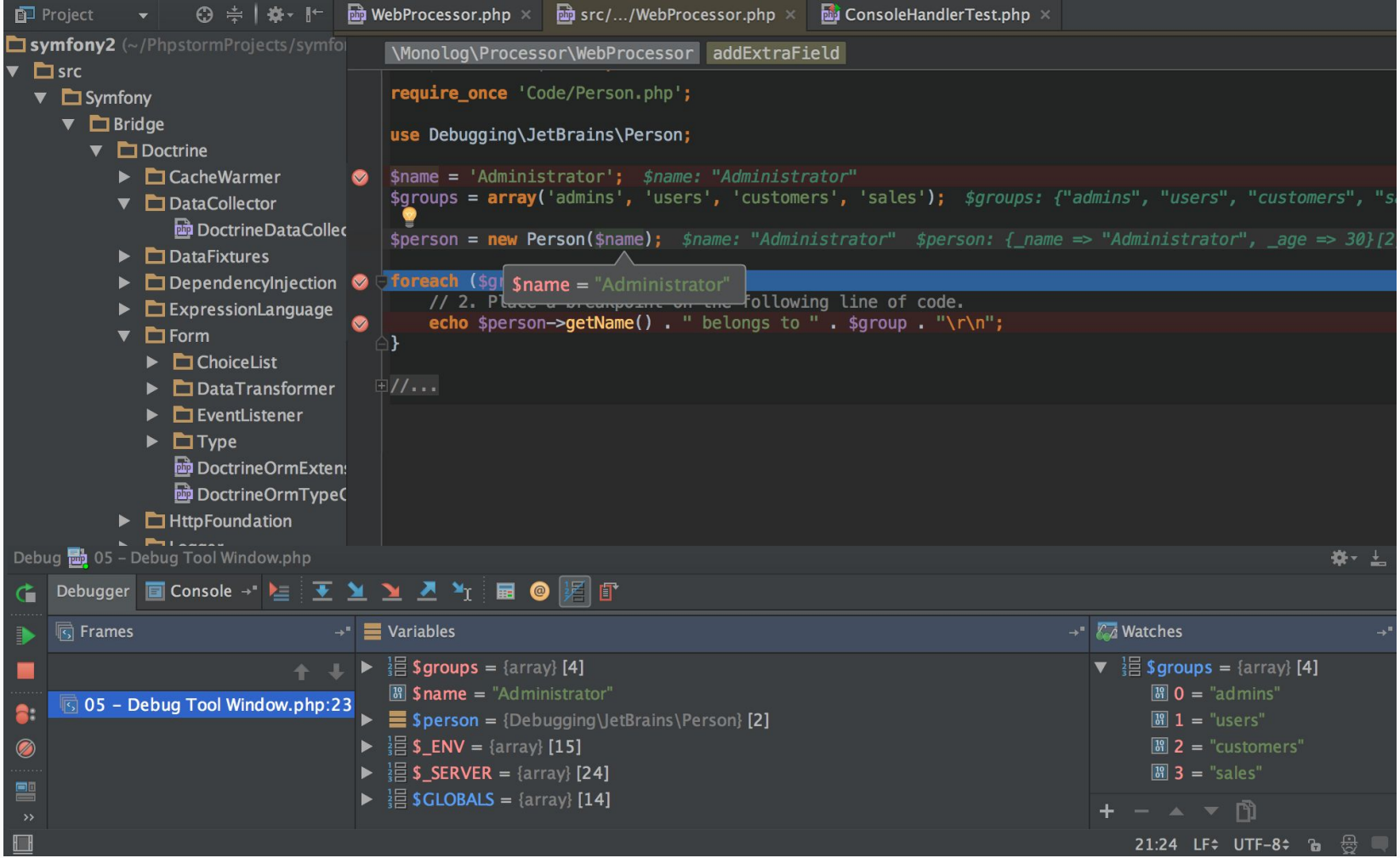
Agilität jenseits von Buzzwords

Timon Fiddike

@TimonFiddike

www.agile.coach

Bilder ohne Hinweis: pixabay.com



https://www.jetbrains.com/phpstorm/

Symfony

https://symfony.com/

https://agile.coach/test-driven-development-tdd-training/

30 Trainings in 30 Firmen später ...

Fallbeispiel 1: 15 Personen Team

1. Hauptaufgabe: Software Entwicklung
    a. Neue Control Panel Software entwickeln
2. Nebenaufgabe: Second Level Support
    a. 1 Support Anfrage / 20 min / Entwickler

30 Trainings in 30 Firmen später ...

Fallbeispiel 1: 15 Personen Team

1. Hauptaufgabe: Software Entwicklung
   a. Neue Control Panel Software entwickeln
2. Nebenaufgabe: Second Level Support
   a. 1 Support Anfrage / 20 min / Entwickler

80 % unzufrieden mit Ergebnissen:

1. Chefs aller Art:
   a. Wir wollen ganz viele Dinge.
   b. Es geht zu langsam.
2. Entwickler:
   a. Zeitdruck, Stress.
   b. Multitasking, wenig Fokus.
   c. Codequalität niedrig.

Knackpunkt sind immer wieder:
Menschen und Prozesse

Fallbeispiel 2: Neues Produkt in 5 Monaten

1. Neues Gesetz: Transfer Pricing für Konzerne.
2. “Wir wollen das erste Produkt dafür auf Messe [...] anbieten.”
3. “Gesetz noch nicht gelesen, aber Produkt muss in 5 Monaten fertig sein.”

Fallbeispiel 2: Neues Produkt in 5 Monaten

1. Neues Gesetz: Transfer Pricing für Konzerne.
2. “Wir wollen das erste Produkt dafür auf Messe [...] anbieten.”
3. “Gesetz noch nicht gelesen, aber Produkt muss in 5 Monaten fertig sein.”

1. 4 Leute im gleichen Raum, 100 % Fokus
2. 1 Wochen Sprints, jew. Feedback von Kunden
3. Überraschungen, aber schnell gelernt

Fallbeispiel 2: Neues Produkt in 5 Monaten

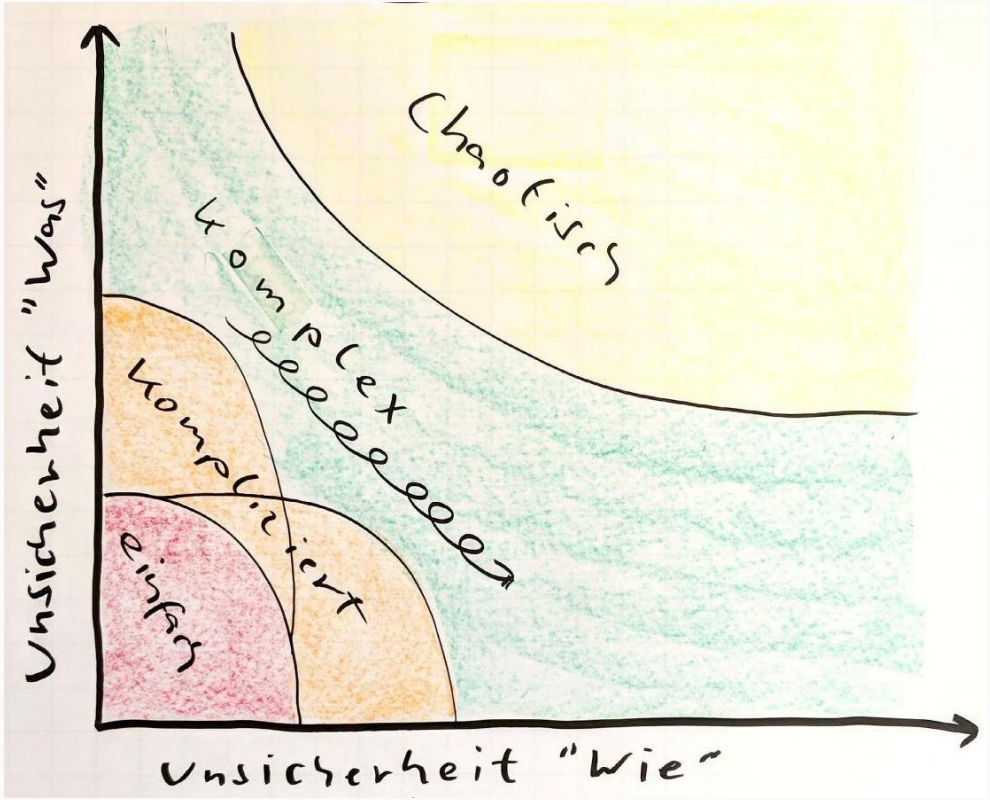
1. Neues Gesetz: Transfer Pricing für Konzerne.
2. “Wir wollen das erste Produkt dafür auf Messe [...] anbieten.”
3. “Gesetz noch nicht gelesen, aber Produkt muss in 5 Monaten fertig sein.”

1. 4 Leute im gleichen Raum, 100 % Fokus
2. 1 Wochen Sprints, jew. Feedback von Kunden
3. Überraschungen, aber schnell gelernt
4. Jeden Tag pünktlich nach Hause
5. Produkt rechtzeitig fertig
6. Großer Konzern kauft Produkt, “Happy €nd”

Wann macht agile (iterativ inkrementelle) Arbeit Sinn?

Stacey Matrix, Original Ralph Stacey,
Variante von Ken Schwaber:

“Was” funktioniert für Kunden / Nutzer?



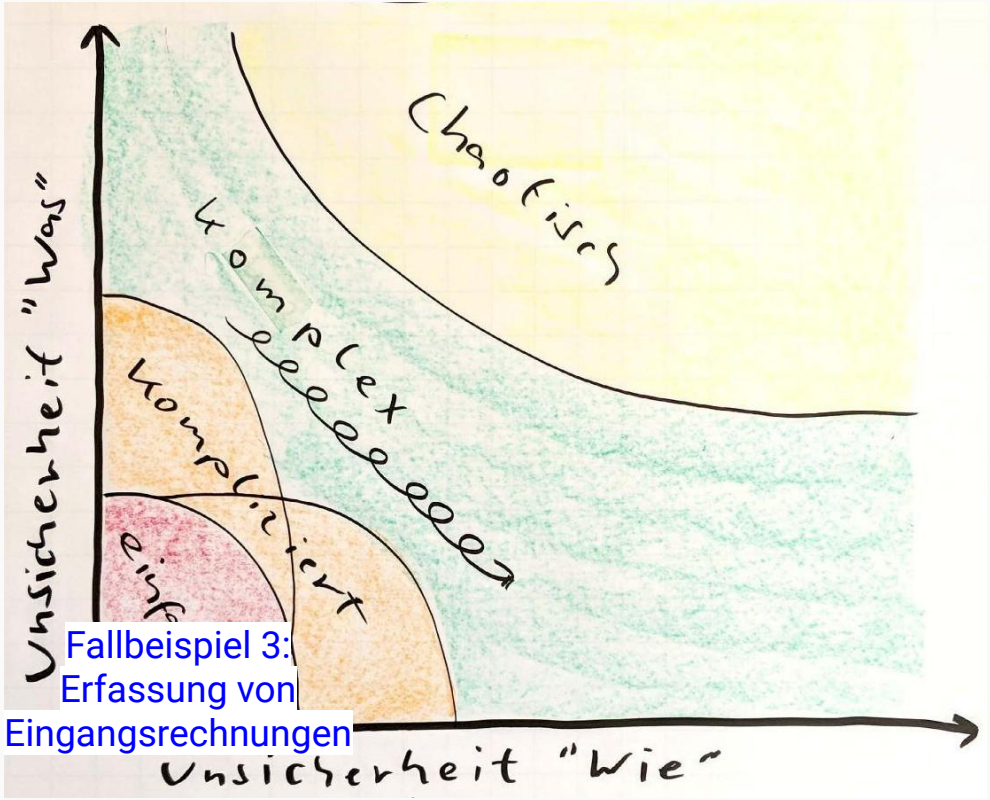
“Wie” schaffen wir das? Menschen, Prozesse, Technik

Wann macht agile (iterativ inkrementelle) Arbeit Sinn?

Stacey Matrix, Original Ralph Stacey,
Variante von Ken Schwaber:

Einfach: Vorher offensichtlich

“Was” funktioniert für Kunden / Nutzer?



Fallbeispiel 3:
Erfassung von
Eingangsrechnungen

“Wie” schaffen wir das? Menschen, Prozesse, Technik

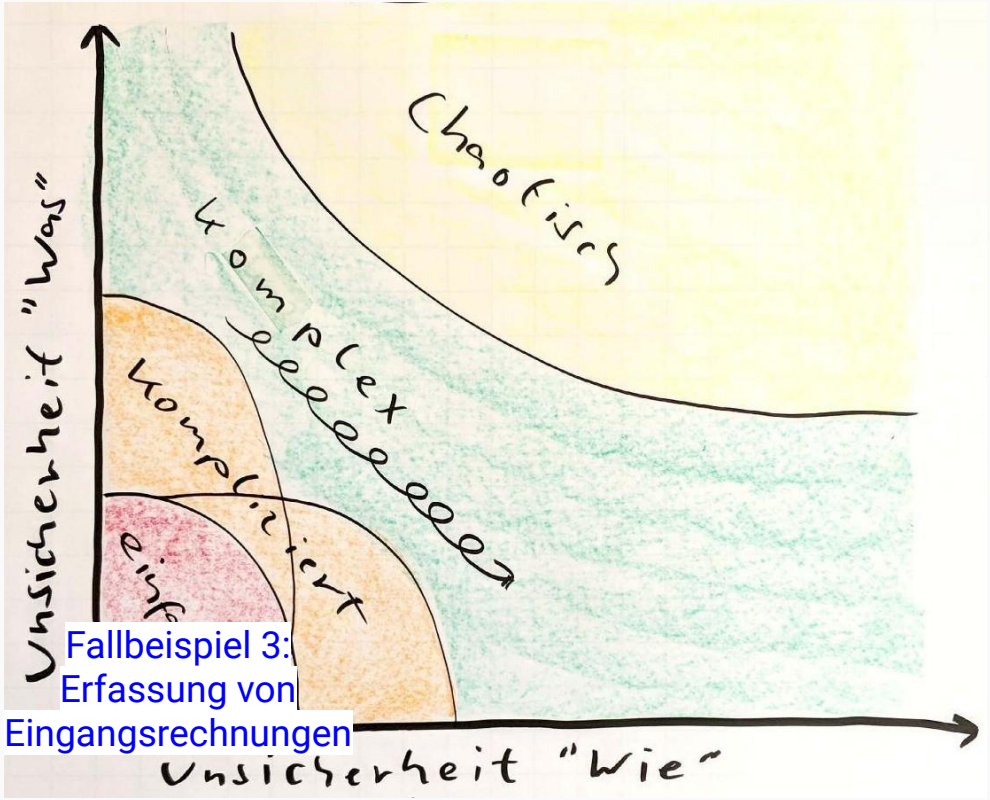
Wann macht agile (iterativ inkrementelle) Arbeit Sinn?

Stacey Matrix, Original Ralph Stacey,
Variante von Ken Schwaber:

Kompliziert: Vorher planbar,
detailliert Planen ist effizient

Einfach: Vorher offensichtlich

“Was” funktioniert für Kunden / Nutzer?



Fallbeispiel 3:
Erfassung von
Eingangsrechnungen

“Wie” schaffen wir das? Menschen, Prozesse, Technik

Wann macht agile (iterativ inkrementelle) Arbeit Sinn?

Stacey Matrix, Original Ralph Stacey,
Variante von Ken Schwaber:

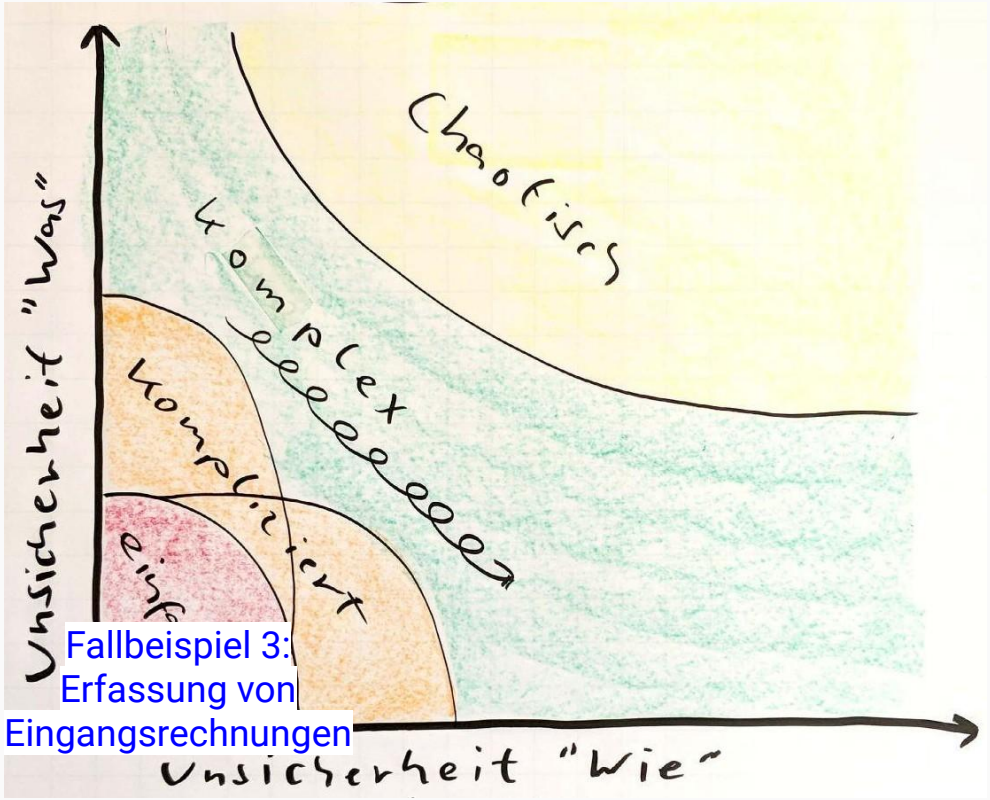
Chaotisch: Nicht replizierbar

Komplex: Hinterher schlauer,
nur iteratives Arbeiten ist effektiv

Kompliziert: Vorher planbar,
detailliert Planen ist effizient

Einfach: Vorher offensichtlich

“Was” funktioniert für Kunden / Nutzer?



Fallbeispiel 3:
Erfassung von
Eingangsrechnungen

“Wie” schaffen wir das? Menschen, Prozesse, Technik

Wann macht agile (iterativ inkrementelle) Arbeit Sinn?

Stacey Matrix, Original Ralph Stacey,
Variante von Ken Schwaber:

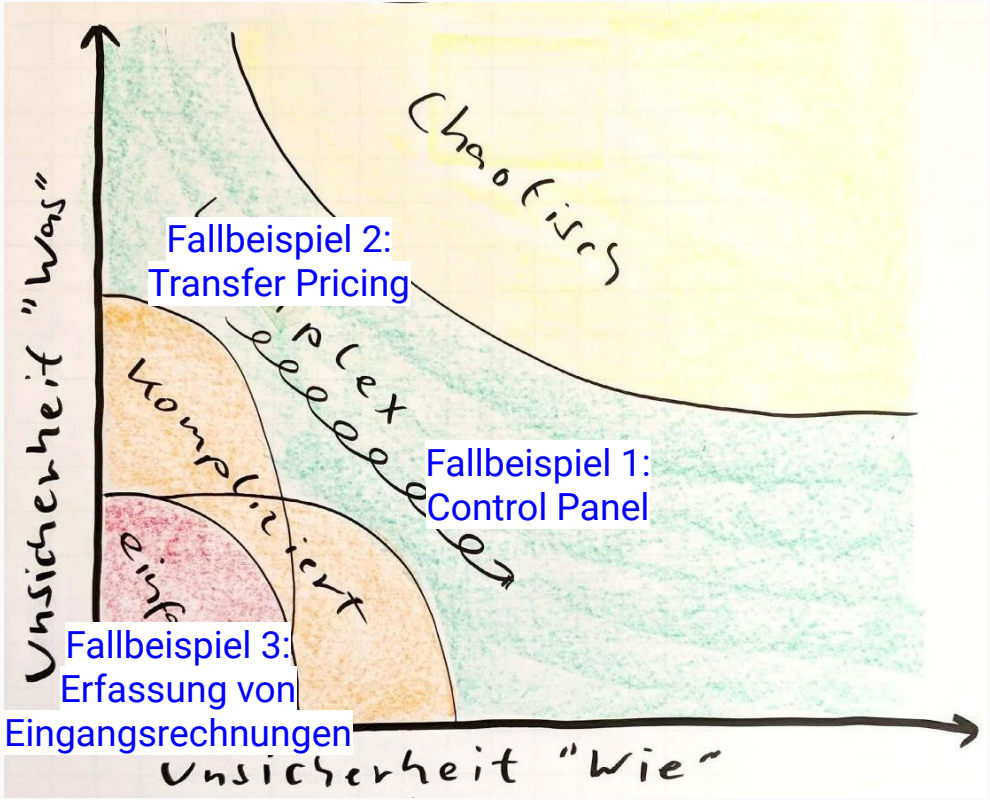
Chaotisch: Nicht replizierbar

Komplex: Hinterher schlauer,
nur iteratives Arbeiten ist effektiv

Kompliziert: Vorher planbar,
detailliert Planen ist effizient

Einfach: Vorher offensichtlich

“Was” funktioniert für Kunden / Nutzer?



“Wie” schaffen wir das? Menschen, Prozesse, Technik

Plan + Festpreis für Effizienz und Kontrolle??

Implizite Annahmen AG: Kompliziert: Vorher planbar, Festpreis ermöglicht Kontrolle.

1. AG: Ausschreibung, AN: “Festpreis”
2. Lieferung gemäß Vertrag
3. Change Requests bis AG zufrieden

Plan + Festpreis für Effizienz und Kontrolle??

Implizite Annahmen AG: Kompliziert: Vorher planbar, Festpreis ermöglicht Kontrolle.

1. AG: Ausschreibung, AN: “Festpreis”
2. Lieferung gemäß Vertrag
3. Change Requests bis AG zufrieden

Komplex? Erst Hinterher schlauer:

1. Vieles funktioniert nicht wie geplant, Zeitdruck

Plan + Festpreis für Effizienz und Kontrolle??

Implizite Annahmen AG: Kompliziert: Vorher planbar, Festpreis ermöglicht Kontrolle.

1. AG: Ausschreibung, AN: “Festpreis”
2. Lieferung gemäß Vertrag
3. Change Requests bis AG zufrieden

Komplex? Erst Hinterher schlauer:

1. Vieles funktioniert nicht wie geplant, Zeitdruck
2. Funktioniert für den Kunden so noch nicht, Anpassen ist schwierig, Agenturwechsel wäre sehr schwierig

Plan + Festpreis für Effizienz und Kontrolle??

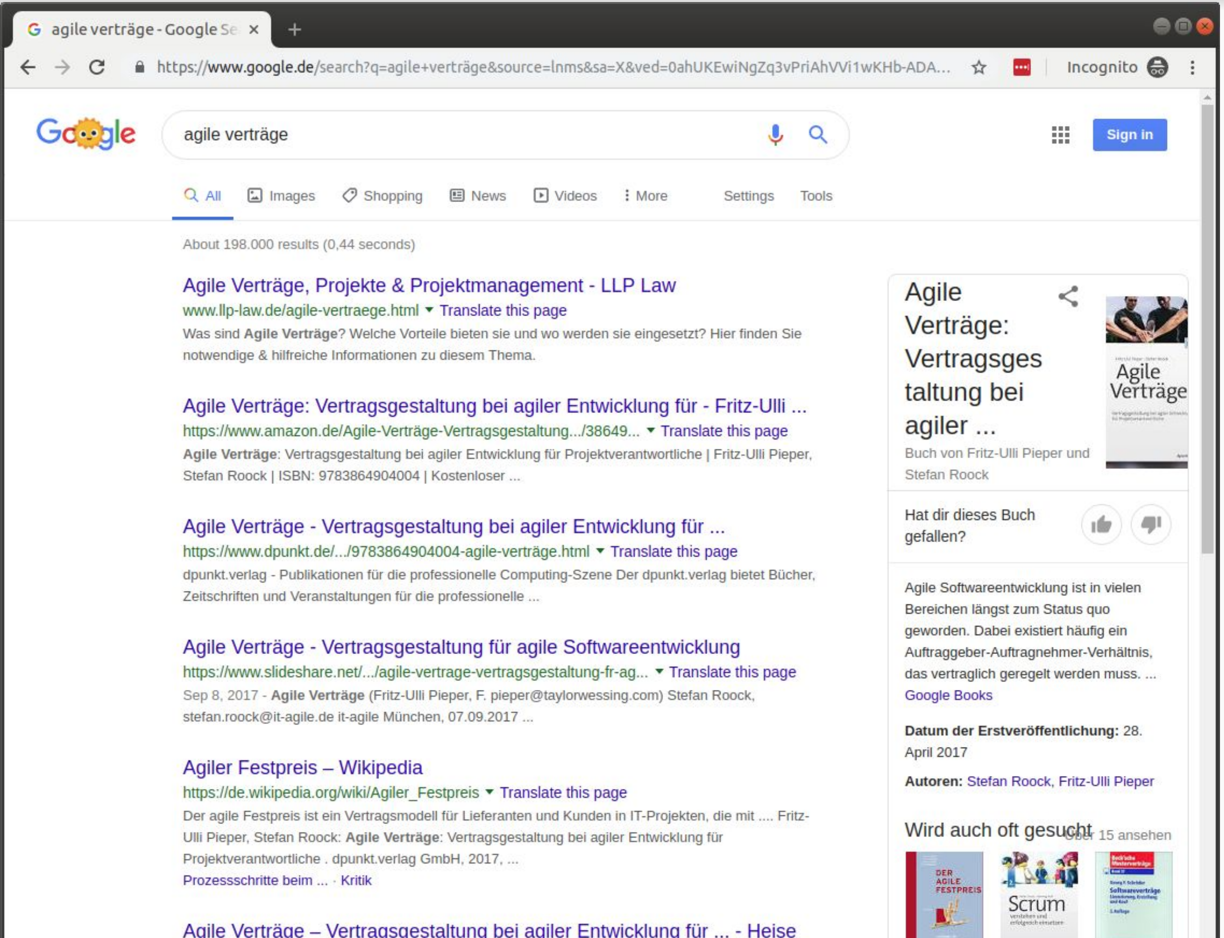
Implizite Annahmen AG: Kompliziert: Vorher planbar, Festpreis ermöglicht Kontrolle.

1. AG: Ausschreibung, AN: “Festpreis”
2. Lieferung gemäß Vertrag
3. Change Requests bis AG zufrieden

Komplex? Erst Hinterher schlauer:

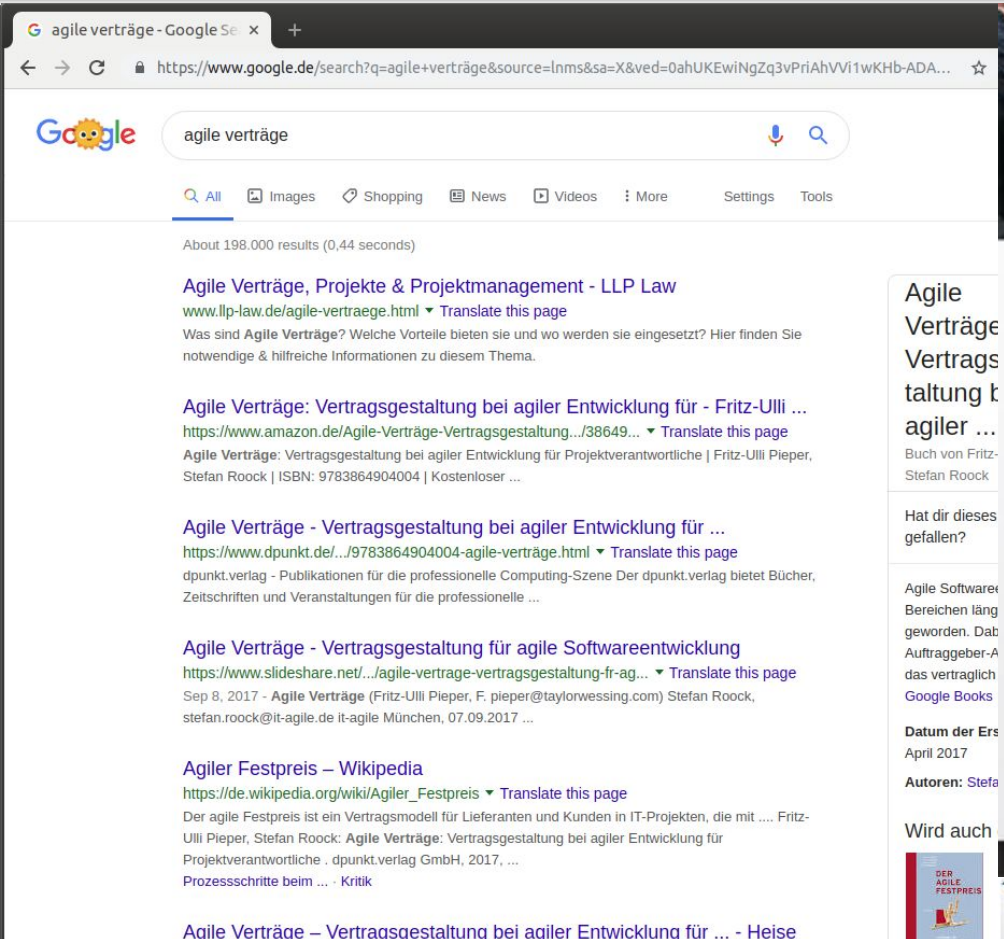
1. Vieles funktioniert nicht wie geplant, Zeitdruck
2. Funktioniert für den Kunden so noch nicht, Anpassen ist schwierig, Agenturwechsel wäre sehr schwierig
3. Super, jetzt verdienen wir endlich Geld!

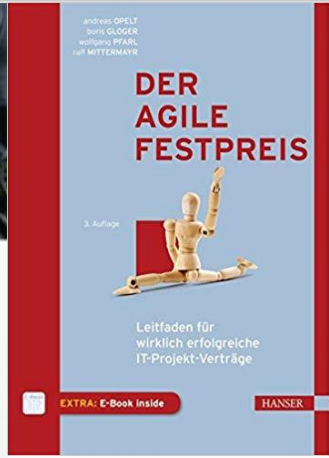
Win-Win gewünscht?



https://www.slideshare.net/roock/agile-vertrage-...

Win-Win gewünscht?

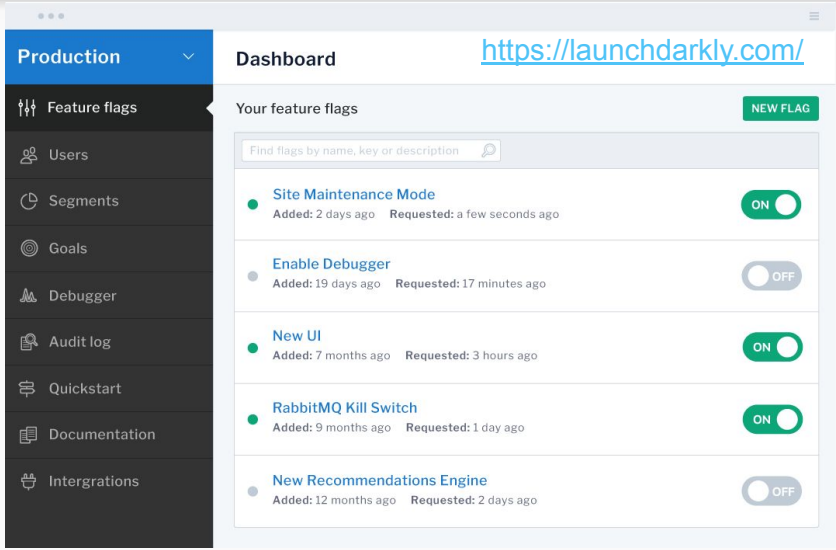




https://www.slideshare.net/roock/agile-vertrage-...

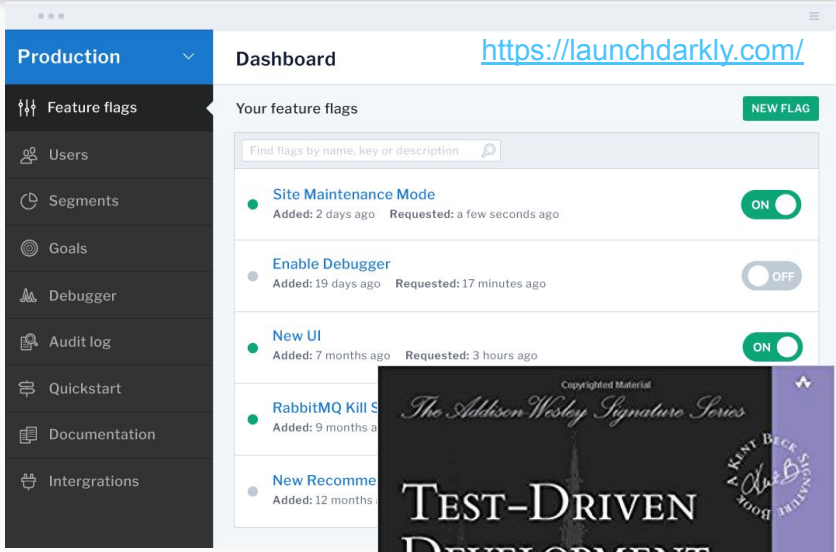
Technik muss Anpassungsfähigkeit des Business ermöglichen ...

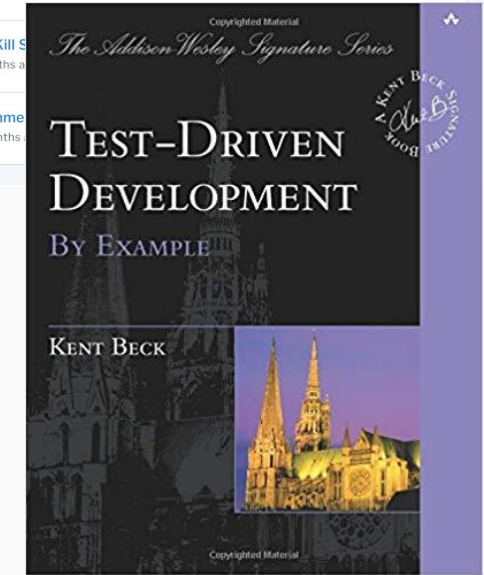
1. Feature Toggles erleichtern
   a. Branch in the codebase,
      a.k.a. Master Branch Development
   b. Continuous Integration, Delivery
   c. AB-Tests mit Kunden



Technik muss Anpassungsfähigkeit des Business ermöglichen ...

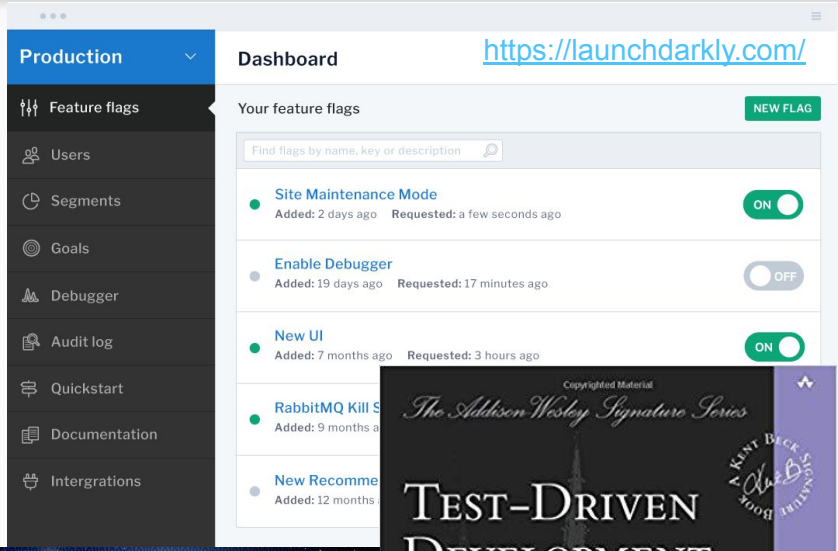
1. Feature Toggles erleichtern
   a. Branch in the codebase,
      a.k.a. Master Branch Development
   b. Continuous Integration, Delivery
   c. AB-Tests mit Kunden
2. Test Driven Development erleichtert
   a. Emerging Architecture
   b. Änderungen aller Art

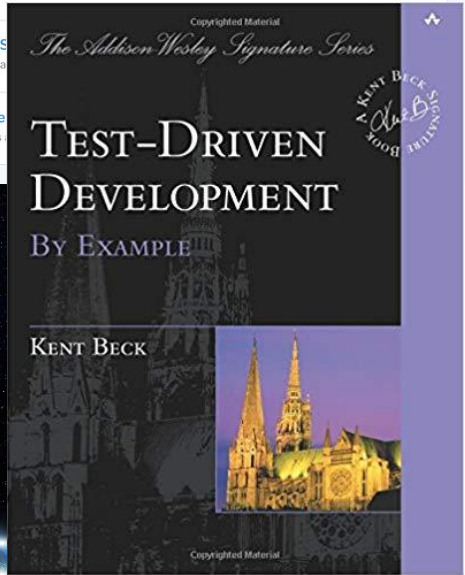




Technik muss Anpassungsfähigkeit des Business ermöglichen ...

1. Feature Toggles erleichtern
   a. Branch in the codebase,
      a.k.a. Master Branch Development
   b. Continuous Integration, Delivery
   c. AB-Tests mit Kunden
2. Test Driven Development erleichtert
   a. Emerging Architecture
   b. Änderungen aller Art
3. Pair Programming stärkt
   a. Fokus
   b. Durchsatz / Flow
   c. Lernen und Weiterentwicklung





Danke für Eure Aufmerksamkeit!

Timon Fiddike

@TimonFiddike

www.agile.coach

Bilder ohne Hinweis: pixabay.com