Ruy Kuhlmann
Schneidemühler Str. 28 E
76139 Karlsruhe
Tel: 0176 - 277 99299
Email: ruystt@web.de
Web: www.ruynk.com
Blog: ruynk.blogspot.de

"Agile" vs. "nicht Agile": Gedanken zu einem Vergleich

Zielgruppe

- Management
- Abteilungsleiter
- Projektleiter
- Business Analysten

Einleitung

Ist Ihnen eine Studie bekannt mit dem Ergebnis, dass eine "*agile*" Methode besser funktioniert als eine nicht "*agile*"?

Ich meine hiermit, einen „echten" Vergleich.

> Keine Annahmen.
>
> Keine Vermutungen.
>
> Keine Beteuerung: "... theoretisch funktioniert das so und so".

Ich suche eine Studie auf (nahezu) wissenschaftlicher Basis. Auf harten Fakten.

Ich suche einen Vergleich aus der Praxis mit (messbaren) Ergebnissen und nicht das, was man (leider viel zu) oft vorfindet:

1. Ein Vergleich nach „Prinzipien" (theoretischer Ursprungs also; wie z.B. Mut, Respekt, Einfachheit - Softskills halt), oder

2. Ein Vergleich nach bestimmten Punkten, wie z.B. hier: http://www.computerwoche.de/a/agile-methoden-im-vergleich,2352712 - Computer Zeitschrift Artikel (hier werden nur agilen Methoden untereinander verglichen - nach einer mir unbekannten Bewertungsmethode -). Siehe hierzu auch mein Vergleich: http://www.ruynk.com/downl/Vergleich_zwischen_Agilen.pdf

Das 'Wissen' über Agile wird weitgehend durch „Hören-Sagen" verbreitet. Genaueres ist nicht wirklich bekannt (und das scheint gewollt zu sein). Es gibt eine Unmenge an Buecher, doch es sind alle "agil" geschrieben, damit meine ich; schwammig und ungenau.

Aber was mir sehr nachdenklich stimmt ist es, dass anscheinend so gut wie kein Interesse daran besteht, einen solchen Vergleich durchzufuehren!

Folgende Probleme bei "*agilen*" Entwicklungsumgebungen habe ich selbst gesehen:

1. Qualitätsmanagement so gut wie nicht existent: Bugs werden einfach ignoriert, oft mit dem Vermerk: „not a bug" oder „would not fix". Die einfache Tätigkeit des "Testens" wird sehr oft mit "Qualitätsmanagement/Qualitätssicherung/Qualitätskontrolle" vermengt und weitgehend ignoriert (bis auf etwas, was irgendwie automatisch laeuft)

2. Risikomanagement wird oft (stark) vernachlässigt. Wenn überhaupt, wird das Thema "Risiko" auf dem Team abgeladen und die Entwickler dürfen sich bei den Retros (oder anderen 'Laber'meetings) gegenseitig verichern, dass Risken "voll" unter Kontrolle sind.

3. Hohe Fluktuation (Mitarbeiter die regelrecht „fliehen")

4. SW-Entropie; die SW wird von Sprint zu Sprint unübersichtlicher und voller (unbekannten und ignorierten) Bugs

5. Religiöses Eifer: Wider jeder Vernunft werden Rituale eingehalten ohne Rücksicht auf Verluste (Mantra: sollen wir etwa wieder nach Wasserfall arbeiten?), und

6. Blick in die Zukunft – sollte die SW tatsächlich fertig gestellt werden und ausgeliefert, ist damit zu rechnen, dass sie so gut wie unmöglich zu warten ist. Nich nur aus Mangel an Dokumentation sondern auch weil das produzierte Ergebnis auf gut Glück zusammengekleistet wurde (hängt auch mit Punkt 4 dieser Liste zusammen).

Ich kann mir nicht wirklich vorstellen, dass so etwas wie Scrum tatsächlich funktioniert.
Ausnahmen ja, aber dass diese Methode in mehr als (maximal) 10% der Teams Gutes tut, kaufe ich nicht ab.

"Scrum sieht in erste Linie die Verbesserungsmöglichkeiten in der Prozessstruktur und macht keinerlei Angaben zur eigentlichen Arbeit des Entwicklers. Boris Gloger, einer der deutschen Scrum-Trainer, berichtet dazu von erheblichen Steigerungen in der Produktivität, die den Faktor vier erreichen können, während die Vision von Scrum-Mitbegründer Jeff Sutherland den Faktor zehn beinhaltet [Gloger, Boris: Scrum – Produkte schnell und zuverlässig entwickeln. München, Hanser Verlag, 2008]" (Siehe weiter unten Bibliographie: Masterarbeit, auf Seite 17).

Faktor 4 ? Faktor 10 ??? Das ist viel zu viel zu viel des Guten.
Zu gut um wahr zu sein? Es gibt Leute die an solchem Hokus Pokus zu "glauben" scheinen.

Mir ist nur eine vergleichende Studie bekannt (siehe Bibliographie: Masterarbeit). Diese hat tatsächlich "Scrum" direkt mit "Wasserfall" verglichen. In wie weit dieses "Wasserfall" das von Royce beschriebenes Modell entsprach, ist mir nicht bekannt: Wahrscheinlich war eher ein "klassisches Modell" gemeint.

Diese Studie hat herausgefunden (Masterarbeit, Seite 88) dass während Scrum 7:19 Personenstunden (als Mittelwert aus 10 "Scrum" Projekten) benötigt hat, hat Wasserfall 6:02 Personenstunden als Mittelwert aus 11 "Wasserfall" Projekten benötigt.

Der Unterschied ist nach meiner Meinung nicht wirklich "bahnbrechend" und würde belegen, dass es überhaupt keinen Vorteil existiert, Scrum einzusetzen.

Es ist empfehlenswert die Masterarbeit (Bibliographie) zu lesen. Diese Arbeit ist jedoch kritisch zu bewerten, denn der allgemeine Tenor versucht aller Anschein nach "Scrum" zu "verkaufen", so z.B. schreibt der Autor (Seite 102, im Fazit): "Man kann aber durchaus davon ausgehen, dass Scrum auf die Softwareentwickler motivierend wirkt, so dass hier langfristig Verbesserungen bei der Umsetzungsgeschwindigkeit zu beobachten sind.".
Kurze Analyse gefällig? Ganz kurz: "Ich gehe davon aus, dass in der Zukunft XYZ besser **ist**." Die Wortwahl ist hier (und anderswo) tendenziös und irreführend. Ich unterstelle dem Autor nicht, dass er das absichtlich gemacht hat; doch sein "Glauben wollen" schimmert ab und an durch.

Vergleich

Um einen einigermaßen tragenden Vergleich zu gewährleisten wäre es wichtig, den Vergleich so genau wie möglich zu machen, und das gelingt nur wenn wir uns ausschließlich auf Fakten begrenzen.

Der Versuch sollte also, meiner Ansicht nach, folgende Punkte beinhalten:

1. Produktivität
Die Produktivität ist einerseits sehr wichtig, andererseits schwer zu messen. Am sichersten scheint dies durch "Function Points" (wurde im Jahre 1977 durch IBM eingeführt), also Punkte für Funktionen. Das funktioniert dergestalt, dass in einem Code man Punkte vergibt für:

- Funktionsaufrufe (Methode, Prozedur, ...); 1 Punkt pro Aufruf
- If Anweisung; 1 Punkt
- Switch Anweisung; 2 Punkte
- Schleifen (for, do-while, while-do); 3 Punkte für jede Schleife
- Bildschirm-Ausgabe (z.B. User Case Bildschirm); 5 Punkte für jede Ausgabe
- Report (z.B. User Case Bericht); 10 Punkte pro Ausgabe-Seite

Man addiert alle Punkte und man dividiert durch die Zeit (Wochen, Tagen, Stunden, je nachdem was Sinn macht): Damit hat man schon eine Grundlage für Vergleiche. Das ist schnell zu erledigen (mit "grep" z.B.) und ergibt durchaus brauchbare Vergleichszahlen.
Viel Arbeit?
Ja. Dafür ist doch das Management da.

2. Zufriedenheit der MA
Gemessen werden sollen nicht Überstunden (Siehe das Buch von Tom DeMarco et al.: Adrenalin-Junkies und Formular-Zombies) sondern Fluktuation.
Ganz wichtig (nicht vergessen): Jedes Mal, wenn ein Mitarbeiter ein Projekt verlässt, geht mit ihm viel Wissen verloren. Daher ist Fluktuation unbedingt zu verhindern.
Zu vergleichen sind:
2.1. Anzahl der MA, die ein nicht-Agile Projekt in einer gegebenen Zeit verlassen haben, und
2.2. Anzahl der MA, die ein Agile Projekt in einer gegebenen Zeit verlassen haben.

3. Last but not least; der Wunde Punkt der SW-Entwicklung: Bugs!
Folgende Werte sind zu erheben nach "Agile" und "nicht Agile" getrennt :
3.1. Anzahl Bugs pro Zeiteinheit
3.2. Kritikalität der Bugs pro Zeiteinheit (Anzahl der Bugs pro Zeiteinheit und Kritikalität)
3.3. Durchschnittliche Verweildauer eines Bugs (wie lange braucht man, einen Bug durchschnittlich zu beheben)
3.4. Durchschnittliche Verweildauer der Bugs nach Kritikalität (wie lange braucht man, einen Bug durchschnittlich nach Kritikalität zu beheben).

Auf Feedback freue ich mich sehr!

Bibliographie

Masterarbeit von Mark Harwardt; Wasserfallmodell versus Scrum (Fernuniversität Hagen - Lehrgebiet Programmiersysteme)

Mit freundlichen Grüßen,

R. C. N.-Kuhlmann
SCJP, ISTQB, REQB, CPM (IAPM)

Karlsruhe, Januar 2017