Ruy C. N.-Kuhlmann Projektkoordination
Schneidemühler Str 28 E
76139 Karlsruhe
Tel: 0176 277 99299
Email: ruystt@web.de

Vergleich zwischen agilen Methoden

'

Zielgruppe

- Management
- Abteilungsleiter
- Projektleiter
- Business Analysten

Einleitung

In diesem Schreiben ziehe ich einen Vergleich zwischen verschiedenen "agilen" Methoden durch.

Als Grundlage für diesen Vergleich dienen die Daten, die man unter:
http://www.computerwoche.de/a/agile-methoden-im-vergleich,2352712
für die verschiedenen agilen Methoden vorfinden kann.

Einleitung

Im oben erwähnten Vergleich wird halt verglichen, was die Methoden theoretisch "abdecken": Das besagt eigentlich so gut wie gar nichts darüber,
a) wie gut die Methoden "funktionieren" und
b) wie sie in einem Betrieb eingesetzt und von den Mitarbeiter erlebt werden.
M.a.W. ist dieser Vergleich mit sehr viel Vorsicht zu genießen und nur sehr bedingt brauchbar; das Gleiche gilt also folgerichtig auch für meine Ausführungen weiter unten.

Diese Vergleichen vergleichen nur nach der Definition der Methoden (besser: Vorgehensweise); die Methodik des Vergleichen (wie wurde gemessen, Voraussetzungen und Annahmen, etc) sind mir nicht bekannt. Über die Praxis der Methoden (praktische Anwendung) gibt es so weit ich das sehen kann, keine Aussagen bzw Erfahrungswerte. Darüber hinaus, so wie es bei mir ankommt, sind viele der Methoden eher Nischenlösungen. Gründe dafür:

- Es sind anscheinend viele Lösungen von eher kleinen Unternehmen, die damit versuchen etwas an Marktanteile zu bekommen.
- Es sind daunter möglicherweise Methoden, die eine Firma für sich selbst entwickelt hat und anschliessend testen sie den Markt, ob vielleicht ein Geschäft daraus zu machen ist.
- Einige Methoden sind aus der Kombination bzw Vermischung von anderen Methoden entstanden, wohl in dem Bestreben, die Vorteile von der einen oder der anderen Methode zu kassieren und die Nachteile zu beseitigen.

Zahlenwerk

Methode	Projekt Managemen	Requirement Engineering	Systemdesign	Qualitätsmanagement	Implementierung	Test	Integration	Wartung	Betrieb	Summe	Durchschnitt	Standard Abweichung
ActiF	3	4	3	2	4	1	3	0	0	20	2,22	1,4741
Adaptive Software Development (ASD)	2	1	1	4	2	1	0	0	0	11	1,22	1,2273
Agile Enterprise	4	4	2	3	4	4	3	1	0	25	2,77	1,3966
Agile Model Driven Development (AMDD)	1	2	2	2	3	4	3	3	3	23	2,55	0,8315
Behavior Driven Development (BDD)	0	4	1	3	2	2	1	0	0	13	1,44	1,3426
Crystal	2	2	0	3	3	4	1	1	0	16	1,77	1,3147
Design Driven Development (DDD)	0	3	1	1	1	1	0	0	0	7	0,77	0.9162
Dynamic System Developmemt Method (DSDM)	2	4	4	2	4	4	4	3	2	29	3,22	0,9162
Eclipse Way Process	4	4	4	2	4	4	3	1	0	26	2,88	1,4487
Evolutionary Process for integrating COTS-Based Systems (EPIC)	4	2	2	3	2	3	3	0	0	19	2,11	1,2862
Evolutionary Project Management & Product Development (EVO)	4	2	3	2	2	4	2	0	0	19	2,11	1,3699
Extreme Programming (XP)	1	3	2	3	4	4	2	1	0	20	2,22	1,3147
Feature Driven Development (FDD)	3	3	4	3	3	1	1	0	0	18	2,00	1,4142
Iconix	3	4	4	4	4	4	3	1	0	27	3,00	1,4141
Internet-Speed Development (ISD)	3	3	1	0	3	3	4	0	1	18	2,00	1,4142

Lean Software Development (LSD)	1	2	3	4	3	4	2	1	1	21	2,33	1,1547
Microsoft Solutions Framework For Agile Software Development (MSF4ASD)	3	2	2	2	3	3	2	0	0	17	1,88	1,0999
Mobile-D	2	2	1	2	2	3	1	0	0	13	1,44	0,9558
Rapid Application Development (RAD)	1	2	0	1	4	3	3	1	0	15	1,66	1,33
Scrum	4	4	1	3	1	1	3	1	0	18	2,00	1,4142
Test Driven Development (TDD)	0	1	0	3	4	4	2	1	0	15	1,66	1.5635
Unified Process	1	4	4	1	4	3	3	1	0	21	2,33	1,4907
Agile Unified Process	4	3	3	1	4	3	3	1	0	22	2,44	1,3426
Essential Unified Process (EssUP)	4	3	4	2	4	3	3	1	0	24	2,66	1,33
Open Unified Process (OpenUP)	4	4	4	1	4	3	3	1	0	24	2,66	1,4907
Usability Driven Development (UDD)	2	3	2	3	2	4	4	1	0	21	2,33	1,2472

Aus dieser Tabelle sind die besten und die schlechtesten Methoden bald gefunden:

Gewinner
Dynamic System Developmemt Method (DSDM) mit 29 Punkten
2. ist Iconix mit 27 Punkten
3. Platz für Eclipse Way Process mit 26 Punkten.

Verlierer sind
Letzter: Design Driven Development (DDD) mit 07 Punkten
Vorverlierer ist ASD mit 11 Punkten.
Besser schnitten BDD und Mobile-D mit jeweils 13 Punkten, wobei Mobile-D unwesentlich besser ist mit einer Standardabweichung von nur 0,9558.

Der Durchschnitt von allen Werten ist 19,3077. Die Verteilung scheint ziemlich normal zu sein, denn wir haben 13 Methoden über den Schnitt und 13 darunter.

Fazit

Scrum mit 18 Punkte ist leicht unterdurchschnittlich.
Test Driven Development (15 Punkte) liegt ca. bei 25%; schlechte Leistung eigentlich für eine Methode, die so viel versprochen hat (Mantra: Erst die Tests schreiben, dann entwickeln).

Extreme Programming hat 20 Punkte bekommen und ist somit besser als Scrum (wundert mich eigentlich; ich hätte XP eher bei ca 15 Punkten erwartet, schlechter als Scrum).

Dass Crystal ziemlich schlecht abgeschnitten hat (16 Punkte) überrascht mich gewissermaßen, denn ich halte Crystal für eine durchdachte Methode, die nicht Unmögliches verspricht und als eher solide erscheint.

Lean Software Development mit 21 Punkten schneidet bescheiden: Ich hätte mehr erwartet. Doch wie oben geschrieben: Wie gemessen wurde ist nicht ersichtlich und somit sind die Ergebnisse auch nicht zu erklären, nur gewissermaßen zu "erfühlen". Doch das gilt aber eigentlich für alle Ausführungen in diesem Fazit.

Über DSDM

"Der Entwicklungsprozess in DSDM umfasst sieben Phasen. Je nach Projektkonstellation können einzelne Phasen auch ausgelassen werden.
Phase 1 - Pre-Project
Phase 2 - Feasability Study
Phase 3 - Business Study
Phase 4 - Functional Model Iteration
Phase 5 - Design and Build Iteration
Phase 6 - Implementation
Phase 7 - Post-Project"
Offensichtlich arbeitet DSDM deutlich nach einem "klassischem" SW-Entwicklungsmodel (vox populi: Wasserfall), zu erkennen an die fest definierten Phasen der Entwicklung.
Erklärt vielleicht diese Tatsache, dass DSDM am besten abgeschnitten hat (29 Punkte)?
Anzumerken ist übrigens, dass eine solche Entwicklungsvorgehensweise, wie hier beschrieben, dem Standard Projekt Management sehr nahe kommt.

Über Iconix

"Der Iconix-Prozess besteht im Allgemeinen aus vier Phasen (siehe Abbildung) und verwendet insgesamt vier, auf der Unified Modeling Language (UML) basierende Diagramme (Use Case, Sequence, Domain, Class), um priorisierte Anwendungsfälle (Use Cases) durch Iterationen in einen lauffähigen Code zu überführen. In jeder Phase erfolgt eine Überprüfung der zuvor abgeschlossenen Arbeit und bei Bedarf eine Anpassung.

- Requirements
- Analysis/Preliminary Design
- Detailed Design
- Implementation"

Offensichtlich arbeitet auch Iconix nach einem "klassischem" SW-Entwicklungsmodel (vox populi: Wasserfall).
Erklärt vielleicht diese Tatsache, dass Iconix als die 2. beste Methode abgeschnitten hat (27 Punkte)?
Im Gegensatz zu dem Gewinner (DSDM) hat Iconix zwar nur 4 Phasen, dafür gilt Iconix als "schlanke" Entwicklungsmethode.
Das Ergebnis des Vergleichs macht also so weit Sinn (ist kongruent).
Anzumerken ist übrigens, dass auch hier die Entwicklungsvorgehensweise sehr ähnlich eines Standard Projekt Managements ist.

Über Crystal

Bei Crystal kann man folgendes lesen:
"Crystal-Entwickler Alistair Cockburn sollte Anfang der 1990er Jahre für die IBM Consulting Group eine Methode für objektorientierte Entwicklung ausarbeiten. Allerdings existierten zu dieser Zeit keine Vergleichsmöglichkeiten, an denen sich das Resultat orientieren konnte. Cockburn analysierte deshalb so viele Projekte wie nur möglich und befragte die jeweiligen Projektteams, was für den Erfolg oder Mißerfolg entscheidend war.
Er kam zu dem überraschenden Ergebnis dass Projekte dann erfolgreich abgeschlossen wurden, wenn das Projektteam keine festgelegte Vorgehensweise befolgte, sondern lediglich regelmäßig

miteinander kommunizierte. Die Ergebnisse blieben dabei von 1991 bis 1999, international wie auch über verschiedene Entwicklungssprachen hinweg, einheitlich.
Cockburn kam somit zu dem Entschluss, dass sowohl jedes Projekt als auch jedes Team einzigartig ist und keine allgemeine Vorgehensweise existiert."

Diese Aussage ist sehr interesant, besagt doch, dass
- entweder ist die Anwendung von Methoden irrelevant für die Abwicklung eines Projektes, oder (wahrscheinlicher)
- alle untersuchten Methoden "liefern" nicht. In diesem Falle sollte man sich fragen warum. Das habe ich mit meinem FlePA getan und ist im Buch (siehe URL www.ruynk.com) zu lesen. Es wird dem Management nicht wirklich gefallen, aber die Wahrheit soll sich aus (harten) Fakten ableiten lassen und soll kein Wunschdenken, nicht glamouros sein.

Evolutionary Project Management & Product Development

"Unter EVO wird das Projekt in kleine Stücke ("chunks") geteilt, wobei jeder chunk höchstens 5 Prozent des Gesamtaufwands umfassen sollte. Dabei werden diejenigen Funktionen zuerst entwickelt, die den größten Nutzen bringen und sich leicht implementieren lassen."
Das unterteilen in Chunks von ca 5% halte ich für unflexibel. Man sollte in Teile teilen, die Sinn ergeben und nicht nach starren Werten. Die Größe der Teile (siehe das FlePA Buch) erfordern leider etwas Nachdenken, nicht "n% wird irgendwie schon passen".

Hall of Shame

Es gibt (leider) Aussagen, die ich bei dem Vergleich der Computerwoche lesen musste, wo ich nicht umhin konnte, hier eine "Hall of Shame" errichten zu müssen. Dermassen verwirrt und hirnverbrannt sind sie.

Unter "Design Driven Development" ist zu lesen:

"D3 basiert auf den folgenden Werten:
* Designfülle und Traumerfüllung sind das neue Mantra für den Geschäftserfolg.
* Design ist ein zufälliges Ereignis, das während der Konzeption eintritt. Die Maximierung dessen Eintrittswahrscheinlichkeit ist der Schlüssel für Innovation"

Das ist der Hammer!
* Geschäftserfolg ist "Designfülle" (was ist das?) und "Traumerfüllung" (?!)
* Design ist ein "Zufall". Na, dann Prost! Buchhaltung, Pruduktivität, Qualität, etc etc etc werden wohl auch alle "Zufall" sein... Richtig? (??!!)

Unter FDD ist zu lesen:

"FDD stellt den Begriff "Feature" in den Mittelpunkt der Entwicklung. Ein Feature ist dabei als etwas definiert, was in den Augen des Kunden nützlich ist. Jedes Feature stellt somit einen Mehrwert für den Kunden dar. Die Beschreibung eines Features erfolgt in einem Satz nach dem Schema:
Action Result Object
Berechne das Saldo des Kontos
Features sind also sehr feingranulare Funktionen. Oftmals bestehen zwischen den Features Abhängigkeiten. Zusammenhängende Features werden zu so genannten Feature Sets gruppiert."
Siehe weiter unten "Kommentar zu FDD und BDD".

Unter Behavior Driven Development ist zu lesen:

"BDD legt den Schwerpunkt darauf, Software zu entwickeln, die für die Anwender von Belang ist. Dazu wird bei der Anforderungsermittlung neben den User Stories
(Schema: "As a <ROLLE> I want <AKTION> so that <MEHRWERT>")
auch das Verhalten (Behavior) in Form von Vor- und Nachbedingung erfasst. Das Hilfsmittel hierfür ist das Schema "Given - When - Then", was mit "Vorbedingung - Aktion - Nachbedingung" beschrieben werden kann. Somit lassen sich für eine User Story mehrere Szenarien mit unterschiedlichen Randbedingungen erstellen. Diese beschreiben in natürlicher Sprache, wie sich die Software dann jeweils verhalten soll."
Siehe weiter unten "Kommentar zu FDD und BDD".

Komentar zu FDD und BDD

Menschen (ich vermag gar nicht von Entwickler zu reden, denn einem Entwickler traue ich viel mehr Können zu, als die Menschen die solchen Stuß von sich geben) , die versuchen die SW Entwicklung nach bestimmte (sträflichen) Vereinfachungen zu definieren und es wagen, daraus eine Entwicklungsmethode zu konstruieren, erinnern mich an die Blinde, die einen Elefanten zu begreifen / definieren versuchen.

Anmerkung:
Einige (sogenannten agilen) "Methoden" scheinen nicht berücksichtigt worden sein, wie z.B. Kanban.

Mit freundlichen Grüßen,

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

Karlsruhe, Juni 2017

Web: www.ruynk.com
Freiberufler: www.amtrs.de
IT-Wissen - Kurse und Seminare: www.ryusui.de
Private HP: www.ruynk.de
Blog: ruynk.blogspot.de