Produktivitätsmessung von Softwareentwicklung im Wandel der Zeit

Macht der Wandel der Zeit eine Anpassung bei der Messung von Produktivität erforderlich

永远忠实的

Das ist ein Scrum

häufiger Untertitel : „Schnell, flexibel, in Teams ohne Hierarchien: So entstehen z.B. Websites und Shops und anderes nach Scrum, einer agilen Methode für eine schnelle Software-Entwicklung.“

Wie nun kam es dazu ?
eine kurze „historische“ Betrachtung

Frühzeiten der betrieblichen IT sind geprägt durch:

- jeder Fachbereich hat „seine“ Anwendung
- jede Anwendung(Fachbereich) hat „seine“ IT Organisationseinheit

sog. Matrixanwendungen (ab 1990er) geprägt durch:

- eine Anwendung bedient verschiedene Fachbereiche
- IT Organisationen organisieren sich gemäß der Technologie und verlieren die Nähe zum Fachbereich

Prä-Agiles Zeitalter (ab 2000er) geprägt durch:

- integrative Lösungen (wie z.B. SAP Produkte) bestimmen die Geschäftsprozesse
- komplexe Releasewechsel beherrschen das Geschehen

gegenwärtig „Agiles Zeitalter“ geprägt durch:

- Kontinuität der Teams und Kontinuität der Softwarelieferung (time to market)
- immer komplexer; immer globaler werdende Lösungen (cloud) beherrschen die IT

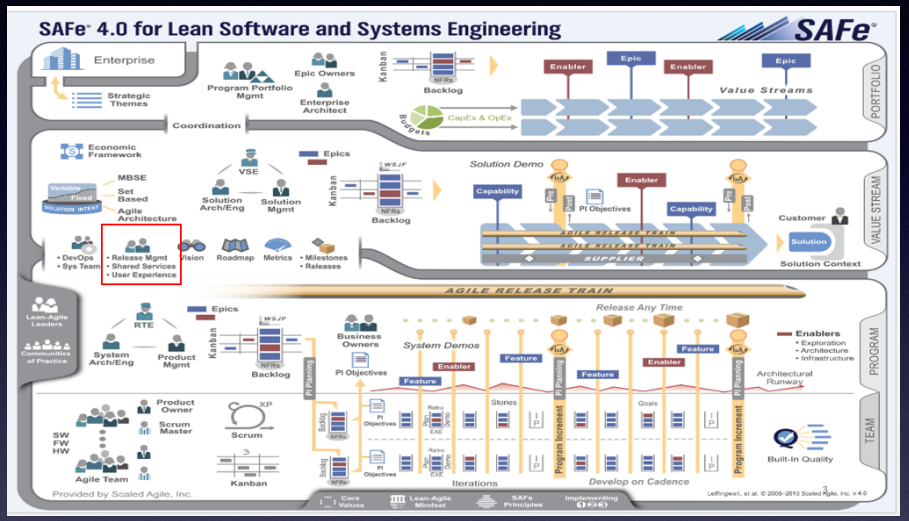
Wie nun kam es dazu eine kurze „historische“ Betrachtung

stellt man der „historischen“ Entwicklung in der Informationstechnologie die jeweils zeitgemäßen Methoden und Frameworks gegenüber, wird man feststellen, dass dem Verschwinden von sog. Greenfield Projekten notwendigerweise eine Reformation des V-Modell Ansatzes folgen mußte.
Ist bei grossen komplexen Neu-Implementierungen ein strukturiertes und interaktives Vorgehen gemäß Wasserfall bzw. V-Modell zielführend, so behindert selbiges Modell die Umsetzung von kleinen Anpassungen im laufenden Betrieb (betrifft sowohl den Betrieb der Technologie wie auch die operativen Prozesse der Fachbereiche)

Stellen wir also fest, dass der Agile Release Train nicht aufzuhalten ist

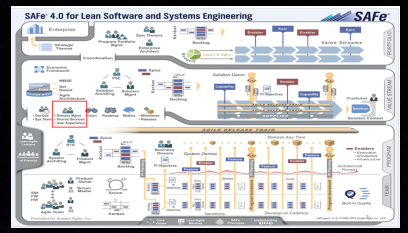
- zumindest für Organisationen, welche ihren Schwerpunkt auf Releases UND auf Geschwindigkeit (time to market) legen, ist die Einführung von agilen Frameworks (SAFe ist ein Beispiel) naheliegend

- erforderliche Metriken entstammen heute allerdings mehrheitlich noch dem Projektgedanken und lehnen sich an unzeitgemäße Methoden an



永远忠实的

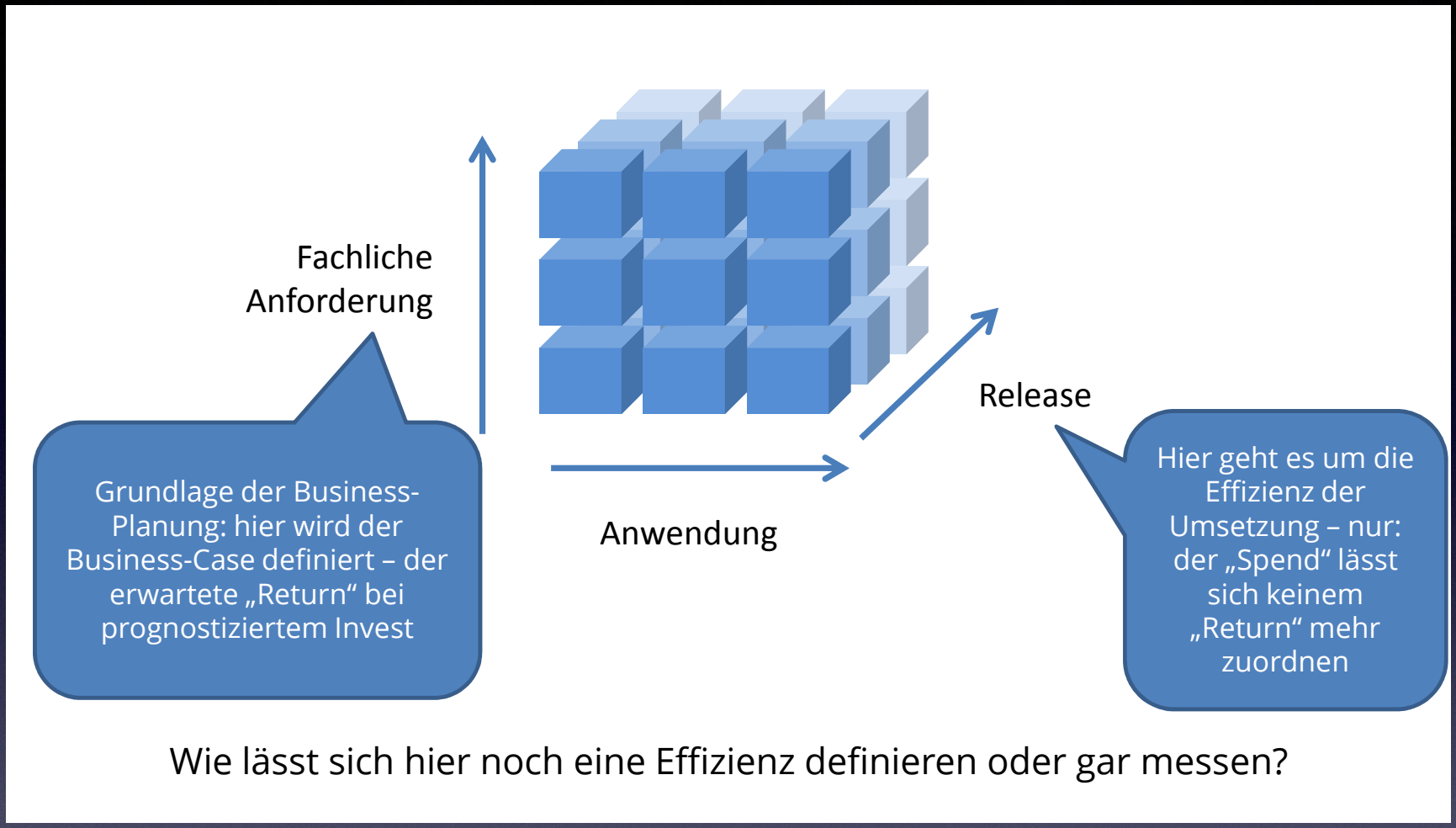
Evolution von Vorgehensmodellen



was lernen wir nun aus dieser Geschichte?; Vorgehensmodelle stellen keine Philosophie dar!
Vorgehensmodelle passen sich der Realität also den vorliegenden Bedarfen der Informationstechnologie also deren Anforderungen an.

Wir dürfen und müssen es also keinesfalls als einen Krieg der Modelle verstehen, sondern vielmehr einen kontinuierlichen Reifungs- und Anpassungsprozess in der Informationstechnologie erkennen.

Warum dies ebenso für Messmethoden und Metriken gilt, soll im weiteren Verlauf klarer werden



Einzelnen fachlichen Anforderungen lassen sich kaum noch Kosten bzw. Aufwand zuordnen

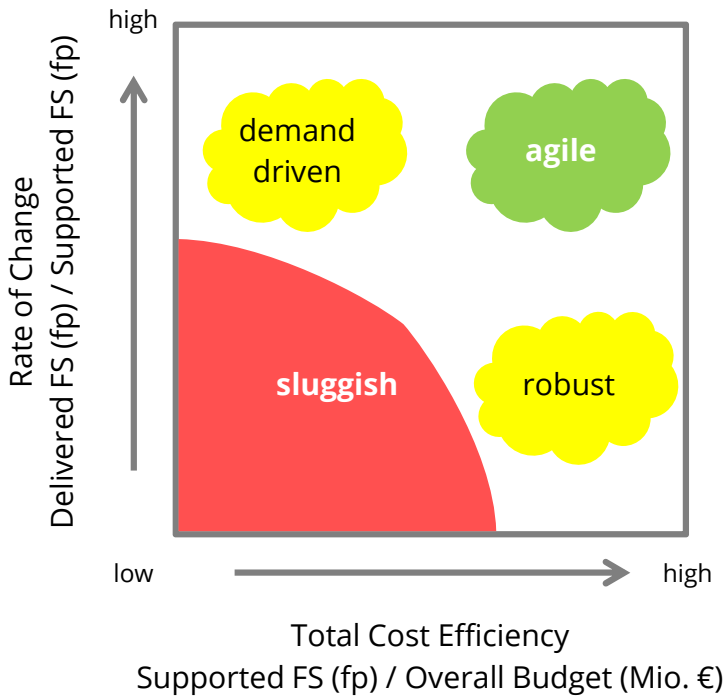
永远忠实的

Produktivitätsmessung durch den Wandel der Zeit

Wenn es keine Projekte mehr gibt, muss der Messgegenstand neu definiert werden

- Statt Projekt der Zeitraum
- Statt Projektergebnis die Anwendung (Baseline und Rate-of-Change)
- nur!was (einfach?) messbar ist, kann auch gemessen werden

Unterstellt wir hier, dass dem Auditorium die Notwendigkeit von Produktivitäts- und anderen Messungen bewußt ist. Alle „Nicht-Eingeweihten“ finden interessante Informationen im Backup



Warum?

- Kosten/Aufwandsseite lässt sich häufig nur schwer nach FS-relevanten Aktivitäten differenzieren
- Changes sind nicht nur Treiber für Entwicklung, sondern auch für Wartung

Einschätzung

- Schnelle, strategische Einordnung der applikationsbezogenen Effizienz
- Nicht unmittelbar zur Steuerung geeignet: Hierfür weitere Analysen notwendig, z.B. Aufwands- bzw. Kostenverteilung, Entwicklungseffizienz, usw.

Vorteile

- Vermeidet Abgrenzungsprobleme („FS-relevant“, Entwicklungs- vs. Unterstützungsleistungen)
- Lässt sich leicht über Applikationen aggregieren

Herausforderungen

- Baselines („Supported FS“) müssen ermittelt oder zumindest abgeschätzt werden

Erläuterungen
Bezugszeitraum := alle Werte sind auf einen Bezugszeitraum (z.B. 12 Monate, ein Release usw.) zu normieren
Delivered FS := Gelieferte Funktionalität in Sinne eines FPA Enhancement Count
Supported FS := Unterstützte Funktionalität im Sinne eines FPA Application Count (Baseline) zum Ende des Bezugszeitraums
Overall Budget := Grundsätzlich sämtliche Kosten, die in AD&M anfallen, konkret zu definieren/abzugrenzen, z.B.: Beratung, Analyse, Design, Entwicklung, Fehlerbehebung, Betriebsunterstützung, Tests, Regressionstests usw.

Bewertung moderner Enterprise Applications

$$Effort(pm) = K_{enh} * (fs_{enh})^{Senh} + K_{change} * (fs_{enh})^{Sce} * (fs_{bas})^{Scb} + K_{bas} * (fs_{bas})^{Sbas}$$

Legende

- $_{enh}$ = steht für „enhancement activities“
- $_{bas}$ = steht für „baseline“
- K_{change} = Kostenfaktor für
- K = Kostenfaktoren – vgl. COCOMO
- fs = Functional Size
- S = Skalentreiber – vgl. COCOMO
- Kalibrierungsfaktoren – sind hier der Einfachheit halber weggelassen

Aufwandsschätzmodell auf Grundlage des Functional Sizing

	Erweiterungen	Nur Wartung	Neuentwicklung
Fs_{enh} (fp)	2.500	0	2.500
FS_{bas} (fp)	15.000	15.000	2.500
k_{enh} (pm/fp)	0,13	0,13	0,13
s_{enh}	1,05	1,05	1,05
k_{change} (pm/fp)	0,50	0,50	0,50
Sce	0,40	0,40	0,40
Scb	0,10	0,10	0,10
k_{bas} (pm/fp)	0,13	0,13	0,13
s_{bas}	0,80	0,80	0,80
RoC (p.a.)	17%	0%	100%
Effort (pm p.a.)	795	285	574
Effort (fte)	66	24	48
P_{dev} (fp/pm)	5,2		5,2
P_{ms} (fp/fte)	570	630	320

P_{dev} = Produktivität, wie in „klassischem“ Projektbenchmark festgestellt
P_{ms} = Wartungseffizienz, wie in „klassischem“ Wartungsbenchmark festgestellt

Szenario 1 - „Durchschnitt“

	Erweiterungen	Nur Wartung	Neuentwicklung
Fs_{enh} (fp)	2.500	0	2.500
FS_{bas} (fp)	15.000	15.000	2.500
k_{enh} (pm/fp)	0,07	0,07	0,07
s_{enh}	1,03	1,03	1,03
k_{change} (pm/fp)	0,50	0,50	0,50
Sce	0,40	0,40	0,40
Scb	0,10	0,10	0,10
k_{bas} (pm/fp)	0,10	0,10	0,12
s_{bas}	0,80	0,80	0,80
RoC (p.a.)	17%	0%	100%
Effort (pm p.a.)	470	219	309
Effort (fte)	39	18	26
P_{dev} (fp/pm)	11,3		11,3
P_{ms} (fp/fte)	720	820	340

P_{dev} = Produktivität, wie in „klassischem“ Projektbenchmark festgestellt
P_{ms} = Wartungseffizienz, wie in „klassischem“ Wartungsbenchmark festgestellt

Szenario 2 - „schlanke Entwicklung“

	Erweiterungen	Nur Wartung	Neuentwicklung
Fs_{enh} (fp)	2.500	0	2.500
FS_{bas} (fp)	15.000	15.000	2.500
k_{enh} (pm/fp)	0,20	0,20	0,20
s_{enh}	1,05	1,05	1,05
k_{change} (pm/fp)	0,70	0,70	0,70
Sce	0,40	0,40	0,40
Scb	0,10	0,10	0,10
k_{bas} (pm/fp)	0,18	0,18	0,18
s_{bas}	0,80	0,80	0,80
RoC (p.a.)	17%	0%	100%
Effort (pm p.a.)	1.176	395	868
Effort (fte)	98	33	72
P_{dev} (fp/pm)	3,4		3,4
P_{ms} (fp/fte)	410	460	230

P_{dev} = Produktivität, wie in „klassischem“ Projektbenchmark festgestellt
P_{ms} = Wartungseffizienz, wie in „klassischem“ Wartungsbenchmark festgestellt

Szenario 3 - „Tanker“

	Schlank	Durchschnitt	Tanker
Fs_{enh} (fp)	2.500	2.500	2.500
FS_{bas} (fp)	15.000	15.000	15.000
k_{enh} (pm/fp)	0,07	0,13	0,20
s_{enh}	1,03	1,05	1,05
k_{change} (pm/fp)	0,50	0,50	0,70
Sce	0,40	0,40	0,40
Scb	0,10	0,10	0,10
k_{bas} (pm/fp)	0,10	0,13	0,18
s_{bas}	0,80	0,80	0,80
RoC (p.a.)	17%	17%	17%
Effort (pm p.a.)	470	795	1.176
Effort (fte)	39	66	98
P_{dev} (fp/pm)	11,3	5,2	3,4
P_{ms} (fp/fte)	720	570	410

P_{dev} = Produktivität, wie in „klassischem“ Projektbenchmark festgestellt
P_{ms} = Wartungseffizienz, wie in „klassischem“ Wartungsbenchmark festgestellt

Vergleich der Szenarien

Ja!Der Wandel der Zeit macht Anpassungen bei der Messung erforderlich

- denn nicht die Software hat sich geändert, sondern die Art und Weise wie diese entsteht
- Effizienz bzw. Produktivitätsmessung moderner Softwareentwicklung fordert Ansätze, die sich an den aktuellen Vorgehensweisen orientieren
- wie nun gesehen funktioniert Functional Sizing auch weiterhin, allerdings ist der Bewertungsgegenstand Projekt sinnvoll mit dem Mess-Zeitraum und „rate of change“ der Anwendung innerhalb des Zeitraumes zu ersetzen

BACKUP

Produktivitätsmessung in der Softwareentwicklung (speziell im agilen-Kontext)

- folgendes ist so oder so ähnlich allenthalben zu lesen (entspring den zahllosen Veröffentlichungen zu Scrum)

„Am Ende jeder Etappe wird neben dem Produkt auch der Prozess genau unter die Lupe genommen. Das Team evaluiert gemeinsam was gut läuft und was verbessert werden kann. Die zu verbessernden Punkte werden priorisiert und in den nächsten Etappen gezielt angegangen. So entwickelt das Team nicht nur das Produkt von Etappe zu Etappe weiter, sondern verbessert sich als Team und die Arbeitsprozesse sowie die Produktivität. Neben der gezielten Kommunikation und Behebung von Hindernissen im “Daily Standup” ist das sogenannte “Sprint Retrospective” Meeting der wichtigste Treiber für die Weiterentwicklung des Teams und der damit einhergehenden Produktivitätssteigerung.“

„Die korrekte Umsetzung von Scrum führt somit nicht nur in der Entwicklung von Software zur Vervielfachung der Produktivität. Wir nutzen Scrum in jeder einzelnen Abteilung und für all unsere Projekte z.B. für die Veranstaltung unserer Scrum Trainings und im Recruiting neuer Mitarbeiter Immer wieder konnten wir die Vorteile von Scrum sehen. Etappe für Etappe lernen wir, wie wir unsere Arbeit besser machen können und steigern so unsere Produktivität.“

Produktivitätsmessung in der Softwareentwicklung (speziell im agilen-Kontext)

Wir finden, dass unabhängig von der Methode; sogar unabhängig vom Fokusbereich (Technologie oder jedweder anderer wertschöpfender Bereich) die Produktivität gemessen werden sollte, einfach um Veränderungen zu bemerken (negativ wäre dies ein Indiz für Kostensteigerung) und den Erfolg von ergriffenen Maßnahmen zu beweisen.
Wir finden aber auch, dass gerade eine Methode (wie Scrum), welche die Produktivitätssteigerung wie keine andere „im Wappen führt“, kontinuierlich den Beweis erbringen sollte, dass dieses Versprechen auch eingehalten wird