Ausgangssituation

Für klassische Projektsituationen existieren bewährte Planungs- und Steuerungsmethoden (CCPM).

Daneben gibt es aber auch eher produktionsähnliche Situationen (z.B. feature- und produktorientierte inkrementelle Innovationen, Testphasen oder Bugbehebung) die grundsätzlich einfacherer (leichtgewichtiger) zu planen und steuern sind.

Beiden gemeinsam ist das Ziel den Fluss (Features/Zeit) oder besser Durchsatz (€/Zeit) zu optimieren. Der Schlüssel hierzu ist die Reduzierung der offenen Aufgaben auf ein Minimum.

Anforderungen

- Die Qualität muss steigen – einwandfrei sein
- Die Durchlaufzeiten müssen sinken
- Die globale Effizienz muss sich verbessern – über Durchlaufzeitverkürzung ergibt sich bei gleichen Ressourcen automatisch ein höherer Durchsatz
- Die Selbststeuerung des Teams muss gefördert werden – das Team braucht Informationen um die täglichen/operativen Entscheidungen sicher richtig selbst treffen zu können
- Es muss klar sein, wann Drum-Buffer-Rope geeignet ist – wo dessen Grenzen sind – so dass das Team sicher selbst wählen kann wann welche Methode zum Einsatz kommen soll
- Die Steuerung muss möglichst einfach sein – damit schnell kommunizierbar und erlernbar
- Die Steuerung muss stabil gegenüber Störungen sein
- Schnell reagieren und Entscheidungsbedarf früh aufzeigen (Frühwarmsystem) und damit erzwingen, dass Entscheidungen früh getroffen werden
- Eindeutiges Signal als Warnung
- Führt zu einer Steuerung des Upstream Prozess – z.B. über „no buffer holes“
- Bei großen Changes einfache Möglichkeit zum Replanning
- Agil – Richtungswechsel einfach/leicht möglich – ohne großen Planungsaufwandsverlust …. ABER … auch zu viele Richtungswechsel unterbindet in dem es eindeutige Signale gibt wenn es ausufert – Trade-Offs explizit macht
- Zuverlässig wenn Kundenzusagen gemacht wurden
- Zeit zum denken gibt

Idee

Wenn es schon in der Softwareentwicklung produktionsähnliche Situationen gibt, warum sollte man dann nicht aus Produktionssteuerungen adaptieren. Das Ziel aller Produktions- und Projektsteuerungen ist den Work-in-Progress so zu begrenzen, so dass der Fluss/Durchsatz optimal wird.

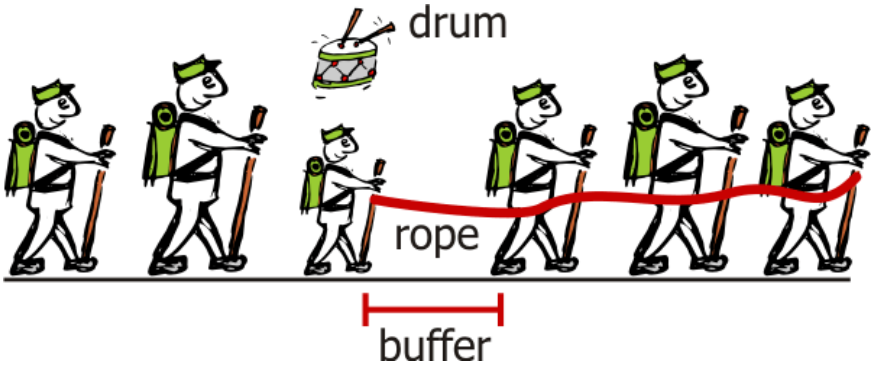
Die erste Generation der Fertigungssteuerungen geht hierbei auf Henry Ford (1913) zurück. Der Trick war damals den Work-in-Progress über die Lagerfläche an Zulieferteilen am Band zu begrenzen. Der

Nachteil war, dass man sinnvoll nur ein einziges Produkt produzieren konnte – Flexibilität war unmöglich.

Die zweite Generation wurde durch Taiichi Ohno/Toyota (1953) gefunden - Kanban. Hier wird der Work-in-Progress über den Bestand (an Paletten/Aufgaben/Arbeitspaketen) vor einer Verarbeitungsstufe begrenzet. Hiermit war es möglich viel mehr Varianten zu produzieren – um den Preis hoher Bestände.

Die dritte Generation stammt von E. Goldratt (2005). Hier taucht zum ersten Mal die Idee der Steuerung anhand des Engpasses auf. Mit diesem Ansatz kann die Steuerung weiter vereinfacht und die Bestände gesenkt werden. Bezeichnet wird diese Steuerung nach ihren Bestandteilen: Drum, Buffer, Rope – Trommel, Puffer und Seil.

Das Prinzip wir oft anhand einer Pfadfindergruppe dargestellt. Das Ziel ist hierbei, dass alle zusammen optimal schnell am Ziel ankommen. Das Problem ist – einer der Pfadfinder ist der langsamste und die Gruppe muss immer wieder Pausen einlegen um sich zu organisieren.



Die Lösung ist einfach: Der langsamste wird identifiziert – er ist der Engpass – er wird zum Taktgeber („Trommel/Drum“) der ganzen Gruppe. Vor dem Engpass muss ein kleiner Abstand bestehen („Puffer/Buffer“), so dass wenn jemand davor (Downstream) stolpert, dass der Engpass im weiter laufen kann. Alle dahinter (Upstream) werden automatisch gebremst, da man am Engpass ja nicht vorbei kommt. Was noch fehlt ist das Signal für den Vordersten, so dass er nicht wegläuft. Daher bekommt dieser ein Seil („Rope“) das ihn mit dem Engpass verbindet.

In der Produktion ist Up-/Downstream vertauscht – aber alles andere bleibt. Diese Steuerung sorgt sicher für eine optimale Work-in-Progress-Begrenzung, minimale Bestände und damit optimalem Durchsatz.

Drum-Buffer-Rope (DBR) in der agilen Softwareentwicklung

DBR geht davon aus, dass es sich um ein Produktionsumfeld handelt. Dies ist gekennzeichnet dadurch, dass:

- viele kleine Features (Artefakte oder Stories) auszuliefern sind,
- die Erstellung nach einem einheitlichen stufenweisen Ablauf erfolgt,
- die netto Dauer an der an einem Artefakt gearbeitet wird viel kleiner (<10%) gegenüber der brutto Dauer der Erstellung ist.

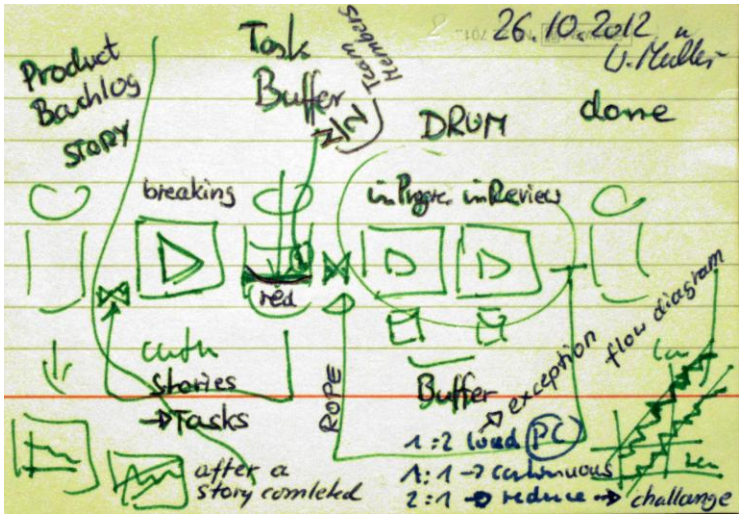
In der Praxis ergeben sich nun zwei Varianten: eine sehr Scrum ähnlich mit wenigen Prozessschritten – sehr einfach und leichtgewichtig und eine zweite, eher verwandt mit Kanban, mit vielen Prozessschritten.

DBR für Scrum-ähnliche Teams

Das oberste Ziel ist es die Sprints loszuwerden um damit die unnatürlichen Brüche am Ende der Sprints zu vermeiden und in einen kontinuierlichen Fluss zu kommen. Der Fluss ist wichtig um die Anzahl der offenen Stories und Tasks zu verringern und somit die Durchlaufzeit zu Verkürzen. Am Schluss steigt natürlich auch der Durchsatz.

Keine Panik - viele Dinge aus Scrum überleben (das meiste davon ist sehr nützlich) - nur die Steuerung wird angepasst. Eine Retrospektive alle 2-3 Wochen – ist immer noch sinnvoll. Es gibt weiterhin ein „Planning 1“ – durch „Reliable Scrum“ wird das Backlog aber in den ersten Sprints weitgehend komplett qualifiziert, so das Backlog vollständig priorisiert und geschätzt vorliegt und nur noch angepasst werden muss. Reviews werden natürlich auch gemacht - aber nicht in einem definierten Rhythmus, sonder alle 5-10 Stories, wenn es wirklich Wert ist, etwas zu zeigen. „Dailies”, „Impediment Logs“ und bleibt natürlich alles erhalten.

So was hat ändert sich dann? Es gibt keine Sprints mehr! Es ist alles ein kontinuierlicher Fluss mit dem Ziel, so wenig wie möglich Stories und Tasks geöffnet haben.



Um dies zu erreichen, wird der Prozess in zwei unabhängige Teile geteilt 1. das Schneiden der Stories in Tasks (auf der linken Seite des Taskpuffer) und 2. das Abarbeiten der Tasks (auf der rechten Seite).

Die Steuerung der linken Seite ist extrem einfach. Das Aufteilen von Stories in Task ist ja für eine Person nur wenige Stunden Aufwand – ungefähr so viel wie ein Task selbst. Daher wird das Schneiden von Stories ausschließlich durch die Menge der Tasks im Taskpuffer gesteuert. Wenn nur noch zwei Tasks übrig sind – wird einer der Entwickler die nächste Geschichte aus dem Backlog nehmen und sie in Task aufbrechen. Es kann sein, dass die Alarmgrenze von 2 Task zu riskant ist. Sie sehen, dass - wenn ein Pufferloch auftritt – also keine Aufgabe in der Aufgabenliste Puffer mehr übrig ist. Wenn Pufferlocher auftreten muss man einfach die Alarmschwelle solange (langsam und schrittweise) erhöhen bis keine Pufferlöcher mehr auftreten. Die Alarmschwelle sollte dabei nicht die

Hälfte der Anzahl der beteiligten Entwickler überschreiten, ansonsten ist es ein deutlicher Hinweis auf Prozessprobleme.

Das Monitoring geschieht über die aus dem „Reliable Scrum“ bekannten Werkzeuge – also das klassische „Product Burndown Chart“ und die neue „Fieberkurve“. Diese Diagramme werden immer aktualisiert, sobald eine Story fertig gestellt wurde. Hierdurch entstehen viel mehr Messpunkte und noch feinere „Echtzeit“ Transparenz. Übrigens ist dieser Teil im strengen Sinne gar keine Drum-Buffer-Rope Steuerung. Hier spricht man von "Vendor Managed Inventory" oder "Just-In-Time". Der Verkäufer/Lieferant (Backlog) überwacht den Puffer vor der Produktion und füllt diese eigenständig auf.

Und jetzt auf der rechten Seite? Das Drum-Buffer-Rope! Ok auch hier muss man etwas genauer hinschauen. Normalerweise hat eine Drum-Buffer-Rope viele Prozessschritte (wie in Kanban). Hier haben wir einen zweistufigen kontinuierlichen Prozess mit zwei Schritten "Entwicklung" und "Review/Test". Das besondere ist aber, dass es nur eine Art von Ressourcen gibt – Entwickler. Diese haben zwar Unterstützung durch QA Mitglieder, die die Tests zu schreiben - aber am Ende sind die Entwickler der begrenzende Faktor. Daher macht es keinen Sinn, zwischen den Prozessstufen zu unterscheiden. Beide werden durch die Verfügbarkeit der Entwickler eingeschränkt.

Die Drum-Buffer-Rope Steuerung besteht nun aus folgenden drei Teilen:

1. **Die Trommel** - dies ist die begrenzende Ressourcen - in diesem Fall die Entwickler. Die Trommel ist wie der Herzschlag der Produktionskette. Sie gibt den Takt vor – nach ihr müssen sich alle richten.
2. Dann benötigen wir **einen Puffer** (in der Regel vor der Trommel), aber in diesem Fall, wenn es nur einen begrenzenden Prozessschritt gibt ist der komplette Bestand (Anzahl der offenen Tasks) selbst der Puffer - spielt aber für die Steuerung keine Rolle ob ein angefangener Task im Bearbeitung ist oder in einem Puffer liegt. Keine dedizierten Puffer ist Letzt endlich sogar ideal.
3. Und **das Seil**? Das ist die Signalleitung vom Puffer um neue Aufgaben zu starten. In diesem Fall ganz einfach - wenn eine Aufgabe abgeschlossen ist, dann darf eine neue Aufgabe gestartet werden.

Das Monitoring für die rechte Seite besteht aus einem "aggregiertes Input-Output-Diagramm", oder manchmal auch "Flussdiagramm" genannt. Das Ziel ist es, den Bestand (Differenz zwischen Ein- und Ausgangslinie) so niedrig wie möglich zu halten.

Dies kann auf sehr einfache Weise erreicht werden. Es werden keine neuen Aufgaben begonnen, bis die ersten "Pufferlöcher" entstehen. Wenn der erste Entwickler keine Task mehr hat dann kann ein zusätzlicher Task gestartet werden und damit wird der Puffer um eins erhöht. Dies sollte aber eine Ausnahme sein und es müssen die Ursachen hierfür untersucht werden. Pufferlöcher sind voll von Informationen über Hindernisse oder verfahrenstechnische Probleme. Aber schließlich, wenn alles getan wurde und immer noch Pufferlöcher auftreten, dann muss man den Bestand erhöhen um den Durchsatz zu sichern.

Wenn für eine lange Zeit keine Pufferlöcher aufgetreten sind, kann man den Bestand Schritt für Schritt wieder zu reduzieren, z.B. durch eine einfach temporäre Regel „Alle 5 Tasks einen weniger starten“ o.ä.

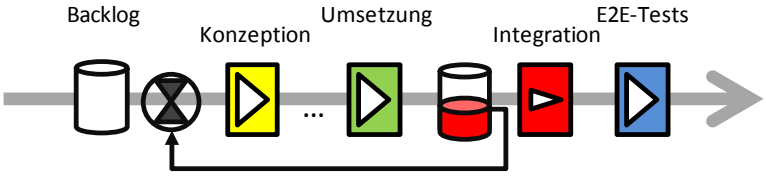
Damit ist „ultimative Scrum“ definiert und zeigt diese Ergebnisse (in der Praxis):

1. kontinuierlichen Fluss
2. mit der minimal möglichen Anzahl offener Tasks
3. und die kürzeste Lieferzeiten
4. und dem Best möglichen Durchsatz

p.s.: es ist mathematisch nachweisbar, dass DBR das Optimum ist – um jetzt noch weiter die Leistung zu steigern muss man an den Fähigkeiten der Entwickler arbeiten und weiterhin störenden Overhead reduzieren … aber die Luft wird deutlich dünner :-)

DBR für Kanban-ähnliche Prozesse mit vielen Stufen

DBR für vielstufe Prozesse verwendet ebenso ein vollständiges Product-Backlog mit Features (s. auch Reliable Scrum). In diesem Product-Backlog sind nur Features enthalten für die der Kunde bereit ist zu bezahlen (Ressourcen zur Verfügung zu stellt) alles andere kommt in einen Ideenspeicher und wird hier nicht betrachtet. Die Features im Product-Backlog weisen eine eindeutige Reihenfolge auf, die durch den Product Owner festgelegt wird.

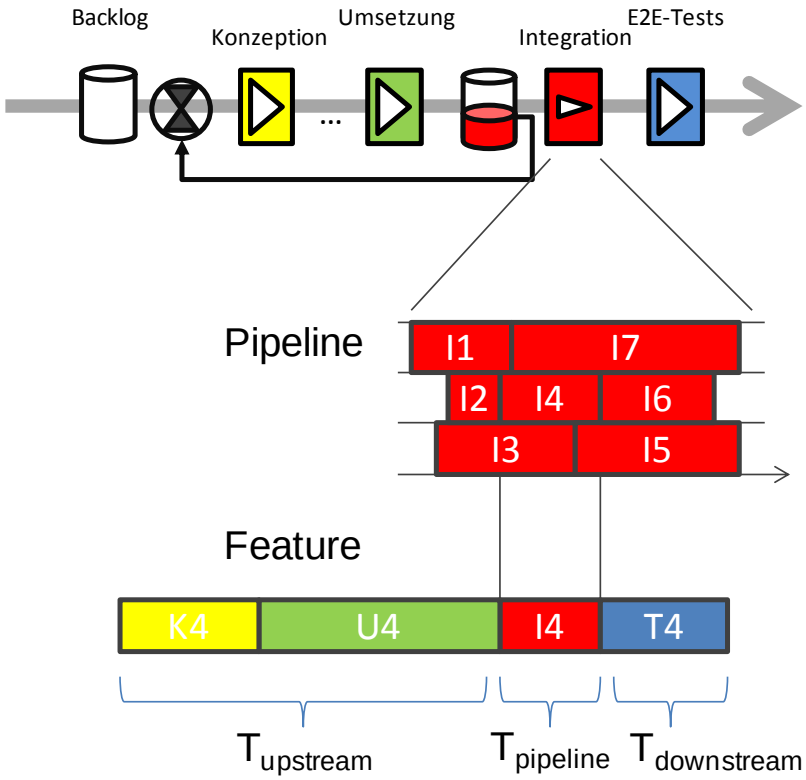


Ausgehend vom Backlog wird eine Prozesskette definiert: in der Softwareentwicklung typischerweise (Konzeption – Umsetzung – Integration – Tests). Bei DBR wird davon ausgegangen, dass eine dieser Stufen wahrscheinlich ein Engpass darstellt – entweder dieser ist offensichtlich – oder er wird strategisch festgelegt. Gut geeignet ist die Integrationsphase – da her viele Experten ungeplant zusammen arbeiten müssen. Die Verfügbarkeit der Experten stellt hierbei eine natürlich Begrenzung dar.

Damit erkennt man auch die Drum = Engpass = Integration. Den Buffer = Puffer an Features vor dem Engpass und die Rope (Seil) Signal zur Freigabe von neuen Features zur Entwicklung.

Eine ganz einfache Variante von DBR wäre in dem Fall – immer wenn der Puffer an Features auf 2/3 einer festzulegenden Menge von Features als Puffer absinkt werden entsprechend neue Features in die Pipeline zugelassen

Der Vorschlag DBR in der Softwareentwicklung geht einen detaillierteren Weg. Der Zeitpunkt wann ein Feature zu freigegeben wird hängt an der Pipelineplanung im Engpass.



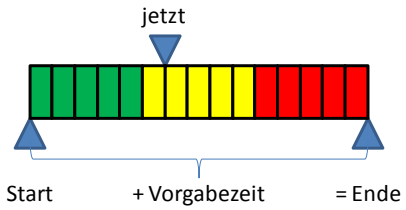
Hierzu wird für jedes Feature der Zeitbedarf, pro Verarbeitungsstufe, geschätzt. Zum Zuge kommt die 3-Punkt-Schätzung mit Wahrscheinlichkeit, Verhältnismäßige Berechnung in Bezug auf den Entwicklungsaufwand oder eine fixe Vorgabe bei uniformen Verarbeitungsstufen.

Eine Verarbeitungsstufe (die Engpassstufe) dient als Planungshilfe. Alle Bearbeitungszeiten der Features in dieser Stufe werden über die Kapazität pro Zeit abgetragen und werden als „Planned Load“ bezeichnet. Im Beispiel ist dies die Integrationsphase und die Kapazität besteht aus drei Features gleichzeitig.

Das Due-Date eines Features berechnet sich aus dem geplanten Ende in der Engpassstufe zuzüglich geplanter Downstream-Dauer und zuzüglich Sicherheitspuffer. Bei Releasezyklen größer eines Tages wird der Termin auf den nächsten Releasetermin aufgerundet.

Das Startdatum der Entwicklung eines Features berechnet sich aus dem geplanten Start in der Engpasstufe abzüglich geplanter Upstream-Dauer. Bei sprintorientiertem Vorgehen wird auf den Anfang des passenden Sprints abgerundet. Die Kapazitäten der Nicht-Engpasstufen werden so eingestellt, dass sie im Normalfall die Durchlaufzeit nicht beeinflussen. Einzelne Due-Dates können als commited definiert werden – sie dürfen bei Neuberechnungen nicht ohne Abstimmung mit dem Product Owner geändert werden.

Die operative Zuordnung der Ressourcen zur Abarbeitung der Features richtet sich nach dem Anteil der vergangen Zeit zwischen Start der Umsetzung und Due-Date des einzelnen Features.



Die Features mit dem größeren Anteil vergangener Zeit sind zu bevorzugen. Zu besseren Transparenz werden die Anteile in vier Farben dargestellt: erste Drittel = grün, zweites Drittel = gelb, rotes Drittel = rot und Due-Date überschritten = schwarz.

Zur Optimierung des Flusses wird das Portfolio der Features auf 30% rot (und 0% schwarz) geregelt. Wenn mehr als 30% der Features rot gemeldet werden muss der Sicherheitspuffer vergrößert und bei kleiner 30% verringert werden.

Für alle Features wird im akribisch ermittelt welche Wartezeiten auftraten und welche Hindernisse (Impediments) dazu führten. Für die Engpassstufe wird ermittelt wie oft es zu Arbeitslücken kam und aus welchem Grund. Hieraus werden Maßnahmen abgeleitet die global die Durchlaufzeit verkürzen.

Erweiterungen

Rapid-Response(RR)-Aufträge (z.B. Marketing Aktivitäten, Bugfixing oder Incidents). RR-Aufträge können mit fixen Bearbeitungszeiten versehen werden. Für die RR-Aufträge wird ein gewisser Anteil der Kapazität der Engpasstufe reserviert werden. Falls diese Kapazität nicht genutzt wird, kommt diese Kapazität den normalen Aufträgen zugute.

Releasezyklen sind in diesem Vorgehen nicht notwendig. Vor allem in Kombination mit RR-Aufträgen macht es Sinn tägliche Releases anzustreben.

Durch die zugesagten Due-Dates können die Features einfach in Rahmenprojekte (Vermarktung) integriert werden. Das Vermarktungsprojekt fungiert hierbei als Container – die Features werden als Zulieferung betrachtet und über den Ampelstatus überwacht.

Sobald die Durchlaufzeitverkürzung greifet und sich der Anteil der Touch-Time >10% entwickelt kann reibungslos auf Critical-Chain-Project-Management umgeschaltet werden. Der einzige Unterschied ist die aufwändigere Modellierung der Planung und die andere Generierung der Ampelfarben.

Alle bewährten Vorgehensweisen von Scrum können hier weiterhin zum Einsatz kommen – man muss aber überlegen, welche wirklich notwendig sind und die Komplexität der Steuerung nicht unnötig verkomplizieren:

- Daily Stand-Ups
- Sprints, Sprint Planning und Sprint Burn-Down – sind grundsätzlich nicht zu Gesamtsteuerung notwendig – können aber weiterhin zu Feinsteuerung verwendet werden
- Impedimentlogs und Impedimentabarbeitung sind sehr willkommen sollten aber systematisch in den KVP einbezogen werden
- Planttafeln mit Laufkarten jederzeit und gerne – vor allem geeignet Zeitfresser systematisch zu erfassen

Funktionale Betrachtung

Funktion	Scrum-NG
1) WIP-Begrenzung	Due-Dates werden über „Planned Load“ kapazitätsecht in der Engpasstufe (CCR) so ermittelt, dass der WIP zur Kapazität passt
2) Engpassidentifikation/zuweisung	Strategische = willentliche Festlegung der Engpasstufe
3) Due-Date Bestimmung	DBR: Ende der Planned Load für das Feature + Downstream-Dauer + Sicherheit + ggf. aufgerundet auf Releasezeitpunkte sDBR: Leadtime halbieren; ein halben lead-time zurück von end planned load – das ist START, Lieferfrist ist dann eine halbe Leadtime nach planned load ende (gilt unter der Vorraussetzung: dass der Engpass ungefähr in der Mitte befindet)
4) Festlegung eine Reihenfolge – strategische Priorisierung	Durch den Product Owner auf Basis der Due-Dates und Busines-Case-Infos
5) Umgang mit Varianzen	Sicherheitspuffer für Due-Dates und Regelung auf 10% rot
6) operative Priorisierung – Ressourcenzuweisung	Anhand Anteil der verbrauchten Zeit von Start Bearbeitung bis Due-Date
7) transparenz – controlling - KVP	Anteil verbrauchter Zeit – Einteilung in grün/gelb/rot/schwarz – Regelung auf 30% rot und Tracking Zeitfresse mit Taskforces

Offene Fragestellungen

- Übersicht über Ähnlichkeiten/Differenzen sDBR, Kanban, Scrum, CCPM → erledigt s. Website
- wie wird der Prozess vor dem Product-Backlog gesteuert → muss ich noch beschreiben – ich arbeite hier mit einem Verfahren „Power Affinity Estimation“ von Patrick Seifried (den ich sehr schätze)
- Meßgröße aus Throughput-Accounting €Tage = opportunitätskosten des Product-Backlog
- Meßgröße Evolvierbarkeit = Verhältnis von internen zu externen Features
- insgesamt welche Meßgrößen müssen erhoben werden – Lead Times, Wartezeiten u.s.w.
- genau Definition Lead-Time … a) Lead-Time ab Übergang von Product-Backlog bis Auslieferung und b) Lead-Time inkl. Lagerzeit im Backlog! … s. €Tage
- agiles Manifest erweitern um „recht auf effizient wertschöpfende Arbeit“ u.a.
- super geniale Bezeichnung für das Ganze finden

Abgrenzungen zu Projekt und Lean Management

Die Abgrenzung der unterschiedlichen Ansätze erfolgt über zwei Achsen:

a) Anteil der Touch-Time zur Lead-Time. Also wie viel % der Durchlaufzeit wird effektiv an dem zu liefernden Artefakt gearbeitet.

Produktion < 10% Touch-Time/Lead-Time

Projekt oder Entwicklung > 20% Touch-Time/Lead-Time

Dazwischen ist ein Graubereich in dem beides der Fall sein kann.

Produktion ist einfacher zu managen, da durch Reihenfolgenvertauschung der Aufträge Terminverzüge einfacher zu kompensieren sind. Daher kommen die Lean Ansätze auch alle aus der Produktion. In der Entwicklungs- und Projektwelt muss explizit die Abhängigkeiten betrachtet werden und mit Puffern gearbeitet werden - der Ansatz hierzu heißt Critical-Chain-Project-Management (CCPM http://de.wikipedia.org/wiki/Critical-Chain-Projektmanagemen...)

b) Stabilität des Umfeldes. Hierbei gibt es mehrere Varianten: Stabilität des Demand/Bedarf, Stabilität der Prozesse und Stabilität des Ressourcenbedarfs pro Produkt

Wenn alles stabil ist dann kann man mit einfachen Fließbandfertigungen (Steuerung über den Platz zwischen zwei Arbeitsstationen) den Fluss optimieren (Erfinder Henry Ford).

Wenn das Produkt sehr vielfältig wird (wie bei heutigen Automobilen) dann eignet sich Kanban (Steuerung über Bestand zwischen zwei Arbeitsstationen) - Erfinder Taiichi Ohno. Demand und Prozesse bleiben aber stabil. Das war die Wurzel von Lean und heißt heute Toyota-Production-System, was aber weit über Kanban hinaus geht.

Wenn alles instabil ist dann kommt typischerweise eine Steuerung über die Zeit zum Einsatz. Das heisst dann Simplified-Drum-Buffer-Rope (sDBR) und wurde von Elyahu Goldratt beschrieben.

Abgrenzung Projekt gegen Produktion

Projektgeschäft und Produktion sind fundamental unterschiedlich – sie differenzieren sich anhand mehrerer zu beachtender Aspekte:

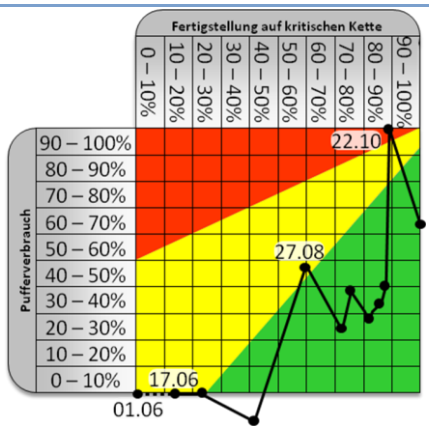
Aspekt	Projekt	Produktion
Touch-Time (das Verhältnis der Zeit an dem am Produkt gearbeitet wird zur gesamten Durchlaufzeit)	>>20%	<10%, der Vorgang wartet lange bei kurzer Bearbeitungszeit (oft <1%)
Vorgangsdauern	Immer wieder neue Vorgänge, hohe Unsicherheit, sehr hohe Streuung z.T. Faktor 2-3, große	Große Menge an gleichartigen Vorgängen, gutes Wissen über Dauer, erträgliche Streuung

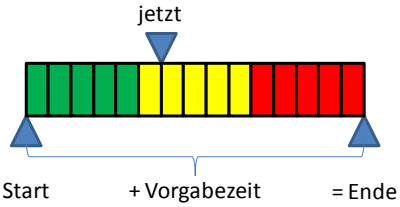
	Risiken	
Komplexität	Netzwerk von Vorgängen mit unterschiedlichen Arbeitsstationen und mit unterschiedlichen Abhängigkeiten	Typische Reihenfolgen von Arbeitsstationen
Ziel	Die Projekte werden alle früher oder genau zur geplanten Zeit fertig	Die Vorgänge werden alle schneller oder genau zu einer vorgegebenen Dauer fertig
Typ. Art der Steuerung	Ressourcen**re**allokation auf kritische Vorgänge	Reihenfolgenvertauschung zur Bearbeitung kritischer Vorgänge
bekannte Steuerungen (u.a.)	Critical-Chain-Project-Management (CCPM), simplified CCPM, Scrum, Scrumban	Kanban, Drum-Buffer-Rope (DBR), simplified DBR (SDBR), Dynamic-Buffer-Management (DBM)
typisch für	Entwicklung	Betrieb

Im Übergangsbereich kann es notwendig sein nahtlos von einer Steuerung in die anderen zu wechseln.

Operative Steuerung

	Projekt	Produktion
Ziel	Alle Projekte sind pünktlich	Alle Vorgänge sind pünktlich
operative Steuerung (Ampelfarben)	Jedes Projekt hat einen Puffer (ca. 1/3 der geplanten Durchlaufzeit). Betrachtet werden der Fortschritt auf der kritischen Kette sowie der Pufferverbrauch.	Für jede Vorgangsart wird eine gewünschte Durchlaufzeit definiert (z.B. critical Bug 4h, major Bug 2 Tage, normal Bug 10 Tage). Betrachtet wird wie viel Zeit der vorgegebenen Durchlaufzeit (Vorgabezeit) schon vergangen ist.



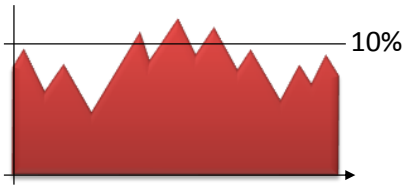


Rot	Pufferverbrauch ist höher als Fortschritt auf der kritischen Kette	>=2/3 der Vorgabezeit ist verstrichen
Gelb	Pufferverbrauch ist im Bereich des Fortschritts	>=1/3 der Vorgabezeit ist verstrichen
Grün	Fortschritt ist deutlich größer als Pufferverbrauch	<1/3 der Vorgabezeit ist verstrichen
Schwarz	Puffer ist vollständig verbraucht	Die komplette Vorgabezeit ist verstrichen
Operative Handlungsanweisung	**für beide gleich:** **schwarz vor rot vor gelb vor grün**	
Handlungsprinzip	Der Vorgang mit schlechtesten Verhältnis von Fortschritt zu Pufferverbrauch muss vorranging und so schnell wie möglich bearbeitet werden	Der Vorgang mit weitest fortgeschrittenen Vorgabezeit muss vorrangig bearbeitet werden

Durch die identische operative Handlungsanweisung können in einer Arbeitsstation auch Vorgänge aus dem Projektegeschäft und der Produktion gemischt bearbeitet werden ohne sich gegenseitig zu kanibalisieren.

Strategische Steuerung

	Projekt	Produktion
Ziel	>95% der geplanten Termine	>95% der Vorgabezeiten

	werden gehalten	werden gehalten
Primäre Regelung (zu viel Rot)	Work-In-Progress: das Eingangstor (Projektfreigabe) wird so eingestellt, dass ca. 10% der Projekte „rot“ sind.  Mittel- und Langfristig werden die Ressourcen angepasst.	Ressourcen: die Ressourcen zu Bearbeitung der Vorgänge werden so eingestellt, dass ca. 30% der Vorgänge „rot“ sind
Sekundäre Regelung (zu wenig rot)	Wenn zu viele Projekte grün sind werden Maßnahmen zur Verkürzung der geplanten Durchlaufzeiten ergriffen	Die Vorgabezeiten werden verkürzt oder Ressourcen werden verlagert/freigegeben

Die strategische Steuerung ist in beiden Fällen praktisch identisch. Im Projektgeschäft ist die Steuerung des Eingangstors (Projektfreigabe) von großer Bedeutung, da die Ressourcenelastizität begrenzt ist. In der Produktion kann der Eingang von Vorgängen typischerweise nicht beeinflusst werden – der korrekten Dimensionierung der Ressourcen bekommt daher eine zentrale Bedeutung.

Über die Speed4Projects.Net

Unternehmensgeschichte

Die Speed4Projects wurde im Jahr 2006 von Wolfram Müller gegründet und bietet seither Beratung zur Einführung von Critical-Chain-Projektmanagement und High-Speed-Projektmanagement an.

Wolfram Müllers langjährige Tätigkeit im Bereich Entwicklung und Projektmanagement in unterschiedlichsten Unternehmen hatte ihn mit einer Vielzahl unterschiedlicher Ansätze des Projektmanagements in Berührung gebracht. Viele dieser Ansätze hat er scheitern sehen und erst die Methoden rund um Critical Chain und High-Speed-Projektmanagement brachten sichtbare Erfolge. In Zusammenarbeit mit der Firma VISTEM konnte

Erfahrung aus über 530 Projekten in Bereichen:

- > Softwareentwicklung
- > Medizintechnik
- > IT-Projekte
- > Organisationsprojekte

Für namhafte Unternehmen:

- > BOSCH
- > AT&T
- > BARD/angiomed
- > Porsche
- > S+P, web.de, GMX, 1&1

Wolfram Müller diesen Ansatz in vielen Unternehmen einführen. Im Bereich Erfolgsgeschichten finden Sie Stimmen der Kunden, die dies bestätigen.

Philosophie

Mit pragmatischen Ansätzen schnelle sichtbare Erfolge erzielen, das ist der Grundgedanke von Speed4Projects. Nicht von ungefähr kommt es, dass die Wahl des passenden Instruments auf die Methode Critical-Chain und High-Speed-Projektmanagement fiel. Viele Projektmanagement-Methoden setzen voraus, dass eine Firma ihre gesamten Prozesse umstellt und sich zusätzlich noch teure Software anschafft. Speed4Projects ermöglicht es Ihnen mit Ihrer bestehenden Projektmanagement-Infrastruktur weiter zu arbeiten und durch gezielte punktuelle Optimierung Ihre Projektsteuerung deutlich zu verbessern.

Geschäftsführung

Wolfram Müller, Jahrgang 1969, beschäftigte sich als Dipl.-Ing. Mechatronik und Dipl.-Ing. Maschinenbau zunächst mit Themen der Entwicklung, Fertigung und Prozessoptimierung. Die Werkzeuge des klassischen Projektmanagements lernte er im Rahmen seiner Tätigkeit als Projektmanager in der Medizintechnik kennen. Seit 1987 und parallel zum Studium entdeckte er als Freelancer in zahlreichen Software-Entwicklungsprojekten den Spaß an schnellen Projekten. Bis heute konnte er beide Seiten erst als Entwickler und später als Manager des Project Office der 1&1 Internet AG (mit ihren Marken Schlund+Partner, GMX sowie web.de) ausleben. Seit 2006 steht seine Erfahrung im Critical-Chain und High-Speed-Projektmanagement in Vorträgen, Veröffentlichungen und vor allem in Form von Beratung zur Einführung jedermann, zu Verfügung.

Wenn sie Fragen haben:

Telefon: **+49 171 565 1821**

eMail: Wolfram.Mueller@Speed4Projects.Net