Dirk M. Sohn

(dirk.sohn@oio.de)
ist einer der Gründer der OIO Orientation in Objects GmbH in Mannheim und sammelte langjährige Erfahrung als Projektleiter, Berater agiler Methodik und Personal-disponent.

Agile Ressourcenplanung

Klassisches Projektmanagement kümmert sich als eigenständige Disziplin schon seit über 50 Jahren um die Planung und Steuerung von Mitarbeitern und "anderen Ressourcen". Im Kreise von Experten für Agilität wird allerdings ein Diskussionsbeitrag mit dem Namen "Agile Ressourcenplanung" für Stirnrunzeln sorgen. Schlanke Vorschriften, hohe Eigenverantwortung, räumliche Nähe aller Vollzeit-Teammitglieder eines kleinen Teams und „haptische Wahrnehmung" von Aufgaben als Kärtchen sind Beispiele für Ideen und Werte, mit denen enorme Verbesserungen von Produktivität und Zufriedenheit aller Parteien erzielt wurden. Der Artikel zeigt an einigen einfachen Beispielen aus der Praxis auf, warum sich aus unserer Sicht agiles Vorgehen und Personaleinsatzplanung zusammenraufen müssen.

„Estimation is waste"

Dies ist wohl einer der radikalsten, seit Jahren diskutierten und auch hofierten kernigen Sätze in der agilen Szene. Wer dies nicht glaubt, der versuche nur einmal, in dieser Szene einen Beitrag mit dem Namen „Agile Ressourcenplanung" zu platzieren.

Da liegt man in doppelter Hinsicht im Kreuzfeuer: Zum einen ist es dort, wo „agil" draufsteht, wohl ein Tabu von „Ressourcen" zu sprechen, wenn man „Mensch" meint. Und zum anderen macht man sich eben auch mit der Nennung des Begriffs „Planung" zumindest eines Anfangsverdachts schuldig, in diesem Umfeld nicht ganz auf dem Laufenden zu sein.

Bei meiner Suche nach bekannten und öffentlich zugänglichen Ursprüngen dieser Assoziation von Verschwendung mit dem Begriff Planung verbanden sich Elemente des berühmten Toyota-Produktionssystems (TPS) aus den 1970ern mit Veröffentlichungen von David J. Anderson über das japanische Wort „muda" (engl. waste, deutsch Verschwendung). So schreibt Toyota einem der Väter des TPS, Taiichi Ohno, das Zitat zu: „Eliminate muda, mura, muri completely" [Toyo04].

Und ich vermute, dass die Veröffentlichungen von David J. Anderson – nicht zuletzt seine Blogbeiträge „Why Estimates are Muda" [And05(1)] und "Stop Estimating" [And05(2)] – dazu beigetragen haben, dass die Schätzung von Aufwand im Bereich der Softwareentwicklung ihren bis dato postulierten Selbstzweck verloren hat und nun nur noch nach entsprechender Begründung gerechtfertigt ist. Eine ähnliche Aussage findet sich auch in jüngster Vergangenheit im Blog von Ken Schwaber:

I suspect, unless convinced otherwise, that any time we spend worrying about velocity or capacity is waste, not adding a whit of value [Schw12].

Das Motiv der Kernigkeit von Anderson's Aussagen lag allerdings schon damals hauptsächlich darin, die Aufmerksamkeit seiner Leser zu erreichen:

So a few of you have realized that I was being mildly flippant (well perhaps more than mildly) when I suggested that you should STOP ESTIMATING! But I hope I got your attention [And05(03)].

Ebenso mag es sieben Jahre danach umgekehrt einigen Lesern unseres Titels „Agile Ressourcenplanung" ergehen – jedenfalls hoffen wir genau das. Zufällig hat Anderson genau jetzt im September 2012 seine damaligen Aussagen neu erläutert:

It's at the root of the myth that you don't estimate in Kanban. Such dogmatic, context free guidance has never been part of Kanban. The truth is that effective Kanban implementations will have people who make considered decisions about the economic and risk management value of estimating and decide whether or not it is appropriate for any given type of work or class of service [And12].

Entlang dieser schönen Erklärung eines Grundgedankens von Agilität, nämlich nicht blind auf Vorgaben und Mythen zu hören, sondern genau hinzuschauen, was man braucht, möchten wir im Folgenden einige Entscheidungsgrundlagen unserer Art der „agilen Ressourcenplanung" darlegen und erläutern, warum wir es für angebracht halten, so vorzugehen.

Hierzu weisen wir kurz auf grundlegende Quellen zum Thema Werte und Arbeitszeit hin, erläutern unser Verständnis der Arbeitsweise eines idealen Scrum-Teams, bilden unsere Arbeitswirklichkeit darauf ab, erläutern sodann auf dieser Basis den Kern unseres Ansatzes und schließen mit einer Auflistung von Rahmenbedingungen, unter

denen wir diesen Ansatz für überflüssig halten.

Agilität, Werte und Arbeitszeitgesetz

Der kommerzielle Erfolg von Scrum hat dem Interesse an Agilität zum endgültigen Durchbruch verholfen. Mittlerweile ist aus einer Bewegung „vom Entwickler für den Entwickler“ namens Extreme Programming mit einem starken Fokus auf Werte und einem kleinen alten Methodenkoffer namens Scrum aus dem letzten Jahrtausend ein Buzzword-Bingo geworden, an dem auch das höchste Management und die einschlägigen wirtschaftlichen Pressenachrichten nicht mehr vorbeikommen.

So kommentierte beispielsweise Focus im Mai 2012 die entsprechenden Anstrengungen der SAP in einem Artikel „Das Comeback“ mit der Aussage: *Unter der Überschrift „Agiles Software Engineering“ stellt SAP das Programmieren grundlegend um – und gewinnt wieder an Schlagkraft* [Foc12].

Im englischen Ursprung steht *“to be agile”* für *“to be able to move quickly and easily“*. Die Verwendung im Kontext Softwareentwicklung entstammt dem sogenannten Agilen Manifest [AgM01]. Welche Kernaussagen lassen sich im Extreme Programming und im Agilen Manifest zu unserem Thema finden?

In der deutschsprachigen Ausgabe des Buches Extreme Programming von Kent Beck wird als eines der Hauptverfahren von Extreme Programming definiert: *40-Stunden-Woche – Man arbeitet prinzipiell nicht mehr als 40 Stunden in der Woche. Überstunden werden nie länger als eine Woche geleistet* (vgl. [Bec00], S. 54).

Auf Seite 60 derselben Veröffentlichung wird dies wie folgt erläutert und relativiert: *Ich möchte jeden Tag frisch und tatkräftig beginnen und müde und zufrieden beschließen. […] Ob sich diese Vorstellung in eine Arbeitswoche von genau 40 Stunden übersetzen lässt, ist nicht so wahnsinnig wichtig. […] Einer mag 35 Stunden konzentriert arbeiten können und ein anderer 45. Niemand kann jedoch über viele Wochen hinweg 60 Stunden in der Woche arbeiten und dann immer noch frisch, kreativ, sorgfältig und voller Selbstvertrauen sein. Arbeiten Sie nicht so.*

Hier geht es also auf dem knappen Platz einer möglichst kurzen Sammlung von wichtigen Verfahrensvorschriften für Softwareentwickler tatsächlich um die Festschreibung einer Arbeitszeit für mündige Entwickler! Gerne sei an dieser Stelle an die „Hours of Labour“ von Sidney Chapman erinnert (vgl. [Cha09]) – eine über 100 Jahre alte Erkenntnis, dass im freien Markt sowohl der Arbeitgeber wie der Arbeitnehmer dazu tendieren werden, einen Arbeitsvertrag zu schließen, der mehr als die optimal produktive Arbeitszeit vereinbart.

Beim Agilen Manifest steht zu diesem Thema: *Individuals and interactions over processes and tools* und außerdem in der seltener zitierten Unterseite mit den Prinzipien hinter dem Manifest: *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*

Man solle also, egal in welcher Geschwindigkeit man arbeitet, sicherstellen, dass man diese Geschwindigkeit „für immer“ beibehalten kann. Nicht zuletzt steht in in § 3 im deutschen Arbeitszeitgesetz:

Die werktägliche Arbeitszeit der Arbeitnehmer darf acht Stunden nicht überschreiten. Sie kann auf bis zu zehn Stunden nur verlängert werden, wenn innerhalb von sechs Kalendermonaten oder innerhalb von 24 Wochen im Durchschnitt acht Stunden werktäglich nicht überschritten werden [Arb12].

Hierzu muss man allerdings wissen, dass die Samstage immer noch zu den Werktagen zählen können. Hier wird also eine langfristige Wochenarbeitszeit von 48 Stunden angenommen, die sich kurzfristig auch bis zu 60 Stunden pro Woche ausdehnen kann – deutlich über den Vorgaben von Extreme Programming hinaus. Allen gemeinsam ist der Gedanke, dass es für Arbeitnehmer mehr als Selbstverantwortung braucht, um sich selbst optimal einzusetzen – es braucht Leitplanken.

Das ideale Scrum-Team

Ein notwendiger Anker für die Erklärung des Bedarfs an „agiler Ressourcenplanung“ ist ein gemeinsames Verständnis der Kernelemente von Scrum. Was wissen wir von den Lehrmeistern von Scrum über eine ideale Implementierung dieser kleinen Methode für das Management von Softareentwicklung?

Die Softwareentwicklung findet in zeitlich abgegrenzten Etappen (Sprints) oft konstanter Dauer von wenigen Wochen bis hin zu einer Woche statt. Die häufigste Angabe zur Teamgröße ist 7 +/- 2 – je nach Interpretation rein für das Entwicklungsteam oder inklusive eines dem Team zugeordneten Vertreters der Business-Seite (Product Owner) und eines „Aufpassers“ für einen reibungslosen Scrum-Ablauf (Scrum Master).

Scrum lohnt sich also nicht für die Entwicklung im 2er-Team, Scrum funktioniert auch nicht mit 10 und mehr Entwicklern pro Team. Das Team sollte keine Fluktuation aufweisen und die Mitglieder des Entwicklungsteams sollten ein „Vollzeit-Engagement“ im Team haben. Übereinstimmend wird davon ausgegangen, dass das Team in Co-Location arbeitet – hierbei meint man weniger die Zeitzone oder die Stadt des Teams, sondern den gemeinsamen Arbeitsraum.

Das Team sollte in der Lage sein, „cross-funktional“ zu arbeiten, was bedeutet, dass es dem Team autark von anderen Personen möglich sein sollte, am Ende jedes Sprints fertige nützliche Software zu erstellen. Die Herausforderung liegt hier in den zahlreichen Qualifikationen, die hierfür von einem kleinen Team abgedeckt werden müssen

Das Entwicklungsteam soll seine Aufgaben selbstorganisiert an sich ziehen (Pull-Prinzip) und abwickeln, die Festlegung der Arbeitsinhalte einer Arbeitsetappe (Sprint) geschieht in gemeinschaftlich getragener Verantwortung aller Teammitglieder. Zu Beginn jedes Sprints wird hierfür im sogenannten „Sprint Planning Meeting“ der Inhalt des Sprints gemeinsam mit dem Product Owner festgelegt. Am Schluss jedes Sprints werden die Ergebnisse öffentlich präsentiert und teamintern eine Retrospektive durchgeführt.

Nun stellt sich planerisch die Frage: Woher weiß das Team in einem Sprint Planning Meeting, wie viele der Anforderungen im Rahmen des nächsten Sprints bearbeitet werden können? Wenn alle oben genannten Voraussetzungen gegeben sind, mag das Team ein paar wenige Sprints lang falsch liegen, aber dann (nach wenigen Wochen) wird es diese Frage immer besser beantworten können und somit immer verlässlicher am Ende eines Sprints das ausliefern, was am Anfang des Sprints geplant war.

Hierzu muss ein solches Team keine explizite Metrik entwickeln – ein so kon-

stantes, ortsgebundenes, langfristig voll engagiertes und hochvertrautes, hochqualifiziertes Team in gemeinsamer Verantwortung ist in der Lage, eine entsprechende Einplanung von Anforderungen sozusagen im kollektiven Unterbewusstsein korrekt vorzunehmen.

Hier endet die Darstellung des idealen Scrum-Teams. Ein paar kleinere Fallstricke aus der Praxiserfahrung unserer idealen Scrum-Teams sind: Ein 7-Personen-Team hat nicht immer den Beginn der 3-monatigen Elternzeit des einzigen Entwicklers, der sich mit JavaScript auskennt, im Unterbewusstsein; es sind nicht immer alle Personen zum Sprint Planning Meeting anwesend (Krankheit, Urlaube) und genau diese haben aber vergessen, dem Unterbewusstsein gewisse künftige Abwesenheiten zu übermitteln.

Kleineren Fallstricken könnte man mit kleineren taktischen Maßnahmen begegnen, nun beginnt aber erst der Katalog der ernsthaften Fragen, dem man sich dann stellen muss, wenn diese Bedingungen für ein ideales Scrum Team nicht stimmen – hier beschränkt auf Ressourcenplanung:

1. *Mangelnde Teamkonstanz:* Kann man die Projekte im Projektgeschäft (nicht die Entwicklungsaufgaben in der Produktentwicklung) tatsächlich so schneiden, dass Teams immer konstant beieinander bleiben?
2. *Fehlender Vollzeiteinsatz:* Kann man Skalierungsaufwand im Falle größerer Mannschaften („Scrum of Scrums“ / sonstige Querschnittsfunktionen wie Architektur / hier wird häufig zu einem Matrix-Ansatz geraten) abdecken, ohne die Grundidee des Vollzeiteinsatzes im Team zu kompromittieren?
3. *Fehlende Co-Location:* Kann man wirklich immer sicherstellen, dass zumindest alle Teammitglieder, wenn schon nicht alle Projektmitglieder, im selben Zimmer arbeiten?
4. *Falsches Geschäftsmodell:* Sind wirklich alle Entwickler ausschließlich mit dem aktuell zur Betrachtung stehenden Projekt in Vollzeit betraut und müssen keinerlei andere Arbeiten wahrnehmen, oder, falls doch, sind diese anderen Arbeiten zumindest sehr konstant in ihrer Belastung bezüglich der Arbeitszeit und -leistung?

Und wenn eine dieser Fragen mit „Nein“ beantwortet werden müsste, wie funktioniert dann noch das kollektive Unterbewusstsein?

Unser Hintergrund

Um es gleich vorwegzunehmen: Bei uns funktioniert das kollektive Unterbewusstsein nicht. Wir haben zumindest schon ein dickes „Nein“ bei Frage 4 anzukreuzen: OIO Orientation in Objects – seit 1998 Ihr Expertenhaus für Softwareentwicklung mit Java und XML – das ist in Kurzform seit fast 15 Jahren unsere Mission.

Was ist ein Expertenhaus und wie funktioniert so etwas bei der Softwareentwicklung? Schon der scheinbar kleine Bereich rund um die Entwicklung mit Java und XML hat aufgrund der Größe der Domäne Bedarf an Dutzenden von Experten: Experten für zahlreiche Technologiestandards wie beispielsweise JavaServer Faces und ihren scheinbar proprietären Konkurrenten Google Web Toolkit (GWT), Experten für agile Methodik wie z. B. Scrum und ihre scheinbaren Konkurrenten im klassischen Projektmanagement, Experten für hoch-komplexe Geräte wie beispielsweise einen Java Application Server, usw.

Wie entsteht die entsprechende Expertise? Nur im mehrjährigen zielgerichteten praktischen Handeln in der Materie unter gleichzeitiger Reflektion durch Weitergabe des Erlernten. Also nach unserer Überzeugung durch eine ideale Mischung von Einsätzen als Softwareentwickler, Berater und Trainer/Coach.

Kennt man diesen Hintergrund, so kann man sich vorstellen, dass unsere Entwicklungsteams praktisch nie mit Vollzeitentwicklern haushalten können. Nicht selten ist einer der Entwickler ein paar Tage abwesend, zum Beispiel in seiner Tätigkeit als Trainer, und zwar je Sprint unterschiedlich.

Und wenn man zusätzlich berücksichtigt, dass wir ein Dienstleister sind, so kommt hinzu, dass unsere Aufträge typischerweise Projektcharakter aufweisen, also einen klar abgrenzbaren zeitlichen und budgetierten Rahmen für eine recht einmalige Herausforderung aufweisen. Und nicht zuletzt brauchen unsere Experten Freiräume zur Forschung und Weiterbildung in ihrer Expertise, dies mit wechselnder Belastung. Also auch ein „Nein“ auf Frage 1.

Unser betrieblicher Hintergrund hält auf der anderen Seite aber auch ein paar Erleichterungen für agiles Handeln bereit: Unsere Herangehensweise an jedes Projekt muss so modern wie möglich sein, um einen maximalen Lerneffekt für unsere Expertisen zu erzielen, unsere Verhaftung mit Vorgaben an Werkzeuge und Technologien muss minimiert werden. Das regt zu Agilität im Sinne obiger Definition an: „To be able to move quickly and easily“.

In Anlehnung an erste Quellen von Extreme Programming haben wir 1998 die Grundwerte unseres Unternehmens als „Einfachheit, Mut, Kommunikation, Feedback und Respekt“ beschrieben. Man darf also als Hintergrund über uns auch noch wissen,

- wir seit Gründung versuchen „einfach mutig mit dem nötigen Respekt zueinander sagen, wenn uns etwas nicht gefällt,
- dass wir dies auch mit unseren Geschäftspartnern so pflegen,
- dass sich daraus im Innenverhältnis eine Kultur der Arbeitszeiterfassung ergab – unser Kollegium kennt sein Arbeitszeitkonto und geht davon aus, dass die im Arbeitsvertrag individuell festgelegte Arbeitszeit genau diejenige ist, die jeder für sich langfristig gesehen anbieten kann,
- dass das Verfehlen von Schätzwerten ein Angebot zum Lernen ist und nicht ein Mittel zu dem Zweck, Scheitern zu betonen,
- dass unsere Kunden davon ausgehen dürfen, dass wir uns nur Dinge vornehmen, die wir auch in diesem Rahmen bewältigen können und darüber hinaus auch bereit sind zu erklären, warum bestimmte Dinge für uns nicht möglich sind.

Agile Ressourcenplanung

Auf dieser Basis ist unsere Idee der „agilen Ressourcenplanung“ eigentlich schnell erklärt. Die Abwicklung von Softwareentwicklung findet generell in einem Issue Tracker statt (hier: Atlassian Jira) – falls haptisch greifbare Wandkarten bevorzugt werden, steht einem Ausdruck nichts im Wege. Es muss jedoch jeden Abend die verbrauchte Zeit je Issue und der geschätzte Restaufwand je aktivem Issue erfasst werden.

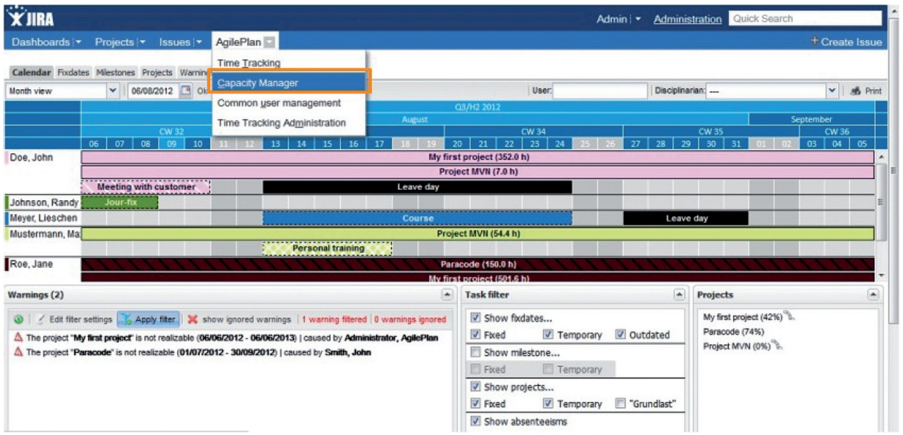
Über die bekannten Scrum-Rollen hinaus haben wir zusätzlich die Rolle „Disponent“

eingeführt. Jeder Mitarbeiter des Unternehmens wird von genau einem Disponenten verplant. Diese halten in ein und demselben zentralen Werkzeug (ein von OIO entwickeltes und vertriebenes Jira Plugin „Braintime AgilePlan“ (vgl. [Agi11])) die entsprechenden Einsatzpläne für ihren Pool vor.

Die Granularität ist entweder „ein ganzer Sondertag“, wie beispielsweise Urlaube und sonstige, einen Projekteinsatz verhindernden Sachverhalte (Krankheiten, Schulungen als Teilnehmer oder Trainer, …) oder eine prozentuale Zuweisung der verbleibenden Zeit auf ein oder potenziell mehrere Projektabschnitte (typischerweise Sprints).

Agile Plandaten sind allen Beteiligten im notwendigen Umfang in Form einer Webanwendung zugänglich und ständig aktuell (das System ist verbunden mit der Schulungseinsatzplanung und der Abwesenheitsverwaltung, ansonsten ist der Disponent ohnehin letzte Instanz). Für die grob geschätzten Backlogs von Softwareprojekten hält der Disponent vorläufige Kapazität der beteiligten Entwickler möglichst langfristig vor, um für eine möglichst stabile Teamzusammensetzung zu sorgen. Auf der Basis dieses Wissens erfolgt je Sprint folgende Feinplanung:

- **Schritt 1:** Jedes Teammitglied stellt vor Beginn jedes Sprints auf einer Team-Wiki-Seite seine erhoffte Kapazität je Sprint und deren Grundlagen bereit (Arbeitszeit minus bekannte Abwesenheiten minus persönlicher Puffer für die Geburtstagsfeier der Patentante in Gleitzeit). Jeder kann dabei auf seine „offiziellen Verplanungen“ in AgilePlan zugreifen und diese dadurch auch möglicherweise hinterfragen. Dieser Schritt dauert wenige Minuten und ermittelt in Summe eine Team-Kapazität für den Sprint.
- **Schritt 2:** Der „normale“ Scrum-Ablauf eines Sprint Planning Meetings findet statt – das Team plausibilisiert die akzeptierten User Stories bis die auf Ebene geschätzter Aufgaben gegen die ermittelte Teamkapazität durch. Die hierbei geschätzten Aufgaben sind selten kleiner als zwei Stunden und selten größer als zwei Tage – größerer Aufwand deutet bei uns zumeist auf ein noch recht unverstandenes Arbeitspaket hin, das nicht in einen Sprint aufgenommen werden sollte. Die zusätzlichen Kosten gegenüber einem rein User Story / Feature Point / Business Value getriebenen Ansatz sind wenige Stunden für das ganze Team pro Sprint und erhöhen das Teamverständnis über das Sprint Goal drastisch.
- **Schritt 3:** Sollten dabei Ideen zur Abweichung vom gedachten Dispositionsrahmen entstehen (zu viel oder zu wenig Kapazität), werden diese in einem nächsten Schritt dem Disponenten zugetragen. Dieser muss jederzeit in der Lage sein „übrige Kapazität“ sinnvoll aufzunehmen (dies gelingt in einer lernenden Organisation geradezu spielend) und kann bisweilen auch noch zusätzliche Kapazität anbieten. Hierbei ist es von großer Hilfe, wenn die Kapazitätsanfragen aller an einer Person interessierten Parteien gleichzeitig auflaufen, wir versuchen dies durch unternehmensweit synchrones Timeboxing zu unterstützen. Auch dieser Schritt ist in wenigen Minuten erledigt.
- **Schritt 4:** Auf der Basis der endgültig abgestimmten Kapazität wird das Sprint Goal fixiert.



OIO Jira Plugin für Kapazitätsplanung.

Warum nennen wir diesen Vorgang agile Ressourcenplanung? Weil wir keinen schlankeren Prozess finden konnten, um eine von den Mitarbeitern geforderte also notwendige Frage zu beantworten: Welche meiner zeitlichen Ressourcen soll ich grob in welche Richtung lenken und wie viel Freiheitsgrade würden mir dabei bleiben. Und weil wir hierbei nicht auf mögliche Vorgaben möglicher Prozessexperten gehört haben, und seien diese auch aus dem Umfeld der Agilität, sondern genau das minimal umgesetzt haben, was Not tut.

Ergebnisse und Übertragbarkeit

Sie brauchen aus unserer Sicht keine agile Ressourcenplanung, wenn Sie über die Rahmenbedingungen für ein wie oben beschriebenes ideales Scrum-Team verfügen, am besten nicht im Projektgeschäft sind, sondern über den langen Atem der Produktentwicklung verfügen, und wenn Sie die Mitarbeiterlaufbahn-Entwicklung in Aufgaben innerhalb des Teams leisten.

Es kann auch sein, dass Sie die Idee zwar für interessant aber nicht auf Ihr Unternehmen übertragbar halten – und zwar weil der Wertehintergrund nicht stimmt. Beispielweise könnte die personenbezogene Zeiterfassung auf Arbeitspakete vor dem Hintergrund der betrieblichen Mitbestimmung in Deutschland bei mangelndem Verständnis des Grundes auf Seite der Arbeitnehmervertreter durch den Betriebsrat abgelehnt werden.

Das Betriebsverfassungsgesetz regelt in § 87 „Mitbestimmungsrechte“ (BetrVG (vgl. [Bet])) die Mitbestimmung des Betriebsrats bei *„Einführung und Anwendung von technischen Einrichtungen, die dazu bestimmt sind, das Verhalten oder die Leistung der Arbeitnehmer zu überwachen“*. Allerdings würde ja eine solche technische Einrichtung gar nicht zu dieser Bestimmung eingeführt werden, und selbst wenn, dürfte der Betriebsrat trotzdem zustimmen, wenn er vom Nutzen überzeugt wäre.

Man muss sich auch im Klaren sein, dass im Falle solcher Probleme die Einführung von Scrum generell zur Disposition steht, denn die Mitbestimmungsrechte betreffen

auch die *„Grundsätze über die Durchführung von Gruppenarbeit; Gruppenarbeit im Sinne dieser Vorschrift liegt vor, wenn im Rahmen des betrieblichen Arbeitsablaufs eine Gruppe von Arbeitnehmern eine ihr übertragene Gesamtaufgabe im Wesentlichen eigenverantwortlich erledigt.“*

Wenn Sie in dieser Hinsicht Interesse haben: Wir haben schon Konzernbetriebsräte entsprechend aufgeklärt und damit zu einer positiven Entscheidung beitragen dürfen – manchmal finden es Arbeitnehmervertreter sehr nützlich, wenn Überplanungen transparent werden. Auch könnte der Vortrag (Betriebsverfassungsgesetz und Scrum, 2012) von den Scrum Days 2012 von der Deutschen Telekom von Interesse für Sie sein.

In allen anderen Fällen sehen wir die folgenden Argumente für eine solche Lösung sprechen:

- Schätzungen sind wertvoll! Wenn Schätzungen schwer fallen und viel Zeit kosten, dann zeigt dies ein mangelndes gemeinsames Verständnis – seien es unklare Anforderungen oder eine schlechte Architektur – und die entstehenden Arbeiten sind kein „waste“, sondern sehr wichtig.
- Konstantes Feedback zum Ausgang von Schätzungen lehrt wechselseitiges Verständnis zwischen Auftraggeber und Auftragnehmer und sogar im Entwicklerteam – und darum geht es hauptsächlich bei Softwareentwicklung. Man braucht also auch Zeitmessungen. Experimentieren Sie gerne mit Story Points oder anderen Metriken. Aber Sie dürfen auch Stunden-basierte Schätzungen nutzen – diese sind eine ehrliche Währung. Jeder kennt sie und sie passt auch zu den arbeitsvertraglichen Pflichten.
- Transparente und verbindliche Zusagen der Ressource Arbeitszeit sind ebenso wichtig. Nur so kann man bei agilem Vorgehen Schwerpunkte, wie Arbeiten am konkreten Projekt, Arbeiten in zusätzlichen Arbeitsgruppen und das Verfolgen einer eigenen Mitarbeiterlaufbahn, für alle Beteiligten klar steuern.
- Unsere Vorschläge sind also die konsequente Schätzung und Zeiterfassung, die Einführung einer Rolle „Disponent“ und der für alle transparente Zugang auf eine zentral gepflegte Dispositionsplanung. Diese Lösung ist aus unserer Sicht minimal invasiv für notwendiges Lernen und Feedback und somit agil. ■

Literaturverzeichnis

[APW11] AgilePlan Website. (2011). Abgerufen am 17. September 2012 von http://agileplan.oio.de

[AgM01] Agiles Manifest. (2001). Abgerufen am 18. September 2012 von Manifesto for Agile Software Development: http://agilemanifesto.org/

[Arb12] Arbeitszeitgesetz (ArbZG). (08. September 2012). Abgerufen am 17. September 2012 von Juristischer Informationsdienst dejure.org: http://dejure.org/gesetze/ArbZG/3.html

[Bec00] Beck, K. (2000). Extreme Programming. München: Addison-Wesley Verlag.

[Bet] www.scrum-day.de/vortraege/betriebsverfassungsgesetzscrum.html.

[Bet12(1)] Betriebsverfassungsgesetz und Scrum. (2012). Scrum Days 2012. http://www.scrum-day.de.

[Bet12(2)] BetrVG. (18. September 2012). Abgerufen am 18. September 2012 von http://dejure.org/gesetze/BetrVG/87.html

[And05(1)] Blog David J. Anderson. (27. September 2005). Abgerufen am 17. September 2012 von Why Estimates are Muda: http://www.agilemanagement.net/index.php/blog/Why_Estimates_are_Muda

[And05(2)] Blog David J. Anderson. (13. März 2005). Abgerufen am 17. September 2012 von Stop Estimating: http://www.agilemanagement.net/index.php/blog/Stop_Estimating/

[And05(3)] Blog David J. Anderson. (17. März 2005). Abgerufen am 17. September 2012 von Agile Estimating: http://agilemanagement.net/index.php/site/comments/agile_estimating/

[And12] Blog David J. Anderson. (09. September 2012). Abgerufen am 17. September 2012 von A Taste of Lessons #11: Agile Practices: http://agilemanagement.net/index.php/Blog/a_taste_of_lessons_11_agile_practices/

[Schw12] (25. Mai 2012). Abgerufen am 17. September 2012 von http://kenschwaber.wordpress.com/2012/05/25/waste-not/

[Foc12] Focus Online. (26. Mai 2012). Abgerufen am 17. September 2012 von SAP Das Comeback: http://www.focus.de/finanzen/boerse/aktien/tid-26061/wirtschaft-das-comeback_aid_757907.html

[Toyo4] Toyota Traditions. (Juli 2004). Abgerufen am 17. September 2012 von Toyota Website: http://www.toyota-global.com/company/toyota_traditions/quality/jul_aug_2004.html

[Cha09] Wikipedia - Sydney Chapman. (September 1909). Abgerufen am 17. September 2012 von Hours of Labour: http://en.wikipedia.org/wiki/Sydney_Chapman_%28economist%29