Projektmanagement: Schlagkräftigere Projektorganisation durch „agilen Anforderungsmanager“

Andrej Loncar, Senior Manager, aloncar@it-economics.de
Daniel Reis, Manager, dreis@it-economics.de
it-economics GmbH, Steinstr. 52, 81667 München

Einleitung

Klassisches Projektmanagement hat zum Ziel, die Umsetzung von Projekten innerhalb vorgegebener Rahmenbedingungen wie Termin-, Ressourcen- und Budgetvorgaben planbar, kalkulierbar und kontrollierbar zu machen. In systematisierter und wissenschaftlich untermauerter Form gibt es solche Verfahren in der IT schon seit den 50er Jahren, ab Ende der 90er Jahre entstanden agile Methoden wie Extreme Programming (Beck, 1999) oder Scrum (Schwaber & Beedle, 2002). Sie alle setzen auf eine klar definierte Rollenverteilung. So kennen die klassischen Methoden zum Beispiel für die Aufnahme der Anforderungen an das zu erstellende Produkt einen „Requirements Engineer“ (RE). Bei agilen Methoden ist dies die Aufgabe des „Product Owners“ (PO). Im folgenden Artikel möchten wir aufzeigen, welche Vorteile im Vergleich dazu das Konzept eines „agilen Anforderungsmanagers“ bietet. Er basiert auf Beobachtungen und Erfahrungen, die unsere Berater bei der Einführung agiler Methoden bei mittelständischen Kunden aber auch Konzernen, insbesondere im Finanzbereich, sammelten.

Vom Requirements Engineer zum agilen Product Owner

In der von uns beobachteten und zum Teil begleiteten Projektpraxis beschränkt sich die Rolle des klassischen Requirements Engineers in linearen Projektorganisationen wie zum Beispiel dem traditionellen Wasserfallmodell (Benington, 1983) als auch in iterativen Vorgehensmodellen wie etwa dem Rational Unified Process (Kruchten, 2003), meist auf folgende Tätigkeiten:

- Erstellung von Anforderungsdokumenten/Spezifikationen in Abstimmung mit anderen Beteiligten („Stakeholdern“)
- Anpassung dieser Dokumente aufgrund von Fragen, die während der Implementierung auftreten
- Unterstützung in System- oder Abnahmetests

Dies führt oftmals zu Problemen, insbesondere:

- Die Kommunikation mit dem Implementierungsteam ist dokumentenbasiert und damit anfällig für Missverständnisse – abhängig vom Grad der Komplexität, Abstraktionsfähigkeit sowie Erfahrungen und Kenntnissen
- Lange Entscheidungswege, da der RE oftmals zu wenig Autorität hat, um selbst Entscheidungen zu treffen
- Aufwand abhängig von der Projektphase. Dies führt häufig dazu, dass während der Implementierung das Team der REs verkleinert wird („Brain-Drain“) oder sich um andere Projekte kümmern muss („Context Switching“)
- Überforderung der Fachbereiche in der Designphase, weil deren Mitglieder bezüglich Verfügbarkeit und Abstraktionsvermögen zu stark beansprucht werden
- Überzogene Anforderungen, die nicht praxisrelevant sind oder später nicht wirklich genutzt werden

Dies und die strikten Timelines in der klassischen Projektorganisation verleiten den RE in der Praxis häufig dazu, sein Hauptaugenmerk auf den Sign-Off von Anforderungen/ Spezifikationen zu legen, und nicht auf die effiziente Umsetzbarkeit der Anforderungen.

Im Gegensatz dazu hat der Product Owner (PO) anders gelagerte Verantwortungen:

- Erarbeiten und Kommunikation einer Produktvision, die dem Team einen Überblick über die Anforderungen gibt
- Definition der Anforderungen über „Themen“ (allgemeine Beschreibung der Anforderung), „Epics“ (Präzisierung der Beschreibung aus den „Themen“, aber immer noch auf einer höheren Abstraktionsebene) und „User Stories“ (konkrete Präzisierung mit Blick auf die Implementierung)
- Festlegen von Prioritäten und Reihenfolge bei der Implementierung
- Abnahme der implementierten „User Stories“

Somit ist der Product Owner maßgeblich dafür verantwortlich, Funktionalitäten zu definieren und diese dem Entwicklerteam zu erklären. Hinzu kommt Verantwortung für die Release-Steuerung und die Maximierung des **R**eturn **o**n **I**nvestment (ROI). Dadurch bringt seine Rolle eine deutlich größere Gestaltungsverantwortung mit sich als die des RE.

Im Gegensatz zur klassischen Projektorganisation findet beim „agilen Projektmanagement“ schon die Planung unter etwas anderen Maximen statt:

- Die Planung sollte als kontinuierliche Aktivität verstanden werden. Als Werkzeug dazu dient das Führen eines priorisierten „Product Backlogs“
- Auch kurzfristige Änderungen müssen akzeptiert werden
- Auf die Ausarbeitung von Details verzichtet man, bis diese wirklich benötigt werden
- Die Planung soll auf das Nötige beschränkt werden.

Dieses Konzept führt dazu, dass Anforderungen in „Themen“ und „Epics“ nur mit einem sehr niedrigen Detailgrad festgehalten werden. Die Planung dieser Elemente soll gerade noch hinreichend sein, um eine Budget-, Zeit- und Ressourcenplanung zu Beginn des Projekts zu gewährleisten. Eine Detaillierung der Anforderungen erfolgt dann erst im Verlauf des Projekts („Just-in-Time-Philosophie“). Dies bringt folgende Vorteile mit sich:

- Kurzfristige Änderungen führen nicht direkt zu unnötig vorgenommenen Ausgaben, die aus alten Anforderungen resultieren („Sunk Costs“)
- Die Detaillierung in User Stories erfolgt erst dann, wenn ausreichende Informationen über die zu implementierende Funktionalität und das zu Grunde liegende System vorliegen.

Dabei muss man im Blick behalten, dass die „User Stories“ im Vergleich zu klassischen Anforderungsdokumenten und Systemspezifikationen einen deutlich niedrigeren Detailgrad haben. Sie stellen eigentlich nur ein Versprechen für eine kontinuierliche Kommunikation zwischen dem PO und dem Implementierungsteam dar. Dies ist sinnvoll, weil verbale (bidirektionale) Kommunikation zwischen den Beteiligten

einen effizienteren Informationsaustausch erlaubt als die unidirektionale Kommunikation über schriftliche Dokumente.

Da Planung und Detaillierung der Anforderungen kontinuierliche Aktivitäten sind, muss der PO dem Implementierungsteam durchgängig zur Verfügung stehen. Dies verhindert die bereits erwähnten Probleme „Brain-Drain" und „Context-Switching" aus klassischen Projektsituationen.

Allerdings erlaubt der agile Ansatz nur eingeschränkt einen Abgleich zwischen den Erwartungen der „Stakeholder" und der bereitzustellenden Funktionalität im Vorfeld der Implementierung. Deshalb wird die tatsächliche Implementierung im Rahmen von „Demo-Sessions" präsentiert, die im Abstand von wenigen Wochen stattfinden. Sie erlauben es, Abweichungen von den Kundenwünschen mit geringem Aufwand zu korrigieren.

Die Erfahrung zeigt, dass die Einführung agiler Methoden sehr gut bei kleineren Projekten funktioniert, in denen nur ein PO oder eine geringe Zahl von POs ein Produkt verantworten. Problematischer ist ihre Einführung bei Großprojekten – besonders in Fällen, bei denen kein verkaufsfähiges Produkt erstellt wird, sondern eine Software zum Beispiel von internen „Kunden" als Service zur Erledigung ihres Tagesgeschäfts eingesetzt wird.

Eine potentielle Schwachstelle von agilen Methoden besteht zudem darin, dass durch die Just-in-Time-Philosophie und die Fokussierung auf einzelne Teams der Blick auf das (komplexe) Gesamtprojekt verloren gehen kann – oder sich gar nicht erst entwickelt. Dies führt dann häufig dazu, dass die Implementierungsteams samt dem PO ihren Durchsatz an User Stories ausschließlich für ihr eigenes Team optimieren, ohne die Auswirkungen auf andere Teams im Blick zu behalten. Abstimmungsprobleme zwischen den Teams führen dann zu einer Verringerung der Softwarequalität und/oder einer Erhöhung der Entwicklungskosten – obwohl die Einführung agiler Methoden dies ja gerade verhindern soll.

Dieser Effekt tritt verstärkt in Organisationen mit gewachsenen IT-Systemlandschaften auf, in denen ein PO mit „seinem" Implementierungsteam eine Funktionalität nicht vollumfänglich betreuen kann. Da der agile Ansatz maßgeblich auf Vertrauen aller Beteiligten zueinander basiert, führt eine solche lokale Optimierung langfristig zu einem Vertrauensverlust aller Beteiligten und damit zu einem Scheitern der agilen Vorgehensweise.

Der agile Anforderungsmanager

Die gerade aufgeführten negativen Effekte bei agilen Großprojekten lassen sich durch eine Kombination agiler Vorgehensweisen mit Methoden aus dem klassischen Requirements Engineering beheben.

Dazu erfolgt zu Beginn des Projekts eine klassische Anforderungsanalyse, in der alle Anforderungen auf der Detailebene von agilen Themen aufgenommen werden. Daran schließt sich eine Zuordnung der Themen/ Anforderungen zu den Produkt- oder Systemkomponenten an. Sie dient als Basis für eine Grobschätzung der „Themen" sowie zur Identifikation von Lücken in der Themenbeschreibung.

Sofern möglich, sollte dann ein agiler Anforderungsmanager mit seinem Team in die Lage versetzt werden, jeweils komplette Anforderungen zu implementieren. Falls dies aufgrund der Organisationsstruktur nicht möglich ist, sollten die Aufgabenbereiche so zugeschnitten werden, dass die Anzahl der Schnittstellen möglichst gering gehalten wird.

Um die Arbeit der Teams möglichst stark voneinander zu entkoppeln, sollte die Schnittstellenfunktionalität über automatisierte „Interface Contract Tests" festgelegt werden. Dies erlaubt den einzelnen Teams, ihre Änderungen relativ unabhängig voneinander zu liefern.

Der agile Anforderungsmanager agiert in diesem Kontext nicht nur als PO für sein Team, sondern ist wiederum Teil eines agilen Teams von Anforderungsmanagern. Dieses Team entwickelt gemeinsam Funktionalitäten und plant unter Einbindung der Entwicklungsteams High-Level-Meilensteine. Gegenüber seinem Implementierungsteam tritt der agile Anforderungsmanager wie ein Product Owner auf – und hat somit im Vergleich zum klassischen Requirements Engineer einen größeren Gestaltungsspielraum.

Das Team von agilen Anforderungsmanagern wird wiederum von dem Projektleiter oder dem Leiter des Anforderungsteams gesteuert. Letztere Rolle lässt sich als „Chief Product Owner" (CPO) definieren. Wie ein einzelner PO ist der CPO auch für die Releasesteuerung und die Maximierung des ROI verantwortlich. Er fokussiert seine Aufmerksamkeit jedoch auf das Gesamtprojekt und dabei insbesondere auf teamübergreifende Anforderungen auf der Ebene von „Themen" und „Epics". Im Gegensatz zu den „User Stories" der Implementierungsteams wird hierbei keine Software entwickelt. Vielmehr zerlegen die betroffenen agilen Anforderungsmanager teamübergreifende Anforderungen in getrennte Epics und User Stories, von denen jede einzelne später von genau einem Implementierungsteam umgesetzt werden kann.

Eine solche – möglichst weitgehende – Entkopplung der Implementierungsteams bei gleichzeitiger Verzahnung und Steuerung der agilen Anforderungsmanager verstärkt das Vertrauen zwischen den Teams und den agilen Anforderungsmanagern, da Abstimmungsprobleme zwischen den Teams minimiert werden. Dies wiederum schafft die Basis für eine erfolgreiche Umsetzung agiler Großprojekte. Dass der beschriebene Ansatz zu positiven Ergebnissen führt, konnten die Autoren im Projekteinsatz bei unseren Kunden feststellen.

Literaturverzeichnis

Beck, K. (1999). *Extreme Programming Explained: Embrace Change (1st Edition).* Addison Wesley.

Benington, H. D. (1. Oktober 1983). Production of Large Computer Programs. *IEEE Annals of the History of Computing (IEEE Educational Activities Department) 5 (4)* , S. 350 - 361.

Kruchten, P. (2003). *The Rational Unified Process: An Introduction (3rd Edition)* . Addison-Wesley.

Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum.* Prentice Hall.