Stefan Luckhaus

Aufwandsschätzungen in der agilen Softwareentwicklung

Einsatz von Methoden zur Messung des funktionalen Umfangs

tredition

Druck und Distribution im Auftrag des Autors:
tredition GmbH, Halenreie 40-44, 22359 Hamburg, Deutschland

ISBN
Softcover 978-3-732-36593-7
Hardcover 978-3-732-36594-4
E-Book 978-3-732-36595-1

Inhaltsverzeichnis

Einleitung

Indirekte Aufwandsschätzungen, bei denen ein methodisch nach festen Regeln ermittelter Wert für den geplanten Entwicklungsumfang mit einem präzise gemessenen Erfahrungswert für die eigene Produktivität ins Verhältnis gesetzt wird, sind eine bewährte Methode zur verlässlichen Planung von Softwareentwicklungsprojekten. Ihre Anwendung erfordert jedoch einen minimalen Spezifikationsgrad, das heißt aus einem Satz formulierte User Stories müssen durch Anwendungsfälle und Elementarprozesse präzisiert werden. Danach ist ihre „Vermessung" möglich – und sie erfordert bei entsprechenden Methodenkenntnissen keinen großen Aufwand.

Dieses Buch beschreibt in kurzen Ausführungen und basierend auf eigenen praktischen Erfahrungen des Autors die Grundlagen methodischer Aufwandsschätzungen. Es zeigt, dass diese Vorgehensweise nicht nur gut mit agiler Softwareentwicklung vereinbar ist, sondern gerade Grundprinzipien wie beispielsweise

- die flexible Berücksichtigung geänderter Anforderungen oder
- regelmäßige Verbesserungen als Folge von Nachbetrachtungen

unterstützt.

Merkmale und Bedeutung agiler Softwareentwicklung

Entstehungsgeschichte

Anfang der 90er Jahre gerieten viele Großprojekte aufgrund ihrer langen Prozesslaufzeiten und den währenddessen häufig wechselnden Anforderungen, starren Rollenverteilungen und anderen Problemen in Schieflage. Häufig genannt wird in diesem Zusammenhang das nach dem Wasserfallmodell begonnene Großprojekt C3 des Chrysler-Konzerns (Chrysler Comprehensive Compensation). In dieser Zeit experimentierte man in den USA mit leichtgewichtigen Entwicklungsprozessen und fand heraus, dass beispielsweise durch kürzere Laufzeiten, mehr Eigenverantwortung und Zusammenarbeit in den Projektteams oder einen unkomplizierten Umgang mit Änderungsanforderungen viele Risiken besser bewältigt und die Projekte häufiger zum Erfolg geführt werden konnten - Erfolg im Sinne einer frühzeitigen Verwertbarkeit der Ergebnisse durch den Auftraggeber. Es entstanden Vorgehensmodelle wie Scrum oder Crystal. Das Chrysler-C3-Projekt wurde durch die Einführung verschiedener Methoden aus dem Kontext leichtgewichtiger Vorgehensmodelle vor dem Scheitern bewahrt, die anschließend unter der Bezeichnung Extreme Programming populär wurden [Wells 2009].

Auf einem Treffen im Februar 2001 in Utah (USA) tauschten Experten ihre Erfahrungen mit Softwareentwicklungsprozessen aus und formulierten ein Wertesystem, das das Fundament für die künftig als agil bezeichnete Art der Softwareentwicklung bildete – das Agile Manifest [Agiles Manifest 2001]:

Wir erschließen bessere Wege, Software zu entwickeln,
indem wir es selbst tun und anderen dabei helfen.
Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

Individuen und Interaktionen

mehr als Prozesse und Werkzeuge

Funktionierende Software

mehr als umfassende Dokumentation

Zusammenarbeit mit dem Kunden

mehr als Vertragsverhandlung

Reagieren auf Veränderung

mehr als das Befolgen eines Plans

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein.

Das Agile Manifest wird oft falsch interpretiert. Beispielsweise dahingehend, dass man auf Dokumentation verzichten kann. Insbesondere der letzte Absatz macht jedoch deutlich, dass es dabei nur um Prioritäten geht, eine Tätigkeit wie Dokumentation aber durchaus für wichtig befunden wird.

Präzisiert wurde dieses Wertesystem durch zwölf Prinzipien agiler Softwareentwicklung [Agiles Manifest Prinzipien 2001]:

1. *Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.*
2. *Heiße Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.*
3. *Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.*
4. *Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.*
5. *Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.*
6. *Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.*
7. *Funktionierende Software ist das wichtigste Fortschrittsmaß.*
8. *Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.*
9. *Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.*
10. *Einfachheit -- die Kunst, die Menge nicht getaner Arbeit zu maximieren -- ist essenziell.*
11. *Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.*
12. *In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.*

Der Erfolg agiler Methoden sprach sich herum. Nach einer Ende 2005 von Forrester Research durchgeführten Untersuchung entwickelten bereits 14% der Unternehmen in Europa und Nordamerika ihre Software unter Zuhilfenahme von agilen Prozessen, während weitere 19% über die Nutzung nachdachten [Forrester 2005].

Status quo

Der auf agile Methoden spezialisierte Technologieanbieter VersionOne stellte in seiner siebten jährlichen Umfrage zu agilen Methoden fest, dass 2013 bereits 84% aller Unternehmen agile Entwicklung betreiben [VersionOne 2013]. Inzwischen hat die Zahl vermutlich weiter zugenommen. Jedoch auch wenn keine agilen Vorgehensmodelle wie Scrum eingesetzt werden, sondern hybride Modelle, beispielsweise die Kombination agiler Methoden mit Projektmanagementstandards, so findet man in erfolgreichen Projekten immer wieder die folgenden drei typischen Merkmale agiler Entwicklung:

- **Inkrementell**: Teile des Systems werden zu verschiedenen Zeiten entwickelt und das System jeweils um die fertig gestellten Teile erweitert.
- **Lernend**: Das Team lernt aus Retrospektiven. Die Organisation lernt durch einen kontinuierlichen Verbesserungsprozess, beispielsweise auf Basis regelmäßiger Fehlerursachenanalysen.
- **Unmittelbar**: Alle Beteiligten arbeiten eng und direkt zusammen. Das Team zeichnet sich durch eine flexible Aufgabenverteilung ohne starre Rollenzuordnung und eine kollektive Verantwortung für das Produkt aus. Auftraggeber und/oder Produktverantwortliche bringen sich aktiv und kontinuierlich in den Entwicklungsprozess ein.

Generell lässt sich feststellen: Die „frühe und kontinuierliche Auslieferung wertvoller Software“ sowie das Willkommenheißen

von „Änderungsanforderungen selbst spät in der Entwicklung“ - beides sind Prinzipien des Agilen Manifests - stellen heute kritische Erfolgsfaktoren für Software entwickelnde Unternehmen dar. Gründe dafür sind die schnell fortschreitende Durchdringung unserer Welt mit Software, die Digitalisierung und Virtualisierung in Kombination mit dem globalen Wettbewerb. Kommerzielle Entwicklungsprojekte, bei denen der Auftraggeber länger als ein halbes Jahr auf sein Produkt warten muss und während dieser Zeit keine Möglichkeiten hat, geänderte Anforderungen einfließen zu lassen, sind heute nahezu undenkbar.

Zuverlässigkeit

Wird Software nicht nur zum Spaß entwickelt, sondern werden mit einem Entwicklungsvorhaben kommerzielle Ziele verfolgt, ist es essentiell, die Zielerreichung zu planen und zu kontrollieren. Insbesondere, wenn die Ziele in einem Werkvertrag nach §§ 631 ff. BGB vereinbart wurden.

Werkverträge und agile Softwareentwicklung schließen sich nicht aus. Viele Prinzipien agiler Softwareentwicklung resultieren aus Nachbetrachtungen von Problemen meist schwergewichtiger Entwicklungsprojekte und sind somit gute Maßnahmen zur Minderung von Risiken, denen Entwicklungsprojekte mit festem Endtermin häufig ausgesetzt sind. Nachfolgend sind einige Beispiele aufgeführt – ohne Anspruch auf Vollständigkeit:

Risiko 1:

Invalide Planung hinsichtlich des erforderlichen Personalaufwands, um den Entwicklungsauftrag in der notwendigen Zeit erfüllen zu können.

Maßnahme:

Diesem Risiko kann durch die Verwendung von Aufwandsschätzmethoden begegnet werden, deren Genauigkeit sich mit jeder Retrospektive verbessert. Solche Methoden werden in den nachfolgenden Kapiteln beschrieben.

Risiko 2:

Ziele und Anforderungen des Auftraggebers verändern sich, bevor die Entwicklung abgeschlossen ist, beispielsweise infolge von Veränderungen des Marktes.

Maßnahmen:

Ein Prinzip des Agilen Manifests lautet: „Heiße Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden." Agile Softwareentwicklung zeichnet sich somit grundsätzlich durch die Bereitschaft aus, neue oder geänderte Anforderungen zu berücksichtigen. Jedoch schließt dies nicht eine Einigung zwischen Vertragspartnern bezüglich der kaufmännischen Auswirkungen von Veränderungen auf ein Entwicklungsprojekt aus.