Mögliche Messansätzen innerhalb agil durchgeführter Softwareentwicklungsprojekte

Andreas Schmietendorf
Hochschule für Wirtschaft und Recht Berlin
Email: andreas.schmietendorf@hwr-berlin.de

Zusammenfassung: Im Mittelpunkt des Beitrags stehen Messansätze, die im Kontext agil durchgeführter Industrieprojekte benötigt werden. Nach einer kurzen Einordnung und Abgrenzung zu wissenschaftlich benötigten Messansätzen (empirische Softwaretechnik) wird auf industrielle Bedürfnisse eingegangen. Ausgehend vom aktuellen Verständnis agiler Methoden (speziell Scrum und eXtreme Programming - XP) werden die benötigten Messansätze exemplarisch aufgezeigt. Da derartige Messansätze selbst dem agilen Paradigma unterliegen, bedarf es für eine erfolgreiche Verwendung einer entsprechenden Werkzeugunterstützung bzw. Automatisierung der korrespondierenden Messprozesse. Auf ausgewählte Aspekte dieses Sachverhalts soll innerhalb des letzten Abschnitts eingegangen werden.

1 Motivation

Der vorliegende Beitrag untersucht die Bedürfnisse, Sinnfälligkeit und Praktikabilität der Verwendung von Softwaremetriken im Umfeld agil durchgeführter Entwicklungsprojekte. Die Motivation zur Auseinandersetzung mit dieser Themenstellung liegt in der allgemein geführten Diskussion benötigter Mess- und Bewertungsansätze im Kontext agiler Paradigmen zur Softwareentwicklung. Unter [Boehm 2003] finden sich metrisch zu erfassende Einflussfaktoren, anhand derer die Auswahl agiler Vorgehensweisen entschieden werden kann:

- Methodenkompetenz der Mitarbeiter – gering vs. hoch;
- Konstanz der Anforderungen – gering vs. hoch;
- Kritikalität der Anwendung – gering vs. hoch;
- Größe der Organisation – gering vs. hoch;
- Kulturelle Eigenschaften – geringe Ordnung vs. hohe Ordnung.

Eine hohe Bewertung der jeweiligen Eigenschaft wurde durch [Boehm 2003] eher als ein Indiz für die Verwendung klassischer Methoden der Softwareentwicklung gesehen. Aus Sicht des Autors kann diese sehr allgemeine Einschätzung nicht ohne eine weiterführende Interpretation der Merkmale übernommen werden. Aktuelle Ansätze wie Scrum und XP erfordern durchaus eine hohe Methoden- und Sozialkompetenz. Ebenso sollte die Größe der Organisation für die Durchführung agiler Projekt nicht grundsätzlich ein Hindernis darstellen. Allgemein kann sicherlich bei Entwicklungen im Kontext mobiler Endgeräte anders als bei gewachsenen Altsystemen wie z.B. Host- oder auch ERP-Anwendungen eine klare Tendenz für agile Vorgehensweisen beobachtet werden.

2 Ausgangssituation und Abgrenzung

Bei einer Untersuchung der Bedürfnisse einzusetzender Softwaremetriken muss zwischen wissenschaftlichen und industriellen Interessen differenziert werden. Im Kontext der empirischen Softwaretechnik (Wissenschaft) gilt es, theoretische Ansätze zur Softwareentwicklung hinsichtlich ihrer Eigenschaften zu bewerten. Typische Zielstellungen beziehen sich auf die Falsifikation von Theorien und Hypothesen. Im Zusammenhang mit dem agilen Paradigma zur Softwareentwicklung interessieren insbesondere die Validität der angenommenen Vorteile wie z.B. höhere Produktivität, kürzere Projektlaufzeiten, verbesserte Qualität.

- Die ökonomische Bewertung agiler Vorgehensweisen (speziell XP) findet sich z.B. unter [Müller 2003]. Dabei werden insbesondere die paarweise Programmierung und die testgetriebene Vorgehensweise im Kontext der resultierenden Projektlaufzeit, die Kosten sowie die erreichbare Qualität untersucht. Unter Verwendung eines speziell entwickelten Modells (8 metrisch untersetzte Bewertungskriterien) gehen die Autoren auf eine durchgeführte Kosten-/Nutzen-Analyse ein.
- Unter [Arisholm 2007] finden sich die Ergebnisse eines kontrollierten Experiments zum Pair Programming. Mit Hilfe von 3 bewerteten Programmieraufgaben unterschiedlicher Komplexität wurde die paarweise Entwicklung (98 Paare) mit der Einzelentwicklung (99 Entwickler) verglichen. Im Kontext der jeweils erstellten Ergebnisse wurden die Dauer, der Aufwand und die Korrektheit ermittelt.
- Eine Analyse gegenwärtiger Ansätze zur agilen Qualitätssicherung und –bewertung findet sich unter [Janus 2012]. Insgesamt werden 36 Ansätze untersucht und im Kontext der Kategorien Methode, Modelle, Wartung & Weiterentwicklung, Software-Qualität und agile Softwareentwicklung einer grundlegenden Bewertung unterzogen. Nur wenige dieser Ansätze verwenden aktuell metrisch untersetzte Bewertungskriterien.
- Unter [Dyba 2008] findet sich eine umfangreiche Analyse empirischer Studien zur agilen Softwareentwicklung, wobei folgende Themen berücksichtigt wurden:
  - Einführung und Verwendung agiler Methoden,
  - Menschliche und soziale Faktoren,
  - Bekanntheit agiler Methoden,
  - Verfügbare vergleichende Studien.
- Zusammenfassend verweisen die Autoren auf einen weitergehenden Forschungsbedarf im Umfeld der agilen Softwareentwicklung. Schwerpunkte werden dabei insbesondere im Bedarf empirischer Analysen, der Verbreiterung der berücksichtigten Vorgehensweisen (bisher beziehen sich Studien zumeist auf XP-Ansätze) aber auch im stärkeren Bezug auf realistische Projekte aus dem industriellen Umfeld gesehen.
- Unter Verwendung der aus dem objektorientierten Paradigma bekannten Entwurfsmetriken (u.a. Kopplung, Kohäsion, Vererbung) wird unter [Roden 2007] eine Untersuchung zur Stabilität (System Design Instablility with Entropie – kurz SDI_e) agil entwickelter Projekte durchgeführt. Mit Hilfe der Entwurfsmetriken entwickeln die Autoren ein Qualitätsmodell, welches zu einem Total Quality Index (kurz TQI) führt.

3 Industriell motivierte Anforderungen

Die Verwendung von Softwaremetriken innerhalb eines industriellen Projekts erfolgt zur qualitativen und quantitativen Bewertung der aktuellen Projektsituation und der dabei bereitgestellten Ergebnisse. Im Zusammenhang mit einem agil durchgeführten Softwareentwicklungsprojekt gilt es, Metriken einzusetzen, die den Projekterfolg unterstützen und dem agilen Paradigma nicht entgegenstehen. Im Rahmen dieses Beitrags soll eine Konzentration auf die industriellen Bedürfnisse des Einsatzes von Metriken bei agilen Projekten erfolgen.

Die Notwendigkeit zur Verwendung messtechnischer Ansätze soll mit Hilfe einiger Eckdaten einer industriellen Projektfamilie aufgezeigt werden. Vor 13 Jahren begann die parallele Entwicklung benötigter Softwaresysteme zur Unterstützung der kaufmännischen und technischen Bereitstellung eines telekommunikationstechnischen Massenprodukts.

Im Rahmen einer Analyse innerhalb des Jahres 2009 konnten folgende Eigenschaften der fünf involvierten Projekte festgehalten werden:

- 120 bis 150 Anforderungen der Kunden werden jährlich in entsprechende Aufträge überführt.
- Aufwandsschätzungen werden pro Jahr ca. 300-mal benötigt. Nicht alle Schätzungen führen allerdings zu korrespondierenden Aufträgen.
- Pro Jahr erhalten die 5 Projekte zusammen ein Auftragsvolumen von ca. 500 Personen-Monaten.
- An den Realisierungen innerhalb aller Projekte sind je nach anfallender Arbeitslast zwischen 15 und 30 Mitarbeiter beteiligt.
- Bei kleineren Aufträgen nehmen beteiligte Projektmitarbeiter gleichzeitig mehrere Rollen ein.
- Mehrere Versionen werden parallel, überlappend und in enger Interaktion zur Kundenseite (Feedback) realisiert.
- Die Realisierung einer Version erfolgt auf der Basis eines iterativen Vorgehens. Typische Projektlaufzeiten liegen bei 2 Monaten.
- Entwickelt wurde mit Hilfe der Microsoft ASP- bzw. ASP.NET-Technologie unter Verwendung von RDBMSn (SQL-Server, Oracle).
- Existierende Dokumente zeigen häufig eine geringe Detaillierung. Darüber hinaus erfolgte ein hoher Grad an Inline-Dokumentation.

Nach der langen Projektlaufzeit beziehen sich neue Anforderungen insbesondere auf bereits implementierte Komponenten, auf die Neuentwicklung bzw. Anpassung zu unterstützender Schnittstellen bzw. auf den versionsbedingten Austausch von Standard-Komponenten (z.B. Datenbankzugriffsschicht). Implizierte Aufwände derartiger Anforderungen resultieren maßgeblich aus der Wartbarkeit der bestehenden Systeme. Diese gilt es einzuschätzen und schrittweise zu verbessern.

Obwohl der Agilitätsbegriff zum Zeitpunkt der Initiierung noch keine Bedeutung hatte, fallen vielfältige Merkmale agiler Vorgehensweisen bei der betrachteten Projektfamilie auf. Zu nennen sind insbesondere das iterative Vorgehen, die priorisierte Umsetzung von Kundenanforderungen, die kurzen Projektlaufzeiten, die Art der Dokumentation und das unmittelbare Kundenfeedback. Die messtechnische Untersuchung eines Systems zeigte allerdings auch die für gewachsene Systemarchitekturen typischen Problembereiche (z.B. enge Kopplung von Komponenten, Verzicht auf generische Lösungsansätze) auf.

4 Messtechnische Anforderungen agiler Projekte

4.1 Allgemeine Anforderungen (agiles Manifest)

Agile Methoden zur Softwareentwicklung werden seit etwa 10 Jahren verwendet. Der Agilitätsbegriff wird zumeist mit einer schnellen, kostengünstigen und qualitativ hochwertigen Implementierung von Software in Verbindung gebracht. Nicht selten wird in diesem Zusammenhang von einer Entbürokratisierung der Softwareentwicklung und dem Verzicht einer dokumentengetriebenen Vorgehensweise gesprochen. Allen agilen Vorgehensweisen gemein ist die Verwendung der folgenden Praktiken:

- Iterative bzw. zyklische Vorgehensweisen mit einem Wechselspiel kurzer Planungs- und Entwicklungsphasen.

- Berücksichtigung eines zeitnahen und wenn möglich unmittelbaren Feedbacks durch den frühzeitigen Systemeinsatz.

Benötigte Ansätze für Softwaremessungen lassen sich insbesondere im Zusammenhang mit dem Anforderungsmanagement (Planung/Aufwandsschätzung), bei der Projektverfolgung (Fortschritt) und bei der Qualitätsbewertung (Architektur und Implementierung) vermuten. Das agile Manifest [Manifest 2001] greift die Frage der Messung in folgender Weise auf:

"Working software is the primary measure of progress."

Entsprechend dieser Aussage wird der Projektfortschritt vordergründig auf der Grundlage der arbeitsfähigen Software bewertet. In dieser Aussage steckt allerdings auch die Vermutung, weitere, offensichtlich dann sekundäre Messgrößen, zu benötigen. Auf die Bedeutung der Aufwandsschätzung für agile Vorgehensweisen wurde bereits unter [Schmietendorf 2009] eingegangen. Dementsprechend soll dieser Aspekt hier nicht vordergründig behandelt werden.

4.2 Anforderungen im Kontext von Scrum

Mit Hilfe selbstgesteuerter Entwicklungsteams, d.h. ohne Projektleiter, erfolgt im Rahmen von Sprints (2-4 Wochen) die ungestörte Implementierung der für diesen Zyklus geplanten Anforderungen (Quelle: Sprint-Backlog). Mit Hilfe des „daily scrum" erfolgt eine tägliche Feedbackrunde zur Darstellung des Projektfortschritts, potentieller Probleme und anstehender Aufgaben. Nach Abschluss eines Sprints erfolgt ein Review gegenüber dem Auftraggeber, aber auch eine Retrospektive im Sinne des Zusammenfassens der gewonnenen Erfahrungen. Aufgrund der Einfachheit dieser Managementmethode findet sich diese sehr häufig im industriellen Umfeld.

Scrum-Projekte bedürfen einer objektiven Bewertung des Projekterfolgs. Insbesondere im Zusammenhang mit dem ersten Sprint eines Projekts ergibt sich die Notwendigkeit zur Einführung einer messtechnisch untersetzten Erfolgskontrolle, was [Gloger 2009] in folgender Weise formuliert:

> „Ein wichtiger Schritt ist daher die Einführung diagnostischer Metriken. … Entscheidend dabei ist, dass diese Metrik vom Team selbst kommen muss. Dafür kann man es fragen, wie es messen will, ob der Sprint erfolgreich war."

Scrum orientiert sich mit der zyklischen Planung, Implementierung, Bewertung und Lieferung nutzbarer Software am aus dem Qualitätsmanagement bekannten kontinuierlichen Verbesserungsprozess. Es liegt nahe, diese Vorgehensweise mit dem PDCA-Zyklus[1] (Plan - Do - Check - Act) zu vergleichen (vgl. z.B. [Gloger 2010]). Unter [Heiramo 2010] findet sich das folgende Statement in Bezug auf den Aspekt der Messung/Bewertung:

> „However, the PDCA cycle provides us some insight into how we would benefit from Scrum more. An integral part of the PDCA cycle is measuring the output (or generally, evaluating it against some criteria) and comparing that to an established expectation."

Sehr deutlich wird hier der Bedarf an messtechnisch untersetzten Bewertungen, die auch aus vertraglicher Sicht verwendet werden können. Speziell sieht [Heiramo 2010] die folgenden benötigten Messansätze:

- Erwartungshaltung in Bezug auf Nutzen/Ergebnis eines Sprints.
- Messbarkeit der Ergebnisse bei Sprint-Reviews/-Retrospektive.

[1] auch als Deming-Cycle bezeichnet, nach Dr. W. Edwards Deming

Eine umfängliche Diskussion im Kontext verwendbarer Scrum-Metriken kann unter [Behrens 2009] nachvollzogen werden.

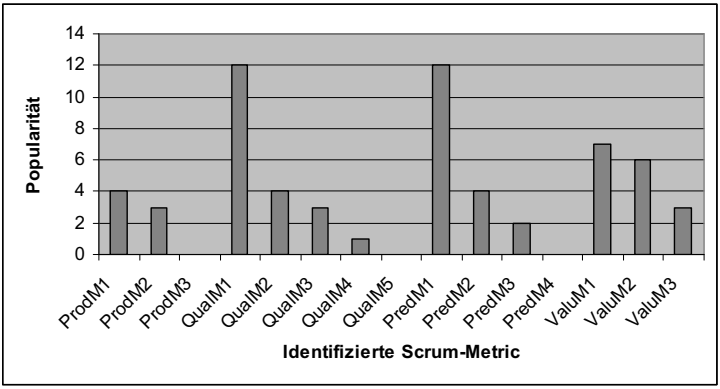

Abbildung 1: Scrum-Metriken und deren Popularität (unter Verwendung von [Behrens 2009])

Ca. 30 Scrum-Experten identifizierten, diskutierten und bewerteten die Popularität von Messansätzen im Kontext folgender Kategorien:

$ProdM_x$ - Productivity (3 Messansätze, 6 Mythen)

- $ProdM_1$ - In Sprint Cycle Time
- $ProdM_2$ - In Sprint Work in Process
- $ProdM_3$ - Sprint Completion Bar

$QualM_x$ - Quality (5 Messansätze, 4 Mythen)

- $QualM_1$ - Technical Debt Points
- $QualM_2$ - Running Automated Tests
- $QualM_3$ - Post Sprint Defect Arrival
- $QualM_4$ - Post Release Defect Arrival
- $QualM_5$ - Root Causes Fixed

$PredM_x$ - Predictability (4 Messansätze)

- $PredM_1$ - Enhanced Burndown Chart
- $PredM_2$ - Velocity
- $PredM_3$ - Cost per Story Point
- $PredM_4$ - Hours per Story Point

$ValuM_x$ - Value (3 Messansätze)

- $ValuM_1$ - Business Value Delivered
- $ValuM_2$ - Customer Satisfaction Survey
- $ValuM_3$ - Employee Satisfaction Survey.

Aus den vorgenannten Messansätzen wurden die folgenden drei Vertreter hinsichtlich ihrer Bedeutung am stärksten bewertet (siehe Abb. 1):

- Technical Debt Points ($QualM_1$) – Bewertung der qualitativen Projektentwicklung anhand der Menge offener architektonischer und quellcodebezogener Problembereiche. Typische Probleme beziehen sich auf fehlende Dokumentationen, hohe Abhängigkeiten zwischen den Systemkomponenten oder aber nicht eingehaltene Namenskonventionen. Die Problembeseitigung erfolgt z.B. beim Refactoring. Angestrebt wird eine geringe Anzahl offener (schuldiger) Probleme.
- Enhanced Burndown Chart ($PredM_1$) – Die Verwendung so genannter Burndown-Charts kann im Kontext agiler Methoden (nicht nur bei Scrum) vielfältig beobachtet werden. Bei der hier aufgezeigten Kennzahl handelt es sich um die Bewertung der je Iteration umgesetzten Anforderungen unter Berücksichtigung der Realisierungsgeschwindigkeit (velocity) und ggf. veränderten Zielstellungen (scope change). Mit Hilfe dieser Ergebnisse kann die Fertigstellung weiterer Anforderungen abgeschätzt werden.
- Business Value Delivered ($ValuM_1$) – Jeder umgesetzten und gelieferten Anforderung aus dem Sprint-Backlog wird ein Geschäftswert zugeordnet. Typischerweise wird man hier hohe Werte anstreben. Allerdings sollte dieser Wert bei einer korrekten Priorisierung der Anforderungen in späteren Iterationen sinken.

[SmokeJumper 2012] empfiehlt die Nutzung von 3 Metriken innerhalb von Scrum-Projekten. Eingegangen wird auf Burn-Down-Charts als grafische Präsentation der noch umzusetzenden Aufgaben (task to do) entlang des täglich zu überwachenden Projektverlaufs. Als zusätzliche Informationen werden die durch den Auftraggeber bereits akzeptierten User Stories (zumeist mit Hilfe von Story Points gemessen) mitgeführt. Darüber hinaus verdeutlicht die Trendlinie, wie weit die Realisierungsgeschwindigkeit vom gesetzten Ziel abweicht. Bei einer Überschreitung von großer 20% wird der Verzicht auf niedrig priorisierte User Stories empfohlen; bei einer Unterschreitung von mehr als 20% wird eine Aufnahme neuer User Stories empfohlen.

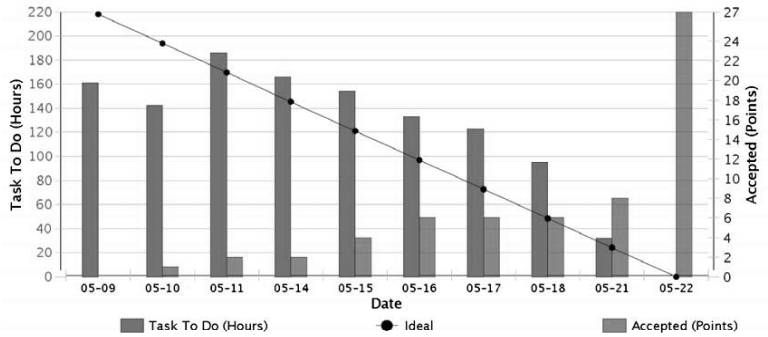

Abbildung 2: Burn-Down-Chart (Quelle: [SmokeJumper 2012])

Eine weitere Kennzahl bezieht sich entsprechend [SmokeJumper 2012] auf die teambezogene Entwicklungsgeschwindigkeit (Velocity). Dabei geht es um den sukzessiven Erfahrungsaufbau im Umgang mit der Menge an je Sprint umzusetzenden Anforderungen. Auf dieser Grundlage lassen sich die Aufwände zukünftiger Softwareentwicklungen sicherer und für alle Beteiligten (Stakeholder) nachvollziehbarer abschätzen. Der Kern dieser Kennzahl bezieht sich auf die erfolgreich gelieferten Anforderungen (User Stories), die zu Beginn des Sprints typischerweise mit Hilfe von Story Points geschätzt wurden.

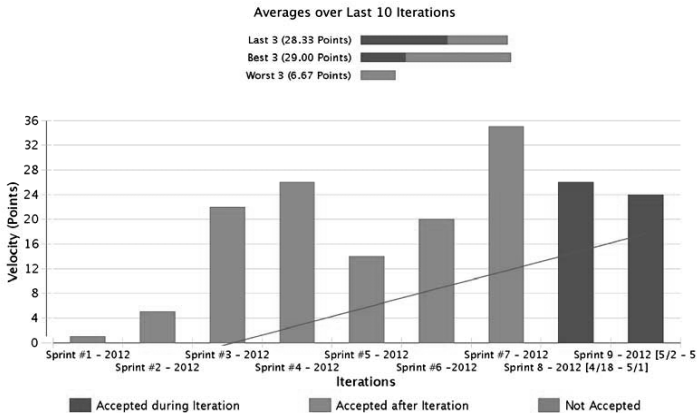

Abbildung 3: Velocity (Quelle: [SmokeJumper 2012])

Mit Hilfe der Abbildung 3 kann z.B. auf eine Entwicklungsgeschwindigkeit von 20 bis 30 Story Points geschlossen werden.

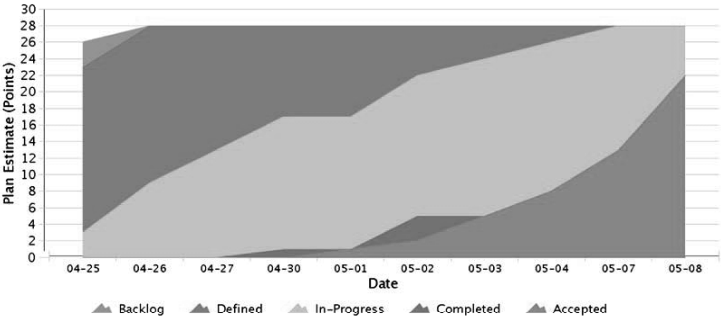

Abbildung 4: Iteration Cumulative Flow (Quelle: [SmokeJumper 2012])

Iteration Cumulative Flow – Beobachtung des je Iteration erreichten Projektbearbeitungsstatus. Innerhalb des Diagramms (vgl. Abbildung 4) werden die Zustände „Anforderungen im „Backlog“, „Anforderungen definiert“, „Anforderungen in Bearbeitung“, „Anforderungen erledigt“ und „Anforderungen akzeptiert“ verwendet.

4.3 Anforderungen im Kontext von XP

Viele der im Kontext des Scrum-Ansatzes aufgezeigten Messansätze können auch für andere agile Vorgehensweisen verwendet werden. Dementsprechend soll im Folgenden nur auf die messtechnischen Besonderheiten des XP-Ansatzes eingegangen werden.

Das „eXtreme Programming“ bietet gegenüber Scrum eine stärkere Detaillierung der vorgeschlagenen Prinzipien und Techniken (best practices). Darüber hinaus werden ineinander greifende Rückkopplungsmechanismen vorgeschlagen. So soll z.B. ein unmittelbares Feedback im Rahmen des für XP typischen „Pair Programming“ zur Qualitätssicherung beitragen. Das Feedback der Kundenseite soll mit Hilfe wöchentlicher Iterationen (inkl. lauffähige Version)

und kurzen Releasezyklen (wenige Monate) gewährleistet werden. Unter [Jeffries 2001] wird der XP-Ansatz in folgender Weise charakterisiert:

> „Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation."

Benötigte Messansätze werden z.B. im Zusammenhang mit den folgenden Prinzipien bzw. Primär- und Sekundar-Praktiken benötigt:

Prinzipien:

- XP-Prinzip der Wirtschaftlichkeit in Bezug auf das Kosten-/Nutzen-Verhältnis einer Softwareentwicklung.
- XP-Prinzip der Qualität in Bezug auf Software mit einem „guten" und damit erweiterbaren Design (Grundlage flacher Aufwandskurven).
- XP-Prinzip der Reflektion in Bezug auf gewollte Lern- und Verbesserungsprozesse (erfordert messtechnischen Nachweis).

Praktiken:

- XP-Praktik der testgetriebenen Entwicklung in Bezug auf die Erfassung des Entwicklungsstands der Software (Burndown-Chart).
- XP-Praktik des Wochenzyklus - es sind die umsetzbaren Anforderungen auf der Grundlage von Erfahrungen zu planen (Aufwandsschätzung).
- XP-Praktik des Quartalszyklus - es ist der Geschäftswert der gelieferten Software einzuschätzen (Grundlage nutzungsbezogener Bezahlung).

Viele der für ein XP-Projekt benötigten Messansätze korrespondieren mit den bereits beim Scrum-Ansatz aufgezeigten Anforderungen.

Der Zeitpunkt des Eintreffens einer neuen Anforderung spielt für den daraus resultierenden Aufwand bei einem XP-Projekt im Vergleich zu konventionell durchgeführten Entwicklungsprojekten keine Rolle. In diesem Zusammenhang wird auch von einer flachen Aufwandskurve gesprochen. Diese Annahme bildet die Grundlage für die Akzeptanz unvollständiger Anforderungen, wie diese typischerweise bei Projektbeginn vorliegen. Die dafür benötigten Voraussetzungen können nur mit Hilfe eines qualitativ hochwertigen und flexiblen Designs der Implementierung erreicht werden. Die messtechnische Bewertung dieser Eigenschaft ist im Kontext des Entwicklertest (Unit Test) und der kontinuierlichen Integration (Codeintegration) aus Sicht des Autors zwingend. Im Kontext einer kontinuierlichen und dementsprechend automatisierten Vermessung des Quellcodes sollten folgende Sachverhalte aufgegriffen werden:

- Bewertung der Softwarearchitektur (Stabilität/Wartbarkeit),
- Identifikation von duplizierten Codeteilen,
- Bewertung der Komplexität,
- Identifikation verletzter Regeln zur Codierung,
- Vermeiden von Abhängigkeiten (ripple effect).

Zur Erfassung der vorgenannten Qualitätseigenschaften bedarf es der Untersetzung mit konkreten Kennzahlen. Aus Gründen der Motivation, Zielorientierung und Akzeptanz sollten

diese durch das Projektteam selbst festgelegt und mit Hilfe von Messwerkzeugen zyklisch aufgenommen werden.

4.4 Risk Management für agile Softwareprojekte

Industrielle Softwareentwicklungsprojekte berücksichtigen das Risikomanagements als projektbegleitende Aufgabenstellung. Bei agilen Projekten können die vielfältigen Feedbackmechanismen die Gefahren von Risiken erheblich reduzierten, keinesfalls aber ausschließen:

- tägliche Stand-Up Meetings,
- Sprint- und Release-Planung (vgl. Planing Game),
- Retrospective,
- Reviews.

Im Zusammenhang mit agilen Projekten wird ggf. auch von einer risikogetriebenen Softwareentwicklung gesprochen, d.h. die Aufgaben des Risikomanagement sind verkürzt auf jeden zu planenden Sprint anzuwenden. Zur Verdeutlichung möglicher Risiken bietet sich abermals ein Burn-Down-Chart an, welches identifizierte Risiken über den zeitlichen Projektverlauf verfolgen lässt. Darüber hinaus sollten agil arbeitende Projektteams gewonnene Erfahrungen im Umgang mit Risiken im Sinne einer Erfahrungsdatenbank (z.B. innerhalb eines Projekt-Wikis) nachhalten. Auf einen komplizierten Risikomanagementprozess bzw. auf die strukturierte Ablage entsprechender Informationen sollte dabei aber verzichtet werden.

5 Werkzeugunterstützung

Die Erhebung, Verarbeitung, Speicherung und Auswertung von Softwaremetriken bedarf im Kontext agiler Softwareentwicklungsprojekte einer entsprechenden Werkzeugunterstützung. Dabei kommt einer unmittelbaren Integration in verwendete Softwareentwicklungswerkzeuge bzw. der automatisierten Vermessung, Verarbeitung und Auswertung eine besondere Bedeutung zu. Aus funktionaler Sicht sollten folgende Aspekte berücksichtigt werden:

- Zielorientierte Konfiguration benötigter Metriken,
- Unterstützung einer kontinuierlichen Analyse,
- Bereitstellung grundlegender Interpretationen,
- Visualisierung identifizierter Problembereiche,
- Automatisierte Schnittstellen zur Weiterverarbeitung.

Aufgrund der Konzentration auf die möglichst schnelle Bereitstellung lauffähiger Software fokussieren sich benötigte Messansätze insbesondere auf die Bewertung des Quellcodes und die Bewertung durchgeführter Tests. Bedürfnisse für die Verwendung von Kennzahlen finden sich allerdings auch beim Team- bzw. Projekt-Management. Dabei kommen sehr häufig einfache Werkzeuge (z.B. Excel-Charts, Projekt-Wikis) zum Einsatz. Mit zunehmender Projektlaufzeit finden sich aber auch spezielle Lösungen.

6 Zusammenfassung

Erfolgreiche Projekte im agilen Umfeld erfordern aus Sicht des Autors den gezielten, aber leichtgewichtigen (agilen) Einsatz von Softwaremetriken. Die Feedback- bzw. Rückkopplungsmechanismen auf allen Projektebenen sind nichts anderes als ein allgegenwärtiges bzw.

zyklisches Messen. Im Rahmen der einschlägigen Literatur, aber auch durch die entsprechenden Interessengruppen zur agilen Softwareentwicklung werden diese Anforderungen keinesfalls negiert! Allerdings wird die Verantwortung für benötigte Messungen den entsprechenden Teams übertragen, was zu den Grundprinzipien agiler Vorgehensweisen passt. Diese müssen sich agile Messansätze aneignen, welche die Zielstellungen des Projekts unterstützen. Ggf. kann dabei auf entsprechende Frameworks zur Softwaremessung, oder aber auf Messservices innerhalb des Internets zurückgegriffen werden. Die aktuell verwendeten Metriken zeigen zumeist ein geringes Skalenniveau, was im Kontext einer industrialisierten Herangehensweise nur bedingt befriedigen kann. Nur die empirische Betrachtung agil durchgeführter Softwareentwicklungen bietet die Möglichkeit, im Sinne des Skalenniveaus höherwertige Messgrößen mit einem stärkeren Informationsgehalt bereitzustellen. Abschließend noch eine passende Aussage von [Goldstein 2011], einem Praktiker aus der agilen Community, zu (Scrum-) Metriken:

> "Like it or not, metrics are a big part of life and an even bigger part of working life."

Verwendete Quellen

[Arisholm 2007] Arisholm, E.; Gallis, H.; Dyba, T.; Sjøberg, D. I. K.: Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise, IEEE Transactions on Software Engineering, Vol. 33, No. 2, February 2007

[Behrens 2009] Behrens, P. (Ed.): scrumorlando09: Orlando: Scrum Metrics and Myths, URL: http://scrumorlando09.pbwiki.com (Abruf: Februar 2012)

[Boehm 2003] Boehm, B. Turner, R.: Balancing Agility and Discipline: A Guide for the Perplexed, Addison Wesley, Boston 2004

[Dyba 2008] Dyba, T.; Dingsøyr, T.: Empirical studies of agile software development: A systematic review, Elsevier B.V., 2008

[Gloger 2009] Gloger, B.: Scrum - Produkte zuverlässig und schnell entwickeln, Carl Hanser Verlag, München 2009

[Gloger 2010] Gloger, B.: Scrum – Der Paradigmenwechsel im Projekt- und Produktmanagement, Informatik Spektrum, Band 33 Heft 2, Springer Heidelberg 2010

[Goldstein 2011] Goldstein, I.: Scrum metrics and reporting – measure what you manage, August 2011, URL: http://www.scrumshortcuts.com/blog (Abruf: März 2012)

[Heiramo 2010] Heiramo, P.: PDCA Cycles and Scrum http://agilecraft.wordpress.com/2010/11/24/pdca-cycles-and-scrum/

[Janus 2012] Janus, A.: Konzepte für Agile Qualitätssicherung und –bewertung in Wartungs- und Weiterentwicklungs-Projekten, Magdeburger Schriften zum Empirischen Software Engineering, Shaker-Verlag, Aachen Oktober 2013

[Jeffries 2001] Jeffries, R.: XProgramming.com - What is Extreme Programming?, URL: http://www.xprogramming.com/xpmag/whatisxp (Abruf: Februar 2012)

[Löper 2005] Löper, S.; Schmidt, G.: Softwareprozessmetriken und agile Methoden – eine industrielle Fallstudie, in Proc. Software Engineering 2005 (LNI), Gesellschaft für Informatik, Bonn 2005

[Manifest 2001] Manifesto for Agile Software Development - Principles behind the Agile Manifesto, URL: http://agilemanifesto.org/principles.html (Abruf Februar 2012)

[Müller 2003] Müller, M.; Padberg, F.: On the economic evaluation of XP projects. In ESEC / FSE, Helsinki, Finland, September 2003.

[Roden 2007] Roden, P. L.; Virani, S.; Etzkorn, L. H.; Messimer, S.: An Empirical Study of the Relationship of Stability Metrics and the QMOOD Quality Models Over Software Developed Using Highly Iterative or Agile Software Processes, Seventh IEEE International Working Conference on Source Code Analysis and Manipulation, 2007

[Schmietendorf 2009] Schmietendorf, A.: Aufwandsschätzung bei Projekten nach dem eXtreme Programming-Paradigma, Magdeburger Schriften zum Empirischen Software Engineering, Shaker-Verlag, Aachen Oktober 2009

[SmokeJumper 2012] The Three Most Important Metrics For Agile Teams, http://smokejumperstrategy.com/archive/the-three-most-important-metrics-for-agile-teams/ (Abruf Februar 2013)