Agilität und Prozessreife: Erfüllbarkeit der CMMI-Prozessgebiete durch agile Methoden am Beispiel von XP

Martin Fritzsche, Patrick Keil

Lehrstuhl IV: Software & Systems Engineering
Technische Universität München
Boltzmannstr. 3
85748 Garching
fritzscm@in.tum.de
keilp@in.tum.de

Abstract: In den letzten Jahren wurden agile Methoden wie eXtreme Programming zunehmend populär. Parallel dazu stützen sich mehr und mehr Unternehmen auf Reifegradmodelle, um ihre eigenen Prozesse oder die der Zulieferer zu analysieren und zu verbessern, nachdem sich immer mehr die Ansicht durchsetzt, dass viele Projektmisserfolge undisziplinierten, inkonsistenten Prozessen zugeschrieben werden können. In dieser Situation ist es notwendig, die Zusammenhänge und gegenseitigen Einschränkungen von agilen Methoden und Verfahren zur Softwareprozess-Analyse und -Verbesserung zu analysieren. Dieser Beitrag untersucht, in welchem Maß die CMMI-Prozessgebiete durch XP abgedeckt werden und wo XP angepasst werden muss. Darauf aufbauend beschreiben wir die Grenzen von CMMI im agilen Umfeld und zeigen, dass Level 4 und 5 mit den aktuellen Spezifikationen von CMMI und XP nicht realisierbar sind.

1 Einleitung

Indikatoren für Qualität und Reife organisatorischer Abläufe wie CMMI-Levels, SPICE-Ratings oder ISO-Standards werden zunehmend wichtiger in der Softwareentwicklung. So verlassen sich z. B. Auftraggeber, oft auf diese Indikatoren, wenn sie Zulieferer aussuchen, da die Ergebnisse der Analysen und Audits als „Signal“ für die Prozessreife dienen können [Do06, Ke05]. Daneben gibt es in einigen Großunternehmen Richtlinien, die verlangen, dass alle Unternehmensteile bestimmte Reifegrade erreichen.

Gleichzeitig erlangen agile Methoden weiterhin zunehmend Verbreitung. Dies gilt auch für größere Projekte; z. B. beschreiben Cockburn und Highsmith erfolgreiche agile Projekte mit bis zu 250 Personen [CH01]. Dies führt zu der Herausforderung, dass auf der einen Seite Unternehmen auf CMMI als Indikator für die Prozessreife (von der geglaubt wird, dass sie sich in der Produktqualität niederschlägt) setzen, auf der anderen Seite agile Methoden [Be01] wie XP [Be04, Hi05], Scrum [SB01], Lean Development [PP03] oder Crystal [Be04] immer bedeutender werden. Einige Ansätze agiler Methoden sind sogar schon in „schwergewichtige“ Vorgehensmodelle eingeflossen [Vm06].

In der Literatur wurde gezeigt, dass Projekte, die agile Methoden mit einigen Anpassungen verwendeten, CMMI Stufe 2 oder sogar 3 erreichen können [An05, KA04]. Aber die verschiedenen Berichte über erfolgreiche agile Projekte machen nicht deutlich, wie agile Methoden zur Erfüllung der Prozessgebiete beitragen, wo sie angepasst werden müssen und wo sie mit CMMI-Zielen in Konflikt stehen.

Daher sollte verstärkt untersucht werden, wie agile Methoden angepasst werden können, um bestimmten Reifegraden und Referenzprozessen zu entsprechen. Dieses Papier ist als Startpunkt gedacht, der aufzeigt, wo Anpassungen nötig sind. Nach einem kurzen Überblick über CMMI analysieren wir qualitativ am Beispiel von eXtreme Programming (XP), inwieweit agile Methoden CMMI-Prozessgebiete unterstützen oder mit ihnen im Konflikt stehen, wo Anpassungen gemacht werden müssen und ob Unternehmen, die agile Methoden einsetzen, Konformität mit bestimmten CMMI-Levels erreichen können.

2 Kompatibilität der agilen Methoden mit den Anforderungen von CMMI

2.1 CMMI – eine Übersicht

Das Capability Maturity Model for Software (CMM) [So02] wurde vom Software Engineering Institute (SEI) der Carnegie-Mellon University in Pittsburgh entwickelt. CMMI – die „de facto Methode“ der Softwareprozessverbesserung [Va04] – beschreibt planende und steuernde Prozesse zur Bewältigung von Schwierigkeiten in der Softwareentwicklung. Dazu wurden fünf Reifegrade von 1 bis 5 definiert. Diese heißen initial, gemanagt, definiert, quantitativ gemanagt, und optimierend.

Es ist wichtig zu bemerken, dass die CMMI-Prozessmodelle keine präskriptiven Prozesse enthalten, die direkt verwendet werden können. Stattdessen bietet CMMI einen Weg, um die Fähigkeit einer Organisation zu ermitteln, Software auf wiederholbare, vorhersehbare Weise zu entwickeln [Do06].

Um eine bestimmte Stufe zu erreichen, muss ein Unternehmen alle Prozessgebiete dieser Stufe erfüllen, genauso wie die niedrigerer Stufen. Ein Prozessgebiet ist eine Zusammenfassung aller Anforderungen zu einem bestimmten Thema, z. B. Projektplanung, Organisationsweites Training oder Quantitatives Projektmanagement. Um ein Prozessgebiet zu erfüllen, müssen alle assoziierten Ziele – spezifische oder generische – erreicht werden. Spezifische Ziele betreffen genau ein Prozessgebiet und beschreiben, was erfüllt werden muss, um das Prozessgebiet zu erfüllen. Um ein spezifisches Ziel zu erreichen, schlägt CMMI spezifische Praktiken vor. Eine spezifische Praktik ist eine Aktivität, die beim Erreichen eines spezifischen Ziels für wichtig befunden wird.

Generische Ziele beschreiben die Institutionalisierung, sollen also sicherstellen, dass Prozesse effektiv, wiederholbar und dauerhaft sind. Sie werden „generisch“ genannt, weil sie sich auf eine Reihe von Prozessgebieten beziehen. In der stufenförmigen Darstellung hat jedes Prozessgebiet nur ein generisches Ziel. Um ein generisches Ziel zu erreichen, schlägt CMMI generische Praktiken vor.

2.2 Analyse der Abdeckung der Prozessgebiete durch agile Methoden

Unser Ziel ist es, festzustellen, welche der CMMI Prozessgebiete von agilen Methoden unterstützt werden, wo Anpassungen gemacht werden müssen und welche Prozessgebiete im Konflikt stehen. Dazu analysieren wir jedes Prozessgebiet und alle seine spezifischen Ziele im Detail. Die spezifischen Praktiken sind nur erwartete Modellkomponenten, d.h. ihre Verwendung ist empfohlen, aber nicht notwendig. CMMI gibt an, dass sie durch alternative Praktiken ersetzt werden können. Tatsächlich verwenden agile Methoden oft andere Ansätze, als die, die von CMMI vorgeschlagen werden. Deswegen konzentrieren wir uns auf die Analyse der Ziele, betrachten die Praktiken nur als Richtlinien und behalten alternative Vorgehensweisen stets im Auge.

Im Folgenden konzentrieren wir uns auf die sogenannte „staged representation“ von CMMI, da die Prozessgebiete und spezifischen Ziele denen der „continuous“ Variante entsprechen. Unterschiede gibt es nur bei den generischen Zielen und vor allem bei der Assessment-Methode – beide Aspekte werden in diesem Beitrag aber nicht thematisiert. Zu den generischen Zielen siehe u. a. [Fr05].

Für die Abdeckung der spezifischen Ziele und Prozessgebiete wird das folgende Bewertungssystem verwendet:

- im Konflikt (-): das entsprechende CMMI-Ziel kann nicht erreicht werden; d.h. es gibt keine denkbaren Erweiterungen, die nicht den grundlegenden Mechanismen der Methode oder den agilen Prinzipien zuwiderlaufen
- nicht adressiert (0): es ist keine Abdeckung vorhanden
- teilweise unterstützt (+): stark eingeschränkte Abdeckung
- unterstützt (++):eingeschränkte Abdeckung
- wesentlich unterstützt (+++): die agile Methode, wenn sie richtig angewendet wird, erfüllt die jeweilige Modellkomponente zum großen Teil

Die Bewertungen 0 bis ++ unterstellen nicht, dass die jeweiligen CMMI Ziele nicht erreicht werden können. Sie betonen lediglich, dass zusätzliche Praktiken eingeführt werden müssten, um die CMMI Anforderungen vollständig zu erfüllen. Um zwischen „nicht adressiert” und „im Konflikt“ zu unterscheiden, muss geprüft werden, ob die agile Methode möglicherweise erweitert werden kann, ohne dass ihre Grundmechanismen gestört werden oder den Prinzipien aus dem agilen Manifests widersprochen wird.

2.3 Anwendung des Ansatzes auf eXtreme Programming

In diesem Kapitel wenden wir unseren Ansatz auf XP an und zeigen Beziehungen und Konflikte zwischen XP und den CMMI Prozessgebieten und all ihrer assoziierten spezifischen Ziele. Um nicht den Rahmen zu sprengen, halten wir die Analyse kurz. [Fr05] bietet eine detailliertere Analyse und zusätzlich eine Erörterung von Scrum.

2.3.1 Analyse der Prozessgebiete und ihrer spezifischen Ziele

2.3.1.1 Anforderungsmanagement – Anforderungen managen (+++)

Durch die Integration des Kunden in das Team und durch die daraus resultierende intensive Kommunikation mit dem Kunden wird das Verständnis für die Anforderungen gefördert. Das Commitment der Projektteilnehmer auf die Anforderungen erfolgt in der Planungsphase. Änderungen an den Anforderungen werden schnell ausgetauscht und diskutiert. Auch wenn das Tracing der Anforderungen nicht ein explizites Ziel von XP ist, wird es unterstützt durch Stories, Tasks, funktionale Tests, die Inkonsistenzen zwischen der Projektarbeit und Anforderungen aufdecken, und durch Komponententests. Die Praktik von XP, Storycards nach deren Implementierung wegzuwerfen, kann sich als problematisch erweisen. Um dieses Prozessgebiet besser zu unterstützen, sollten die Storycards aufgehoben werden. So können die Möglichkeiten des Tracings verbessert werden, indem Storycards und alte Versionen der Dokumentation archiviert werden.

2.3.1.2 Projektplanung (+++)

Schätzungen aufstellen (+++): Schätzungen für Stories und Aufgaben werden aufgestellt und können während des Projektverlaufs korrigiert werden. Die Präzision der Schätzungen steigt durch einen kurzen Planungshorizont dank kurzer Iterationen.

Projektplan erstellen (+++): Der Projektplan wird durch XP's Release- und Iterationspläne realisiert, die während des Projektes weiterentwickelt werden. Deswegen bleiben Langzeitpläne vage und nur Kurzzeitpläne sind detailliert. Risiken werden identifiziert, Schulungsbedarf wird geplant und die Einbeziehung aller relevanten Stakeholder ist gesichert, wenn XP richtig angewendet wird.

Verpflichtung auf den Plan herbeiführen (+++): Die Einhaltung der Release- und Iterationspläne wird durch die starke Einbindung und Verantwortung aller Teammitglieder erreicht.

2.3.1.3 Projektverfolgung und -steuerung (+++)

Projekt gegen den Plan überwachen (+++): Ablaufplan und Schätzungen werden durch den Tracker überwacht. Der Projektfortschritt wird durch die Verwendung von Messdaten erfasst. Die intensive Kommunikation zwischen den Teammitgliedern und mit dem Kunden hilft, die Information zu übermitteln. Meilensteine werden zeitlich anhand des Terminplans und inhaltlich anhand der Funktionstests überprüft. Das System kurzer Iterationen und laufender Abstimmungen vereinfacht die Projektüberwachung.

Korrekturmaßnahmen bis zum Abschluss managen (+++): Angelegenheiten, die Korrekturen erfordern, werden gesammelt und analysiert. Korrekturen können Anpassungen des Verfahrens oder der umzusetzenden Funktionalität sein. Zusätzlich bieten neue Iterationen immer gute Möglichkeiten, Anpassungen vorzunehmen.

2.3.1.4 Management von Lieferantenvereinbarungen (0)

Dieses Prozessgebiet wird nicht von XP behandelt. Wir glauben, dass die Methode erweitert werden kann, um die Ziele des Prozessgebiets zu erfüllen. Einen ähnlichen Standpunkt vertreten Turner und Jain [TJ02]. Jedoch kann es auch problematisch für die Agilität sein, Zulieferer einzubeziehen, wenn dies die iterative Entwicklung behindert. Es gibt Fälle, in denen gelieferte Komponenten am Ende einer Iteration gebraucht werden, um eine funktionierende Software zu erhalten. Es kann ein kritisches Problem entstehen, wenn diese nicht rechtzeitig verfügbar sind.

2.3.1.5 Messung und Analyse (+)

Ausrichtung der Mess- und Analyse-Aktivitäten festlegen (+): Die einzige Messtätigkeit ist die Fortschrittskontrolle. Messungs- und Analyseprozeduren werden durch den Tracker definiert. XP stellt für diese Aufgaben keine besonderen Richtlinien bereit.

Messergebnisse bereitstellen (++): Die Messdaten werden durch intensive Kommunikation im Team gesammelt. Der Tracker analysiert die Daten und gibt diese durch sogenannte Wall Charts an das Team weiter. Die Daten werden normalerweise nicht permanent gespeichert. Es sind jedoch eine Reihe von Tools für Aufwandsschätzungen und Tracking für agile Teams verfügbar. Durch den Einsatz dieser Tools können Messdaten und Ergebnisse ohne allzu großen Aufwand gespeichert werden.

2.3.1.6 Qualitätssicherung von Prozessen und Produkten (+)

Prozesse und Arbeitsergebnisse objektiv bewerten (+): XP fordert keine explizierte Bewertung der Prozesse, Arbeitsprodukte und Dienstleistungen. Das einzige Kontrollinstrument, das die Methode verwendet, ist der Coach, der das Team in der Anwendung von XP leitet.

Objektiven Einblick liefern (+): Qualitätsprobleme können in einem XP-Projekt innerhalb des Teams einfach kommuniziert werden. Der Coach unterstützt eine entsprechende Korrektur. XP erstellt jedoch keine Aufzeichnungen über Qualitätssicherungsaktivitäten, wie dies von CMMI vorgeschlagen wird. Die Methode basiert auf der Annahme, dass dies wegen des intensiven Austauschs innerhalb des Teams, z. B. durch Pair Programming und das Arbeiten im selben Raum, nicht nötig ist.

2.3.1.7 Konfigurationsmanagement (+++)

Baselines erstellen (+++): Konfigurationselemente sind Code, Design, Tests und Anforderungen. Der Einsatz eines Konfigurationsmanagementsystems wird empfohlen, da kontinuierliche Integration stark darauf basiert. Baselines werden regelmäßig durch funktionale Tests festgelegt und zusätzlich am Ende jeder Iteration erstellt.

Änderungen verfolgen und steuern (+++): Änderungen werden durch Praktiken wie Pair Programming, Tests, Kundenpartizipation usw. gesteuert und nachverfolgt.

Integrität erzeugen (+++): XP fordert kontinuierliche Integration. Der Code ist aufgrund von Coding Standards einfach zu lesen und damit seine eigene Beschreibung. Audits werden informell durch Pair Programming, Kundenpartizipation und Tests durchgeführt.

2.3.1.8 Anforderungsentwicklung (++)

Kundenanforderungen entwickeln (++): Der Kunde ermittelt die Anforderungen und spezifiziert sie mittels Storycards und funktionaler Tests. Die Entwickler unterstützen ihn bei diesen Aufgaben. Die Anforderungsspezifikation bleibt jedoch relativ vage. Details müssen während der Entwicklung direkt mit dem Kunden besprochen werden.

Produktanforderungen entwickeln (++): Benutzeranforderungen werden in Produktanforderungen verfeinert und mittels Taskcards spezifiziert. Auch sie bleiben relativ vage.

Anforderungen analysieren und validieren (++): Die Programmierer beraten den Kunden während der Anforderungserhebung. Zusätzlich erlaubt die Akzeptanz sich verändernder Anforderungen und die Verwendung von Iterationen, dass die Anforderungen konstant analysiert und validiert werden. Die Erstellung von Betriebskonzepten und Szenarios wird durch das Schreiben von Funktionstests seitens des Kunden abgedeckt. Es findet jedoch keine erschöpfende Anforderungsanalyse zu Beginn des Projektes statt.

2.3.1.9 Technische Umsetzung (+++)

Lösungen für Produktkomponenten auswählen (+++): Alternative Lösungen werden am Anfang des Projekts durch Prototypen und später mittels Refactoring und iterative Entwicklung erprobt.

Design entwickeln (+++): Ein möglichst einfaches Design wird entwickelt. Code wird als Designdokument benutzt. Das Design wird iterativ durchgeführt.

Produktdesign implementieren (+++): XP verwendet eine Vielzahl an Implementierungspraktiken, z. B. Refactoring, Coding Standards, Pair Programming. Eine unterstützende Produktdokumentation wird entwickelt, wenn es der Kunde fordert.

2.3.1.10 Produktintegration (+++)

Produktintegration vorbereiten (+++): Durch kontinuierliche Integration und weil Integrationsschritte in XP sehr oft durchgeführt werden, ist eine sorgfältige Vorbereitung kritisch.

Schnittstellenkompatibilität sicherstellen (+++): Schnittstellenkompatibilität wird dadurch sichergestellt, dass bei jedem Integrationsschritt alle Tests ausgeführt werden.

Produktkomponenten zusammenbauen und Produkt ausliefern (+++): Komponentenzusammenstellung und Auslieferung werden ausgeführt. Kontinuierliche Integration und direkte Kundenbeteiligung helfen zusätzlich, dieses Ziel zu erreichen.

2.3.1.11. Verifikation (+++)

Verfikation vorbereiten (+++): Verifikation wird durch intensives Testen durchgeführt. Die Vorbereitung konzentriert sich deswegen auf dieses Thema. Ein Test-Framework sollte verwendet und daher entsprechende Vorbereitungen getroffen werden. Des Weiteren verwendet XP einen Test-First-Ansatz, d.h. alle Tests müssen vor dem Code geschrieben werden.

Partnerreviews durchführen (+++): Partnerreviews sind immer impliziter Teil von XP. Pair Programming, Refactoring und das Prinzip des kollektiven Eigentums des Codes implizieren konstante Partnerreviews.

Ausgewählte Arbeitsergebnisse verifizieren (+++): Überprüfungsmethoden sind hauptsächlich Partnerreviews und Tests, die beide kontinuierlich durchgeführt werden.

2.3.1.12. Validierung (+++)

Validierung vorbereiten (+++): Validierung erfolgt in XP-Projekten durch Kundenbeteiligung und häufige Auslieferungen. Hauptkriterium ist die Akzeptanz durch den Kunden.

Produkt- oder Produktkomponenten validieren (+++): Validiert wird das Produkt durch den Kunden. Dies ist durch seine Integration in das Team möglich und durch die kurzen Releasezyklen. Der Kunde kann die Anforderungen ändern oder neue aufstellen und bestimmt, welche im nächsten Release realisiert werden sollen. Der starke Einfluss des Kunden erhöht die Wahrscheinlichkeit, dass ihn das Produkt zufrieden stellt.

2.3.1.13. Organisationsweiter Prozessfokus (-)

Dieses Prozessgebiet wird nicht behandelt, weil es die Organisation betrifft, während XP nur das Projekt betrachtet. Es steht sogar mit XP im Konflikt: Wie bei anderen agilen Methoden auch werden während des Projekts oft Anpassungen gemacht – diese Verbesserungen sind jedoch auf das jeweilige Projekt beschränkt, da sie nicht dokumentiert werden sollen. Wissen über Verbesserungen ist damit an Personen gebunden und andere Projekte können nur davon profitieren, wenn Personen zwischen Projekten verschoben werden. Das Problem ist, dass in großen Organisationen zu viele Projekte parallel laufen. In diesem Fall können nicht alle Projekte von den speziellen Erfahrungen eines Projekts profitieren. Dazu kommt, dass die Informationen nicht permanent sind, da Mitarbeiter das Unternehmen verlassen können. Dieser Konflikt kann gemildert werden, indem unternehmensweite Sammlungen von bewährten Praktiken angelegt werden oder durch den Austausch von Erfahrungen zwischen Projekten.

2.3.1.14. Organisationsweite Prozessdefinition (0)

Dieses Prozessgebiet wird nicht von XP adressiert, da es die Organisationsebene behandelt, XP jedoch die Projektebene. Die Methode steht einer organisationsweiten Prozessdefinition jedoch nicht im Wege.

2.3.1.15. Organisationsweites Training (++)

Fähigkeit zum organisationsweiten Training aufbauen (++): Training findet bei XP zunächst während der Erforschungsphase statt. Deswegen benötigt ein XP-Projekt organisationsweite Trainingsmöglichkeiten. Pair Programming und Coaching können auch als Training angesehen werden, so dass XP die Trainingsmöglichkeiten einer Organisation weiter verbessert.

Benötigtes Training zur Verfügung stellen (++): Wie erwähnt wird die Weiterbildung explizit in der Erforschungsphase durchgeführt und implizit während des ganzen Projektes durch Coaching und Pair Programming. Bei Letzterem gibt es jedoch Mängel in Bezug auf die Dokumentation des Trainings und die Bewertung seiner Effektivität.

2.3.1.16. Integriertes Projektmanagement (++)

Den für das Projekt definierten Prozess nutzen (0): XP adressiert dieses auf die Organisationsebene bezogene Ziel nicht.

Mit relevanten Betroffenen koordinieren und zusammenarbeiten (+++): XP integriert und koordiniert Entwickler, Kunden, Tester und Management.

Die gemeinsame Vision des Projektes für IPPD[1] nutzen (+++): XP trägt viel zur Integration der Projektmitglieder und deren enger Zusammenarbeit bei. Dies und die intensive Kommunikation im Team helfen, eine gemeinsame Vision zu formen.

Integrierte Teams für IPPD organisieren (0): XP sieht nur ein einziges Team vor und trifft daher keine Aussagen zur Organisation von komplexen Teamstrukturen.

2.3.1.17 Risikomanagement (+++)

Risikomanagement vorbereiten (+): XP gibt nicht explizit an, wie Risikomanagement betrieben wird. Aber XP-Projekte treffen zweifelsohne gewisse Vorbereitungen.

Identifizieren und analysieren von Risiken (+++): XP fordert die Identifikation und Analyse von Risiken während der Planungsphase.

Risiken mindern (+++): Die Flexibilität, die durch kurze Iterationen erreicht wird, ist ein mächtiges Instrument in der Minderung von Risiken. Turner [Tu02] weist jedoch darauf hin, dass durch die fehlende spezifische Dokumentation Probleme bei langfristigen Risiken auftreten können.

2.3.1.18 Integrierte Teambildung (+++)

Team zusammenstellen (+++): XP bildet ein sich selbst organisierendes, interdisziplinäres Team, in dem alle relevanten Stakeholder integriert werden.

---

[1] IPPD steht für „Integrated product and process development"

Arbeit des Teams leiten (+++): Teamabläufe werden durch eine klare Definition der verschiedenen Rollen, Pair Programming, gemeinsame Verantwortung für den Code und den Fokus auf Kooperation und Kommunikation geregelt.

2.3.1.19 Integriertes Lieferantenmanagement (0)

Dieses Prozessgebiet wird von XP nicht adressiert. Analog zu „Management von Lieferantenvereinbarungen“ steht die Methode nicht im Konflikt mit dem Prozessgebiet und könnte durch entsprechende Praktiken erweitert werden.

2.3.1.20 Entscheidungsanalyse und -findung (-)

Laut [Tu02] ist die Fähigkeit, auf neue Situationen schnell reagieren zu können, bei agilen Methoden gegenüber einem formalen Bewertungsprozess größer. XP identifiziert und erprobt Alternativen informell und nicht auf die Weise, wie sie CMMI vorschlägt.

2.3.1.21 Organisationsweite Umgebung für die Integration (+)

Die Themen dieses Prozessgebiets werden von XP auf Projektebene, nicht jedoch auf Organisationsebene adressiert.

IPPD Infrastruktur bereitstellen (++): XP schafft Grundlagen für die Erreichung dieses Ziels mit der Einführung von Tools, intensiver Kommunikation und Kooperation. Zusätzlich wird dieses Ziel gefördert durch die Förderung der Fähigkeit zur Kommunikation und Kooperation sowie von Führungsfähigkeiten.

Personen für die Integration managen (+): Führungsmechanismen in den Entwicklungsteams sind demokratisch. Jedoch sind die Kunden und der „Big Boss“ mit Entscheidungsbefugnissen auf höherer Ebene ausgestattet.

2.3.1.22 Performanz der Organisationsweiten Prozesse (-)

XP setzt seinen Fokus bewusst auf Individuen und nicht auf Themen, die eine starke Orientierung auf den Prozess aufweisen. Laut [TJ02] steht die Idee, Prozesse zu messen und dafür Baselines und Modelle zu pflegen, im Konflikt mit dem agilen Manifest.

2.3.1.23 Quantitatives Projektmanagement (-)

Statistische Methoden legen ihren Fokus auf Prozesse und nicht auf Individuen. Daher stehen sie im Konflikt mit den agilen Prinzipien (laut [Tu02] sind sie „für Maschinen, aber nicht für Menschen“ geeignet). Weiterhin beruhen sie auf dem Gesetz der großen Zahlen und auf Mittelungseffekten in großen Teams. Da die meisten agilen Projekte klein sind, ist die Anwendung von statistischen Methoden damit fragwürdig.

2.3.1.24 Organisationsweite Innovation und Verbreitung (-)

Prozessverbesserungen und -anpassungen werden innerhalb des Projekts vorgenommen und nicht dokumentiert, daher können sie nicht in der gesamten Organisation verbreitet werden.

Dieser Punkt baut stark auf dem Prozessgebiet „Organisationsweiter Prozessfokus“ auf, das mit XP im Konflikt liegt (siehe 2.3.1.13).

2.3.1.25 Ursachenanalyse und Problemlösung (0)

Dieses Prozessgebiet wird von XP nicht adressiert. XP steht jedoch nicht im Konflikt mit dem Prozessgebiet und könnte durch entsprechende Praktiken erweitert werden.

2.4 Abdeckung der CMMI-Prozessgebiete durch agile Methoden

In 2.3 haben wir im Detail gezeigt, welche CMMI Prozessgebiete von XP unterstützt werden und welche im Konflikt stehen. Im Folgenden fassen wir die Abdeckung von CMMI Prozessgebieten durch XP zusammen.

Alle sieben Prozessgebiete des Levels 2 können mit der Methode erreicht werden. Von den 14 Prozessgebieten von Level 3 stehen nur zwei im Konflikt. Drei von den vier Prozessgebieten von Level 4 und 5 sind ebenso im Konflikt. Von den zwölf generischen Praktiken wurde nur eine als im Konflikt stehend bewertet.

Damit zeigen die Ergebnisse, dass Level 2 leicht und sogar Level 3 nahezu erreicht werden kann. Es ist allerdings praktisch unmöglich, Level 4 und 5 mit XP (und Scrum) zu erreichen ohne Anpassungen vorzunehmen die der Agilität zuwiderlaufen. Hauptsächlich Prozessgebiete, die sich explizit mit Prozessverbesserung befassen („Organisationsweiter Prozessfokus“, „Performanz der Organisationsweiten Prozesse“, „Quantitatives Projektmanagement“ und „Organisationsweite Innovation und Verbreitung“), stehen im Konflikt mit agilen Methoden. Außerdem existiert ein Konflikt bei dem Level 3 Prozessgebiet „Entscheidungsanalyse und -findung“, weil dort ein formaler Bewertungsprozess gefordert wird.

Der größte Teil der Prozessgebiete kann erreicht werden. Oft muss die Methode allerdings um zusätzliche Praktiken erweitert werden, um die Prozessgebiete vollständig zu erfüllen. Des Weiteren zeigt unsere Analyse, dass XP nur projektbezogene, aber nicht prozessbezogene Prozessgebiete abdeckt.

2.5 Zusammenhänge zwischen agilen Methoden und Reifegradmodellen

CMMI bewertet eine Organisation sowohl als Ganzes als auch ihre einzelnen Entwicklungsprozesse. Demgegenüber ist eine agile Methode ein individueller Entwicklungsprozess, oder auch nur ein Methoden-Framework oder Teil eines Entwicklungsprozesses. Daher sind die Konzepte nicht eins zu eins vergleichbar. Ihre Blickwinkel sind verschieden, aber es bestehen dennoch Zusammenhänge. Paulk fasst zusammen, dass CMM eine Methode zum Softwaremanagement ist, wohingegen agile Ansätze Methoden zur Softwareentwicklung sind [Pa01]. Sie können nicht nur nebeneinander bestehen, sie unterstützen sich sogar gegenseitig [Gl01].

Wir sind davon überzeugt, dass CMMI ein geeigneter Weg zur Verbesserung von Prozessen ist – auch im agilen Umfeld. Durch die Überprüfung der Abdeckung der Prozessgebiete einer agilen Methode werden Mängel und damit Verbesserungspotentiale aufgedeckt. Allerdings kann die Prozessverbesserung mit CMMI nur bis zu einem gewissen Grad erfolgen, da es einige Prozessgebiete gibt, die mit agilen Prinzipien im Konflikt stehen. Einige Prozessgebiete von Level 3 und die meisten von Level 4 und 5 sind nicht umsetzbar, ohne gegen Grundprinzipien agiler Methoden zu verstoßen. Dadurch würde die agile Methode geschwächt und einige ihrer Vorteile würden verloren gehen. Auch würde ein solches Vorgehen im Widerspruch zu dem Ziel von CMMI stehen, Prozesse zu verbessern. Dies geschieht, indem die agile Methode so gut wie möglich umgesetzt wird und nicht, indem man aus ihr eine völlig andere Methode macht, die nicht mehr agil ist.

Daraus folgern wir, dass der beste Ansatz zur Prozessverbesserung im agilen Umfeld darin besteht, sich auf CMMI Level 3 zu beschränken. Grundsätzlich sollte eine Organisation die CMMI-Levels im Hinblick auf die von ihr eingesetzten Prozessmodelle bewerten. Level 5 zu erreichen muss, abhängig von der eingesetzten Methode, nicht immer sinnvoll sein. Wie bei jedem traditionellen Prozess muss das Verfeinern von agilen Methoden auch als fortlaufende, erfolgskritische Aufgabe angesehen werden.

3 Zusammenfassung

Wir haben analysiert, welche CMMI Prozessgebiete von XP abgedeckt werden können. Dazu haben wir Prozessgebiete identifiziert, bei denen die Methode angepasst werden muss, um die CMMI Ziele zu erfüllen. Einige Prozessgebiete stehen im Konflikt mit der Methode und mit agilen Prinzipien im Allgemeinen. Die meisten Prozessgebiete können erfüllt werden, wenn man agile Methoden verwendet, und durch CMMI können Mängel der agilen Methoden identifiziert werden.

Wir kommen deshalb zum dem Schluss, dass Prozessverbesserung mittels CMMI auch bei agilen Methoden erfolgen kann. Allerdings kann sie nur bis zu einem bestimmten Grad durchgeführt werden, da einige Prozessgebiete, hauptsächlich aus Level 4 und 5, im Konflikt mit agilen Prinzipien stehen. Eine Erweiterung des Blickwinkels agiler Methoden auf organisationsweite Themen würde helfen, bestehende Konzepte beständiger Prozessverbesserung zu nutzen.

Wenn diese Konzepte im agilen Umfeld eingesetzt werden, wird dies die Akzeptanz agiler Methoden fördern. Es ist jedoch noch immer schwierig, CMMI-Assessments für agile Methoden durchzuführen. Die von CMMI vorgeschlagenen spezifischen Praktiken unterscheiden sich oft von den Vorgehensweisen agiler Methoden. Deshalb haben Assessoren beträchtliche Probleme bei der Analyse eines agilen Projektes. Um diese Situation zu verbessern, sollte ein Katalog von Praktiken und Subpraktiken entwickelt werden, der typischerweise von agilen Methoden eingesetzt wird, um die CMMI-Ziele zu implementieren.

In diesem Beitrag wurde beispielhaft XP untersucht. Um unsere Ergebnisse zu verallgemeinern, sollten auch andere agile Methoden analysiert werden. Daneben sollten konkrete Leitfäden entwickelt werden, um agile Methoden so zu erweitern, dass diejenigen Prozessgebiete, die nicht im Konflikt stehen, von ihnen vollständig erfüllt werden können. Dafür kann unsere Arbeit als Ausgangspunkt dienen.

Literaturverzeichnis

[An05] Anderson, D. J.: Stretching Agile to fit CMMI Level 3 - the story of creating MSF for CMMI Process Improvement at Microsoft Corporation. Agile Development Conference (ADC'05), 2005, 193-201.

[Be04] Beck, K.: Extreme Programming explained: Embrace change. Addison-Wesley, 2004.

[Be01] Beck, K. et. al.: Manifesto for Agile Software Development. 2001, http://AgileManifesto.org, zuletzt besucht am 14.12.2006.

[CH01] Cockburn, A.; Highsmith, J.: Agile Software Development: The People Factor. Computer, Nov. 2001, 131-133.

[Do06] Doernhoefer, M.: Surfing the Net for Software Engineering Notes. ACM SIGSOFT Software Engineering Notes 31 (1), Jan. 2006, 5-13.

[Fr05] Fritzsche, M.: Agile Methoden im industriellen Umfeld. Diplomarbeit, Technische Universität München, 2005.

[Gl01] Glazer, H.: Dispelling the Process Myth: Having a Process Does Not Mean Sacrificing Agility or Creativity. CrossTalk, Nov. 2001, 27-30.

[Hi05] Highsmith, J.: Extreme Programming: Agile Project Management Advisory Service White Paper. 2000, http://www.cutter.com/freestuff/ead0002.pdf, zuletzt besucht am 03.10.2005.

[KA04] Kähkönen, T.; Abrahamsson, P.: Achieving CMMI Level 2 with Enhanced Extreme Programming Approach. Lecture Notes in Computer Science, Vol. 3009, Jan. 2004, 378-392.

[Ke05] Keil, P.: Principal Agent Theory and its Application to Analyze Outsourcing of Software Development. Proc. Int. Workshop on Economics-Driven Software Engineering Research (EDSER 2005). IEEE Computer Society, 2005.

[Pa01] Paulk, M. C.: Extreme Programming from a CMM Perspective. IEEE Software, November/Dezember 2001.

[PP03] Poppendieck, M.; Poppendieck, T.: Lean Development – An Agile Toolkit. Addison-Wesley, 2003.

[SB01] Schwaber, K.; Beedle, M.: Agile Software Development with SCRUM. Prentice Hall, 2001.

[So02] Software Engineering Institute: Capability Maturity Model Integration (CMMI), Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1). Software Engineering Institute, 2002.

[TJ02] Turner, R.; Jain, A.: Agile Meets CMMI: Culture Clash or Common Cause? Extreme Programming and Agile Methods – XP/Agile Universe 2002: Second XP Universe and First Agile Universe Conference. Springer, 2002, 153-165.

[Tu02] Turner, R.: Agile Development: Good Process or Bad Attitude? Product Focused Process Improvement. Proc. 4th Int. Conf., PROFES 2002. Springer, 2002, 134-144.

[Va04] Van Solingen, R.: Measuring the ROI of Software Process Improvement. IEEE Software, Mai 2004, 32-38.

[Vm06] V-Modell XT Portal, http://www.v-modell-xt.de, zuletzt besucht am 24.09.2006.