Seminar Software Engineering (Master)
Sommersemester 2005

Agile Softwareentwicklung

Das agile Manifest

von

Jochen Neuhaus	422030
Christian Baun	511653

Dienstag, 3. Mai 2005

Agile Softwareentwicklung

Was ist Agil?

Das agile Manifest / agile Prinzipien

Agile Prozesse

Bewertung agiler Methoden

Diskussion / Fragen

Jochen Neuhaus, Christian Baun

Agile Softwareentwicklung

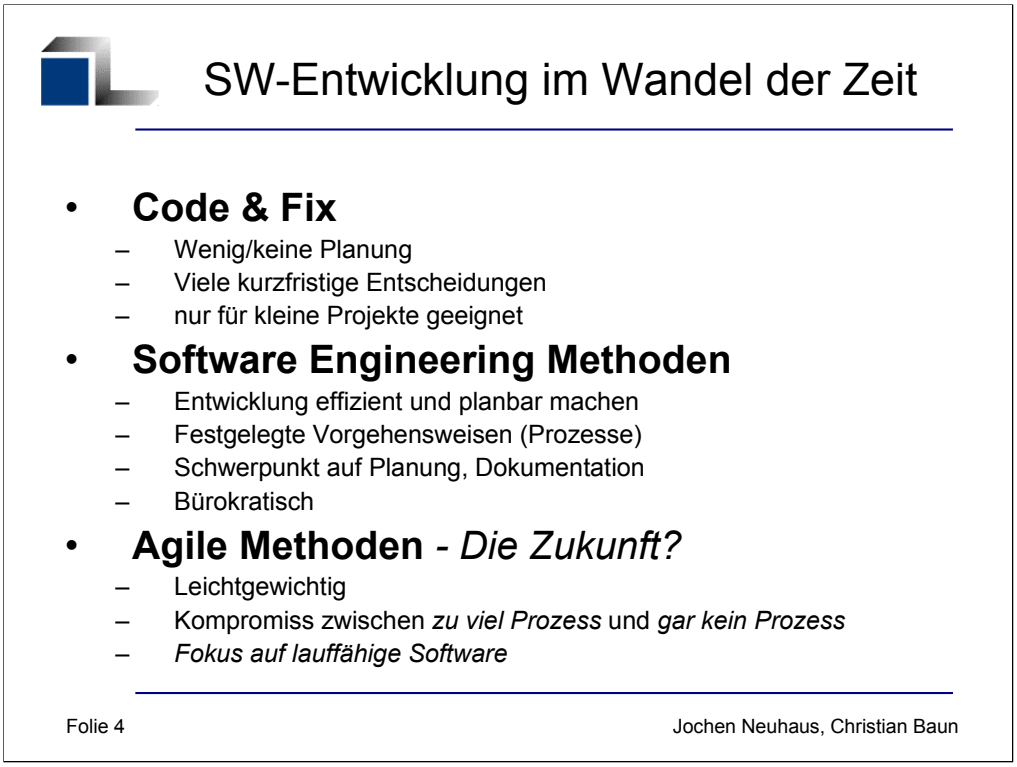
Was ist Agil?

Das agile Manifest / agile Prinzipien

Agile Prozesse

Bewertung agiler Methoden

Diskussion / Fragen



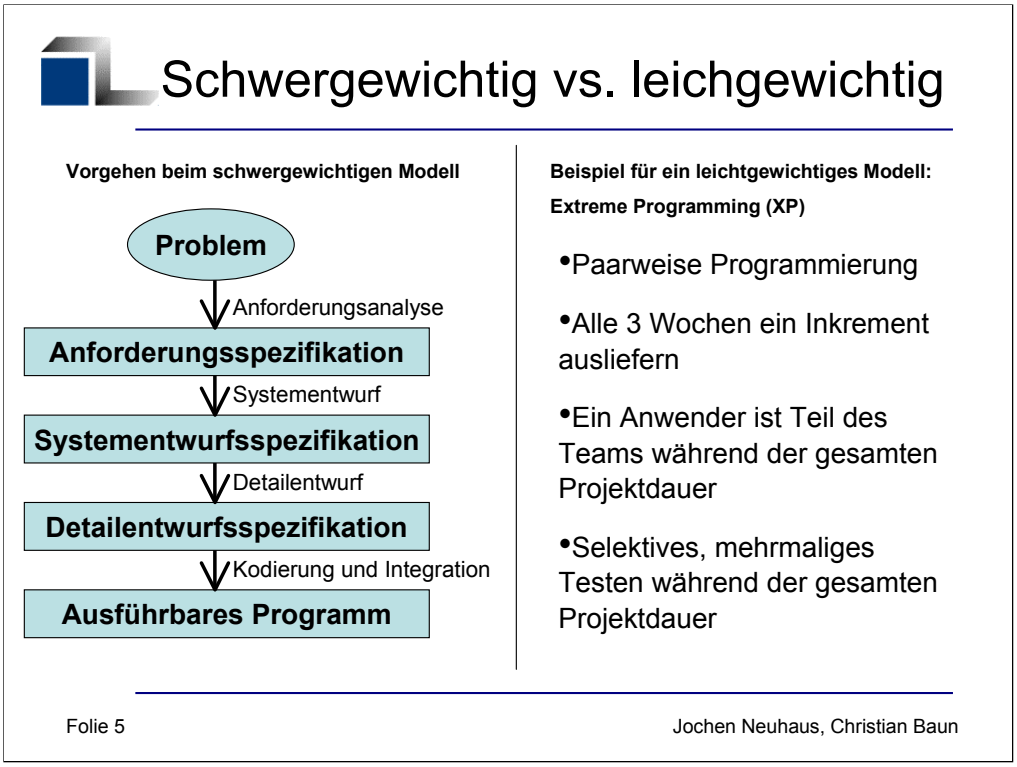
Als **Code & Fix** wird das planlose erstellen einer Software bezeichnet. Man beginnt einfach mit der Implementierung und trifft Designentscheidungen wenn sie auftreten ohne auf langfristige Effekte zu achten. Je größer ein Projekt wird, desto komplizierter wird die Wartung und Weiterentwicklung. Am Ende der Entwicklung sind oft sehr lange Testphasen nötig.

Ingenieurtechnische Methoden: Das Software Engineering hat seinen Ursprung in den Ingenieurswissenschaften. Man geht von der Idee aus, dass Software geplant werden kann wie ein Bauwerk. Sobald ein exakter Plan erstellt ist, kann die Konstruktion von weniger qualifizierten Mitarbeitern ausgeführt werden → Beispiel Brückenbau.

Agile Methoden sind eine Abkehr von den festgelegten Vorgehensweisen der schwergewichtigen Prozesse. Sie versuchen die Bürokratie zu vermindern und stellen nicht den Plan sondern die lauffähige Software in den Fordergrund.

Kodierung wird nicht als minderqualifizierte Konstruktionsarbeit angesehen, sondern ebenfalls als Design (Kompilieren und Linken entsprechen der Konstruktion).

Der Weg ist flexibel. Das Ziel ist es nicht. (Martin Fowler)
Plans are nothing, planing is everything. (Dwight D. Eisenhower)



Jede Phase beim schwergewichtigen Modell hat eine

- **definierte Eingabe** (Produkte/Dokumentation) und
- **definierte Ausgabe** (Produkte/Dokumentation).

Agile Softwareentwicklung

Was ist Agil?

Das agile Manifest / agile Prinzipien

Agile Prozesse

Bewertung agiler Methoden

Diskussion / Fragen

Das Agile Manifest wurde 2001 von Vertretern verschiedener agiler Softwareentwicklungsmethoden als Definition gemeinsamer Werte veröffentlicht und führte zur Gründung der Agile Alliance (*http://agilealliance.org/*).

Das **Agile Manifest**:

• **Individuen und Interaktion** sind wichtiger sind als Prozesse und Werkzeuge.

• **Lauffähige Software** ist wichtiger ist als umfangreiche Dokumentation.

• Die **Zusammenarbeit mit dem Kunden** ist wichtiger ist als Vertragsverhandlungen.

• Auf **Änderungen reagieren** ist wichtiger als einem Plan zu folgen.

Prinzipien des Agilen Softwareentwicklungsprozesses (1/12)

Stelle den Kunden durch frühzeitige und regelmäßige Auslieferung nützlicher Software zufrieden

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Folie 8 Jochen Neuhaus, Christian Baun

Oft weiß ein Kunde zu Beginn eines Projekts noch nicht genau, was er eigentlich haben möchte. Durch wiederholte Auslieferung lauffähiger Software (mit vermindertem Funktionsumfang) kann der Kunde herausfinden, welche Software er eigentlich benötigt. Das Ziel ist, für den Kunden die Software zu erstellen, die er benötigt und nicht die, die er vor Projektbeginn in Auftrag gegeben hat.

Bei traditionellen schwergewichtigen Prozessen ist die Auslieferung von Software an den Kunden mit einem hohen Verwaltungsaufwand verbunden. Deshalb tendiert man leicht dazu Software nur einmal auszuliefern: Am Ende der Entwicklung.

Prinzipien des Agilen Softwareentwicklungsprozesses (2/12)

Sich ändernde Anforderungen werden begrüßt, selbst wenn diese spät in der Entwicklung auftreten. Agile Prozesse machen Änderungen als Wettbewerbsvorteil für den Kunden nutzbar

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Folie 9 Jochen Neuhaus, Christian Baun

Es sind nicht möglich alle Anforderungen im Vorfeld zu erkennen und zu klären. Gerade bei längeren Projekten können sich Anforderungen im Laufe der Zeit ändern. Solche Änderungen sollen nicht abgeblockt werden, sondern angenommen werden, da die Software so für den Kunden von größerem Nutzen wird.

Vorraussetzung für diese Art der Entwicklung ist eine andere Art der Zusammenarbeit mit dem Kunden (Kein traditionelles Lastenheft, Kunde ist in Entwicklung eingebunden, im Vertrag können nicht Kosten, Dauer und Umfang festgelegt werden).

Einige agile Prozesse fordern, dass ein Endanwender immer im Projektteam integriert ist.

Prinzipien des Agilen Softwareentwicklungsprozesses (3/12)

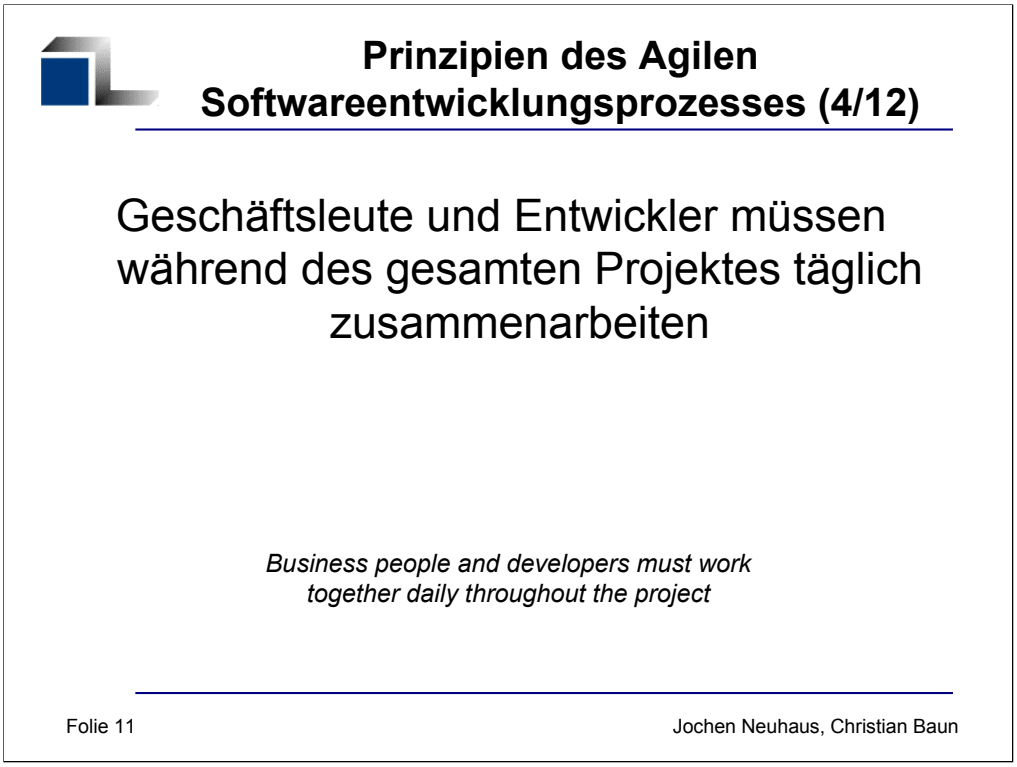
Lauffähige Software soll häufig, in Abständen von wenigen Wochen bis Monaten ausgeliefert werden. Kurze Zeitspannen sollten bevorzugt werden

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale

Folie 10 Jochen Neuhaus, Christian Baun

Je kürzer die Abstände sind, desto besser kann auf Änderungswünsche reagiert werden und desto besser kann der Kunde Änderungswünsche äußern.

Durch häufige Integration aller Subsysteme wird vermieden, dass unabhängige nicht zusammen lauffähige Komponenten entstehen (kann bei großen Entwicklungsteams vorkommen).



Ein agiler Entwicklungsprozess darf dem Team nicht vom Management aufgezwungen werden, das Team muss den Prozess selbst wählen dürfen. Ist das nicht der Fall, so kann es leicht zu Widerstand bei den Entwicklern kommen. Ein agiler Prozess kann aber nur funktionieren, wenn die Mitarbeiter motiviert bei der Sache sind.

Ein weiterer Punkt ist, dass Entwickler in der Lage sein müssen, alle technischen Entscheidungen zu treffen, während das Management für die geschäftlichen Entscheidungen zuständig ist. Hierbei ist eine enge Zusammenarbeit nötig, um jeweils die besten Lösungen zu finden. Jeder Entwickler muss mit Experten für betriebswirtschaftliche Dinge in Kontakt treten können.

Die Verantwortung für das Projekt liegt also nicht nur bei den Führungskräften, sondern wird zwischen Entwicklern und Managern aufgeteilt. Beide sind gleichberechtigt bei der Führung des Projekts.

Prinzipien des Agilen Softwareentwicklungsprozesses (5/12)

Baue Projekte um motivierte Individuen herum auf. Stell ihnen die benötigte Umgebung und Unterstützung zur Verfügung und vertraue auf sie

Build projects around motivated individuals.
Give them the environment and support they need,
and trust them to get the job done.

Folie 12 Jochen Neuhaus, Christian Baun

„*Putting People First*“ ist eine der Schlüsselaussagen agiler Methoden.

Traditionelle Prozesse sehen Entwickler oft nur als Ressourcen. Es werden lediglich Rollen wie Entwickler, Tester usw. festgelegt, die Individuen sind in diesen Rollen austauschbar. Es herrscht die Meinung, dass die Menschen, die die Arbeit tun nicht diejenigen sind, die am Besten entscheiden können, wie die Arbeit getan werden soll (Brückenbau, traditionelles Engineering).

Diese Sicht der Mitarbeiter als austauschbare Codegeneratoren wirkt sich negativ auf die Moral aus und senkt die Bereitschaft sich zu engagieren. In agilen Prozessen werden die Mitarbeiter sie dagegen als förderungs- und vertrauenswürdig angesehen. Damit ein agiler Prozess funktioniert, benötigt er qualifizierte Mitglieder, die gut zusammenarbeiten können.

Vertrauen in den Einzelnen ist selten ein Merkmal großer Unternehmen (Abteilungen). Ein Grund, warum große Unternehmen in der Regel in viele Abteilungen unterteilt sind ist, dass diese sich gegenseitig kontrollieren können. Gegenseitige Kontrolle wirkt sich negativ auf die Moral aus.

Prinzipien des Agilen Softwareentwicklungsprozesses (6/12)

Die effizienteste und effektivste Methode um Wissen innerhalb des Entwicklungsteams zu verteilen oder an das Team heranzutragen ist ein persönliches Gespräch von Angesicht zu Angesicht

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Folie 13 Jochen Neuhaus, Christian Baun

Wenige Projekte schlagen nur aufgrund spezifischer Technologien, Werkzeugen oder ähnlichem fehl. Ein Hauptgrund ist meist fehlende oder nicht funktionierende Kommunikation. → *„Wenn Kommunikation funktioniert, setzen technische Probleme keine Projekte aufs Spiel.“* (Nicolai Josuttis)

Kommunikation umfasst außer der Interaktion zwischen Teammitgliedern auch die Interaktion zwischen verschiedenen Teams, mit dem Management und mit dem Kunden. Direkte Kommunikation von Angesicht zu Angesicht ist dabei die effektivste Form.

Die Kommunikation innerhalb des Teams kann durch entsprechende räumliche Gegebenheiten gestaltet werden. Dies hat direkte Auswirkungen auf den Projektfortschritt.

Indirekte Kommunikation bedeutet einen höheren Aufwand als direkte Kommunikation (Antwortzeiten bei Email/Anrufbeantworter). Dadurch wird leicht auf die Kommunikation verzichtet und Klarstellungen werden durch Annahmen ersetzt, was schnell zu Fehlern führen kann.

Alistair Cockburn bewertet die Effektivität von Kommunikationsformen folgendermaßen (in aufsteigender Reihenfolge):
1.Papier
2.Tonaufzeichnungen
3.Zwei Personen via Email
4.Videoband
5.Zwei Personen am Telefon
6.Zwei Personen am Whiteboard
Papier ist eine der am meisten verbreiteten Kommunikationsformen (vielleicht weil es leichter zur Dokumentation verwendet werden kann?)

Prinzipien des Agilen Softwareentwicklungsprozesses (7/12)

Lauffähige Software ist das wichtigste Maß für den Projektfortschritt

Working software is the primary measure of progress.

Folie 14 Jochen Neuhaus, Christian Baun

Höchstes Ziel ist immer die lauffähige Software, die für den Kunden den Geschäftswert darstellt. Deshalb werden bei der agilen Entwicklung Maßzahlen wie Anzahl der Dokumente, Lines of Code oder ähnliches nicht zum Messen des Projektfortschritts verwendet, sondern nur die für den Kunden nützliche Software.

Durch das Prinzip der häufigen Integration zu einem lauffähigen Produkt und durch die kurzen Inkremente entsteht im Projektverlauf immer wieder lauffähige Software, die auch an den Kunden ausgeliefert wird. Dadurch kann der Kunde den Projektfortschritt direkt bewerten.

Dadurch verhindern agile Softwareentwicklungsprozesse, dass einzelne Entwickler sich hauptsächlich mit Dingen beschäftigen, die nicht primär dem Ziel einer lauffähigen Software dienen, wie übermäßige Dokumentation zu erstellen, die nicht direkt benötigt wird um Software zu erstellen.

Prinzipien des Agilen Softwareentwicklungsprozesses (8/12)

Agile Prozesse fördern die dauerhafte Entwicklung. Geldgeber, Entwickler und Nutzer sollten in der Lage sein eine konstante Fortschrittsgeschwindigkeit dauerhaft beizubehalten

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Softwareentwicklung ist eher ein Marathon als ein Sprint.

Das Ziel muss nicht in möglichst kurzer Zeit erreicht werden.

Es müssen dauerhaft substanzielle Verbesserungen einfließen können.

Extreme Programming hat beispielsweise als Prinzip eine Beschränkung der Wochenarbeitszeit, um eine Überarbeitung der Mitarbeiter zu verhindern.

Prinzipien des Agilen Softwareentwicklungsprozesses (9/12)

Ständige Aufmerksamkeit auf die technische Güte und gutes Design verbessert die Agilität

Continuous attention to technical excellence and good design enhances agility.

Folie 16 Jochen Neuhaus, Christian Baun

Das Design ist nicht nur zu Beginn eines Projekts wichtig und die Qualitätssicherung nicht nur am Ende eines Projekts.

Das gesamte Team muss sich für die Qualitätssicherung und ein gutes Design verantwortlich fühlen.

Prinzipien des Agilen Softwareentwicklungsprozesses (10/12)

Einfachheit – die Kunst die Menge der Arbeit, die nicht getan wird, zu maximieren - ist entscheidend

Simplicity – the art of maximizing the amount of work not done – is essential.

Es ist nicht von Vorteil, sich zu Beginn eines Projekts zu sehr auf Verallgemeinerungen zu konzentrieren. Diese führen zu einer unnötig komplizierten Architektur.

Man will vermeiden, Code zu implementieren, der nicht direkt für das aktuelle Inkrement verwendet wird. Es soll nur das Programmiert werden, was mit den aktuellen Anforderungen des Kunden zu tun hat und nicht das, was vielleicht in Zukunft gebraucht werden könnte.

Prinzipien des Agilen Softwareentwicklungsprozesses (11/12)

Selbstorganisierende Teams sind der Schlüssel zu guten Architekturen, Anforderungsspezifikationen und Entwürfen

The best architectures, requirements, and designs emerge from self-organizing teams.

Bei Unternehmen, die schwergewichtige Prozesse einsetzten, gilt in der Regel die These, dass Teams sich nur schwer kontrollieren können. Der gängige Lösungsansatz ist eine Steigerung der Kontrolle der Mitarbeiter.

Nachteilig ist, dass Kontrolle von den Mitarbeitern oft als Misstrauen verstanden wird.

➔Mitarbeiter werden durch Kontrolle entmutigt.

➔Motivation der Mitarbeiter sinkt.

Agile Softwareentwicklungsprozesse ermutigen die Mitarbeiter sich innerhalb der Teams selbst zu organisieren, da diese selbst über ihre Fähigkeiten am besten bescheid wissen. Von der Unternehmensleitung getroffene Teamorganisationen gehen oft an den Fähigkeiten und Vorlieben der Mitarbeiter vorbei.

Durch die Selbstorganisation der Teams in agilen Softwareentwicklungsprozesse kann die Motivation der Mitarbeiter gesteigert werden.

Prinzipien des Agilen Softwareentwicklungsprozesses (12/12)

In regelmäßigen Abständen sollte das Team überlegen, wie es effektiver werden kann und dann sein Verhalten entsprechend ändern und anpassen

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Die Teams finden durch Selbstreflektion in regelmäßigen Abständen eigene Entwicklungsprozesse und steigern so ihre Effektivität.

Es existiert kein Entwicklungsprozess, der für alle denkbaren Teams und Projekte optimal ist.

Besser: Entwicklungsprozesse, die den individuellen Bedürfnissen eines Teams und Projekts angepasst werden können.

Agile Softwareentwicklung

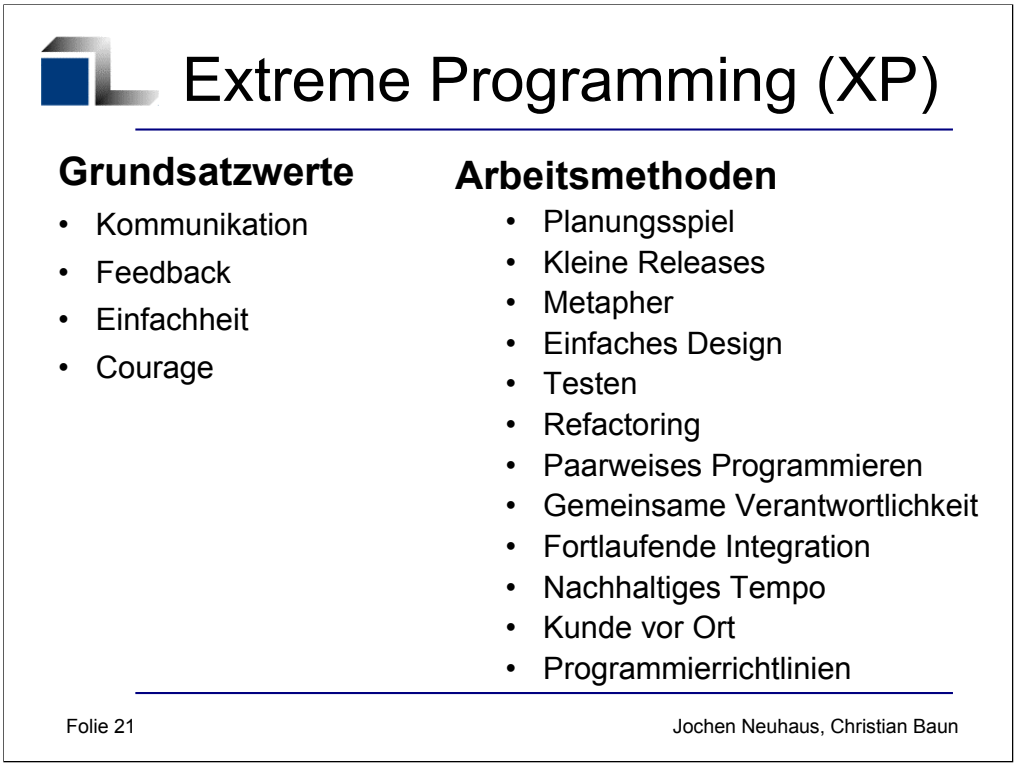
Was ist Agil?

Das agile Manifest / agile Prinzipien

Agile Prozesse

Bewertung agiler Methoden

Diskussion / Fragen



Extreme Programming (XP) ist die bekannteste agile SE-Methode.

Sie wurde von Kent Beck und Ward Cunningham Ende der 80er Jahre entwickelt.

Zum ersten mal wurde XP im C3-Projekt von Chrysler eingesetzt. (Lohnabrechnungen – Payroll)

Die 12 Arbeitsmethoden müssen nicht immer alle angewendet werden, das Team kann je nach den Erfordernissen des Projekts einzelne Methoden weglassen. Es wird empfohlen, dass diese Anpassung des Prozesses erst durchgeführt werden, wenn alle Mitarbeiter Erfahrungen mit XP gesammelt haben (z.B. nach einigen Iterationen).

Crystal

- Familie von Vorgehensmodellen:
  - Crystal Clear (bis 6 Mitarbeiter)
  - Crystal Yellow (bis 20 Mitarbeiter)
  - Crystal Orange (bis 40 Mitarbeiter)
  - …
- Inkrementelle Entwicklung - Releases alle 1-4 Monate
- Wenig Angaben zu detaillierten Prozessschritten
  → Vertrauen in Selbstorganisation der Entwickler
- Regelmäßige Anpassungen des Prozesses durch die Mitarbeiter

Die Crystal Familie von Prozessen wurde Anfang der 90er Jahre von Alistair Cockburn bei IBM entwickelt.

Weitere Prozesse für bis zu 500 Mitarbeiter sollen vorhanden sein, wurden aber bisher nicht publiziert. Im Farbcode spiegelt sich die „schwere“ des Prozesses wieder – je dunkler, desto höher die Zahl der geforderten Dokumente und Rollen im Projekt.

Alle Crystal-Vorgehensmodelle sind inkrementell. Releases an den Kunden finden alle 1 bis maximal 4 Monate statt.

Jedes Release wird in mehreren Reviews (User Viewings) durch die Anwender qualitätsgesichert.

Scrum

- Einfaches Modell, wenige Festlegungen
- Teams organisieren sich weitgehend selbst
- Vorgehen und Methoden werden fortlaufend angepasst
- Entwicklungsprozess ist unterteilt in Sprints (Iterationen)
  - Dauer je Sprint: 30 Tage
- Vor jedem Sprint werden die zu entwickelnden Funktionalitäten festgelegt
- Keine Einmischung von außen (Management) während eines Sprints
- Tägliche Teambesprechung (*Scrum*)

Scrum ist ein einfaches Vorgehensmodell, das wenige Festlegungen trifft. Das zentrale Element von Scrum ist der Sprint. Ein Sprint ist die Umsetzung einer Iteration. Als Iterationslänge schlägt Scrum ca. 30 Tage vor.

Vor dem Sprint werden Anforderungen des Kunden an das Produkt in einem Sprint-Backlog gesammelt. Dieser enthält auch technische Aufgaben. Die Anforderungen werden informell skizziert und während des Sprints umgesetzt. Der Kunde legt die Prioritäten der Anforderungen fest. Entsprechend der Prioritäten werden die Anforderungen während des Sprints umgesetzt. Zum Sprint organisiert sich das Entwicklungsteam selbst. Es sind also keine detaillierten methodischen Vorschriften notwendig. Die Anforderungen aus dem Sprint-Backlog teilt das Team selbständig auf und jeder Entwickler schätzt den Aufwand der ihm zugewiesenen Aufgaben ab.

Jedem Tag findet ein kurzes Scrum-Meeting statt. Dieses dauert ca. 15 Minuten. Während des Meetings berichtet jeder Entwickler über seinen aktuellen Arbeitsfortschritt seit dem vergangenen Meeting und macht Zusagen für die Ergebnisse bis zum nächsten Meeting. Aufgetretene Probleme werden nach dem Meeting geklärt. Diese täglichen Kurzmeetings dienen dem Austausch von Informationen im Team und der zusätzlich zur Disziplinierung. Die beteiligten Entwickler fühlen sich in der Regel an die von ihnen während des Scrum-Meetings gegebenen Zusagen gebunden. Während eines Sprints dürfen die Anforderungen aus dem Sprint-Backlog nicht geändert werden.

Vorteil: Das Team kann sich ungestört auf die Umsetzung konzentrieren.
Nach einem Sprint wird das Sprint-Ergebnis einem informellen Review durch Team und Kunden unterzogen. Dazu wird das Ergebnis des Sprints (die laufende Software) vorgeführt, technische Eigenschaften werden präsentiert. Der Kunde prüft, ob das Sprint-Ergebnis seinen Anforderungen entspricht, eventuelle Änderungen werden im Sprint-Backlog dokumentiert.

DSDM (Dynamic System Development Method)

Prinzipien

1. Aktive Einbeziehung des Kunden in das Team
2. Die Entscheidungsgewalt liegt beim Team
3. Eine regelmäßige Lieferung von fertigen (Teil-) Produkten wird angestrebt
4. Jede (Teil-) Lieferung muss einen Geschäftswert für den Anwender darstellen, dies ist Abnahmekriterium (*Business-Value-Driven*)
5. Inkrementelle Entwicklung ist notwendig, um adäquate Lösung zu erzielen
6. Während der Entwicklung sind alle Änderungen zurücknehmbar oder umkehrbar
7. Anforderungen werden auf einem relativ hohen Niveau festgeschrieben
8. Testen ist integraler Bestandteil des gesamten Prozesses
9. Die kooperative Zusammenarbeit aller Projektbeteiligten ist überaus wichtig

DSDM entstand Mitte der 90er Jahre in Großbritannien. Es wurde von einem Konsortium aus 17 Firmen entwickelt. Mittlerweile sind mehr als tausend Firmen Mitglied im DSDM Konsortium. Allerdings wird es selten außerhalb von Großbritannien eingesetzt.

DSDM beinhaltet ähnlich der schwergewichtigen Prozesse mehr Infrastruktur, folgt aber den agilen Prinzipien.

DSDM (Dynamic System Development Method)

- Projektphasen:
  1. Machbarkeitsstudie
  2. Geschäftsstudie
  3. Functional Model Iteration (*Spezifikation und Prototypen*)
  4. Design und Konstruktion (*Entwicklung und Test*)
  5. Implementierung (*Übergabe an Kunden*)
- Phasen 3-6 werden iterativ durchlaufen
- Für die Nutzung von DSDM sind Lizenzgebühren zu entrichten

Folie 25 Jochen Neuhaus, Christian Baun

In der Machbarkeitsstudie wird geprüft, ob DSDM für das Projekt in Frage kommt.

Die *Geschäftsstudie* besteht aus mehreren Workshops für die Entwickler um das Geschäftsfeld zu verstehen.

Functional Model Iteration: Es wird eine detaillierte Spezifikation des Produktes sowie eine Architektur entwickelt.

Design and Build Iteration: Das Produkt wird entwickelt und getestet.

Implementation: Das fertige Produkt wird an die Anwender übergeben.

Die Phasen 3 bis 5 sind iterativ zu durchlaufen. Analysiert, entwickelt und ausgeliefert wird also jeweils ein Teilprodukt.

DSDM ist in Deutschland kaum verbreitet (Dem DSDM Konsortium ist nur eine Anwendung bekannt). Dies kann daran liegen, dass das DSDM Konsortium für die Nutzung von DSDM Lizenzgebühren erhebt.

Adaptive Software Development

- *„Planen ist etwas paradoxes in einer adaptiven Umgebung“* (Jim Highsmith)
- Abgeleitet aus der Chaos-Theorie
- 3 nicht-lineare überlappende Phasen:
  1. Spekulation
  2. Zusammenarbeit
  3. Lernen
- Resultate sind unvorhersagbar
- Änderungswünsche zeigen den Weg zum gewollten Ergebnis

Folie 26 Jochen Neuhaus, Christian Baun

Entwickelt von Jim Highsmith

Bei traditionellen Modellen werden Änderungswünsche als Fehler angesehen die korrigiert werden sollten.

Chaos und Unordnung gelten bei ASD als Chance. Es gilt einen Balanceakt zwischen Freiheit und Regelung zu finden.

Es besteht die Gefahr, dass Projekte in die Anarchie ausarten können.

Die 3 Phasen sind die Bestandteile des Adaptive Development Lifecycle.

- Spekulation statt Plan
- Zusammenarbeit statt Implementierung
- Lernen statt Review

Die drei Phasen bilden einen Zyklus. Jeder Zyklus liefert ein für den Kunden nützliches Stück Software.

Mehrere Zyklen werden wiederholt, allerdings nicht nacheinander sondern mit Überschneidungen.

Feature Driven Development

- 5 Prozesse:
  1. Entwickeln eines Gesamtmodells (*develop an overall model*)
  2. Erstellen einer Feature-Liste (*build a feature list*)
  3. Planung pro Feature (*plan by feature*)
  4. Entwurf pro Feature (*design by feature*)
  5. Implementierung pro Feature (*build by feature*)
- 2 Arten von Entwicklern:
  1. Klassenverantwortliche (*class owner*)
  2. Chefprogrammierer (*chief programmer*)
- Iterationen haben eine Dauer von 2 Wochen

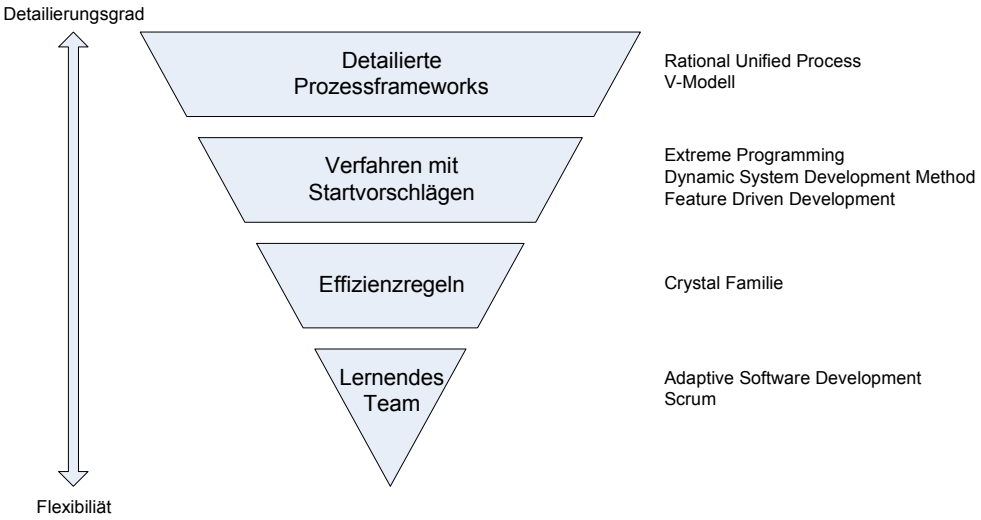
FDD wurde von Jeff De Luca und Peter Coad entwickelt.

Prozesse 4 und 5 werden pro Feature wiederholt.

Der **Chefprogrammierer** leitet ein Feature-Team während zwei Wochen bei der Implementierung eines Features. Für die Position des Chefprogrammierers kommen überdurchschnittliche produktive Programmierer einer Unternehmung in Frage. Er trägt die Verantwortung für die Implementation und das Testen eines Features.

Für jede Klasse der Applikation gibt es einen **Klassenverantwortlichen**, der die Klasse entwirft und implementiert. Diese Organisation führt zu einer konsistenten Implementation einer Klasse und erhöht die Motivation der Mitarbeiter.

Einordnung agiler Methoden



Quelle: Jens Coldewey . Einführung in Agile Entwicklung

Agile Softwareentwicklung

Was ist Agil?

Das agile Manifest / agile Prinzipien

Agile Prozesse

Bewertung agiler Methoden

Diskussion / Fragen

CMM(I) - Capability Maturity Model (Integration)

- Werkzeug, um Softwareprojekte *schneller*, *preiswerter* und mit *besseren Ergebnissen* durchzuführen
- Modell zur Bewertung von
  - Capability – Fähigkeit

  und
  - Maturity – Reifegrad

  eines Softwareentwicklungsprozesses bzw. einer Softwareentwicklungsorganisation
- Entwickelt vom Software Engineering Institute der Carnegie-Mellon University im Auftrag des Department of Defense (*DOD*) 1986 - 1991

Folie 30 Jochen Neuhaus, Christian Baun

Capability Maturity Model Integration (CMMI) dient der Verbesserung von Softwareprozessen und ist der Nachfolger von CMM.

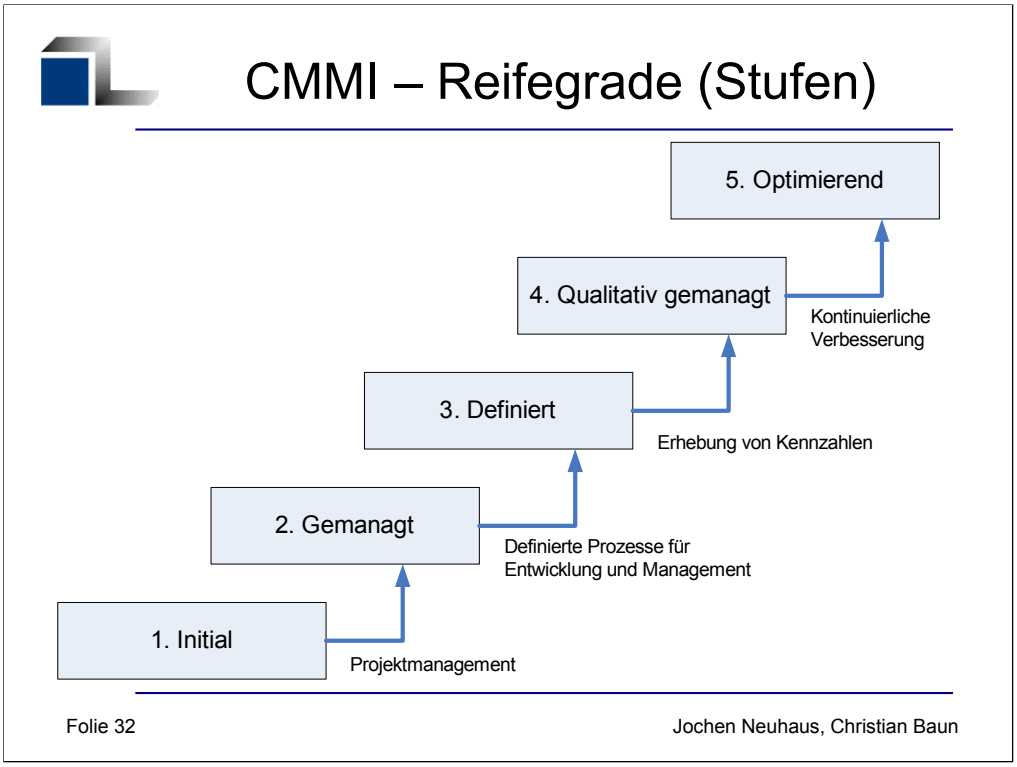
• Ziel von CMM: Leistungsfähigkeit von Software-Lieferanten beurteilen.
• TQM auf Software-Entwicklung übertragen.
• Reifegrade des Software-Prozesses systematisch bis zum höchsten Reifegrad erhöhen.
• Basiert auf den Grundgedanken des modernen Qualitätsmanagement.

Geschichtliche Entwicklung

•1986 begann das Software Engineering Institute (SEI) an der Carnegie Mellon University in Pittsburgh, das dem US-Verteidigungsministerium untersteht, mit der Entwicklung eines Systems zur Bewertung der Reife von Softwareprozessen.
•1991 wurde das Modell als Capability Maturity Model 1.0 herausgegeben.
•1993 wurde CMM überarbeitet und in der Version 1.1 bereitgestellt.
•1997 wurde CMM 2.0 kurz vor der Verabschiedung vom DoD zurückgezogen, statt dessen wurde das CMMI-Projekt gestartet.
•2000 wurde CMMI -- damals noch unter dem Namen Capability Maturity Model Integrated -- als Pilotversion 1.0 herausgegeben.
•2002 wurde CMMI unter dem neuen Namen Capability Maturity Model Integration (kurz CMMI) freigegeben.
•Ende 2003 ist die Unterstützung des SEI für CMM ausgelaufen.

CMM(I)

- Hilfreich bei der Verbesserung eigener Prozesse
- 5 Stufen beschreiben den Reifegrad einer Organisation und ihrer Prozesse
- Einteilung in Stufen ist ein Kriterium zur Beurteilung der Kompetenz einer Organisation, einen erteilten Auftrag erfolgreich durchzuführen



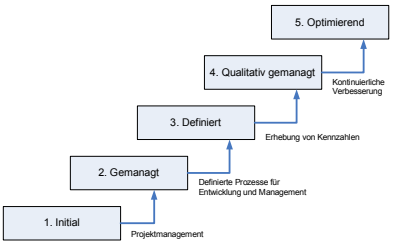
CMMI definiert **Reifegrade** (*maturity levels*). Ein Reifegrad umfasst eine Menge von Prozessgebieten mit konkreten Anforderungen. Die Erfüllung dieser Anforderungen unterstützt jeweils einen wichtigen Aspekt der Softwareentwicklung.

Jeder Reifegrad ist ein Entwicklungsplateau in der Prozessverbesserung der Organisation. Die Reifegrade sind:

1. **Initial**: Keine Anforderungen, diesen Reifegrad hat jede Organisation automatisch.
2. **Managed**: Die Projekte werden gemanagt durchgeführt und ein ähnliches Projekt kann erfolgreich wiederholt werden.
3. **Defined**: Die Projekte werden nach einem angepassten Standard-Prozess durchgeführt, und es gibt eine kontinuierliche Prozessverbesserung.
4. **Quantitatively Managed**: Es wird eine statistische Prozesskontrolle durchgeführt.
5. **Optimizing**: Die Prozesse werden mit den Daten aus der statistischen Prozesskontrolle verbessert.

CMMI – Reifegrade (Merkmale)

- Stufe 1: Initial
  - ungenügende Steuerung der Projekte (keine wirksamen Führungsmechanismen)
  - unklare, wechselnde Anforderungen
  - unvollständige / unrealistische Planung
  - starke Abhängigkeit von einzelnen Mitarbeitern
- Stufe 2: Gemanagt
  - minimale Prozesskontrolle/-führung
  - hohes Risiko bei neuen Herausforderungen
- Stufe 3: Definiert
  - Prozesse für Entwicklung und Management definiert
- Stufe 4: Qualitativ gemanagt
  - Minimum von Qualitäts- und Produktivitätsmetriken und Kennzahlen
- Stufe 5: Optimierend
  - kontinuierliche Prozessverbesserung und -evolution
  - automatische Datensammlung, um Schwachpunkte zu finden
  - rigorose Defekt-Analyse und -verhütung



Folie 33 Jochen Neuhaus, Christian Baun

•Stufe 1: Initial

Hier ist die Ausführung des Prozesses gekennzeichnet als 'ad hoc' und bisweilen chaotisch. Wenige Prozesse sind definiert und die Organisation bietet häufig eine instabile Umgebung für die Projektausführung. Der Erfolg ist abhängig von *individuellem Heldentum* und nicht von der Ausführung bewährter Prozesse. Der Erfolg von Projekten ist den Fähigkeiten weniger oder nur eines einzelnen *Helden* zu verdanken. Wenn dieser die Organisation verlässt oder z.B. durch Krankheit ausfällt, sind die Erfolgsaussichten eines Prozesses gering. Organisationen auf dieser Stufe tendieren dazu, zu optimistisch zu planen, in Krisenzeiten gefährliche Abkürzungen zu nehmen und vergangene Erfolge nicht zuverlässig wiederholen zu können.

•Stufe 2: Managed

In dieser Stufe sind grundlegende Projektmanagement-Prozesse bezüglich Kosten, Zeit und Funktionsplanung gefestigt. Die notwendige Prozessdisziplin, um frühere Projekterfolge bei ähnlichen Projekten zu wiederholen, ist vorhanden. Projekte werden anhand ihrer dokumentierten Pläne ausgeführt. Das heißt dass für die Projekte der Organisation sichergestellt ist, dass ein Anforderungs- und Änderungsmanagement in Kraft ist. Prozesse werden geplant, durchgeführt, gemessen und überprüft. Auf dieser Stufe ist die Qualität der Arbeitsergebnisse der einzelnen Prozessschritte geprüft und dem Management sichtbar anhand von Meilensteinen.

•Stufe 3: Defined

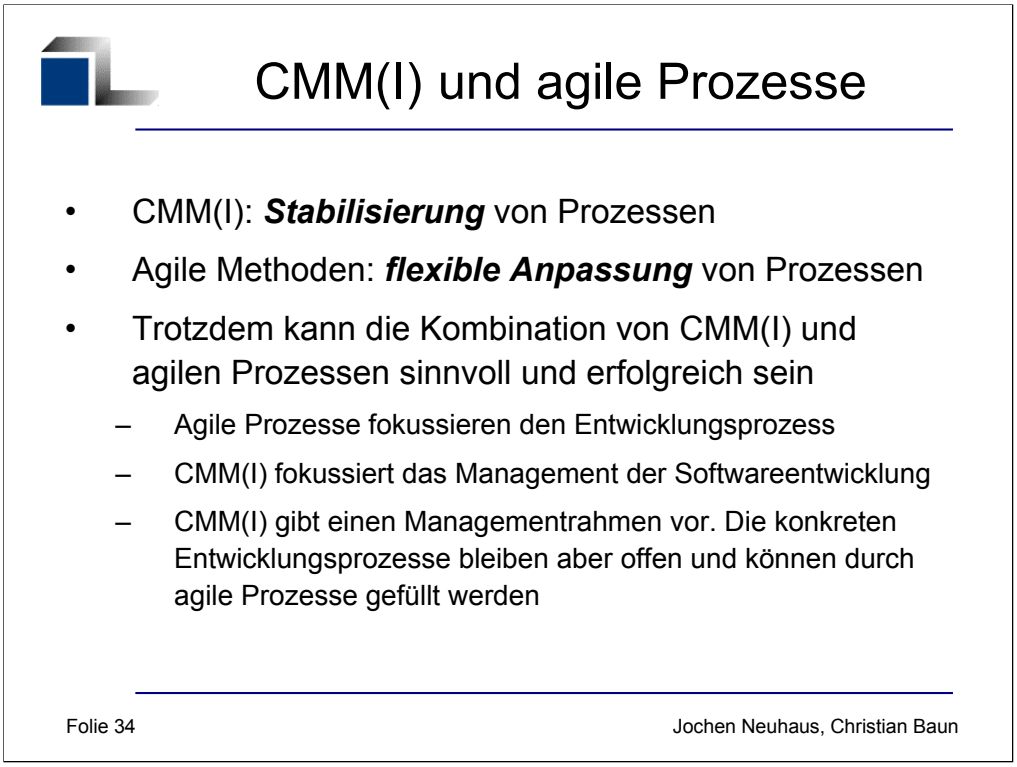
Die Prozesse für das Management und Engineering sind in dieser Stufe dokumentiert, standardisiert und in die organisationsweiten Standardprozesse integriert. Alle Prozesse verwenden eine anerkannte und zugeschnittene Version der organisationsweiten Standardprozesse für die Entwicklung und Wartung von Software. Dadurch wird eine Konsistenz der in den Projekten angewandten Prozesse erreicht. Eine weitere Unterscheidung zur Stufe 2 ist, dass Prozesse detaillierter beschrieben sind und ein Verständnis der Zusammenhänge zwischen den Prozesstätigkeiten, anwendbaren Metriken und weiteren Attributen der Arbeitserzeugnisse aufgebaut wird.

•Stufe 4: Quantitatively Managed

Diese Stufe beinhaltet ausführliche Messungen der Qualität der Prozesse und Produkte. Die Prozesse und Produkte sind quantitativ verstanden und beherrscht.

•Stufe 5: Optimizing

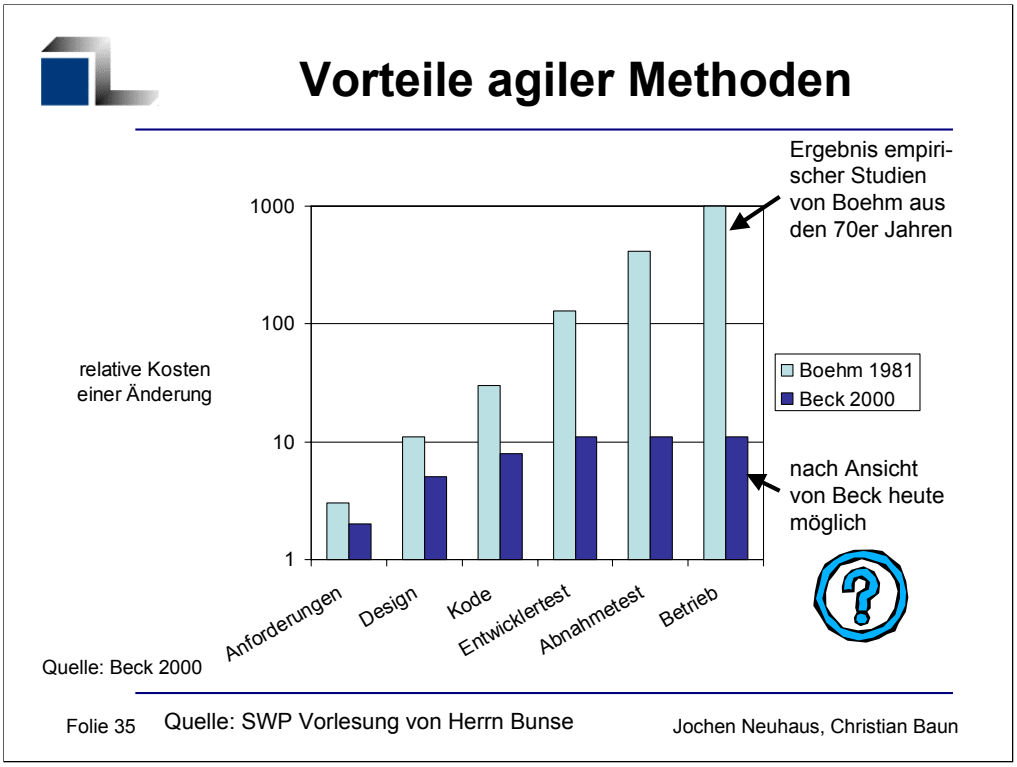
Diese Stufe bezeichnet die kontinuierliche Prozessverbesserung. Diese wird ermöglicht durch quantitativen Rückfluss aus den Prozessen und innovativen Ideen und Technologien.



→12. Prinzip beim agilen Manifest: regelmäßige Anpassung des Prozesses an Team und Projekt

Extreme Programming eignet sich am Ehesten für eine Kombination mit CMMI, da es am stärksten diszipliniert ist

Die anderen agilen Prozesse lassen mehr Freiheit und sind damit weiter von CMMI entfernt.



Was bringt der Einsatz agiler Methoden? Laut Kent Beck (→XP) z.B. eine deutliche Reduktion der Änderungskosten. Dieses Diagramm zeigt allerdings keine Studie, sondern nur eine Annahme von Beck.

Vorraussetzungen und Annahmen

- Kunden sind am selben Ort untergebracht wie das Entwicklungsteam und sind bei Bedarf bereitwillig für die Entwickler verfügbar
- Die Anforderungen an Software und die Umgebung, in der Software entwickelt wird, ändern sich im Laufe der Softwareentwicklung
- Entwickler besitzen die nötige Erfahrung, um ihre Prozesse angemessen definieren und anpassen zu können
- Die Kosten von Änderungen steigen nicht mit der Zeit dramatisch an
- Software kann in Inkrementen entwickelt werden

Quelle: Turk / France / Rumpe 2002

Ein agiler Prozess betrifft nicht nur die Entwickler – Kunde und Führungskräfte müssen ebenfalls mitarbeiten und das Projekt muss sich für den Einsatz eines agilen Prozesses eignen.

Einschränkungen

Agile Methoden bieten nur eingeschränkte Unterstützung für:

- verteilte Softwareentwicklung
- Konstruktion wiederverwendbarer Artefakte
- Entwicklung unter Beteiligung großer Teams
- Entwicklung sicherheitskritischer Software

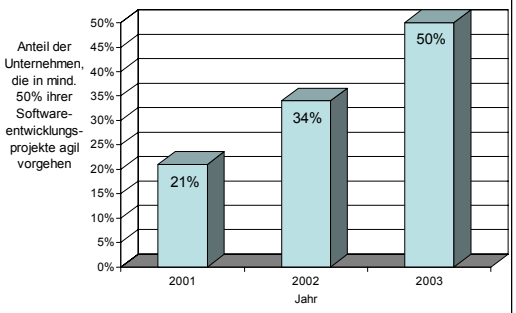
Quelle: Turk / France / Rumpe 2002

Einsatz agiler Methoden

Aktuelle Stellenanzeige von Siemens:

Company: Siemens Corporate Research
Position Title: ***Agile Development Coach***
Required Education: Masters Degree
Job Description: Ideal candidates should have 3 years of software / systems development or product management experience in an **agile product development environment**. The candidates have to be **Certified SCRUM Master** (CSM), […] and knowledge about process modeling and description. The ideal candidates should have solid knowledge about **agile development methods** such as **extreme programming** and **test-driven development** […] and CMMI knowledge.

Quelle: http://careers.peopleclick.com/



Quelle: Cutter Consortium 2001

Quelle: SWP Vorlesung von Herrn Bunse

-Agile Methoden werden in der Industrie eingesetzt. Der Einsatz ist nicht in allen Bereichen sinnvoll, aber einige Anwendungsgebiete können vom Einsatz agiler Methoden profitieren.

-Im Diagramm fehlen absolute Zahlen, aber Steigerung ist erkennbar


Fragen?

Quellen

- K. Beck et al. Manifesto for Agile Software Development.
  http://www.agilemanifesto.org
- Martin Fowler. The New Methodology. 2003
  http://www.martinfowler.com/articles/newMethodology.html
- Christiane Gernert. Agiles Projektmanagement. Carl Hanser Verlag. 2003
- Roman Pichler. Agiles Projektmanagement. 2004
- Projekt Magazin. Agiles Projektmanagement. 2005
  http://www.projektmagazin.de/glossar/gl-0775.html
- K. Beck. Extreme Programming Explained. Addison Wesley. 2000
- Ralf Kneuper. CMMI. dpunkt Verlag. 2003
- Prof. Dr. Wolfgang Schramm. CMMI. 2005
- Christian Bunse Vorlesung Software Prozesse. FH-Mannheim und Fraunhofer IESE. SS 2005
- Virtuelles Software Engineering Kompetenzzentrum
  http://www.software-kompetenz.de
- Jens Coldewey. Einführung in Agile Entwicklung. 2002
  http://www.coldewey.com