

JavaTMmagazin

Internet & Enterprise Technology

XML
extra included

Enterprise Portale

Jahia, Jetspeed, mySAP, WebLogic Portal, Frameworks

Neu: JavaServer Faces

GUI-Entwicklung fürs Web

XML in MS Excel transformieren

Einsatz von dom4j und Apache POI

Enterprise JavaBeans 2.1

Neue Features für Web Services

ECperf-Tests durchführen

Eigene Konfigurationen testen

J2EE Patterns

Idiome der Präsentationsschicht

Angewandte Prozessmodelle

RUP und XP im Vergleich

www.javamagazin.de



mit CD!

D 45867



Rational Unified Process (RUP) und Extreme Programming (XP) im Vergleich

von Andreas Lux

Angewandte Prozessmodelle

Die ganze (IT-)Welt ruft nach Prozessmodellen. Der folgende Artikel legt dar, was es bedeutet, ein Prozessmodell in einem Unternehmen oder in einem Projekt einzusetzen und welche Probleme dabei zu bewältigen sind. Die IT-Industrie krankt derzeit an der mangelhaften Durchführung von IT-Projekten. Als Ausweg aus diesem Dilemma wird die Anwendung von Prozessmodellen diskutiert. Dabei stellt sich die Frage, was denn die Gründe für diese Prozesseuphorie sind. Welche Vorteile bringt die Verwendung eines, wie auch immer gestalteten, Prozessmodells? Was erwarten wir konkret von einem Prozessmodell für unsere Entwicklung, für unser Unternehmen?

Das Warum...

Das Ziel eines definierten Softwareentwicklungsprozesses ist die Erhöhung der Qualität der zu erstellenden Software durch bessere Transparenz und Steuerbarkeit der Softwareentwicklung. Hierbei ist es gleich, ob es sich um die Entwicklung und Pflege einer Inhouse-Lösung oder um die Erstellung und Wartung einer Auftragsarbeit handelt. Mit der Verwendung eines definierten Prozesses ist die gestellte Aufgabe wesentlich besser handhabbar und zwar unter Berücksichtigung aller relevanter Rahmenparameter wie Auslieferungstermine, Budget und verwendete Ressourcen.

Was versteht man eigentlich unter Qualität? Werfen wir einen Blick auf die Qualitätssicherung der fertigen Industrie. Vor der Auslieferung wird durch Kontrollen sichergestellt, dass das ausgelieferte Produkt den Anforderungen des Kunden entspricht. Ist das nicht der Fall, wird das Produkt entweder nachgebessert oder weggeworfen.

Sicherlich muss hier die Produktqualität von der Prozessqualität abgegrenzt werden. Das reine Controlling der Produktqualität und das Wegwerfen von Ergebnissen, die den Qualitätsrichtlinien nicht genügen, ist für die Softwareent-

wicklung sicher nicht brauchbar. Deshalb setzen wir auf die Qualität in den Entwicklungsprozessen. Dieses Vorgehen beruht auf der Annahme, dass qualitativ hochwertige Prozesse auch qualitativ hochwertige Produkte hervorbringen.

Qualität definiert sich je nach Phase unterschiedlich. In der Planungsphase bedeutet Qualität, auf verlässliche Planungsparameter zurückzugreifen. Weiterhin bedeutet es, Produktionsergebnisse auszuwerten, an spezifische Belange der aktuellen Entwicklung anzupassen und damit eine Rückkopplung zu schaffen.

Qualität während der Umsetzung eines Projekts hingegen besitzt eine gänzlich unterschiedliche Definitionen. In dieser Phase bedeutet Qualität, Code zu erzeugen, der einfach wartbar, erweiterbar, stabil und (natürlich!) gut verständlich ist. Es bedeutet weiterhin, dass alle Anforderungen des Kunden mit der gewünschten Funktionalität umgesetzt wurden bzw. dass keine Funktionalität implementiert wurde, die von Kundenseite nicht gefordert wurde. Um die gesetzten Ziele zu erreichen und zwar so, dass sie reproduzierbar sind, ist es dringend erforderlich, nach einem vordefinierten „Leitfaden“ vorzugehen und nichts anderes sind Prozessmodelle.

Die Einführung eines Entwicklungsprozesses ist als kurzfristige Maßnahme zur Steigerung der Qualität in der Softwareentwicklung denkbar ungeeignet. Der Einführungsaufwand eines neuen Prozessmodells ist viel zu hoch, als dass man hiermit einen raschen Erfolg erzielen könnte. Mittel- und langfristig betrachtet ist die Verwendung eines Prozessmodells jedoch unverzichtbar. Die Auswahl des konkreten Prozessmodells kann aber nicht pauschal erfolgen, sondern hängt von der einzelnen Projekt- und Unternehmenssituation ab.

Das Wie...

Bei der Suche nach dem idealen Prozessmodell, soweit es das überhaupt gibt, stellt sich sehr schnell die Frage, was denn solch eine Einführung für das Unternehmen und/oder das Projekt bedeutet. Welche Umstrukturierungsmaßnahmen müssen im Team oder im Unternehmen getroffen werden? Welche Kompetenzen müssen aufgebaut werden und welche Rollen gilt es zu besetzen? Ist das erforderliche Know-how überhaupt im Haus verfügbar oder muss das Wissen erst aufgebaut werden? Wie schafft man die Akzeptanz für das neue Vorgehen bei den Mitarbeitern und beim Kunden? Und nicht zuletzt,

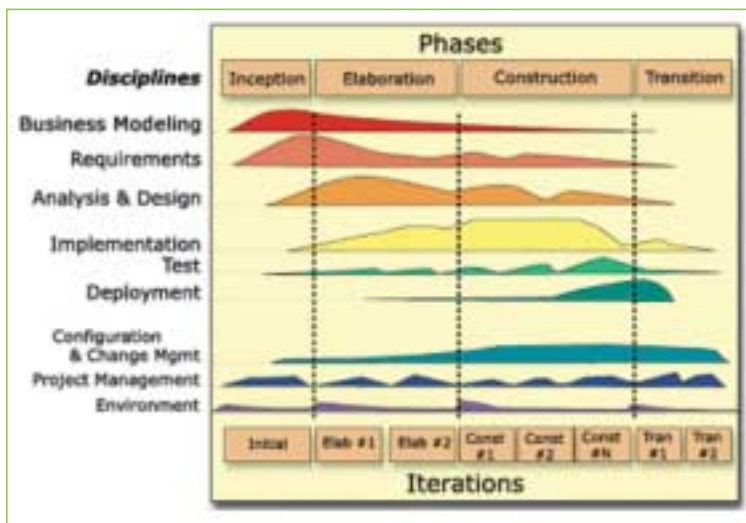


Abb. 1: Prozessmodell

werden vom Kunden erstellt und beschreiben Szenarien, die die Software abbilden können muss. User Stories beschränken sich aber nicht allein auf Bedieneroberflächen, sondern beschreiben bei Bedarf auch die Funktionalitäten des Backends. Aus den User Stories und aus der angestrebten Architektur ergibt sich eine grobe Release-Planung. Diese wird vom Entwicklungsteam in Zusammenarbeit mit dem Kunden erstellt und enthält, wenn auch nur sehr grob, den gesamten Umfang des Projekts. Neben der Funktionalität beinhalten die User Stories ebenfalls die Testszenarien, unter denen die User Story geprüft werden müssen. User Stories und Testszenarien dienen den Entwicklern als Basis zum Implementieren der Funktionalität.

Für jedes Release werden die zu erreichenden Ziele mit dem ständigen Vertreter des Kunden und dem Team diskutiert. Ziel der Diskussion ist das Finden der notwendigen Iterationen zum Erreichen des Releases. Eine Iteration umfasst hierbei in der Regel einige wenige Wochen. Das Ergebnis einer jeden Iteration ist ein lauffähiges Programm, das dem Kunden zur Beurteilung übergeben wird. Auf Basis dieser Beurteilung werden dann die nächsten Iterationsziele definiert. Mit der Entwicklung nach Plan wird nur fortgefahren, wenn der Kunde mit dem Iterationsergebnis zufrieden ist. Im anderen Fall wird die Iterationsplanung oder auch die Release-Planung angepasst.

Das Kodieren erfolgt bei XP immer paarweise (zwei Entwickler, ein Rechner). Hierdurch soll die Qualität des Codes bereits beim Niederschreiben deutlich erhöht werden. Der entstandene Code ist im Gegensatz zur herkömmlichen Teamarbeit nicht Eigentum eines einzelnen, sondern er gehört dem gesamten Team. Ein Programmierpaar bleibt bei XP nicht über die gesamte Projektlaufzeit beisammen, sondern die Teams rotieren zyklisch; sowohl die Teams über die Komponenten als auch die Teams in sich. Auf diese Weise lernen die Entwickler auch die Arbeitsweise der anderen Projektmitglieder kennen. Des weiteren arbeitet (im Idealfall) jeder Mitarbeiter im Laufe des Projekts einmal mit jeder Komponente des Systems und lernt diese kennen. Auf diese Weise ist das

wenn man all diese Punkte betrachtet, ist das Engagement wirtschaftlich überhaupt vertretbar?

All dies sind Fragen, die nüchtern betrachtet und fern von jedem Hype, klar beantwortet werden müssen. Unternehmen müssen wirtschaftlich agieren und das gilt natürlich auch bei der Entscheidung zur Einführung eines Prozessmodells.

Was leisten und fordern die aktuell viel diskutierten Prozessmodelle? Um dies beantworten zu können, werden zwei Vertreter in diesem Bericht kurz vorgestellt.

RUP und XP

Der Rational Unified Process definiert für neun Kernaufgaben (Geschäftsprozessmodellierung, Anforderungsanalyse, Analyse und Design, Implementierung, Test, Softwareverteilung, Configuration- und Change Management, Projektmanagement und Environment) Workflows auf Basis von Aktivitätsdiagrammen. Den Prozessbeteiligten werden Rollen zugewiesen, die definieren, welche Aufgaben eine Person zu bewältigen hat und welche Ergebnisse (Artefakte) zu erzielen sind. Das gesamte Projekt ist in vier Phasen unterteilt. Die „Inception Phase“ beschreibt das Projektsetup, die „Elaboration Phase“ die Ausarbeitung des Projekts, die „Construction Phase“ den Teil, in der der größte Teil der Software implementiert wird. Die „Transition Phase“ schließlich beschreibt das Projektende, in der die er-

stellten Projektteile in Betrieb genommen und in Produktion überführt werden.

In jeder Phase werden eine oder mehrere Iterationen durchgeführt. Je Iteration werden die für die aktuelle Iteration notwendigen Workflows in der Art eines Wasserfallmodells abgearbeitet. Der RUP versteht sich als anwendungsfallgetriebener (basierend auf UseCases) und architekturzentrierter Prozess. Bevor der RUP in einem Projekt oder in einem Unternehmen eingesetzt werden kann ist es notwendig, ihn auf das jeweilige Umfeld und die gestellten Bedürfnisse anzupassen.

Während der RUP versucht, die Komplexität einer Softwareentwicklung durch Strukturierung und eine den aktuellen Zielen angepasste Vorbereitung der zu erledigenden Arbeiten zu kompensieren, versucht Extreme Programming (XP) die Probleme bei der Softwareerstellung durch unkonventionelles, aber zielgerichtetes Arbeiten zu bewältigen.

XP versucht das Risiko einer Fehlentwicklung dadurch zu minimieren, dass ein kompetenter Vertreter des Kunden permanent vor Ort und somit für das Entwicklerteam ständig erreichbar ist. Hierdurch wird vermieden, dass die Beantwortung offener Fragen in die Länge gezogen und dadurch das Projekt gefährdet wird. Der Kundenvertreter besitzt Idealerweise eine ausreichend große Kompetenz, um alle Fragestellungen möglichst schnell zu einer Entscheidung zu bringen.

Bei XP stellen User Stories die Basis der zu erstellenden Software stellen dar. Sie

Detailwissen über die Applikation breit über die gesamte Entwicklungsmannschaft verstreut.

Im ersten Schritt einer jeden Entwicklung werden die Testfälle (Unit-Tests) kodiert. Die Tests sind so zu implementieren, dass sie automatisch durchgeführt werden können. Dies erlaubt vor jeder Auslieferung einen automatisch durchgeführten Test des gesamten Releases. Die eigentliche Funktionalität wird erst kodiert, nachdem die Tests implementiert wurden.

In XP ist jeder Programmierer angehalten, jede beliebige Stelle im Code sofort zu ändern, falls er auf ein wie auch immer geartetes Problem stößt. Lediglich Performanceprobleme werden ausschließlich am Projektende behoben.

Programmiert werden nur Funktionalitäten, die direkt gefordert sind. Eine hohe Flexibilität und Konfigurierbarkeit wird nur dann implementiert, wenn sie direkt zur Erfüllung der Anforderung benötigt wird. Das evolutionäre Design ist

hierbei geprägt von der Prämisse, die geforderte Funktionalität auf dem einfachsten Weg herzustellen. Es gilt das Motto: „Keep things as simple as possible“.

Der Vergleich ...

Der Vergleich der beiden Verfahren soll folgende Fragen klären:

1. Sind die beiden Modelle überhaupt vergleichbar
2. Welche Gemeinsamkeiten gibt es und worin unterscheiden sich die beiden Modelle?
3. Welche Rahmenbedingungen sind ideal für welches Modell oder unter welchen Bedingungen macht der Einsatz keinen Sinn.

Beide Prozesse befassen sich mit der Erstellung von (objektorientierter) Software. Der Prozess wird eingesetzt, um die Qualität zu steigern und um die Entwicklung effizienter zu machen. Das bedeutet,

beide Modelle verfolgen die selben Ziele mit unterschiedlichen Hilfsmitteln. Verglichen werden können somit zwar nicht die Prozesse als Ganzes, wohl aber die verwendeten Hilfsmittel und deren Ergebnisse.

RUP und XP haben relativ wenig Gemeinsamkeiten. Die Basis für beide Modelle stellen UseCases (RUP) bzw. User Stories (XP) dar. Diese definieren die Anforderungen an die Applikation aus Sicht des Benutzers. Beide beschreiben ausschließlich die fachliche Anforderung, ohne sich mit Implementierungsdetails zu beschäftigen.

RUP und XP basieren beide auf einem iterativen Ansatz. Dieses Vorgehen erlaubt es, sich der Lösung eines Problems in mehreren Schritten zu nähern, ohne sich sehr früh in Details zu verstricken.

Die Unterschiede zwischen RUP und XP sind aber weitaus gewichtiger als ihre Gemeinsamkeiten. XP ist ein leichtgewichtiger Prozess. Dies bedeutet, dass die Modellierer des Prozesses sehr viel Wert

Anzeige

darauf gelegt haben, den Entwicklern der Applikation nicht zuviel organisatorischen Mehraufwand zu generieren. Erreicht wurde dies durch die Präsenz des Kunden vor Ort. Bei der Entwicklung von XP-Projekten ist es weniger wichtig, die spezifizierten Funktionalitäten zu implementieren als vielmehr, dem Kunden die Software zu liefern, die er braucht (dies ist in den wenigsten Fällen deckungsgleich mit den Inhalten des Anforderungsdokuments). Wenn während der Entwicklung erkannt wird, dass Funktionalitäten abgeändert werden müssen, wird das direkt mit dem Kunden vor Ort vereinbart. Anhand der neuen Erkenntnisse wird der Iterationsplan/Release-Plan abgeändert und die Entwicklung in der neuen Richtung fortgesetzt.

Bei RUP sind solche volatilen Projektentwicklungen nicht erwünscht. Es stehen unterschiedliche Planungselemente (z.B. Software Development Plan, Phase Plan, Iteration Plan, QA Plan) zur Verfügung, mit deren Hilfe steuernd auf die Softwareentwicklung eingewirkt werden kann. Durch ein vordefiniertes Eskalationsschema und detailliertes Risikomanagement können Probleme und Fehlentwicklungen frühzeitig erkannt und dagegen vorgegangen werden.

RUP liefert für jeden Zeitpunkt im Projekt Leitlinien (Checklisten, Best Practices) die angeben, in welcher Detailtiefe welche Artefakte von wem (Rolle) zu erstellen sind. Es wird definiert, welche Artefakte notwendig sind, um eine bestimmte Aktivität durchführen zu können und welche Artefakte während der Aktivität erstellt werden müssen.

Zu den Milestones am Ende einer jeden Phase werden dem Kunden die in der Phase entstandenen Artefakte vorgestellt und das weitere Vorgehen bewertet (Prüfen von Risikolisten, Abnahme von Iterationsplänen, Prototypen, Toollisten, etc.). Erst nach Freigabe des Milestone durch den Kunden kann in die Durchführung der nächsten Phase eingetreten werden.

Die Qualität aller Artefakte wird während der gesamten Projektlaufzeit durch unterschiedliche QA (Quality Assurance) Maßnahmen wie zentrale Richtlinien, regelmäßige Audits und Reviews, Test, und Trainings sichergestellt.

Die Qualitätssicherung in XP basiert im Gegensatz zum RUP nicht auf Formalismen, sondern auf Selbstkontrolle und Kommunikation mit dem Kunden.

Der Kunde bekommt zu jedem Iterationsende ein lauffähiges Stück Software geliefert. Durch die sehr kurzen Iterationszyklen von einigen wenigen Wochen ist der Kunde ständig über den aktuellen Projektstatus informiert und kann rechtzeitig steuernd in die Entwicklung eingreifen. Die Entwicklung ihrerseits sieht anhand der Kundenresonanz, in welche Richtung die Softwareentwicklung weiter vorangetrieben werden muss.

Durch Pair-Programming wird zudem erreicht, dass bereits bei der Erstellung des Codes Unsauberkeiten und Fehler vermieden werden. Durch die paarweise Programmierung wird jede Zeile Code augenblicklich durch den Partner kontrolliert. Es wird davon ausgegangen, dass durch die enge Verbundenheit beider Entwickler Diskussionen entstehen, die zu sehr viel mehr durchdachten Strukturen und Algorithmen führen.

Beim RUP wird versucht, die Komplexität der zu erstellenden Software durch intensive Planung zu kompensieren. Hierdurch wird vermieden, dass durch die Fluktuation von Schlüsselfiguren aus dem Team das Wissen über die Applikation mit verschwindet.

Eine andere Möglichkeit, dem entgegenzuwirken, zeigt XP. Durch eine Rotation der Entwickler über alle Komponenten der Applikation wird eine breite Wissensstreuung erreicht.

Bei XP geht man davon aus, dass die Qualität der Software im Laufe der Zeit so stark steigt, dass der Mehraufwand des Pair-Programming und der Mitarbeiterrotation durch steigendes Know-how und breiteres Wissen über die gesamte Applikation aufgefangen wird.

Aufgrund der häufigen Releases bei XP ist es notwendig, automatische Tests durchführen zu können. Die Implementierung der Unit Tests vor der eigentlichen Funktionalität führt dazu, dass sich die Entwickler bereits frühzeitig über die Plausibilität und Testbarkeit der Funktionalität Gedanken machen. Durch automatisch ausgeführte Tests wird eine sehr hohe Funktionalitätstreue erreicht, die

vor jeder Auslieferung erneut bewiesen werden kann. Hierbei ist wichtig, dass nur Funktionalität ausgeliefert wird, die alle definierten Unit-Tests zu 100 Prozent bestanden hat.

Durch die regelmäßige Integration neuer oder geänderter Funktionalitäten ist bereits sehr früh ein funktionierendes System vorhanden, mit dem der Kunde erste Erfahrungen sammeln kann.

Die Ferse des Achilles ...

XP ist ein sehr implementierungsnaher Prozess. Durch die geringe Anzahl an Dokumenten, die zu erstellen sind, wird dem Entwickler ein scheinbar ideales Feld geboten, sich in das Projekt einzubringen. Vor allem weniger erfahrene Programmierer sind in diesem Prozess gut aufgehoben, da sie direkt von den Erfahrungen ihres Partners profitieren können. Der Kunde bekommt eine Software, die exakt auf seine Wünsche und Bedürfnisse zugeschnitten ist, sofern er über ausreichend Budget verfügt. Denn der Funktionsumfang der Software wurde niemals exakt bestimmt. Diese Art der Softwareerstellung mag für die Erstellung von Inhouse-Applikationen wohl funktionieren. Bei Auftragsarbeiten dagegen existiert das konkrete Problem,

Kernfeatures XP

- Leichtgewichtiger Prozess
- Entwicklungsnah
- Basiert auf User Stories
- Starke Kommunikation mit Kunden
- Sourcecode gehört allen
- Pair Programming
- Tests als Basis für Funktionalität
- Ein Minimum an Organisation
- Dokumentationsarm

Kernfeatures RUP

- Schwergewichtiger Prozess
- Unterteilt in 4 Phasen
- Phasen sind unterteilt in Iterationen
- Abläufe werden in Workflows beschrieben
- Artefakte sind Ziel aller Aktivitäten
- Rollenbasiert
- UML-Gestützt
- Organisationslastig
- Dokumentenorientiert

dass zu einem sehr frühen Zeitpunkt ein Angebot abgegeben werden muss, das die Implementierung eines definierten Funktionsumfangs umfasst. Dies ist aber für den Kunden ebenso wichtig. Er muss in der Lage sein, die erbrachten Leistungen auf Qualität und Inhalt zu prüfen. Um ein Projekt im Sinne von XP durchzuführen, ist somit ein sehr hohes Maß an Vertrauen zwischen Kunde und Dienstleister notwendig.

Eine (theoretisch) sehr effektive Maßnahme ist die Präsenz des Kunden vor Ort. Doch Mitarbeiter, die über die notwendige Sachkompetenz und das notwendige Urteilsvermögen verfügen, um Fragen vor Ort beurteilen zu können bzw. Entscheidungen treffen zu können, sind vermutlich sehr rar. Es dürfte sehr schwer fallen, den Kunden davon zu überzeugen, diese höchstwahrscheinlich bereits ständig überlastete Person viele Monate exklusiv für die Erstellung einer neuen Software abzustellen.

Ein, meiner Meinung nach, sehr gelungener Ansatz in XP stellt das Pair Programming dar. In konzeptionellen Phasen kann es nur von Vorteil sein, wenn Konzepte, Architekturen und Richtlinien nicht von einzelnen Personen definiert werden. Je mehr Erfahrungen in dieser Phase zusammenkommen, desto schlüssiger sind die Ergebnisse, die dabei entstehen und desto weniger Fehler schleichen sich ein.

Weniger erfahrene Mitarbeiter können aus Diskussionen über eine Architektur weitaus mehr lernen, als wenn sie diese fertig vorgegeben bekommen und verinnerlichen sollen. Ob man diese Art der Teamarbeit aber in allen Phasen der Softwareerstellung weiterleben muss, ist sehr stark abhängig vom zu behandelnden Problem und vom Team selbst.

Was ich bei XP als sehr kritisch ansehe ist die Tatsache, dass so weit wie möglich auf jegliche Art von Dokumentation verzichtet wird. Dass nur der Code selbst die einzige aktuelle Dokumentation darstellt, ist schon richtig. Übergeordnete Zusammenhänge sind üblicherweise daraus aber nicht abzuleiten. So ist es für jede zu erstellende Software notwendig, die implementierten Anforderungen, die bestehende Infrastruktur sowie die verwendeten

Konzepte für Architektur und Design zu notieren und zu archivieren. Denn in jedem Projekt kommt irgendwann einmal der Zeitpunkt, an dem der Mitarbeiter des Kunden wieder nach Hause fährt und die Entwicklerteamschaft sich auflöst. Und auch danach sollte es möglich sein, die Applikation weiter zu pflegen und zu erweitern, ohne ein komplettes Reengineering durchführen zu müssen.

Die Probleme bei der Verwendung des RUP liegen in einer ganz anderen Ebene. Für die Erstellung von Softwaresystemen in Teams kleiner und mittlerer

Erwartungen an Prozessmodelle

Größe (bis ~10 Personen) ist RUP eindeutig zu umfangreich. Es müssen 31 Rollen vergeben werden, die mehr als 100 unterschiedliche Typen von Artefakten erzeugen sollen. Dies bedeutet, dass RUP, bevor er überhaupt eingesetzt werden kann, zuerst sehr aufwendig angepasst werden muss. Dies aber kann nur in Projekten geschehen, in denen auch genügend Budget vorhanden ist, das dieses Vorhaben tragen kann.

Ist das Projekt aber groß genug, um die Anpassung und das Leben vom RUP verkraften zu können, ist RUP sicherlich eine sehr gute Basis, um mehr Struktur und Disziplin der Softwareentwicklung zuzuführen. Eine ausgezeichnete Möglichkeit zur Reduzierung der initialen Aufwände besteht in der Verwendung von bereits vordefinierten und reduzierten RUP-basierten Modellen, die bereits auf ein bestimmtes Themengebiet zugeschnitten wurden.

Erwartungen

Was erwarten wir denn von einem Prozessmodell? Welche Kriterien muss es erfüllen, damit es das Unternehmen weiter bringt?

Ein Prozess soll als Leitfaden für die Erstellung eines Softwaresystems dienen. Er soll uns Dinge bereitstellen, die uns helfen, das gesteckte Ziel zu erreichen, soll uns aber nicht mit zusätzlicher Arbeit be-

lasten. Es soll uns leiten und einschränken, wo es für die Aufgabe nützlich ist und soll uns Freiräume erhalten, wo es möglich ist. Er soll uns an das aktuelle Problem heranzuführen, wenn wir neues Terrain betreten und uns sehr spezielle Hilfen bieten, wenn wir uns mit Details beschäftigen.

Dies alles kann von einem Standardprozess nicht bewerkstelligt werden. Es wird immer eine Vielzahl Wege geben, die zum selben Ziel führen. So ist jedes Unternehmen und jedes Projekt in der Pflicht, sich den optimalen Prozess selbst zu definieren.

Ausblick

Im 2. Teil dieses Artikels wird die Implementierung eines Prozesses namens XXProcess beschrieben, der es erlaubt, Softwareprojekte mit reproduzierbarem Erfolg in allen Phasen durchzuführen. XXProcess kann als eine Implementierung des RUP aufgefasst werden, der jedoch sehr stark auf die Erstellung von Individualsoftware angepasst wurde. Nur so ist es möglich, in ihm bei Einsatz in realen Projekten eine echte Hilfe zu sehen. XXProcess enthält neben großen Teilen vom RUP viele Ideen, die man in XP wieder finden kann. ■

Andreas Lux (a.lux@excellent.de) ist Projektleiter und Gesellschafter der excellent solutions GmbH aus Ulm (www.excellent.de). Er befasst sich seit vielen Jahren mit der Erstellung von Individualsoftware für Businessapplikationen im Umfeld JAVA und Internettechnologien und ist Trainer und Consultant für UML und OO-Themen.

Links & Literatur

Rational Unified Process:

- www.rational.com/products/rup/
- The Rational Unified Process: An Introduction, Philippe Kruchten, Addison Wesley, 2000
- Projektmanagement mit dem Rational Unified Process, Gerhard Versteegen, Springer Verlag, 2000

Extreme Programming:

- Extreme Programming, Kent Beck, Addison-Wesley, 2000
- Extreme Programming . Das Manifest, Kent Beck, Addison-Wesley, 2000
- www.extremeprogramming.org/
- www.xprogramming.com/
- c2.com/cgi/wiki?ExtremeProgrammingRoadmap