

OOP: Agilität und Programmierdisziplin ergänzen einander gut

Entwicklung rückt ans Business

Programmiertechniken und -methoden orientieren sich stärker an der geschäftlichen Realität. Dieser vorherrschende Trend ließ sich bei der diesjährigen Kongressmesse OOP beobachten. Insbesondere die wachsende Akzeptanz domänenorientierter Modellierungssprachen belegt diese Tendenz.



Terry Quatrani, Evangelistin für Softwaremodellierung bei IBM Rational, skizziert in ihrer OOP-Keynote zentrale Aspekte künftiger Entwicklungsprozesse: Vor allem die Transparenz zwischen Softwarehersteller und Anwender steige.

Serviceorientierte Architekturen (SOA) haben Hochkonjunktur – dies war auch auf der OOP 2008 zu spüren. Dabei mahnt Frances Paulisch, die fachliche Leiterin der Münchner Konferenzmesse, den Rummel um SOA von den möglichen Vorteilen zu trennen, die dieser Architekturansatz bietet. „Es gibt viele Marktteilnehmer, die ein SOA-Label auf ihre Produkte kleben, weil es ein Hype-Thema ist“, klagt Paulisch.

Dennoch: „Der Vorteil ist aus meiner Sicht, dass man näher am Business ist. Bedingt durch die Services kann man eher mit den Geschäftspersonen über die Funktionalität reden. Und weil man mit ihnen in deren Terminologie spricht, ist IT für sie kein solches Fremdwort mehr.“

Insgesamt sieht Paulisch einen generellen Trend zur stärkeren Orientierung der Softwareentwicklung an Geschäftsprozessen. Ein Indiz dafür sei der Aufwind, den derzeit domänenspezifische Techniken zur Softwaremodellierung erleben. Fachspezifische Beschreibungssprachen, so genannte DSLs (Domain Specific Language), böten im Vergleich zur generischen UML (Unified Modeling Language) den Vorteil einer größeren Praxishöhe sowie den der stär-

keren Eindeutigkeit. Letzteres sei nicht zuletzt für die automatische Generierung von Programmcode aus den Modellen von Bedeutung.

Allerdings betonen UML-Verfechter, dass sich auch aus ihrer oft als komplex bezeichneten Sprache mit ihrer Vielzahl an Diagrammtypen DSLs erstellen lassen. „Es gibt keine Notwendigkeit, sich eine eigene Grammatik auszudenken, mit der ich eine domänenspezifische Sprache aufbaue“, ereifert sich Andreas Ditze, Mitglied der Geschäftsleitung des Nürnberger Modellierungsspezialisten MID.

Ditze: „Ich gebrauche die UML wie einen Duden, der den Grundwortschatz festlegt. Und ich nutze für meine spezielle Domäne ein Subset der UML, so wie ich es brauche.“ Für fachspezifische Probleme eine Sprache mit eigener Grammatik zu konstruieren, kommt für Ditze dem Versuch gleich, neue Buchstaben zu definieren, „nur weil ich Erlebnisberichte anstelle von Romanen schreiben möchte.“

Scott Ambler, Practice Leader Agile Development bei IBM, stimmt Ditze zu: „Mitte der 90er Jahre hat man sich darum gestritten, ob man Rechtecke oder Wolkensymbole in den Diagrammen verwendet. Im Prinzip ist das doch egal. Ich

habe noch kein Modell in einer domänenspezifischen Sprache gesehen, das sich nicht auch in UML darstellen ließe. Wen juckt's also?“ Zudem weist Ambler darauf hin, dass es keine leichte Aufgabe darstellt, fachspezifische Notationen zu entwerfen: „Es gibt nicht viele Leute, die das machen können.“

Fachspezifische Erweiterungen – so genannte Profile – tragen jedoch nicht notwendigerweise dazu bei, die Komplexität der UML selbst zu verringern, bemerkt Wolfgang Neuhaus, Vorstandsmitglied des Beratungsunternehmens Itemis. „Man kann durch Profile im Grunde genommen nur etwas hinzufügen, was die Komplexität aber weiter steigert.“ Die UML sei nämlich an sich nicht auf Reduktionen des Sprachmodells hin ausgelegt.

Stattdessen favorisiert der Itemis-Vorstand die folgende Methodik: „Man kann beispielsweise die Konzepte, die man benötigt, aus der UML extrahieren, aber ein deutlich reduziertes Metamodell verwenden.“ Damit könne zum Beispiel der Zustandsautomat – ein Diagrammtyp – auf das Notwendige konzentriert und gleichzeitig um zwei oder drei fachspezifische Eigenschaften ergänzt werden. „Es hilft immer, wenn Menschen etwas weit Verbreitetes und Bekanntes wiederfinden – insbesondere, wenn es wie die UML auch in der Ausbildung gelehrt wird. Das bedeutet aber nicht, dass man sie in der reinen Form des Standards verwenden muss“, erläutert Neuhaus seine Präferenz.

Auf diese Weise könne sich die Modellierung von Softwarearchitekturen durchaus der Realität von Geschäftsprozessen annähern, meint der Itemis-Experte. Ob sich der Aufwand für eine eigene fachspezifische Beschreibungssprache lohnt, hängt aber aus seiner Sicht nicht zuletzt von der Tool-Unterstützung ab. Er verweist dabei auf das Graphical

Modeling Framework (GMF) aus dem Eclipse-Umfeld, mit dem sich verhältnismäßig einfach Editorenwerkzeuge für grafische Modellierungssprachen erstellen lassen. Ein Pendant für textbasierte Beschreibungstechniken ist Xtext, das ebenfalls aus dem Eclipse-Lager stammt. „Man kann nach einer eintägigen Einweisung mit dem Tool umgehen“, lobt Neuhaus. „Im grafischen Bereich, bei GMF, könnte dagegen die Dokumentation noch ein wenig besser sein.“

Generell werden drei grundlegende Aspekte die Softwareentwicklung der Zukunft prägen, ist Terry Quatrani, UML-Evangelist bei IBM Rational, überzeugt. Als ersten nennt sie Transparenz, als zweiten die Kollaboration und als dritten die Automatisierung der Entwicklungsprozesse. Vor allem die Transparenz spielt für sie eine wichtige Rolle: „Die Anwender wollen mehr Einfluss haben. Man kann nicht mehr einfach sagen: In einem halben Jahr bekommen Sie Software, mit der Sie etwas anfangen können oder auch nicht.“ Deswegen sollten sich die Entwickler öffnen und dem zukünftigen Nutzer Einblick in den Entwicklungsprozess erlauben, meint Quatrani.

Aus ihrer langen Erfahrung als Programmiererin weiß Quatrani überdies, dass es den oft propagierten Gegensatz zwischen herkömmlichen Vorgehensmodellen und agiler Entwicklung in der Praxis nicht gibt. „Wenn man iterativ – also in sich wiederholenden Zyklen – programmiert, früh und oft testet und seine Kunden in den Prozess einbezieht, dann arbeitet man agil.“ Deshalb arbeite fast jeder Entwickler agil – wenn auch in unterschiedlichem Maße. Agilität nötigt Entwickler auch zu mehr Disziplin, meint ihr IBM-Kollege Scott Ambler. „Bedingt durch das frühe und häufige Testen kann man sich nicht mehr verstecken.“ fg

Entwicklertage OOP: Methodische Dogmen werden brüchig

Agiles Modellieren ist möglich

München (fg) – Nach Meinung mancher Vordenker vertragen sich agile Vorgehensweisen der Softwareentwicklung nicht mit der Modellierung von Code. Tatsächlich haben sich einige flexible Programmierkonzepte bereits in der alltäglichen Entwicklung etabliert – und dort lassen sie sich auch mit Modellen kombinieren.

Jeder arbeitet agil – zumindest in gewisser Weise“, beobachtet Terry Quatrani, Evangelistin für die Unified Modeling Language (UML) bei IBM Rational. „Wer iterativ – also in sich wiederholenden Zyklen – arbeitet, früh und oft testet sowie seine Kunden regelmäßig einbezieht, nutzt definitiv agile Methoden.“

Ihr Kollege Scott Ambler, Practice Leader Agile Development bei IBM Rational, pflichtet ihr bei. Seiner Ansicht nach erfordern die oft als leichtgewichtig empfundenen Methoden in der Regel mehr Disziplin als traditionelle Entwicklungsansätze. „Agile Projekte weisen mehr Transparenz auf als herkömmliche. Man kann sich nicht verstecken. Wenn man früh und oft testet, kommen die Fehler einfach an den Tag“, so Ambler.



Foto: IBM

UML-Veteranin Terry Quatrani sieht allenthalben flexible Entwicklungsansätze am Werk.

Flexible Entwicklungsansätze schließen Disziplin also nicht aus – und ebenso wenig die Modellierung, etwa mit der UML. „Nur weil man agil arbeitet, wirft man Modelle nicht raus“, ereifert sich die Rational-Veteranin Quatrani. Methodenexperte Ambler zitiert Statistiken, nach denen rund 77 Prozent der flexibel arbeitenden Entwicklerteams Entwürfe nutzen. „Was die Teams sagen, unterscheidet sich oft von dem, was sie tun.“ Von agilen Dogmen wollen viele offiziell nicht abweichen. In der Praxis passiere das aber oft, weiß Ambler.