

# *manage* **it**

[[ IT - S t r a t e g i e n u n d L ö s u n g e n ]]

## **Konsequente Kundenorientierung**

Steigende Gewinne mit CRM

## **Linux und Open Source**

Sekt oder Champagner?

## **IT aus der Steckdose**

Das IT-Kaftwerk

## **Mit (mehr) Sicherheit Geld sparen**

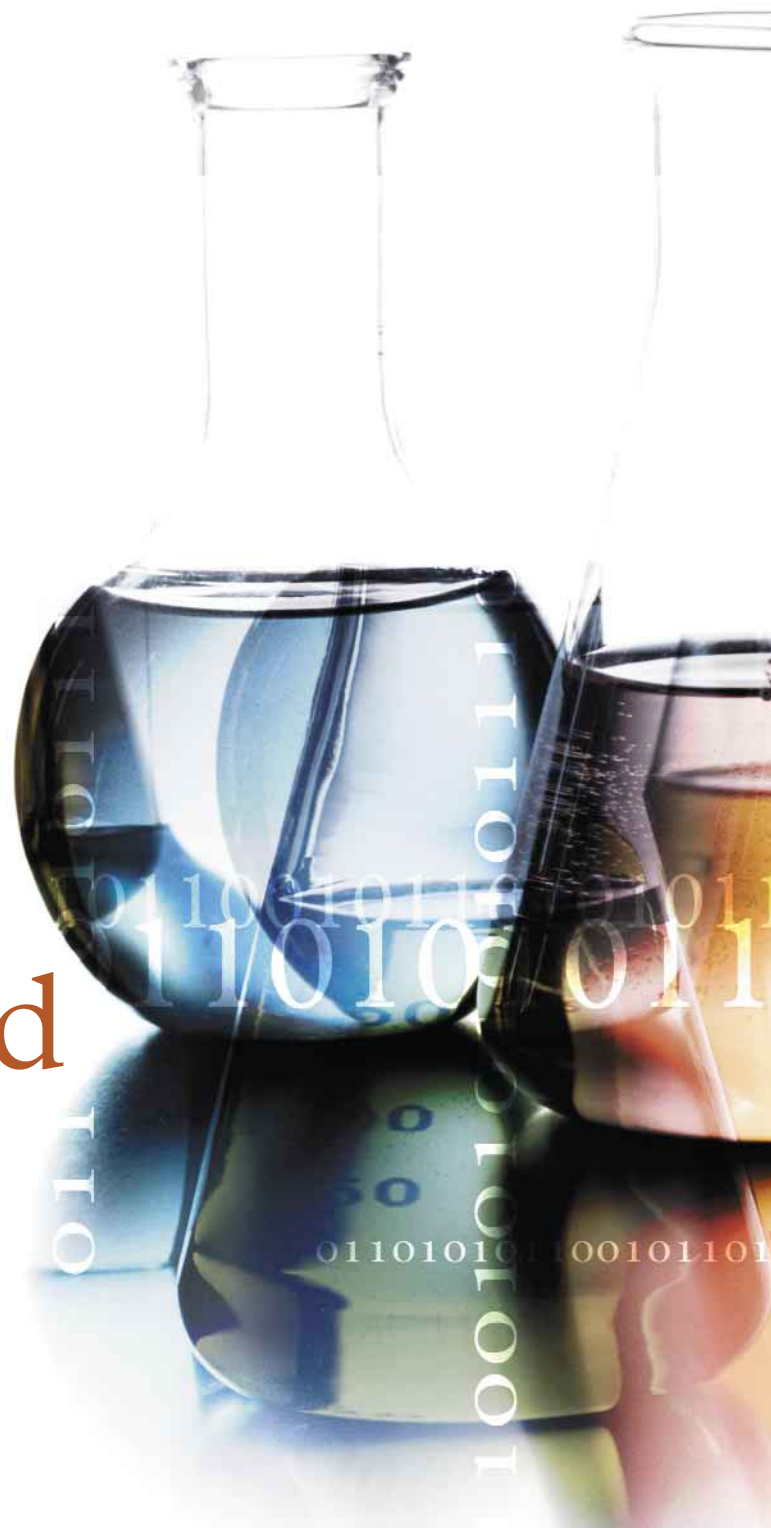
Identitätenmanagement  
und Zugriffskontrolle

Moderne Speicherlandschaften

# Storage Grid

Nutzen Sie unser Angebot für  
Sonderdrucke oder E-Publishing-Dateien  
von Artikeln dieser Ausgabe

Tel.: +49 8092 87543



## UML und die Abkehr von der Code-Orientierung

## Um UML und um UML herUML

Die Softwareentwicklung hat ihre Orientierung am Sourcecode überwunden. Mit UML hat sich eine Modellierungssprache etabliert, die den Entwurf von Modellen mittels standardisierter Verfahren ermöglicht. Moderne UML-Tools erlauben die automatische Generierung von Code und eröffnen so den Weg zu einer hoch-integrierten Softwareentwicklung, die den gesamten Application Lifecycle abdeckt.

Softwareentwicklung hat den Nachteil der späten Geburt. Sie entstand als andere technische Disziplinen wie Maschinenbau, Chemie oder Elektrotechnik längst ihren Platz und ihre anerkannten Verfahren gefunden hatten. Der Softwareentwickler sah sich demgegenüber lange Zeit weniger als Ingenieur denn als Künstler. Und auch wenn es in der Realität der IT-Abteilungen immer anders war, mit dem Image des turnschuhtragenden, stets ein wenig ausgeflippten Softwareentwicklers, mit dem Hang zum Genialischen einerseits und zum Chaos andererseits, hat die IT über lange Zeit kokettiert.

In der Softwareentwicklung hat sich in den letzten Jahren ein überfälliger Paradigmenwechsel vollzogen. Die am Code orientierte Softwareentwicklung der ersten dreißig IT-Jahre war in den 90er Jahren an ihre Grenzen gestoßen. Die Entwicklungsprozesse waren für die traditionellen Verfahren zu komplex geworden – und die Anforderungen zu anspruchsvoll. Man hatte erkannt, dass man ohne methodische Absicherung, ohne stringente und verbindliche Verfahren die anstehenden Aufgaben nicht würde bewältigen können. Darüber hinaus hatten die ersten Versuche, übergreifende Methoden in die Softwareentwicklung einzuführen, auch gezeigt, dass proprietäre Lösungen nicht dauerhaft waren. Hier brachte die UML (Unified Modelling Language)

einen großen Fortschritt, sie hat sich als Grundlage für die Entwicklung von Modellen mittlerweile etabliert und findet in der Branche eine überraschend hohe Akzeptanz. Sie hat sich überdies von einem anfangs noch sehr akademisch geprägten Ansatz zu einer mittlerweile praktikablen Lösung für die modell- und prozessorientierte Softwareentwicklung weiterentwickelt. Die Vereinfachung die UML hier gebracht hat, geht so weit, dass es mit einem entsprechenden Softwarewerkzeug, oder einer UML-Entwicklungsumgebung wie Borland Together mittlerweile möglich ist, den gesamten Entwicklungsprozess von den ersten Business-Anforderungen bis zum Deployment UML-zentriert durchzuführen.

**Keine Ersatz-Programmiersprache.**

UML ist jedoch keine Programmiersprache, sondern eine Modellierungssprache, erst im Anschluss an die Modellierung im jeweiligen Sprachkontext in Code überführt wird, wobei der Entwicklung je nach Anforderungen eine geeignete (objektorientierte) Programmiersprache frei wählen kann. Dabei sind innovative UML-Umgebungen heute durchaus in der Lage, direkt aus den UML-Diagrammen Code zu erzeugen. UML wird damit nicht zu einer; die große Aufmerksamkeit, die die Möglichkeit aus UML direkt Sourcecode zu generieren findet, ist nicht zuletzt auch in Indiz für die weit verbreitete Code-Orientierung der Branche. Meist liegt ein Missverständnis über den Zweck von UML vor: Tatsächlich

**Neuer Besen – UML 2.0**

Nach mittlerweile sieben Jahren UML ist eine neue Version fast schon überfällig und die Version UML 2.0 steht nun bereits unmittelbar vor der Verabschiedung durch die Gremien. Die wesentlichen Verbesserungen wurden in den Bereichen XMI (XML Metadata Interchange), MDA (Modell Driven Architecture), RT (Realtime Diagrammtypen) sowie BPM (Business Process Model) und bei den Veränderungen in den Aktivitätsdiagrammen vorgenommen. Darüber hinaus werden jetzt auch Softwarekomponenten besser unterstützt. Als sehr wichtig wird vor allen Dingen das Diagram Interchange Model angesehen, das die Interoperabilität zwischen den UML-Werkzeugen abdeckt. Die OMG sieht hier als Austauschformat für Diagramminformationen XMI vor. Mit der UML 2.0 wird ebenso ein konsequentes »Vom-Modell-direkt-zum-Code« vorangebracht, ausführbare UML Modelle sind die Folge. Damit rückt dann auch die »Software-Fabrik« ein Stück näher.

geht es um eine methodisch saubere und nachvollziehbare Modellbildung, wenn dann am Ende auch noch Code »ausgespuckt« wird, ist das schön, aber nicht zwingend.

Es handelt sich bei der Code-Generierung somit eher um einen hochintelligenten UML-Aspekt, nämlich aus einem Guss vom Modell zum Sourcecode zu gelangen. Neuere Tools bieten die Möglichkeit durch die »Life Source-Technologie« eine Echtzeit-Verzahnung zwischen Model und Code so zu gestalten, dass beispielsweise eine Modelländerung automatisch im Code abgebildet wird, während umgekehrt Code-Änderungen auch die kontextuierten UML-Modelle verändern. Diese Art der Code-Erzeugung lässt sich direkt aus den Modellen heraus anstoßen – automatisch und tatsächlich quasi »per Knopfdruck«.

**Re-Engineering per UML.** In vielen Fällen besteht das Problem außerdem gar nicht darin, schnell zu Code zu kommen, sondern einen längst vorhandenen Code in ein Modell zu übernehmen, also ältere Applikation so zu strukturieren, dass diese in den Application Lifecycle Prozess passen. Auch hier bringt UML die Lösung: Moderne UML-Tools wie Together können (objektorientierten) Code importieren und daraus automatisch UML-Modelle erzeugen, insbesondere in Form von Klassendiagrammen. Diese Möglichkeiten erleichtern das Re-Engineering von älteren objektorientierten Anwendungen oder Komponenten, die anders oft gar nicht mehr in die Modelle integriert werden könnten. UML verschafft hier mehr Transparenz, insofern wird auch die fast schon sprichwörtliche Kluft zwischen Architektur und Entwicklung durch die Möglichkeiten, die UML in einem integrierten Entwicklungstool bietet, überwunden.

Der Entwickler wird bei der modellorientierten Softwareentwicklung keineswegs überflüssig, er muss sich aber auf neue Arbeitsprozesse einstellen, wie das ja auch bei der Einführung von RAD (Rapid Application Development) oder den IDEs (Integrated Development Environment) der Fall war. Mit den aktuellen UML-Werkzeugen lassen



Das UML-Modellierungstool Together von Borland unterstützt zahlreiche Arten von Diagrammen, so beispielsweise Klassen-, Use-Case- oder Kollaborations-Diagramme

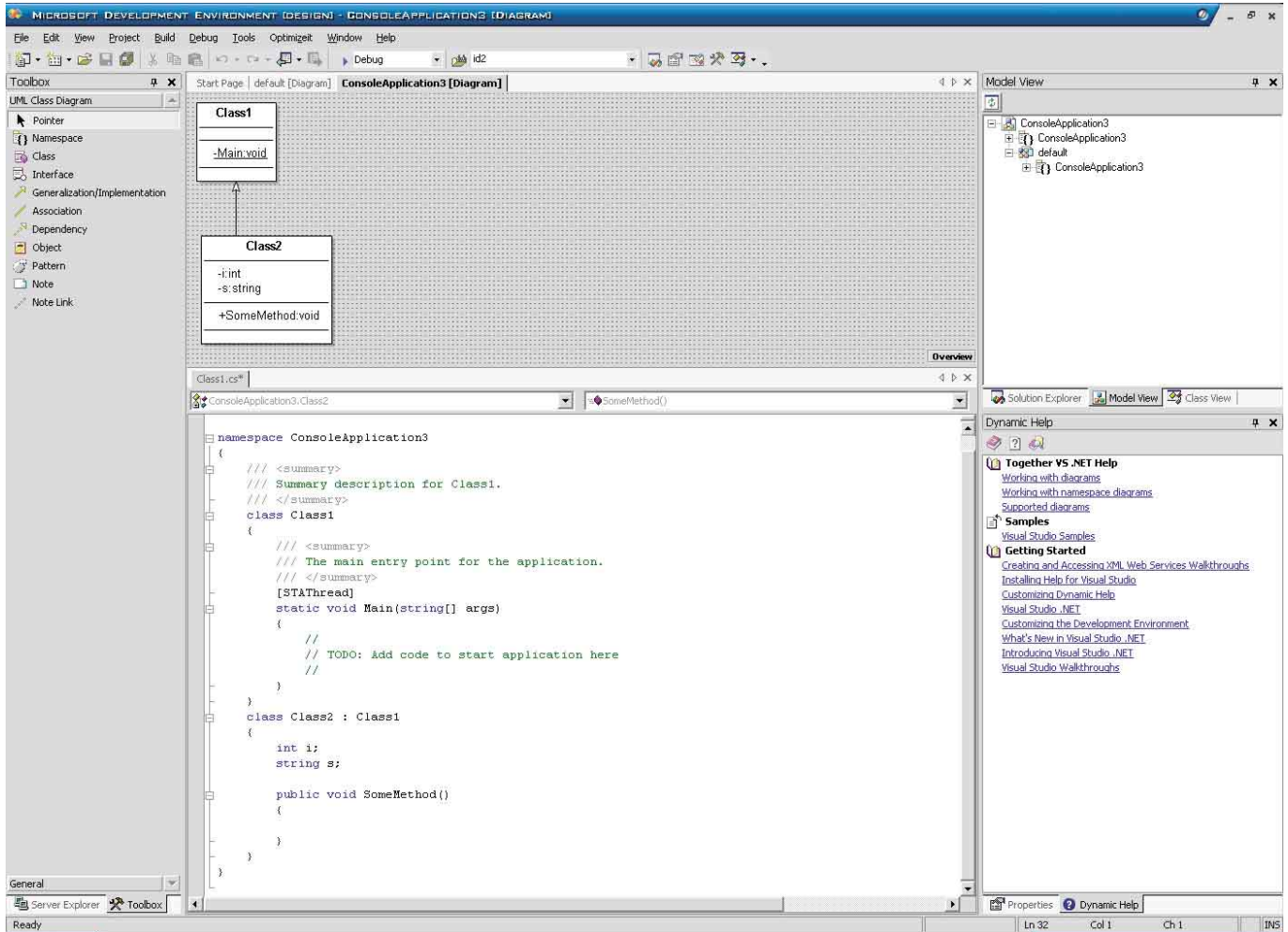
sich mit weniger Aufwand bessere, weil methodisch stringentere Programme oder Komponenten produzieren. Zusätzlich übernehmen diese Tools die eher »lästigen« Aufgaben, wie zum Beispiel die Dokumentation. Insofern führt der Einsatz von UML zu einer Verbesserung der Produktivität der Softwareentwicklung und zu schnellerem ROI.

Die Integration in die unternehmensspezifischen Systemumgebungen lässt sich zwar in der Regel nicht vollständig automatisieren, aber auch sie

wird durch UML-orientierte Entwicklungsschritte gut abgedeckt, so dass sich die Entwicklungsarbeit nicht einfach in spätere Prozessphasen verlagert. Ist zum Beispiel ein SAP-System, eine Dotnet-Architektur und eine J2EE-Architektur vorhanden, so besteht die Integrationsarbeit darin, J2EE und Dotnet zu einheitlichen Prozessen zu verbinden, die dann auf das SAP-System zugreifen. Ein derartiges Entwicklungs- oder Integrationsvorhaben würde man nach der Anforderungsbeschreibung

### Die Phasen des Application Lifecycle

- **Define** Festlegung der Anforderungen an die Software, Definition des Soll-Zustands
- **Design** Entwurf und Modellierung der Programmstruktur entsprechend der Anforderungen
- **Development** Umsetzen des Entwurfs in Source-Code mittels IDE und Programmiersprache
- **Test** Abstimmung von Soll- und Ist-Zustand der Software, Qualitätssicherung
- **Deployment** Implementierung der Software auf einer bestimmten Hard- und Software-Plattform
- **Manage** Verwaltung aller erstellten Softwarebestandteile, sowie Change- und Configuration-Management



Borland Together stellt laufend eine vollständige Übereinstimmung von Code und Modell her.

mit einem Requirements-Management-Werkzeug zunächst einmal in UML modellieren. Gleichzeitig würden die Entwickler den bereits vorhandenen Code als UML abbilden (Reverse Engineering) und so anpassen – Modell und Code sind entsprechend verzahnt –, dass die Kommunikation zwischen den Systemen oder Komponenten über Corba erfolgen kann. Dabei würde die IDL (Interface Definition Language) in UML modelliert und daraus im Anschluss der entsprechende IDL-Code erzeugt werden. Parallel dazu würde der gesamte Build-Prozess anhand der Prozesse, Modelle, Diagramme, Protokolle und Sourcen automatisch dokumentiert und versioniert. Anders ausgedrückt: UML-Entwicklungsumgebungen erleichtern insbesondere Integrationsvorhaben, weil sie definitionsgemäß Technologie-übergreifend funktionieren.

Paradigmenwechsel heißt eben auch, dass sich die Sichtweise auf den Entwicklungsprozess verändert: die Codeerstellung, die Jahrzehnte lang das A und O der Softwareentwicklung war, tritt in den Hintergrund, ist ein (wichtiger) Aspekt unter mehreren.

Dies zeigt sich auch in der Frage der Wiederverwendbarkeit: Objektorientierung und Komponententechnologie sahen diese letzten Endes immer aus Sicht des Codes. Nun befinden sich im berühmten Lego-Baukasten nicht mehr einzelne Softwareobjekte oder funktionell abgeschlossene Code-Teile, sondern Bestandteile von Modellen, die einmal erstellt wurden und die nun unter Einhaltung der UML-Spezifikationen wiederverwendet werden können, was bei Verwendung von XMI (XML Metadata Interchange) sogar Tool-übergreifend erfolgen kann. Auf

diese Weise werden Wiederverwendbarkeit und Pflege nicht nur der Modelle, sondern auch der daran »anliegenden« Sourcen jederzeit gewährleistet. An dieser Stelle ist ein wesentlicher Vorteil des Einsatzes von UML festzuhalten: Ganze Entwicklungsprojekte und deren Zyklen lassen sich nun anhand von dokumentierten und wiederverwendbaren UML-Modellen nachbauen. Gerade unter iterativen Gesichtspunkten eines Softwareentwicklungsprozesses werden beispielsweise die oftmals sehr umständlichen Change-Request-Verfahren erleichtert und der gesamte Lifecycle der erstellten Software gesichert. Dazu gehört auch, dass eine starke Integration – technisch wie fachlich – zwischen einer UML-Entwicklungsplattform wie Borlands Together mit Requirements-Management-Werkzeugen wie Caliber RM usw. möglich ist.



3 Ausgaben

# Einblick Durchblick Ausblick

f ü r d r e i z e h n f ü n f z i g !



[ ] **Ja**, ich bestelle drei Ausgaben » *manage it* « zum Preis von Euro 4,50 pro Ausgabe. Dieses Probeabonnement verlängert sich nicht automatisch.

Schicken Sie diesen Coupon an:

**ap Verlag GmbH  
Postfach 1380  
85554 Ebersberg**

oder faxen Sie die Seite einfach an die Nummer

**+49 8092 87544**

Titel: \_\_\_\_\_

Vorname: \_\_\_\_\_

Nachname: \_\_\_\_\_

Position: \_\_\_\_\_

Firma: \_\_\_\_\_

Straße: \_\_\_\_\_

PLZ: \_\_\_\_\_ Ort: \_\_\_\_\_

E-Mail: \_\_\_\_\_

Telefon: \_\_\_\_\_

Fax: \_\_\_\_\_