Heile Welt

Anforderungsprofile und Innovationsfelder der MDA

Julian Bahrs, Manuel Bork, Tiziana Margaria, Eldar Sultanow

Die MDA (Model Driven Architecture) hat den Anspruch, Softwareentwicklung dank modellgetriebener und generativer Ansätze effizienter, günstiger und fehlerfreier zu gestalten. Doch reichen die bisherigen Erkenntnisse zum erfolgreichen Einsatz in echten Softwareprojekten? So gibt es zwar leistungsstarke Codegeneratoren, doch profitiert man davon auch außerhalb der „heilen Welt", in der generierter Code nicht verändert wird? Außerdem werden zur Codegenerierung oftmals Modelle benötigt, die Implementierungsdetails enthalten und somit fast nur von Entwicklern selbst erstellt werden können. Ein probater Lösungsweg hierfür ist die Domänenspezialisierung, in der Änderungsanforderungen aus der speziellen Problemdomäne stammen.

Einführung

Die modellgetriebene Architektur (engl. Model Driven Architecture, kurz MDA) geht auf grundlegende Gedanken zurück, die James Martin bereits Ende der 70er Jahre formulierte und in den Konzepten der „Information Engineering Methodologie" (IEM) festlegte [Rau07, S. 169]. Insbesondere geht die MDA von der schon lange bekannten Idee aus, die Systemspezifikation von der Implementierung derart zu trennen, dass sie nicht von plattformspezifischen Eigenschaften abhängt. Während als primäre Ziele die Portabilität, Interoperabilität und Wiederverwendbarkeit durch strikte Trennung in der Architektur verfolgt wurden, haben sich aufgrund veränderter Anforderungen, die beispielsweise im Zuge zunehmend verteilter Softwareentwicklungsprojekte entstehen, neue Ziele in den Vordergrund gedrängt.

Dieser Beitrag untersucht vier ausgewählte Innovationsbereiche im Umfeld der MDA, die in modernen Softwareentwicklungsprozessen entstehen:

- Round-Trip-Engineering,
- modellbasierte Kollaboration,
- Wissensmanagement und
- Domänenspezialisierung.

Derzeit oder in Kürze verfügbare Entwicklungswerkzeuge fokussieren diese Bereiche unterschiedlich stark. Wir stellen daher vier Werkzeuge exemplarisch vor und zeigen, wie diese die jeweiligen Innovationsbereiche abdecken.

Round-Trip-Engineering

Round-TripEngineering ist ein Begriff der Softwaretechnik und bezeichnet die Konsistenzhaltung zwischen Programmcode und Diagrammen. Als Basis eines neuen Softwareartefakts greifen Entwickler entweder auf ein gedankliches oder ein explizites Modell in Form von Stichworten oder Diagrammen zurück. Letztendlich werden beide Modelle in Quelltext überführt: Sie werden entweder manuell ausimplementiert oder aber man verwendet sie zur automatischen Codegenerierung.

Die Generierung von Quelltext wird von vielen existierenden Tools unterstützt. Eine Kernanforderung an den Codegenerator ist insbesondere Flexibilität: Wechselnde Anforderungen an den Quelltext, das Einfließen von Best Practices sowie andere Optimierungen verlangen eine leichte Anpassbarkeit des Generators. Um diese Flexibilität zu ermöglichen, haben sich in der Praxis textuelle Codevorlagen, sogenannte Templates, für den letzten Schritt der Quelltexterzeugung bewährt.

In der Praxis wird regelmäßig nur ein Grobgerüst des Quelltexts generiert und vom Softwarearchitekten an die Programmierer zum (manuellen) Ausimplementieren weitergereicht. Dabei wird oftmals noch während der Ausimplementierung das Modell selbst verändert: Das Grobgerüst wird vervollständigt oder das ursprüngliche Design modifiziert. Um nun Modell und Quelltext wieder zu synchronisieren oder um existierenden Quelltext erstmals als abstraktes Modell in Diagramm- oder Textform darzustellen, ist Reverse-Engineering notwendig.

Übliche Ansätze zum Reverse-Engineering verwenden einen auf der Grammatik der Programmiersprache basierenden Parser. Der damit ermittelte Syntaxbaum muss im Anschluss per Modelltransformation auf das bestehende Modell abgebildet werden. Wird durch Reverse-Engineering zuvor aus einem Modell generierter Quelltext eingelesen, darf das ursprüngliche Modell dabei nicht verändert werden. Hierfür ein Beispiel: Wird für ein Attribut einer Klasse Java-Quelltext mit einem privaten Feld und zwei öffentlichen Zugriffsmethoden (Getter und Setter) generiert, so soll dies vom Reverse-Engineering auch wieder auf genau dieses eine Attribut im Modell zurückübertragen werden. Dies ist die Herausforderung beim Round-Trip-Engineering, an der viele aktuelle Entwicklungswerkzeuge scheitern, indem sie dem Modell auch die zwei Zugriffsmethoden hinzufügen. Die resultierenden Modelle werden somit von Implementierungsdetails überladen und für den Anwender unübersichtlich. Die Formulierung entsprechender Modelltransformationen ist dabei nicht trivial – in der Regel werden diese vom Toolhersteller mitgeliefert.

Da die Templates stets weiter entwickelt werden, ändert sich auch der Syntaxbaum, den der Parser nach Einlesen des Quelltexts dem CASE-Tool zur Weiterverarbeitung bereit stellt. Damit nun ein erneutes Wiedereinlesen des Quelltexts ohne unerwünschte Veränderungen des Modells möglich ist, müssten auch die nachgeschalteten Modelltransformationen angepasst werden. Dies ist jedoch erheblich schwieriger als etwa das Anpassen des Templates im Falle des Forward-Engineerings – wenn es denn überhaupt vom Entwicklungswerkzeug angeboten wird. Zudem entsteht ein Wartungsproblem, da Templates und Modelltransformationen stets zueinander passend gehalten werden müssen. Folglich wird in der Praxis gewöhnlich auf ein automatisches Round-Trip-Engineering verzichtet.

Fließen die Änderungen am Quelltext nicht wieder konsequent in die Designdokumente zurück, divergieren Modell und Quelltext im Laufe der Zeit immer mehr, bis Weiterentwicklungen auf Basis des ursprünglichen Designs erschwert oder gar unmöglich werden. Auch für die Dokumentation der Software sind veraltete Modelle nur noch eingeschränkt brauchbar. Das konsequente Round-Trip-Engineering ist daher noch immer Gegenstand aktueller Forschung.

Kollaboration

Kollaboration spielt nicht nur innerhalb von Modellen eine große Rolle in der MDA, sondern auch zwischen den an der Erstellung der Modelle beteiligten Entwicklern und Architekten. Es stellt sich die Frage, wie die gemeinsame Arbeit mehrerer Projektteilnehmer an einem Modell werkzeugunterstützt organisiert werden kann, wenn diese – unter Umständen räumlich verteilt – gleichzeitig erfolgt.

Sind mehrere Entwickler gemeinsam an einem Projekt beteiligt, kann es im Rahmen dieser Zusammenarbeit zu konkurrierenden Änderungen an Fragmenten, also Quelltextdateien oder Modellen, kommen. Zur Vermeidung solcher Konflikte gibt es Verfahren zur Konfliktvermeidung und zur Konfliktlösung. Pessimistische Sperrkonzepte dienen dabei der gänzlichen Verhinderung von

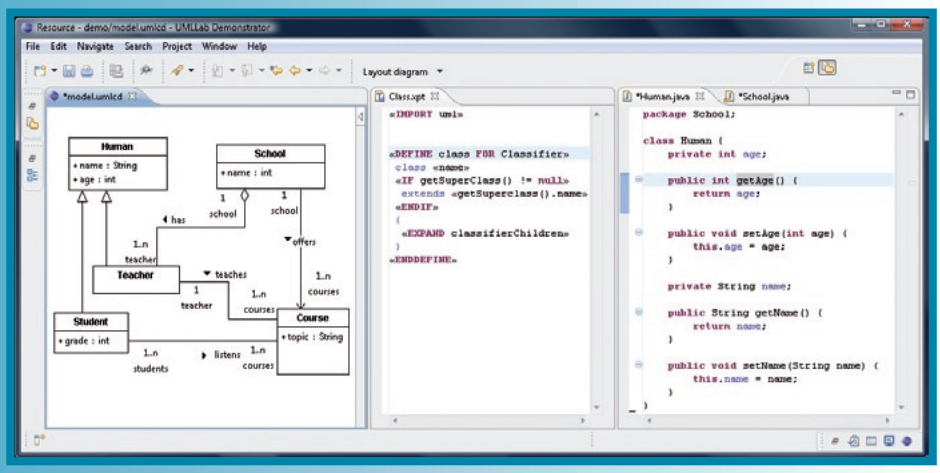

Abb. 1: Echtes Round-Trip-Engineering mit UML Lab: Klassendiagramm, Ausschnitt aus einem Template sowie daraus generierter Code

Konflikten, indem zu ändernde Fragmente zunächst exklusiv für einzelne Bearbeiter gesperrt werden und somit von anderen Entwicklern nicht verändert werden können. In der Praxis wird diese Art der Konfliktvermeidung vom Anwender als unbefriedigend empfunden. Im Gegensatz dazu dienen optimistische Sperrkonzepte der Lösung potenziell aufgetretener Konflikte. Hier werden den Anwendern alle Arten von Änderungen lokal gestattet. Erst beim Senden der lokalen Änderungen an das zentrale Repository wird ermittelt, ob die geänderten Fragmente in der Zwischenzeit von einem anderen Anwender verändert wurden.

In der Praxis haben sich optimistische Sperrkonzepte bewährt und werden in allen gängigen Source-Code-Management-Systemen (SCM), wie beispielsweise CVS und Subversion, umgesetzt. Zur Verwaltung von Modellen ist jedoch weitergehende Funktionalität nötig. So sind Vergleiche unterschiedlicher Versionen von Textdokumenten leicht zeilenbasiert durchführbar. Für Diagramme sind Unterschiede zweier Versionen jedoch erheblich schwieriger zu lokalisieren. Auch sollten rein visuelle Unterschiede wie etwa Veränderungen am Layout gegebenenfalls vernachlässigt werden können, falls dies für die verwendete Modellierungssprache unerheblich ist.

Wissensmanagement

Wissensmanagement bezeichnet die Verwaltung von Wissen mit Konzentration auf die Unternehmens- und Prozessziele. Wissen entsteht in Personen und wird von diesen angewendet. Die Aufgaben des Wissensmanagements umfassen entsprechend die Versorgung der Akteure mit Zugang zu Wissen, wie dies zum Beispiel durch Expertenverzeichnisse, Knowledge-Maps oder durch die Bereitstellung von Information erfolgen kann. Prozessorientierte Ansätze nutzen dabei den Geschäftsprozess als Ausgangs- und Gestaltungsobjekt.

Ein wesentliches Anwendungsgebiet der Modellbildung liegt bei der Beschreibung der Abläufe der Anwendungsdomäne. Dazu gehört neben dem eigentlichen Prozess auch die Interaktion der Akteure mit der zukünftigen Anwendung. Dabei werden in den verwendeten Modellierungsmethoden ausschließlich Informationen betrachtet. Wissensintensive (Teil-)Prozesse, die sich für eine Automatisierung nicht eignen, werden oft nicht näher betrachtet. Dabei ist gerade hier ein höherer Detailgrad der Realitätsabbildung möglich. Gerade im Verständnis der Anforderungen an den Anwender für eine Aktivität liegt jedoch Potenzial für besseres Prozessverständnis und somit auch für die bessere Lösungskonzeption.

Domänenspezialisierung

Domänenspezialisierung bedeutet die Konkretisierung von Modellen/Modellelementen für eine bestimmte Domäne. Dabei dienen plattformunabhängige Modelle als Basis für plattformspezifische Code-Generierung. Bei der domänenspezifischen Modellierung (DSM) werden unmittelbar die Konzepte der Anwendungsdomäne verwendet. Folglich beschränken sich Anforderungen an Sprache und Generatoren lediglich auf diese selbst, sodass die Modellierer direkt mit den Domänenkonzepten arbeiten können. Durch Einschränkung auf aus der Domäne entstammende Sprachkonstrukte ist es dem Modellierer somit idealerweise gar nicht möglich, ungültige Entwurfsmodelle zu spezifizieren.

Im Folgenden betrachten wir das MDA-Entwicklungswerkzeug UML Lab, das modulare Modellierungs-Framework jABC, die Modellierungsmethode KMDL sowie die Modellierungsplattform Innovator.

UML Lab

UML Lab ist ein Entwicklungswerkzeug, das von der Yatta Solutions GmbH aus Kassel entwickelt wird. Die erste Version konzentriert sich vornehmlich auf die Innovationsgebiete Round-Trip-Engineering und modellbasierte Kollaboration. In folgenden Versionen ist Verhaltensmodellierung sowie die Verwendung eigener domänenspezifischer Sprachen vorgesehen. Mit der ersten Version von UML Lab können strukturelle Anteile von Modellen mittels UML-Klassendiagrammen (UML kurz für Unified Modeling Language) modelliert werden. Dafür bietet UML Lab einen Editor mit neuem Eingabekonzept, der das Modellieren stark vereinfacht. Yatta Solutions startet Mitte nächsten Jahres mit der offiziellen Betaphase für UML Lab, für die man sich jetzt unter `www.uml-lab.com` registrieren kann.

Round-Trip-Engineering

Für den finalen Schritt der Quelltexterzeugung werden von UML Lab Templates verwendet, die für Java schon mitgeliefert werden. Templates bündeln Best Practices und andere Optimierungen wie zum Beispiel das Sicherstellen der referenziellen Integrität – je nach Anforderung und auf Wunsch pro Modellelement wählbar. Auch selbst erstellte Templates lassen sich zur Quelltexterzeugung verwenden.

Als offenes Innovationsfeld beim Round-Trip-Engineering wurden die Komplexität des Reverse-Engineerings mittels Parsern und Modelltransformationen sowie das Wartungsproblem im Zusammenhang mit den Templates des Forward-Engineerings identifiziert. Das Problem umgeht UML Lab folgendermaßen: Hier werden die gleichen Templates, die auch für die Quelltexterzeugung verwendet werden, für das Reverse-Engineering genutzt. Die Abbildung des eingelesenen Quelltexts auf das zugrunde liegende Modell erfolgt rein generisch durch Analyse der Templates, mit denen der Quelltext erzeugt werden könnte. Bei Änderungen dieser Templates für das Forward-Engineering wird das Reverse-Engineering gleich automatisch mit angepasst. Das skizzierte Wartungsproblem wird somit vollständig vermieden. Beim erneuten Quelltexterzeugen gehen somit keine manuellen Änderungen verloren. Da sowohl Formatierung des

Quelltexts als auch Kommentare erhalten bleiben, ist generierter Quelltext von handgeschriebenem nicht zu unterscheiden. Mit Hilfe allgemeiner Templates kann auch beliebiger handgeschriebener Quelltext eingelesen werden. Außerdem ist das Erstellen und Verändern von Templates erheblich leichter zu bewerkstelligen als im Falle handcodierter Modelltransformationen.

Egal ob mit Modellieren oder direkt mit Programmieren begonnen wird – UML Lab gewährleistet mit seiner Round-Trip-Funktionalität ein stets zum Quelltext synchrones Modell. Dadurch werden Weiterentwicklungen und Fehlerbehebungen erleichtert und die lückenlose Dokumentation in Form des eingängigen Modells ermöglicht eine verbesserte Qualitätskontrolle.

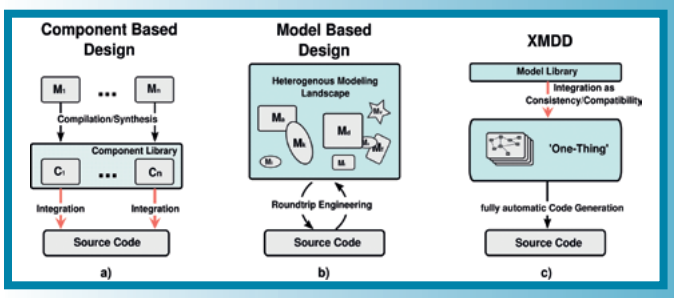

Abb. 2: Evolution der Softwareentwicklungsparadigmen

Modellbasierte Kollaboration

Softwareentwicklung ist Teamarbeit. UML Lab unterstützt das gleichzeitige, parallele Arbeiten an Modellen. Der Entwickler/Softwarearchitekt arbeitet zunächst lokal an Modell bzw. Quelltext und gleicht den Quelltext wie gewohnt per SCM mit seinen Kollegen ab. Zur Versionsverwaltung des Modells bietet UML Lab unterschiedliche Verfahren, wobei jeweils ein optimistisches Sperrkonzept verfolgt wird.

Zunächst kann das Modell per SCM verwaltet werden. Kommt es beim Check-in zum Konfliktfall, überlässt UML Lab dem Entwickler dessen Auflösung. Dazu wird ein Diff-View angeboten, in welchem die konkurrierenden Änderungen im Diagramm farblich markiert sind. Zusätzlich zu dieser SCM-basierten Modellverwaltung bietet UML Lab einen eigenen Repository-Server.

Somit ist eine nahtlose Integration in Projekte bestehend auch aus großen Teams möglich. Egal ob man sich für die Verwaltung über ein bestehendes SCM oder für die Nutzung des eigenen Repository-Servers entscheidet, die Softwareentwicklung kann wie gewohnt weiter betrieben werden.

jABC

Das „Java Application Building Center“ (jABC) verknüpft modellorientierte Verfahren mit der Anbindung von Fachwissen und mit der Sprache, Sichtweise und den Konzepten des Experten, wie es im Wissensmanagement und in der domänenspezifischen Modellierung propagiert wird. Vor allem für Geschäftsanwendungen müssen komplexe Systeme und Prozesse kontinuierlich aktualisiert werden, um sie den sich ständig wandelnden Marktbedingungen anzupassen.

Kontinuierliche, modellgetriebene Softwareentwicklung wird durch das von jABC umgesetzte XMDD-Paradigma (Extreme Model-driven Development, s. [MaSte08]) wiedergegeben, wobei XMDD selbst auf dem Konzept des One-Thing Approach (OTA) basiert, das die Einfachheit des Wasserfallmodells mit höchstmöglicher Agilität verbindet [MaSte09]. So führt XMDD zu einer transparenten, agilen Entwicklung, deren Modelle einem evolutionären Fortschritt unterliegen, der von Qualitätssicherungsmaßnahmen wie Versionierung oder Produktlinienmanagement begleitet wird. Methodisch und technisch schließt XMDD an die Architekturmodellierung an und fokussiert die Abbildung von Geschäftsabläufen. Gegenstand von XMDD sind demnach die Prozessmodellierung und -umsetzung.

Typische Anwendungsdomänen sind jene des *Center of Innovation for Service Centered Continuous Engineering* (SCCE), nämlich u. a. Gesundheitspflege, Biotechnologie, Daten- und Wissensmodellierung, Kollaborations- und Entscheidungsunterstützung.

Round-Trip-Engineering

Die Heterogenität im MDA-Ansatz (getrennt voneinander existierende Komponenten und Modelle) werden in XMDD mittels OTA überwunden. Den Modellen liegt eine Orchestrierungen diverser Dienste zugrunde, was zur (servicebasierten) Virtualisierung und strikten Trennung von Plattform und Architektur führt. Aufgrund der durchgängigen Kompatibilität der Modelle, Anwendungen und Domänen auf Metaebene sind die Modelle bereits auf Korrektheit überprüft, sodass beispielsweise Enterprise Mashups* direkt per Codegenerierung realisiert werden können. Da hierbei keine Änderungen am Code vorgenommen werden, kann durchgängige Konsistenz ohne jedwede Form von Round-Trip-Engineering gewährleistet werden. Entsprechende Beispiele in den Bereichen mobiles Internet mit NGN IMS-Techniken oder Emergency Handling im Ambient Assisted Living wurden bereits auf der IFA 2008, der CeBIT 2009 im Future Parc, der TeleMed und ConHIT 2009 gezeigt.

Abbildung 2 skizziert die Evolution der Softwareentwicklungsparadigmen, vom Wiederverwendungsgetriebenen komponentenorientierten Design, über das Modell-heterogene, modellgetriebene Design, bis hin zum XMDD-getriebenen, kontinuierlichen, modellgetriebenen Engineering. Ziel dieser Entwicklung ist die (konsistente) Überwindung kultureller Gräben.

Modellbasierte Kollaboration

Die modellbasierte Kollaboration gründet auf der Tatsache, dass grafische Modelle die Kommunikation über abstrakte Sachverhalte erleichtern. jABC verwendet für die grafische Modellierung sogenannte SIBs (Service Independent Building Blocks) als unabhängige serviceartige Modellierungs- und Ausführungsbausteine. SIBs sind unabhängige Funktionseinheiten, die sich in gleichen sowie in unterschiedlichen Services wieder verwenden lassen. Sie wurden bereits vor über zwanzig Jahren im „Intelligent Network Conceptual Model“ (INCM) definiert, das der Entwicklung von IN-Standards für Intelligente Netzwerke durch die ITU (International Telecommunications Union) und das ETSI (European Telecommunication Standardization Institute) zugrunde liegt. Weitverbreitete Telefondienste wie die 0800- oder 0180-Nummern, das Televoting oder das VPN sind nach diesen Prinzipien realisiert.

Mit XMDD werden komplexe Wertschöpfungs- und Kommunikationsprozesse in beliebiger Detailtiefe erfasst, vereinfacht, miteinander gekoppelt und auftretende Störungen behoben. So werden Auftraggeber in die Lage versetzt, Transaktionen branchenweit, unternehmensweit und punktuell zwischen zwei beliebigen Organisationen gleichen oder unterschiedlichen Typs in komplexen Prozessmodellen zusammenzuführen. Die so entstehenden Prozessmodelle werden als „Living Model“ bezeichnet, da sie

- jederzeit auf der jeweiligen Realisierungsstufe ausführbar sind,
- selbst die Grundlage für die fortlaufenden Änderungen auf Prozessebene sind und

* Ein Mashup bezeichnet die Kombination bestehender Inhalte oder Daten bzw. Services aus unterschiedlichen Quellen in einer kombinierten und damit neuen Anwendung.

- Veränderungen der involvierten Systeme und Services unmittelbar widerspiegeln.

Das „Living Model" erhält seine Lebendigkeit aus der fortlaufenden Anpassungsfähigkeit an Veränderungen in der Prozesskettenumwelt oder im System selbst. Jeder Entwicklungsschritt der Modellierung wird sofort (quasi nicht hintergehbar für Entwickler und zukünftige Anwender) dokumentiert und grafisch dargestellt, wodurch sich aufwendige Koordinationsverfahren zwischen den Kollaborationsbeteiligten verschlanken.

Wissensmanagement

Obwohl Geschäftsprozesse seit vielen Jahren maschinell abgebildet werden, erfordert die Technologienutzung enorme Zeit- und Personalressourcen für Neuerungen und Anpassungen. Aus diesem Grund thematisiert XMDD das Prozesskettenmanagement (transparente, nachhaltige Prozessverwaltung in verteilten Architekturen), die Sicherstellung gesetzlicher Rahmenbedingungen in Geschäftsprozessen (z. B. Basel II) und die Prozessevolution (Modellierung des Status quo, Analyse und Optimierung). Exemplarisch hierfür sind die Steuerung weltweit verteilter Analyseprozesse im Bereich der Bioinformatik oder das Management komplexer Softwaretests. Die Dokumentation der Prozessmodelle enthält textuelle Spezifikationen, (ggf. formalisierte) Rahmenbedingungen, GUI-Beschreibungen und Rollen mit festgelegten Nutzerrechten.

Domänenspezialisierung

Charakteristisch für den XMDD-basierten OTA ist die Ebenenübergreifende Transparenz. Der relevante Effekt von Designentscheidungen und Detailergänzungen wirkt sich unmittelbar auf die jeweiligen Prozessmodellierungsebenen aus. Das kann z. B. den Simulationscode, die tatsächliche Implementierung, die Rollenfestlegungen oder die Geschäftsregeln betreffen. Auf diese Weise baut sich die Nutzererfahrung für den Prozessmodellierer begleitend mit dem Entwicklungsprozess sukzessive auf und Missverständnisse oder Inkonsistenzen können durch Fachexperten frühzeitig erkannt werden. In diesem Zusammenhang hat sich die Domänen- und Ebenen-spezifische begriffliche und visuelle Flexibilität des jABC als vorteilhaft erwiesen. Komponenten und Modelle können im jeweiligen Fachjargon ontologisch klassifiziert und den jeweiligen Konventionen entsprechend visualisiert werden.

KMDL

Die „Knowledge Modeling and Description Language" (KMDL®) ist eine Modellierungsmethode für wissensintensive Prozesse, mit der auch die Teile des Prozesses abgebildet werden, die nicht durch das zu entwickelnde Informationssystem abgedeckt werden. Darüber hinaus beschreibt die KMDL in der Aktivitätssicht real ablaufende Informations- und Wissensaktivitäten. Diese laufen oft unbewusst und informell ab und werden oft erst im Rahmen der Modellierung transparent. Die KMDL kann daher vor allem Prozesse mit intensiver Informations- und Wissensgenerierung, -verteilung und -anwendung besser als andere Modellierungsmethoden beschreiben. Die menschliche Aktivität und Leistung, insbesondere die eingebrachten Skills und Wissen, ist in diesen Prozessen essenzieller Bestandteil.

In KMDL werden drei Modelltypen unterschieden: Prozess-, Aktivitäts- und Kommunikationssicht. Die Prozesssicht stellt die logische Verknüpfung von Aufgaben dar. Die Aktivitätssicht ist eine Verfeinerung der Aufgaben des Prozesses. Dabei werden Informationen und Wissen sowie gegebenenfalls bestehende Wissens- und Kompetenzanforderungen abgebildet. In einer dritten Sicht werden die Kommunikationsbeziehungen aggregiert auf Basis empirischer Erhebungen für die Menge der Instanzen in der Kommunikationssicht abgebildet.

Zur Modellierung kann der K-Modeler verwendet werden, welcher kostenfrei auf der Website `www.k-modeler.de` verfügbar ist. Das in Eclipse integrierte Werkzeug bietet ein Repository für die Modellobjekte und Modellierungswerkzeuge. Darüber hinaus sind diverse Auswertungen, wie Reports oder die Suche nach Schwachstellen mit Hilfe von Patterns, zur Bewertung und letztendlich zur Gestaltung der Soll-Prozesse vorhanden.

Der K-Modeler ist ein Modellierungswerkzeug für wissensintensive Geschäftsprozesse. Darüber hinaus kann die Nutzung von Funktionen oder Services in der Aktivitätssicht modelliert werden. Diese wiederum beschreiben einen logisch abgegrenzten funktionalen Bestandteil eines Informationssystems. Die Modellierung führt so zunächst zur Beschreibung des Ist-Zustandes und in mehreren Iterationen zum Soll-Modell eines Prozesses und der Anwendungen. Der Softwareentwickler entnimmt dann aus dem Modell sowohl die Funktionsbausteine als auch die Zusammenhänge, besonders die verwendete und erzeugte Information und die beteiligten Akteure.

Die aktuelle Version des K-Modelers kann um Mechanismen zum Forward-Engineering erweitert werden. Dazu werden die Modelle zunächst in die XML-Notation KMML überführt. Die Transformation von KMML zu BPMN (Business Process Modeling Notation) ist in [FrGrSu08] beschrieben. Ein Reverse-Engineering ist aufgrund des betrachteten Modellierungsausschnittes nicht automatisch, jedoch im Rahmen der manuellen Modellerstellung möglich. Die kommende Version des K-Modelers wird die Kommunikationssicht beinhalten und um weitere Analysemöglichkeiten ergänzt.

Innovator

Innovator ist ein Produkt des Nürnberger Softwarehauses MID GmbH und umfasst eine durchgängige Modellierungssuite von Geschäftsprozess-, Anforderungs-, objektorientierter-, daten- und funktionaler Modellierung mit Transformationen zwischen den Modelltypen. Dabei steht die ganzheitliche und nahtlose Modellierung gemäß dem modellgetriebenen Ansatz im Vordergrund. Ziel ist es, die vor allem in größeren Entwicklungsprojekten vorherrschende Komplexität beherrschbar zu machen und für alle Beteiligten eine präzise Kommunikationsgrundlage zu schaffen.

Zu den Vorteilen, die man werkzeugtechnisch mit dem MDA-Paradigma verbindet, gehören die valide Erhebung der funktionalen Anforderungen in maschinenlesbarer Form, die Generierung von größeren Codeteilen statt deren manueller Erstellung, die Automatisierung von Tests und eine leichtere Wartbarkeit des Produkts über dessen gesamten Lebenszyklus.

Round-Trip-Engineering

In modellgetriebenen Ansätzen erfolgt die Codegenerierung aus Elementen einer höheren Abstraktionsebene. Dazu enthält entweder der Generator selbst die Informationen zum Konkretisieren des abstrakten Modells (z. B. in den Templates), oder aber es wird ein Grobgerüst generiert, welches im Anschluss vom Entwickler ausimplementiert wird.

Wie bereits in der Einleitung beschrieben, ist ein Reverse-Engineering schwierig, denn es besteht stets die Gefahr, das Modell mit Implementierungsdetails zu überladen. Moderne Entwicklungsumgebungen wie Eclipse oder Visual Studio bieten hier die Möglichkeit, das Feindesign auf Codeebene zu vi-

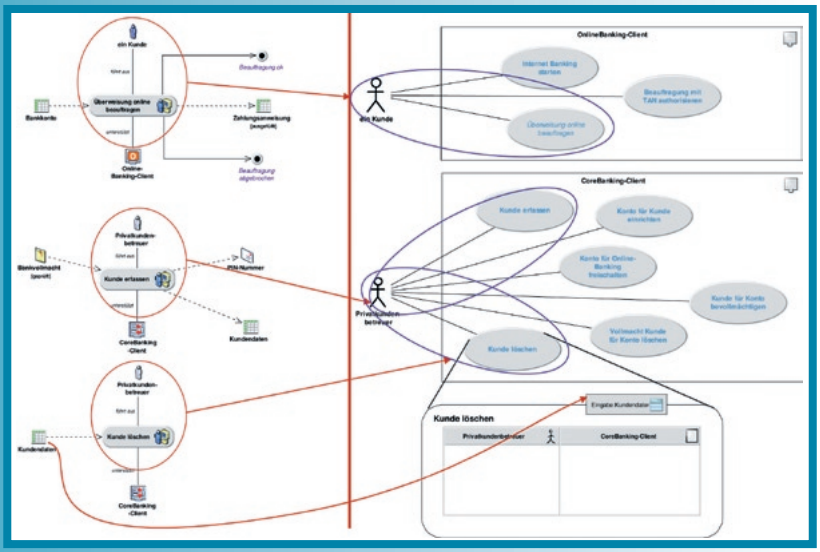

Abb. 3: Fiktiver Bankgeschäftsprozess

sualisieren. In Innovator können Modellierer auf sogenannte Harvesting-Funktionalitäten zurückgreifen, die existierenden Code parsen und die Strukturen als Modellelemente rekonstruieren. Dabei unterstützt Innovator folgende Programmiersprachen: C, Java, C++, C#, WSDL und XSD.

Modellbasierte Kollaboration

Modelle erleichtern die Kommunikation zwischen den Entwicklungsteams, egal ob es sich um kleinere, überschaubare Einheiten oder sehr große und an verschiedenen Orten operierende Personengruppen handelt. Innovator verfolgt bezüglich der Team-Kollaboration den Ansatz eines zentralen Online-Repositorys. Gegenseitiger Ausschluss wird mittels einer feingranularen pessimistischen Sperrstrategie zugesichert. Die Konsistenz der Ergebnisse wird damit von vornherein gefördert und erspart späteren hohen Konsolidierungsaufwand.

Durch einen im Innovator vorhandenen Diff/Merge-Mechanismus können unterschiedliche Modellversionen miteinander verglichen, die Änderungen nachvollzogen und die einzelnen Versionen aufeinander abgestimmt werden. Die Weitergabe an klassische Konfigurations- bzw. Versionierungswerkzeuge bleibt davon unbenommen. Für Interessenträger, die keinen Innovator-Client installiert haben, besteht die Möglichkeit des Read-Only-Zugriffs mithilfe eines Web-Browsers.

Wissensmanagement

Modelle beinhalten das Wissen über die Geschäftsprozesse und Vorgänge im Unternehmen (Prozesswissen). Die Darstellung und die Dokumentation dieses Wissens bzw. dieser Informationen basiert bei Innovator auf der UML und zukünftig auch auf der BPMN. Dies bedeutet, dass Aktivitäten, ihre beteiligten Rollen und die ein- und ausgehenden Informationen hierbei Mittel zum Zweck sind. Mithilfe einer einfachen automatischen Abbildung lassen sich einzelne IT-relevante Aktivitäten z. B. als Use-Cases in das Anforderungsmodell eines IT-Systems übernehmen. Dabei wird mittels einer Trace-Beziehung dieser Zusammenhang dokumentiert und nachvollziehbar. Der Informationsbedarf der Geschäftsprozessaktivität geht ebenfalls in das IT-Modell über, indem aus vorhandenen Geschäftsobjekten korrespondierende Informationsobjekte angelegt werden. Abbildung 3 veranschaulicht dies anhand eines fiktiven Bankgeschäftsprozesses.

Domänenspezialisierung

Im Rahmen des von der Firma MID favorisierten MDSD-Ansatzes (Model-Driven Software Development) setzt man hierbei auf UML-basierte Profile. Eine DSL dient dabei dem Zweck, die Schlüsselaspekte einer Domäne modellierbar zu machen. Dazu besitzt sie ein Metamodell und eine korrespondierende konkrete Syntax. Die Domäne entspricht dem fachlichen Anwendungsbereich der DSL und die konkrete Syntax entspricht bei Innovator der UML.

DSLs werden mithilfe der UML-Profile definiert. Dabei bietet Innovator eine größere Vielfalt an Erweiterungsmöglichkeiten dieser Profile an, als dies innerhalb der UML bzw. der OMG vorgesehen ist. Die UML-basierte DSL wird im Vorfeld als Klassenmodell der Begrifflichkeiten modelliert. Die Domänen sind in der MID-Modellierungsmethodik M³ feingranular bestimmt – für jede einzelne MDA-Modellierungsebene (CIM, PIM, ASM, PSM) existiert jeweils eine eigene DSL.

Literatur

[FrGrSu08] J. Fröming, N. Gronau, E. Sultanow, MDA Werkzeuge – Im Vergleich: jABC, AndroMDA und OpenArchitectureWare, in: iX, 8/2008

[MaSte08] T. Margaria, B. Steffen, Agile IT: Thinking in User-Centric Models, in: Proc. ISoLA 2008, CCIS N.17, pp. 493–505, Springer, 2008

[MaSte09] T. Margaria, B. Steffen, Business Process Modelling in the jABC: The One-Thing Approach, IGI Global, 2009, `http://www.igi-global.com/downloads/excerpts/33287.pdf`

[Rau07] K.-H. Rau, Objektorientierte Systementwicklung: vom Geschäftsprozess zum Java-Programm, Vieweg, 2007

Julian Bahrs ist wissenschaftlicher Mitarbeiter am Lehrstuhl für Wirtschaftsinformatik und Electronic Government der Universität Potsdam. Seine Forschungsschwerpunkte sind die Modellierung und Gestaltung wissensintensiver Geschäftsprozesse sowie Softwarewerkzeuge für das Wissensmanagement.
E-Mail: julian.bahrs@wi.uni-potsdam.de.

Manuel Bork ist Softwareentwickler bei der Yatta Solutions GmbH, die als Spezialist für modellgetriebene Softwareentwicklung ab 2010 das Entwicklungswerkzeug UML Lab sowie Schulungen und Consultingdienstleistungen anbietet.
E-Mail: bork@yatta-solutions.com.

Prof. Tiziana Margaria ist Lehrstuhlinhaberin für Service und Softwareengineering an der Universität Potsdam und Vorsitzende der European Association for Software Science and Technology (EASST), Mitglied der Semantic Web Service Challenge und des SAP Platform Thought Leadership Councils und Advisory Board.
E-Mail: margaria@cs.uni-potsdam.de.

Eldar Sultanow arbeitet als JEE-Entwickler/Architekt bei der Producto AG in Berlin.
E-Mail: eldar.sultanow@testberichte.de.

Interview mit Jochen Seemann und Alexander Bösl (MID GmbH)

Eldar Sultanow sprach mit Jochen Seeman und Alexander Bösl vom Nürnberger Softwarehaus MID GmbH, dem Hersteller der MDA-Modellierungsplattform Innovator.

E. Sultanow: Worin bestehen heute die Hauptschwierigkeiten der modellgetriebenen Softwareentwicklung und wie kann man diesen begegnen?

J. Seemann/A. Bösl: Die Einführung der MDSD stellt einen massiven kulturellen Umstieg innerhalb der Organisation dar. Traditionelle Arbeitsweisen werden nachhaltig verändert und die Mitarbeiter müssen bereit sein, sich auf neue Methoden, Werkzeuge und veränderte Abläufe einzustellen. Klassische Verhaltensmuster weichen einer formaleren Denkweise.

Auf der strategischen Ebene muss es uns einfach noch besser gelingen, den Verantwortlichen klar darzustellen, wie eine modellbasierte IT Geschäftsziele besser unterstützt und den Unternehmen Wettbewerbsvorteile bietet. Hier sehen wir eine klare Korrelation zwischen Business und IT. Auch die wirtschaftliche Transparenz muss noch stärker in den Vordergrund gestellt werden. Will heißen: Es müssen klare Antworten gegeben werden auf Fragen wie: Was kostet mich die Erstellung des Systems? Was kosten mich die Änderungen am System im Laufe des Lebenszyklus einer Applikation? Vor allem muss hierbei auch der Return of Investment als Ganzes stärker in den Mittelpunkt der Betrachtung rücken. Modelle können dabei von großem Nutzen sein.

In Bezug auf technische Aspekte sehen wir die größten Herausforderungen im Umfeld des Changemanagement und in der Umsetzung von Änderungen in verteilten Teams. Auch die Rolle der Datenmodellierung als Aspekt der MDA wurde in der Vergangenheit stark unterschätzt.

E. Sultanow: Welche Entwicklungen und Trends haben sich daraus ergeben?

J. Seemann/A. Bösl: Ein Trend, den wir in unseren Gesprächen und auch am Markt wahrnehmen, ist die zunehmende Modellierung von Prozessen im Fachbereich. Businessanalysten modellieren ihre Geschäftsprozesse und die Unternehmensabläufe. Daraus folgt aber, dass die Werkzeuge einfach und intuitiv bedienbar sein müssen.

Weiter stellen wir fest, dass der Trend zur serviceorientierten Architektur (SOA) die IT näher an die Geschäftsprozesse heranbringt und sich die beiden Disziplinen immer mehr verzahnen.

Schließlich ist die Tendenz in den Unternehmen zur Konsolidierung ihrer Toollandschaft inzwischen stark ausgeprägt. Der Wunsch nach zentralen Repositorys verbunden mit einer überschaubaren Anzahl von Entwicklungstools wird von den Verantwortlichen klar artikuliert.

E. Sultanow: Auf welche konkreten Aspekte legt Ihr Unternehmen den Fokus für die folgenden Jahre?

J. Seemann/A. Bösl: Durch unseren Ansatz einer integrierten Modellierungssuite für die unterschiedlichen Bereiche der Modellierung tragen wir diesen Tendenzen verstärkt Rechnung. Auch Industrieanalysten wie Gartner sehen diesen Trend, der durch Bestrebungen wie Enterprise Architecture Modeling noch bestärkt wird. Wir führen mit unserem Modell-Server verschiedene Blickwinkel auf IT-Systeme zusammen und kommen damit dem Wunsch nach einem zentralen Repository für Informationen rund um die IT-Landschaft sehr nahe. Außerdem integrieren wir in Zukunft noch Systeme zum Anforderungsmanagement und vor allem zur Anforderungsermittlung, um Mitarbeitern in Fachbereichen noch einfacheren Zugang zur modellbasierten Systemanalyse zu geben.

Ein weiterer Schwerpunkt ist sicherlich, dass Modellierungswerkzeuge einfacher und intuitiver benutzbar gemacht werden müssen. Viele Businessanalysten und Fachbereichsmitarbeiter greifen auch heute noch zu einfachen Graphikwerkzeugen, wenn sie ihre Anforderungen oder Prozesse modellieren. Deshalb haben wir für die nächste Produktgeneration eine völlig neue Oberfläche mit neuen UI-Frameworks erstellt, sodass sich Innovator für diese Benutzergruppe im gewohnten „Office"-Look präsentiert und damit unnötige Einstiegshürden vermieden werden.

E. Sultanow: Die Verbreitung von model-driven Development (MDD) trägt noch viele Potenziale in sich. Was wurde in den letzten Jahren falsch gemacht und sollte in Zukunft vermieden werden?

J. Seemann/A. Bösl: Man wollte zu viele Ziele mit einem modellgetriebenen Ansatz gleichzeitig verfolgen: Wiederverwendung von Wissen, Erhöhung der Wartbarkeit, effektivere Softwareentwicklungsprozesse, einfache Skalierbarkeit der Systeme usw. Die MDD/MDA wurde fast zum Allheilmittel deklariert und konnte diese Maximalforderung sicherlich nicht erfüllen.

Der Schwerpunkt lag in der Vergangenheit viel zu sehr auf Entwicklerseite und auf technischen Fragen wie Codegenerierung. Das Potenzial der MDD/MDA für die Kommunikation und Zusammenarbeit wurde dagegen unterschätzt und ist bei Weitem noch nicht ausgeschöpft. Dabei ist es nun mal nicht sinnvoll, auf der einen Seite große Mengen von Code durch MDD/MDA erfolgreich zu erzeugen, während man sich auf der anderen Seite über die genauen Zusammenhänge im Fachbereich nicht ganz klar ist.

E. Sultanow: In welcher Form wird die interdisziplinäre Kollaboration mit Innovator unterstützt?

J. Seemann/A. Bösl: Innovator ist in unterschiedlichen Produktvarianten erhältlich, die spezifisch für die Rolle der jeweiligen Mitarbeiter entwickelt und konzipiert wurden. So nutzen Businessanalysten und Fachbereichsexperten für Geschäftsprozess- und Anforderungsmodellierung eine andere Innovator-Modellierungsumgebung als Softwarearchitekten in der IT und Datenbankarchitekten. Damit tragen wir den unterschiedlichen Blickwinkeln der jeweils Beteiligten entsprechend Rechnung. Dennoch sind alle Modelle im zentralen Innovator-Server miteinander vernetzt und erlauben so die Zusammenarbeit zwischen Fachabteilungen, Applikationsentwicklern und Datenbankverantwortlichen.

E. Sultanow: Inwiefern hat MDD infolge der Globalisierung und der internationalen Zusammenarbeit an Bedeutung gewonnen?

J. Seemann/A. Bösl: Im Zuge der Globalisierung geht der Trend hin zu räumlich verteilten Entwicklungsteams, die immer schneller tragfähige Entwicklungsergebnisse liefern müssen. Der Model-Server des Innovator stellt den Benutzern alle relevanten Informationen zeitnah und auf aktuellstem Stand zur Verfügung. Damit werden Modelle immer mehr zum Kommunikationsmittel für Off-Shore-Entwicklungen und vor allem auch Off-Shore-Testen.

Aber auch die Organisationen selbst sind natürlich immer stärker verteilt, und so gewinnt auch hier die Rolle der Modelle für die interne Kommunikation an Bedeutung. Das hat ganz konkrete Auswirkungen auf unsere Produkte und bedeutet z. B. für Innovator, dass in der nächsten Produktgeneration Modelle mehrsprachig angelegt werden können, d. h. dass das gleiche Geschäftsprozessmodell beispielsweise in Deutsch, Englisch und Französisch bearbeitet werden kann.

Interview mit Klaus Wendland

Eldar Sultanow sprach mit Klaus Wendland, Geschäftsführer der provalida GmbH, einem Beratungshaus für Geschäftsprozessanalyse (BPA), Geschäftsprozessmanagement (BPM), Projekt- und Portfolio-Management (PPM), Identitätsmanagement (IdM) und Qualitätsmanagement (QM), über die gegenwärtigen Herausforderungen und Möglichkeiten im Bereich der Optimierung und Automatisierung von Geschäftsprozessen.

E. Sultanow: Herr Wendland, wie beeinflusst Ihrer Meinung nach die aktuelle Wirtschaftskrise das Management der Geschäftsprozesse?

K. Wendland: Die größten Herausforderungen unserer Zeit sehen Unternehmen in der zeitnahen Anpassung ihrer Geschäftsmodelle und in der Innovationsgeschwindigkeit ihres Unternehmens. Zwei Dinge treten hierbei unter wachsendem Kostendruck noch weitaus stärker als sonst in den Vordergrund: Agilität und Effizienz. Die Fähigkeit, Geschäftsprozesse flexibel an veränderte Marktbedingungen anpassen zu können, bedeutet gerade jetzt einen echten Wettbewerbsvorteil. Insbesondere wenn die Prozesse IT-gestützt zu weiteren Kosteneinsparungen führen.

E. Sultanow: Flexible Geschäftsprozesse auf Seiten des Business. Schön und gut. Aber bedeutet das nicht auch kürzere Zeitpläne und Release-Zyklen auf Seiten der IT, sodass das berüchtigte Spannungsfeld zwischen Business und IT noch weiter belastet wird?

K. Wendland: Diese Gefahr besteht in der Tat. Ein Ausweg aus diesem Dilemma kann nur darin bestehen, die interne Zusammenarbeit zu verbessern. Hierbei empfiehlt es sich, mit einem ganzheitlichen Ansatz, Fachabteilungen und IT an einen Tisch zu holen, und in einer gemeinsamen Sprache Prozesse zu definieren. Die zentralen Aspekte einer solchen Modellierung sind Transparenz und Konsistenz. Idealerweise sollte sich eine solche Modellierung flexibel an die jeweiligen Bedürfnisse eines Unternehmens anpassen lassen, um Missverständnisse und Konflikte weitestgehend zu vermeiden. Ein ganzheitliches BPM-Werkzeug entlastet darüber hinaus die IT-Seite bei der Realisierung der Prozesse. Einfache Änderungen von Geschäftsprozessen sollten von der Fachabteilung selbst durchgeführt werden können, ohne große Implementierungsaufträge für die IT zu generieren. Darüber hinaus bieten sich für die Realisierung serviceorientierte Architekturen und modellgetriebene Softwareentwicklungsansätze an. Zum einen werden so die starren IT-Infrastrukturen der Vergangenheit aufgebrochen, und zum anderen erreicht man auf diese Weise eine klare Trennung von Funktionalität und Technik.

E. Sultanow: Softwareentwicklung findet häufig durch externe Dienstleister und in zunehmendem Maße auch durch die Zusammenarbeit weltweit verteilt arbeitender Experten statt. Wie lässt sich das aus MDA-Sicht beschreiben?

K. Wendland: Ein grundlegendes Problem verteilter Softwareentwicklung besteht darin, dass zeitgleich immer nur ein Entwickler an einem Modell arbeiten kann: Daraus resultieren zwei mögliche Vorgehensweisen. Zum einen bietet es sich an, eine zentrale Versionsverwaltung mit einem darunterliegenden Rechte- und Rollensystem einzusetzen. Zum anderen sind häufig klar definierte Kommunikationsregeln und -wege notwendig, damit jeder Teil des Entwicklungsteams klar definierte Aufgaben hat und darüber informiert ist, wer wann in welchem Bereich zuständig ist. Darüber hinaus muss man insbesondere bei großen verteilten Teams darauf achten, dass gerade die Modelle in sinnvolle Teilmodelle und entsprechende Dateien aufgeteilt sind. Es ist einfach nicht praktikabel, dass einzelne Personen über längere Zeiträume große Teile des Systems exklusiv bearbeiten. Der klassische „Branch&Merge"-Ansatz, der bei Quellcode sehr gut funktioniert, wird bisher bei grafisch erstellten Modellen nur schlecht unterstützt.

E. Sultanow: Welche Fehler wurden in der Vergangenheit gemacht?

K. Wendland: Ein großer Fehler, der immer wieder vorkommt, besteht darin, die Kommunikation zu vernachlässigen und zu sehr daran zu glauben, ein bestimmtes Werkzeug biete die Lösung für alle Probleme. Auch das beste Werkzeug schützt nicht davor, dass Entwicklungen auseinanderlaufen und nicht gut aufeinander abgestimmt sind, wenn die Kommunikation nicht stimmt. Auch kommt es vor, dass Arbeiten doppelt erledigt werden. Ein weiteres Problem stellt die Verwendung von zu vielen verschiedenen Tools dar. Zum einen wird dadurch die Zusammenarbeit erschwert, da sich Entwickler untereinander nicht mehr „mal eben" unterstützen können. Zum anderen sind so Medienbrüche und Inkompatibilitäten vorprogrammiert.

Medienbrüche sind insbesondere ein wesentliches Problem im Bereich der Prozessmodelle. Typische MDA-Werkzeuge richten sich an Entwickler, welche die zugrundeliegenden Technologien kennen und entsprechende Modelle und Generatoren entwickeln können. Diese Werkzeuge sind selten für die Fachabteilungen geeignet, da sie zu viel IT-Ballast voraussetzen. Daher setzt die Fachseite meist andere Prozessmodellierungswerkzeuge ein. Es gibt nur sehr wenige ganzheitliche Tools, die eine medienbruchfreie Entwicklung von Geschäftsprozessen erlauben. Aber gerade im Bereich der Geschäftsprozesse müssen die Fachseite und die Entwickler eng zusammenarbeiten, um gemeinsam optimierte Lösungen entwickeln zu können.

E. Sultanow: Wo setzt jABC/XMDD an, diese Fehler zu vermeiden bzw. Outsourcing zu unterstützen?

K. Wendland: jABC stellt eine ganzheitliche Lösung dar, Geschäftsprozesse von ihrer Analyse und Modellierung, über ihre technische Realisierung und Ausführung, bis hin zum Monitoring vollständig zu unterstützten. Es zeichnet sich durch seine transparente und konsistente Modellierung aus, auf deren Grundlage die fachlichen Anwendungsexperten und die IT gemeinsam Geschäftsprozesse realisieren können. Kommunikationsprobleme zu vermeiden, ist ein zentrales Ziel des jABC. Die Modellierung ist darauf ausgelegt, von der Fachseite verstanden und durchgeführt zu werden. Die IT-Seite ergänzt technische Aspekte und verantwortet die Implementierung. Das jABC eignet sich also insbesondere für agile Softwareentwicklungsmethoden, in denen die modellierten Prozesse immer wieder mit den Wünschen des Kunden abgeglichen werden müssen. Darüber hinaus verwaltet das jABC alle relevanten Informationen in einem zentralen Repository und stellt sie über eine einheitliche Schnittstelle zur Verfügung.