**Raimund Czech** ist Chief of System Architecture im Bereich Business Intelligence der T-Systems International GmbH, Frankfurt am Main.
E-Mail: Raimund.czech@t-systems.com

Modellgetriebene Gestaltung von Software

# Automatismus ersetzt den Künstler

*In modernen Architekturen und Vorgehensmodellen ist der modellgetriebene Entwicklungsansatz (Model-Driven Development, MDD) ein Oberbegriff für Techniken, die aus standardisierten Modellen automatisiert lauffähige Software beziehungsweise Systemkomponenten erstellen. Software-Systeme werden so nicht mehr klassisch künstlerisch programmiert, sondern teilautomatisiert industriell gefertigt. Dabei unterstützt dieses Vorgehen nicht nur den Definitionsprozess der Anforderungen und den Designprozess des Systems, sondern vielmehr die Entwicklung, den Betrieb, die Wartung und die Weiterentwicklung. Um die Möglichkeit zu schaffen, diese beschriebene industrielle Fertigungstiefe ebenso bei Business Intelligence zu erreichen, wird nun auch hier verstärkt MDD eingesetzt – mit vielen Vorteilen.*

Die Grundlage bilden domänenspezifische Sprachen (englisch: Domain-Specific Language, kurz: DSL), die konzernübergreifend definiert werden. Durch die daraus resultierenden maschinenlesbaren Modelle, in denen die Informationen des gesamten Lebenszyklus zur Verfügung stehen, lassen sich der Automatisierungsgrad erhöhen und der Fertigungsaufwand minimieren.
Durch eine DSL wird der Gesamtprozess barrierefrei werkzeugunterstützt abgebildet – im Gegensatz zum klassischen BI-Designprozess (vgl. Abbildung 1), bei dem die einzelnen Prozessschritte autark durchgeführt werden. Somit können alle beteiligten Rollen miteinander kommunizieren und sich auf ein gemeinsames Verständnis einigen. Dadurch werden zum Beispiel die Vertreter der Wirtschaft ohne detailliertes technisches Verständnis und ohne die Verwendung technischer Fachbegriffe (zum Beispiel Bridge-Tabellen) in die Lage versetzt, den Entwicklern die Anforderungen unmissverständlich vorzuschreiben. Auswirkungen in den einzelnen Prozessphasen lassen sich über den Gesamtprozess bewerten. Jeder Anwender erkennt sofort, welche Auswirkungen die Änderungen der konzeptionellen Anforderungen auf das physische Datenmodell haben.
Durch eine fehlende DSL und die autarke Verwendung der Werkzeuge hingegen driften im klassischen BI-Designprozess das fachliche Datenmodell und die physikalische Abbildung auseinander. Die verwendeten Werkzeuge haben keinen Bezug zueinander und die Artefakte, die den Transport der Informationen sicherstellen sollen, besitzen meist proprietären Charakter. Änderungen werden oft nur im physischen Modell durchgeführt, da es keine übergeordnete maschinelle Prozessunterstützung gibt. Das macht die Pflege sehr zeitaufwendig und kompliziert. Die Modelle haben leider nur einen „Dokumentationscharakter“, wobei der „Wahrheitsgehalt“ nicht abzuschätzen ist. Je nach Lebenszyklus lässt sich kein Bezug zwischen Anforderungen, Datenmodell und der implementierten Applikation herstellen. Die Wartung der Systeme erweist sich als kritisch. Neue Anforderungen sind somit oft nicht oder nur mit erheblichem Aufwand realisierbar, beziehungsweise der Aufwand lässt sich im Vorfeld nicht abschätzen. Die Umsetzbarkeitsanalyse findet im eigentlichen Realisierungsprojekt statt.

## Domain-Specific Language

Durch die Verwendung einer domänenspezifischen Sprache können für einen Konzern beziehungsweise ein Projekt standardisierte Modelle strukturiert werden. Unterschiedliche Modellbereiche haben jeweils eigene Regeln. Sie legen fest, welche Elemente wie und in welchem Umfang verwendet werden können. Es empfiehlt sich, dabei international genormte Darstellungselemente zu verwenden, sodass viele DSLs durch UML2-Profile (Unified Modeling Language) umgesetzt werden. Die DSL definiert für verschiedene Sachverhalte Diagramme, die nur bestimmte Sets (Teilmengen) an stereotypisierten Ele-

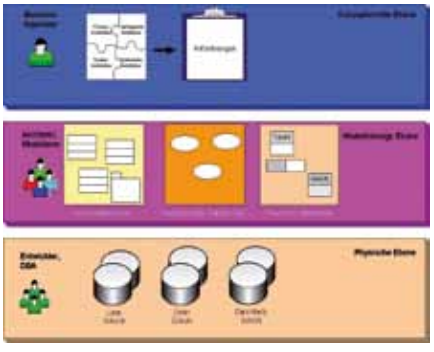

*Abb. 1: Konventioneller BI-Designprozess*

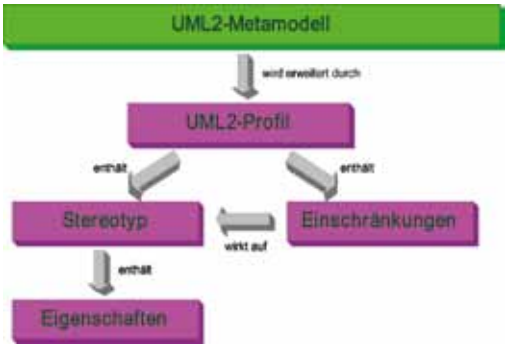


*Abb. 2: UML2-Profil*

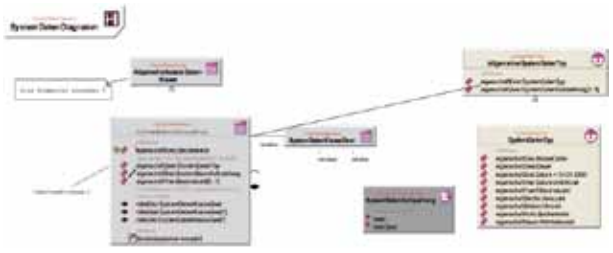

*Abb. 3: Beispiel für ein Systemdatenmodell*

menten mit definierten Eigenschaften zulassen (siehe Abbildung 2). Die DSL stellt somit eine Profilerweiterung der UML dar. Die Verwendung einer DSL garantiert maschinenlesbare Modelle, die eine Model2Model- und Model2Text-Transformation ermöglichen. Die native Verwendung allein der UML freilich reicht in der Regel für eine genaue Beschreibung der Systeme nicht aus. Besonders in großen Konzernen driften die Modelle durch unterschiedliche Vorgehensweisen auseinander. Durch die Verwendung einer zentralen DSL, die das Vorgehensmodell abbildet, ist dies ausgeschlossen, denn der Detaillierungsgrad der Modellierung wird genau festgelegt. Sogar weniger geübte Anwender, die ansonsten mit der UML und Vorgehensmodellen schnell überfordert sind, modellieren nun problemlos. Auch die Mitarbeiterressourcen können somit effizient eingesetzt werden, da sich die Einarbeitungszeit in neue Projekte durch die konzernweite, in der DSL abgebildete Vorgehensweise auf ein Minimum reduziert.

## Konzernweite DSL der DTAG

Bei der Deutschen Telekom AG (DTAG) wird schon seit Jahren eine zentral definierte DSL eingesetzt, die zusätzlich je nach Projektbedarf und Realisierungsvorgehen erweitert werden kann. So lassen sich zum Beispiel für die modellbasierte Entwicklung klassischer Systeme DSL-Erweiterungen hinzuladen, die einen Einsatz verschiedener Code-Generatoren (zum Beispiel AndroMDA) eröffnen. Diese „Erweiterungen" besitzen jedoch als Basis die konzernweit gültige DSL.
Die DTAG-Basis-DSL unterteilt den Engineering-Prozess in eine unternehmensweite Architektursicht und diverse Systemsichten. Zur unternehmensweiten Architektursicht zählt die Prozessarchitektur, in der die unternehmensweiten Geschäftsanwendungsfälle und Prozesse modelliert und definiert sind. Hinzu kommt die fachlogische Architektur, in der das GDM (Group Domain Model), das BOM (Business Object Model), das PBOM (Projektbezogenes BOM) und das FSP (Fachservice-Portfolio) abgebildet werden. Der Benutzer modelliert und definiert in der Systemarchitektur das Zusammenwirken der verschiedenen Systeme. In der technischen Architektur ist das Zusammenwirken der Systeme der technischen Ausprägung vorgegeben. Die Systemsichten dokumentieren die Außenkanten des Systems, die fachliche Systemanalyse, das Datenmodell, die Komponentensicht und die Feindesignsicht.

## Einsatz des Werkzeugs Innovator

Die DSL ist in Form von Profilen in dem Werkzeug „Innovator" der Firma MID GmbH abgebildet. Im Einsatz befindet sich derzeit der „Innovator Object", in dem das Profil DTAG-Basis sowie zahlreiche Erweiterungsprofile geladen sind. Hinzu kommt im BI-Umfeld zusätzlich noch der „Innovator Data". In ihn wird ein zentrales Erweiterungsprofil geladen, das die Anforderungen des BI-Sektors abbildet. Profile, in denen die Dokumentationen, zusätzliche Add-ons und diverse andere Anforderungen an den Entwicklungsprozess definiert werden, erweitern dieses. Die Modellreferenz ermöglicht eine barrierefreie Gesamtsicht. Das Werkzeug bildet im Innovator Data insbesondere die unterschiedlich ausgeprägten Sichten auf die Datenmodellierungen ab. Barrierefreiheit ergibt sich durch die aufeinander aufbauenden DSLs auch innerhalb des Vorgehensmodells. Das Tool mit der hinterlegten DSL sichert die Konsistenz des „Blue Print" und des implementierten Systems durch eine Nachverfolgbarkeit (traceability) über den gesamten Modellierungsprozess von der Anforderung bis hin zur Umsetzung.

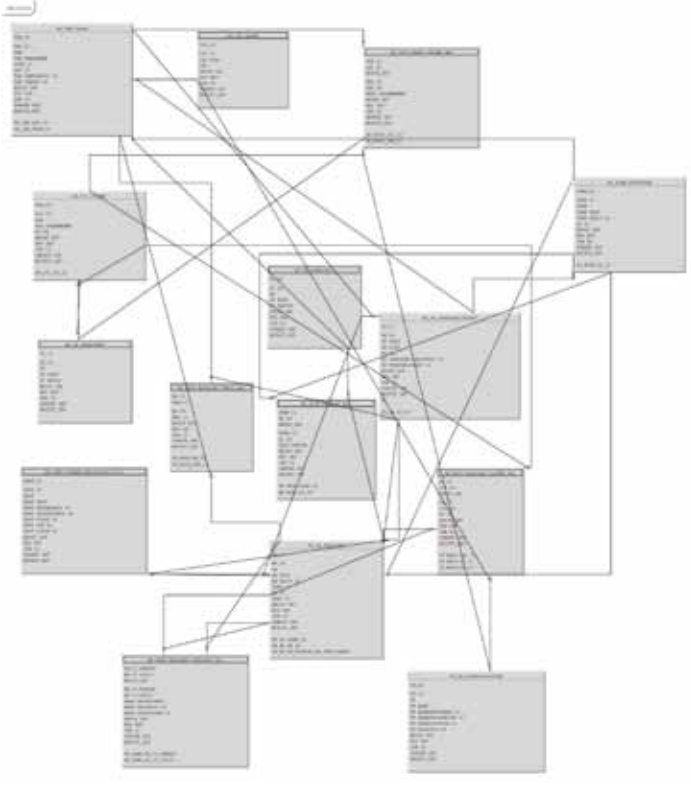

*Abb. 4: Beispiel für ein konzeptionelles Datendiagramm*

## Datenmodell

Im Sektor BI liegt ein besonderer Schwerpunkt auf der Modellierung der Datenmodelle. Durch die verwendete DSL können die Datenbankelemente automatisiert generiert und angelegt werden. Bei den Datenmodellen finden sich die Systemdaten-, die konzeptionelle und die physikalische Sicht. Während in der Systemdatensicht (siehe Abbildung 3) die eigentliche Informationsstruktur abgebildet wird, bildet die konzeptionelle Datenschicht (siehe Abbildung 4) die logische Sicht auf die Informationen ab. Dadurch sind andere Informationen leichter ersichtlich und zusätzliche Informationen lassen sich ergänzen. Diese beiden Sichten sind somit von der gewählten Zielplattform (zum Beispiel der relationalen Datenbank Oracle) unabhängig. Erst in der physikalischen Datensicht (siehe Abbildung 5) wird die Zielplattform berücksichtigt und um weitere wichtige plattformabhängige und technologische Details erweitert. Der Benutzer modelliert im Systemdatenstrukturdiagramm die im System verwendeten Datenstrukturen auf rein fachlicher Ebene. Die so entstandenen Datenmodelle sind noch nicht normalisiert und besitzen keinen technischen Bezug. Das konzeptionelle Datenmodell enthält rein fachliche Entitäten und deren Relationen. Das physikalische Datenmodell, das sich automatisiert durch das geladene Add-on aus der logischen Sicht erstellen lässt, bildet die Tabellen und die technischen Details der verwendeten Datenbank ab. Das physikalische Datenmodell wird durch eine Model2Model-Transformation aus dem konzeptionellen Modell abgeleitet und um die technischen Details (Schlüssel, Datentypen etc.) ergänzt.

Der verwendete Transformationsprozess ist im Detail sehr komplex und kann deshalb im Rahmen dieses Artikels nicht näher erörtert werden, da insbesondere viele konzeptionelle Einschränkungen und technische Randbedingungen Beachtung finden müssen. Diese Details können für mehrere Datenbank-Managementsysteme angelegt werden. Somit handelt es sich beim physikalischen Datenmodell um ein plattformabhängiges Modell (PSM). Die verwendete DSL, die durch das geladene Profil repräsentiert wird, verwendet hierzu als Engineering-Aktion Elemente des OAW (Open Architecture Ware).

Im physikalischen Modell müssen nicht alle Entitäten 1:1 auf Tabellen abgebildet werden. Im Sinne einer Performanzoptimierung lassen sich auch andere Mappings anwenden. Das physikalische Datenmodell beinhaltet sehr viele Detail- und Implementierungsinformationen und hat somit eine zentrale Position bei der Systemarchitektursicht. Der Benutzer kann natürlich auch einen Round-Trip etablieren, denn die physikalische Schicht lässt sich aus der Datenbank automatisiert erstellen. Die logische Schicht wiederum wird daraus generiert. Das System kann außer Tabellen, Views und Prozeduren auch User- und Datenbankrollen modellieren. Die DSL schreibt da-

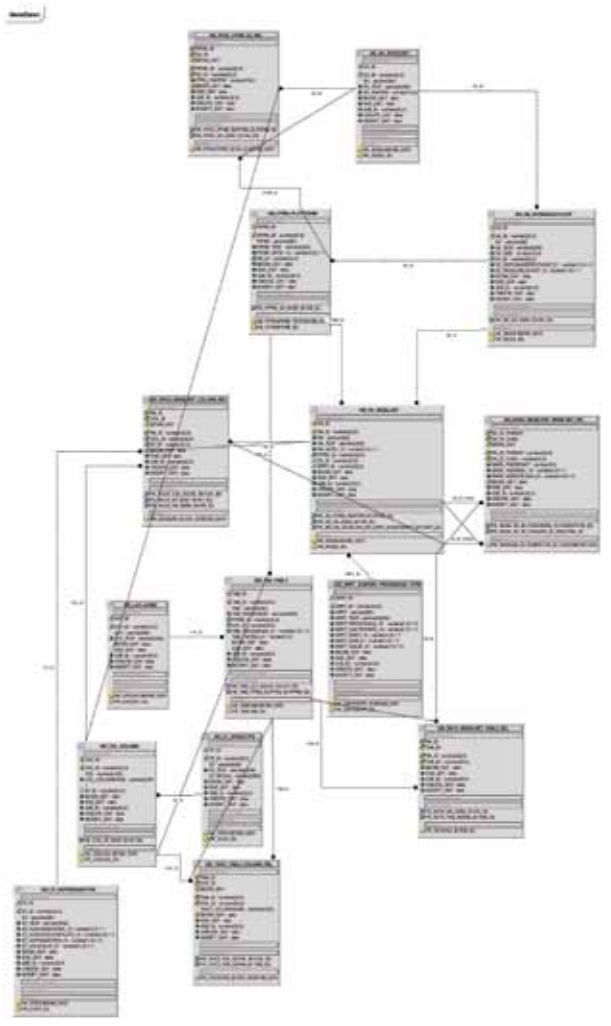

*Abb. 5: Physikalisches Datendiagrammbeispiel*

bei auch den Detaillierungsgrad der Modellierung vor. Durch die unterschiedlichen Sichten und Ausprägungen des Datenmodells können alle beteiligten Rollen des Entwicklungsprozesses optimal unterstützt werden.

## Erfahrungsbericht

Der Einsatz des Werkzeugs und der in den Profilen abgebildeten DSL bewirkt Synergieeffekte für alle Projektgrößen. Die definierte Vorgehensweise und Detaillierungstiefe erhöhen die Qualität der Projekte. Investitionssicherheit resultiert aus dem Bezug der Datenobjekte zum GDM und BOM; die vorgelagerten Prozessketten werden transparent. Das verbessert die Ausprägung der BI-Projekte deutlich. Neben den strategischen und qualitativen Verbesserungen lassen sich Offshore-Ansätze durch die vorgegebene Themenorientierung realisieren. Auch der Wiederverwendungsgrad einzelner Komponenten erhöht sich durch den Bezug zum BOM und zum GDM, da die globale Sicht des Konzerns Berücksichtigung findet. Dabei ist zu beachten, dass der Konzern die Rollen organisatorisch abbildet, wie sie in der Vorgehensmethode hinterlegt sind (zum Beispiel Datenmodellierer, Architekt, Businessingenieur).

## Ausblick zur DSL

Da insbesondere die Profilerweiterung bezüglich der BI-Thematik am Anfang steht, kann davon ausgegangen werden, dass der Automatisierungsgrad noch deutlich erhöht werden kann. Die DSL bildet künftig auch die Testfälle, die Dimensionen und die Logik (zum Beispiel die Prozesse der Ladeschicht) ab. Alle Anforderungen an die Thematik fließen barrierefrei in das Werkzeug ein.

## Ausblick Modellbasiertes BI

Die DSL umfasst außer dem AD- (Application Developer) auch den AM-Prozess (Application Management) und steht damit für das gesamte ALM (Application Lifecycle Management) zur Verfügung. Hierzu gibt es im Konzern bereits fertige Profile, die eine barrierefreie Lösung zwischen dem Werkzeug Innovator und dem Werkzeug „Polarion“ sowie anderen Werkzeugen, welche die Prozesskette unterstützen, ermöglichen. Auch neue Werkzeuggenerationen, die verstärkt das „Modellbasierte BI“ aufgreifen, steigern die Effektivität und die Qualität. Besonders der in Kürze zu erwartende „Innovator DataArchitect“ (siehe Abbildung 6) optimiert das industrielle modellbasierte Vorgehensmodell. Damit lässt sich die Fertigungstiefe erheblich reduzieren. Auch die deutlichen Verbesserungen der Handhabung des Werkzeugs und die bessere Integration in die gesamte Prozesskette führen zukünftig zu einer sehr hohen Akzeptanz; das verbessert wiederum die Gesamteffektivität. Erstaunlich ist, dass der Nutzen sowohl in kleinen als auch in großen Projekten nachzuweisen ist, wobei dieser jedoch primär durch die verwendete DSL und nicht durch das Werkzeug begründet ist.

*Abb. 6: DataArchitect*