1 MDA Tools am Horizont

Die Entwicklergemeinde kennt ein neues Zauberwort, das in traditioneller Form der drei Buchstaben daherkommt: MDA (Model Driven Architecture). Wird man nun weniger programmieren und vermehrt modellieren, oder induziert die Toolindustrie wieder neue Bedürfnisse ? Beides, den eine stärkere Formalisierung ist sowohl der Wunsch von uns geplagten Entwicklern wie auch die Bedingung der Industrie, Tools und Architekturen zu vereinheitlichen. All diese Vorhaben in UML 2.0 laufen unter dem Begriff MDA zusammen. Das Ziel ist es, ausführbare Modelle zu generieren.

1.1 Kernidee

Kernidee der MDA ist also die schrittweise Verfeinerung von der Analyse zum Design ausgehend von einer Modellierung der fachlichen Applikations-/Geschäftslogik, die völlig unabhängig von der technischen Implementierung und der umgebenden IT-Infrastruktur ist. Das Grundprinzip zu MDA lässt sich wie folgt darstellen:

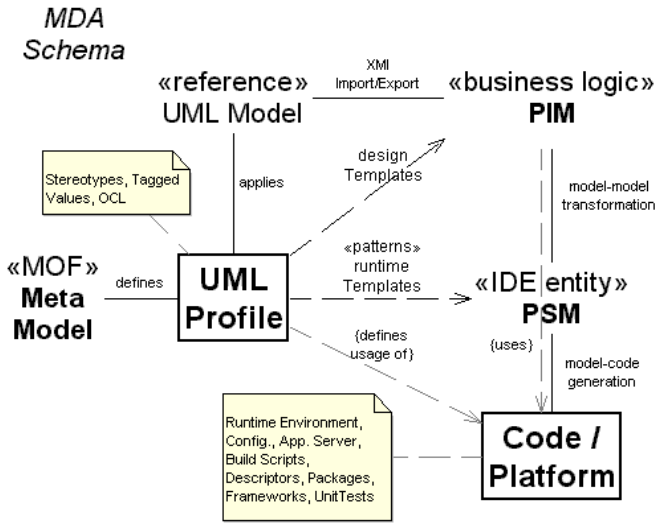

Abb. 1.1: Das MDA Prinzipschema

Das Schema soll den Weg vom abstrakten Metamodell über die unabhängige Fachlogik zum konkreten Code auf der Plattform verdeutlichen, ähnlich einer CORBA-Spezifikation, der IDL (Interface Definition Language) und dem daraus folgenden Code. Was bedeutet dies nun in der Praxis?

Man beginnt mit der Erstellung des Designs oder einer Referenz mit einem konventionellen UML-Tool. Das Tool muss aber fähig sein, mit UML-Profilen und constraints arbeiten zu können. Durch einen XMI-Export erhält man eine austauschbare Designdatei (PIM), die man mit dem gewünschten UML-Profil einem MDA-Generator füttert. Der Generator transformiert die Designdatei mit dem Profil nach spezifizierten Abbildungsregeln (Template) in einen konkreten Architekturrahmen als typisierte Packages. Nun hat man das PSM, das immer noch ein Modell ist.

Mit der Generierung wird dann erstmals Code im Sinne eines Implementationsrahmen erzeugt. Die geschützten Bereiche sind zunächst noch leer. Es erfolgt dann die Entwicklung der eigentlichen Fachlogik in den Klassen, die mit einer weiteren Generierung durch Interpreter oder Compiler zum ausführbaren Code bezüglich der gewählten Plattform führt. Die einzelnen Schritte:

- Modelliere das PIM
- Transformieren auf ein PSM
- Generieren von Code aus dem PSM
- Codieren der Fachlogik
- Integrieren in die Plattform

Wenn nun das Tool aus dem PIM ein plattformspezifisches Modell (PSM) generiert, sollte Einigkeit über die zu implementierende Plattform herrschen. Ist nun eine Plattform CLX, .NET oder J2EE, dann sollten die entsprechenden Klassen und Typen zum Zuge kommen, wie J2EE 1.2compliant oder CORBA 2.3compliant ORB. Was aber ist, wenn die Plattform eine Makro-Sprache mit Objektmodell oder sogar eine Spezifikation wie CORBA ohne Applikationsserver hat?

> *Der PSM UML-Generator muss also nicht nur die Zielsprache, sondern auch die Zielplattform aus den Profilen kennen.*

Im Hinblick auf die Ausarbeitung solcher Details stehen die MDA-Bewegung und die Toolhersteller noch ziemlich am Anfang. Zusätzlich soll in die MDA das Wissen über die zugehörige Businessdomäne einfließen.

Auch das Wissen über die zugehörige Fachlogik soll einfließen. Für Domänen wie z.B. Finanz- oder Versicherungswesen, Telekommunikation, Medizinaltechnik sollten die typischen abstrakten Prozesse schon vordefiniert sein und für die Erstellung der PIM als so genannte „domain-specific core models“ bereitgestellt werden. Ein erster Ansatz ist Statemate, ausführbare Diagramme für Embedded Systeme.[i] Für den Austausch der Modellinformationen über Toolgrenzen hinweg wird XML Metadata Interchange (XMI) eingesetzt. Der Name zeigt es bereits deutlich: XMI ist ein Mitglied der XML-Sprachfamilie. Die XMI-Norm definiert für das Metamodell eine XML Document Type Definition (DTD) oder ein Schema. XMI ist ebenfalls ein Standard der OMG.

Also bei der Transformation von einem abstrakten Modell hin zu einem konkreteren Modell soll man nicht nur Wissen über die technische Infrastruktur verwenden, sondern es wird auch Wissen berücksichtigt über die Businessdomäne, in der die Applikation eingesetzt werden soll.

Die MDA birgt zusätzliches Potential für die Wiederverwendbarkeit von Modellen, da diese ja plattformunabhängig sind. Durch den Einsatz von standardisierter Transformation ist auch die Qualitätssicherung erhöht. Nach OMG sind dies die Hauptvorteile von MDA:

- Reduzierte Kosten bei der Entwicklung
- Erhöhte Qualitätssicherung durch Selbstähnlichkeit
- Schnellere Integration neuer Technologien

Als Nachteil von MDA muss man erwähnen, dass strikte nach Forward Engineering gearbeitet wird. Nimmt man eine Änderung am Code vor oder ändert man das PSM entsprechend, aktualisiert das Tool die jeweils andere Seite nicht automatisch. Typische Refactoring Techniken, die auch in einem Review zum Tragen kommen, sind deshalb bei MDA ein Problem.

Am 6. Januar 2003 wurde die überarbeitete Fassung der OMG übergeben. Am Ende dieses Jahres, da erfahrungsgemäß die Finalization Task Force 9 Monate dauert, erwartet die Entwicklergemeinde eine Freigabe der Version 2. Diese teilt sich auf in drei separate aber zusammenhängende Gebiete:

1. **UML Infrastructure**, welche eine Vereinfachung und Modularisierung des Metamodells vorsieht und keinen Einfluß auf den Benutzer hat. Dieses Gebiet ist für die Toolhersteller und Methodiker interessant, die an Profilen und möglichen Stereotypen Freude und Nutzen haben. Zudem wird das Metamodell von sprachabhängigen Konstrukten (wie C++ oder ADA) gesäubert und sprachneutral, d.h. in OCL definiert.
2. **UML Superstructure**, mit den eigentlichen Erweiterungen bezüglich der Notation und Syntax und somit für den Benutzer sichtbar und verwertbar. Ziemlich aufwerten will die OMG die komponentenbasierte Entwicklung mit z.B. einer präziseren Definition einer Schnittstelle mit

erweiterten Signalen oder Nachrichten. Statische und dynamische Elemente werden auf Echtzeitfähigkeit getrimmt.

3. **UML OCL**, die eine erhöhte Integration in das Metamodell und die Syntax vorsieht, so daß die Spezifikation durch ein Modell zum Code auch eine formale Sprache beinhaltet. Die OCL (siehe Glossar) wird von einer Sprache der Bedingungen zu einer generellen Ausdruckssprache erweitert. Weiter folgt der verbesserte Modellaustausch durch die XMI, z.B. mit Layout Informationen zwischen den einzelnen Tools.

Es ist klar, das auf der einen Seite die Toolhersteller nun gefordert sind. Auf der anderen Seite wird die Vision des „Executable UML“ eine gravierende Änderung des Entwickleralltages zur Folge haben, sofern die Welt und schlußendlich auch unsere Kunden mit „standardisierter Individualsoftware“ und einheitlichen Geschäftsprozessen zufrieden sind. Etwas in dieser Art ist beim BizTalk Server von MS zu finden, der eine Generierung des Dokumenten-Workflows erlaubt.

Mehr noch, bezüglich der Wartung und Pflege einer Anwendung sind dann die selben Fehler zu erwarten, da aus dem weltweiten Modellkatalog in den Codegeneratoren auch ähnliche Fehler entstehen können.

Die wichtigsten Neuerungen (über 540 wurden im Sommer 02 der Revision Task Force überreicht) des Release 2 folgen nun in geraffter Form auf einen Blick:

- Für das Metamodell entsteht ein Sprachkernel
- Zusätzliche Notationen bauen auf dem Kernel auf
- Support für den Einsatz von Komponenten untereinander
- Ausbau der Geschäftsprozessmodellierung (BPM)
- Interne Struktur für Bezeichner, Schnittstellen, Klassen, Typen und Rollen
- OCL Integration als generelle Spezifikationssprache eines Modells
- State Event Generalisierungen und Echtzeiterweiterungen
- Erweiterungen für kompaktere Sequenzdiagramme
- Präziseres Abbilden der Notation zur Syntax

1.2 MDA Tools

Das MDA ein neuer Hype ist, heißt noch lange nicht, das die technische Idee von MDA nicht schon in einigen Tools vorhanden war. Was eben noch fehlte ist eine standardisierte Design-Sprache als Profile, wie OCL, Stereotypen etc., also plakativ gesagt, eine Sprache mit der ich UML-Modelle eindeutig spezifizieren kann. Zudem benötigt MDA eine einheitliche Transformationssprache, welche die Modelle in den Code/Plattform transformiert.

Bisher fielen MDA-Werkzeuge in eine von zwei Kategorien: auf der einen Seite die integrierten Komplettumgebungen mit vollständiger Modellierung, Modelltransformatoren und Generierung, auf der anderen Seite die reinen Generatoren, welche den dateibasierten XMI Input aus anderem UML-Werkzeugen verarbeiten. Man unterscheidet also MDA bzw. MDD (Driven Development) oder auch in Kombination MDA/D genannt.

Die meisten Entwicklungstools sind ja erstellt worden, um Code zu erzeugen und nicht, um Modelle zu implementieren. Patterns und Profile lassen sich hier als eine Art Vermittler zwischen Modell und Code einsetzen, da man den Fokus auf die Architektur legt.

1.2.1 Bold / ECO Framework

Im Folgenden lässt sich der Begriff Bold mit ECO gleichsetzen. Borland hat ja das Framework mit gesamtem Team innerhalb der Boldsoft übernommen, welche auch am ECO Projekt beteiligt sind. Octane Architect wird demzufolge das ECO-Framework enthalten (EnterpriseCoreObjects) wie Delphi Architect, das die Bold-Komponenten beinhaltet. Boldsoft mit seinem UML-Modellierungs- und

Architekturtool für Delphi ist ein weiteres Highlight des MVC Architektur Patterns. Zudem gibt es auch ein anerkanntes MVC Konzept für die Realisierung von Web-Applikationen in der Java-Welt. [ii]

Im Gegensatz zu Bold, benötigt ECO keine eigenen Controls mit der zugehörigen visuellen Anbindung. Achtung, verwechseln Sie den Begriff ECO nicht mit der e-commerce Initiative eCo Framework genannt.

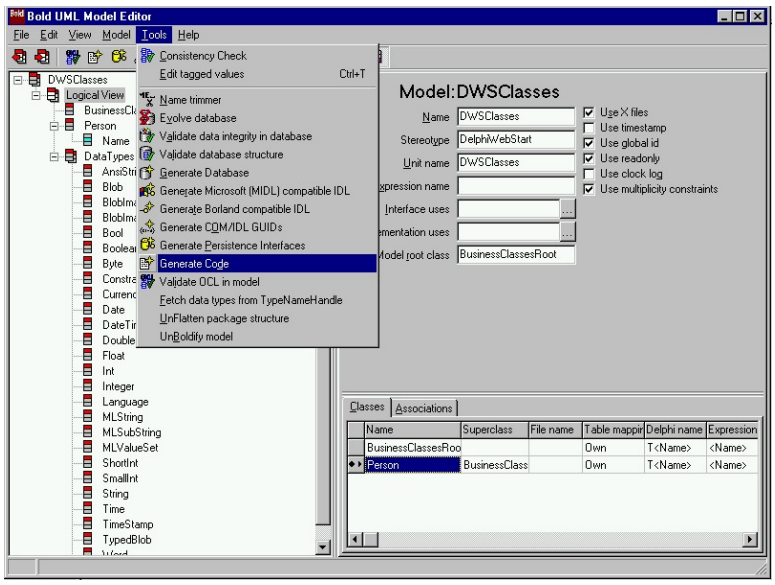

Abb. 1.2: Bold generiert Code und Datenbank

Bold ermöglicht die Implementierung und Steuerung der MVC Architektur, d.h. eine Mittelschicht von Rendering Klassen (Presentation Mapping) vermittelt zwischen den Geschäftsobjekten und der graphischen Repräsentation der Views. Zusätzlich bietet Bold ein durchdachtes Datenbankhandling an, das nebst dem «Klassen zu Tabellen Mapping» in eine relationale Datenbank auch Cached Updates sowie Transaction Handling erlaubt.

Bold benutzt ähnlich den UML-Profilen auch TaggedValues, die das Modell genauer spezifizieren, wie z.B. <PMapper> für die Art der Persistenz von Klassen. Auch mit der OCL möglich.

Das von mir erstellte Modell, enthält bspw. die nötigen Informationen zur automatischen Generierung der Datenbank und den zugehörigen Tabellen. Dieses Klassen zu Tabellen Mapping läßt sich nach der Generierung weiter steuern, da in der Regel die gewünschte Datenbank mit Indexen und Triggern erweitert wird.

Das UML Profil von Bold besteht aus vier Namen, die in den unterschiedlichen Elementen zum Tragen kommen. Bold setzt die Namen automatisch zum zugehörigen Kontext, sei es Modell, Profil, Code oder Plattform:

Modellname	Dieser Name wird im UML Modell eingesetzt.
Expression-Name	Dieser Name kommt in der OCL als Ausdruck oder Profil vor
Code-Name	Wenn Code generiert wird, als konkreter Delphi Code.
SQL-Name	Als spezifische Namen in der Datenbank.

Bold kann in diesem Sinne nur die Client-seitigen Regeln beeinflussen. Der Schemagenerator ist betreffend Änderungen des Modells vorbereitet, die Wahrscheinlichkeit ist groß, daß es Änderungen gibt.

Das direkte Einbinden der Business-Objekte ist wohl der größte Vorteil von Bold. Der Klassencode für die Geschäftsobjekte läßt sich größtenteils automatisieren und auch bei Änderungen aktualisieren, ohne den Einsatz von bekannten Round-Trip Flags im Code, welche auf Veränderungen hin prüfen.

Konkret erzeugt Bold innerhalb der Codegenerierung mindestens drei Units:

- <UnitName>.pas und <UnitName>_Interface.inc werden von Bold stets aktualisiert
- <UnitName>.inc beherbergt den individuellen Code

Der so genannte Objektraum (objectspace) ist eine Instanz des Modells, wie das Objekt eine Instanz der Klasse ist. Bei der Implementierung der GUI sind wir auf die Bold eigenen Komponenten wie Grid oder Navigator angewiesen, da nur sie Verknüpfungen zur Business-Schicht ermöglichen und zudem bei Änderungen auch Modellgetrieben sind. Das Positionieren und Verknüpfen der Komponenten erfolgt größtenteils ohne manuelles Codieren.

Bold/ECO ist eine hervorragende Umsetzung der MDA Grundlagen, die auch während der EKON 7 wieder einige überzeugt hat. Wobei der Support von Bold durch Borland ab und zu seine Grenzen zeigt, man hat es einfach vermisst die frühere Site www.boldsoft.com umzulenken, so dass ein HTTP 500 resultiert ;(.

1.2.2 MDA in StP

Software through Pictures nennt es „Architecture Component Development“ (ACD) Technologie. Auch unter dem Aspekt, daß die OMG dieses Thema unter dem eigenen Namen MDA weiterentwickelt, sind die Ideen hinter ACD und MDA identisch, mit dem Unterschied, daß sich Aonix diesem Thema schon seit Jahren widmet.

Einzigartig ist eben Aonix's ACD Technologie. ACD schlägt eine Brücke zwischen UML und realem Code, so daß mit einer Transformationsregel welche wiederum mit Hilfe von Templates automatisch fast 70% Code aus den Modellen generieren soll. Dahinter steckt eine spezielle Templatesprache, die man dem Transformator füttert.

StP/ACD ermöglicht die Trennung der fachlichen Anforderungen von den technischen Details der Implementierung. Die fachlichen Aspekte sind mit StP/UML Modellen und die technischen Aspekte als Template für den Transformator modellierbar. Dadurch erhält man einen sehr hohen Abstraktionsgrad in den Modellen und ein hohes Maß an Wiederverwendung durch die technischen Musterlösungen in den Templates. Mittels der Templatesprache, welche eine Art Abbildungsregeln definiert, kann der generierte Teil des Codes mehr als die Hälfte betragen.

Die Integration mit weiteren Produkten wie Config-Management-System, IDEs für verschiedene Programmiersprachen und Testwerkzeuge sorgen für die Abdeckung des gesamten Softwarezyklus.

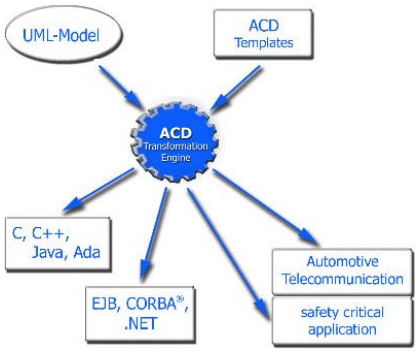

Abb. 1.3: ACD als eine Umsetzung der MDA

Ameos ist bei StP das Modellierungswerkzeug der nächsten Generation. Mit den UML-Profilen bietet Aonix eine einfache Möglichkeit die Standard UML Notation zu erweitern und an projektspezifische

Anforderungen anzupassen. Der Profile Editor von Ameos ermöglicht die Definition von Stereotypen und Tagged Values und die Verknüpfung mit Elementen des Metamodells. Damit wird der Entwurf, die Dokumentation und die einfache Wiederverwendung von UML Profilen sichergestellt. Einzigartig ist die Zuordnung von Farbe in der Profile Definition. Farbe wird dadurch auf semantischer Ebene verwendet und führt zu einer erheblich gesteigerten Lesbarkeit der erstellten Modelle.

Ein weiteres Kernstück von Ameos stellt der Modelltransformator auf Basis der MDA dar. Die PIM's werden dann über Transformatoren in die Zielumgebung überführt. Mit speziell zu diesem Zweck erstellten UML Profilen und den darauf abgestimmten Transformationsregeln, können die genannten Vorteile direkt in die Praxis umgesetzt werden. Neben den Standardlösungen für C, C++, Ada, Java und EJB, werden auch branchenspezifische Lösungen für den Automobilsektor und für sicherheitskritische Anwendungen angeboten. Ameos ist auf Win2000/XP, Solaris und Linux verfügbar und ist somit für den Einsatz in heterogenen Netzwerken prädestiniert.

Die Linux Version von StP 8.3 steht ab sofort zur Verfügung. Mit der neuen Linux Version vervollständigt Aonix das Angebot im Unix Bereich. Die Positionierung von StP wird damit konsequent als Multi-Plattform Modellierungswerkzeug ausgebaut. Zu den unterstützten Plattformen gehören nun Linux (SuSe, Red Hat, Mandrake), Sun Solaris, HP UX und Win. StP lässt sich auch in einem heterogenen Umfeld mit mehreren Plattformen gemeinsam nutzen.

Aonix: http://www.aonix.de

1.2.3 ArcStyler 4.0

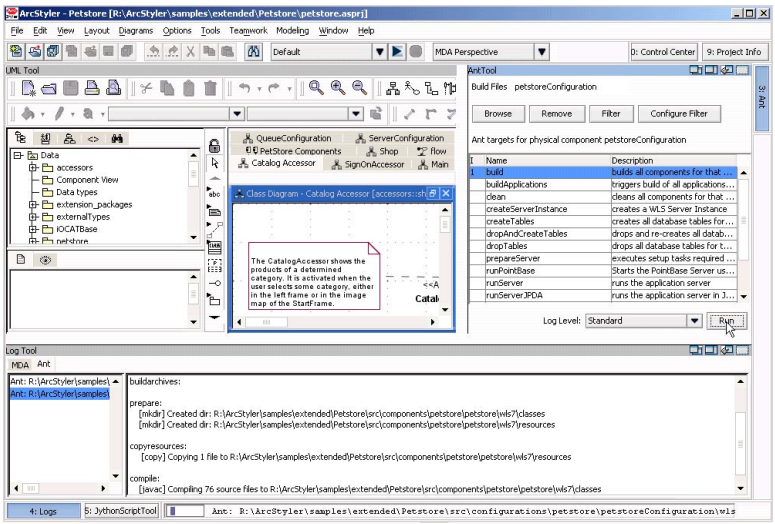

Abb. 1.4: ArcStyler als MDA-Generator

Der neue ArcStyler 4.0 biete jeweils das beste aus beiden Welten. Das Produkt ist in unterschiedlichen Versionen verfügbar, so dass der Anwender die für seine spezifische Aufgabenstellung optimale Lösung wählen kann. Die Standardversion verfügt über eine eigene, leistungsfähige UML-Engine (MagicDraw OEM von Nomagic). Zu den Features gehören u.a. Unterstützung von Profilen, MOF und Meta-Modellierung.

Aus eigener Erfahrung ist der ArcStyler in zwei Prozessen erhältlich: Als Komplettool oder als MDA/D Generator. Viele Unternehmen verfügen bereits über Modellierungswerkzeuge. Um diese Investition zu schützen, wurde von Interactive Objects die Tool Adapter Standard (TAS) API für ArcStyler entwickelt. TAS ermöglicht es , alle Arten von Modellierungswerkzeugen nahtlos in den ArcStyler einzubinden und diese damit zu einer leistungsstarken MDA-Lösung auszubauen.

Wie kommt man nun zu den Profilen ?

Die anpass- und erweiterbaren MDA-Cartridges (eine Art Profile) lösen dieses Problem. Als Template enthalten sie sämtliche technologiespezifischen Mechanismen für die automatischen Transformationen von Modellen. Dazu gehören Mapping und Verifier Engines, Pattern Matchers und erweiterbare Generatoren.

Es können Cartridges für prinzipiell jede Architektur erstellt werden. Die in diesen Cartridges enthaltene Automatisierung wird ebenfalls in UML modelliert. ArcStyler wird mit Standard- und Referenz-Cartridges für Java, J2EE, und .NET ausgeliefert, die bei Bedarf individuell anpassbar sind. Eine Vielzahl weiterer Cartridges (Open Source) steht den Anwendern über die weltweite MDA-Community zur Verfügung: www.mda-at-work.com**.**

Alle ArcStyler-Module verwenden ein gemeinsames und offenes UML/MOF/JMI-Repository. Damit entfällt nicht nur ständiges Importieren, Exportieren und Synchronisieren.

Der ArcStyler deckt weit mehr Aspekte der Softwareentwicklung ab als reine UML-Modellierung und einfache Code-Generierung über simple Patterns. Mit seiner MDA-Engine bietet er nicht nur intelligente Modellierungsunterstützung mit Validierung auf Modellebene (Modell-Debugging unter Berücksichtigung der Plattform), sondern gleichzeitige Unterstützung mehrerer Zielplattformen und Architekturen, flexible und vollständig anpassbare Transformations- und Generierungsfunktionen.

ArcStyler: http://www.iO-Software.com/mda

1.2.4 XCoder von Liantis

Die Liantis GmbH hat ihr MDA-Tool XCoder als Open Source freigegeben. Mit diesem Schritt möchte sie den in zahlreichen Projekten praxisbewährten XCoder einer noch größeren Entwicklergemeinde zur Verfügung stellen. Durch die Freigabe als Open Source haben Unternehmen die Sicherheit, alle notwendigen Ressourcen zur eigenen Anpassung bzw. Weiterentwicklung zur Verfügung zu haben. Anpassbare Standardlösungen für Java, C++, C#, J2EE und .NET liegen bereits in der Standarddistribution vor.

XCoder ist komplett mit MDA erstellt, in Java geschrieben und wird auch mit MDA erweitert. Alle dem XCoder zugrunde liegenden UML Modelle sind ebenfalls Bestandteil der Standarddistribution.

Liantis verspricht sich ein schnelleres und häufigeres Feedback, wenn weitere Entwickler und Anwender an dem Projekt beteiligt sind. Die schnellere Beseitigung von Fehlern wird ebenfalls allen Anwendern zugute kommen.

Durch offene Schnittstellen wie XMI sind praktisch alle marktgängigen UML-Tools unterstützt.

http://www.liantis.com // https://sourceforge.net/projects/xcoder/

1.2.5 Tau von Telelogic

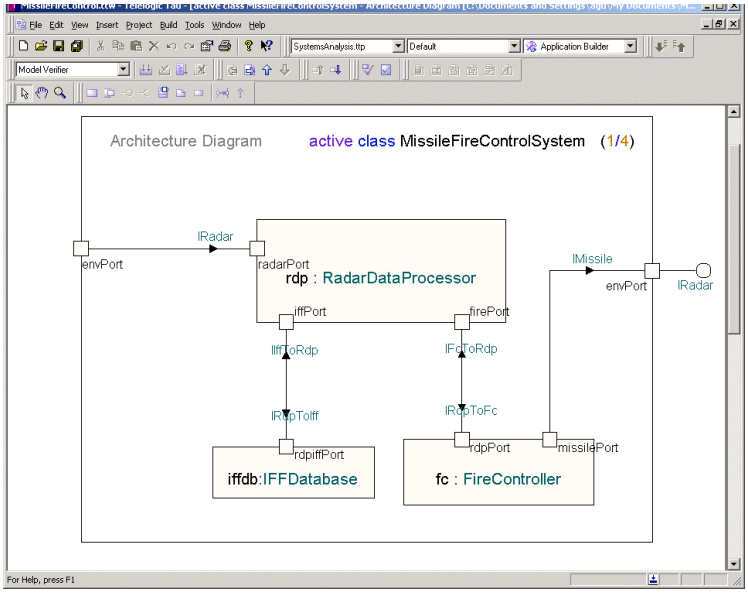

Abb. 1.5: Tau mit Echtzeitfähigkeit

Im Oktober 2002 hat Telelogic bereits Tau eingeführt, das UML 2.0 unterstützt. Tau ist in der Industrie auch als Tool für die Echtzeitmodellierung geschätzt.

Änderungen und immer wieder neue Anforderungen während des Entwicklungsprozesses machen Software immer komplizierter. Gut, wenn man mit dem eingesetzten Tool die Architektur des Systems dann überarbeiten kann, ohne daß sich das nach außen sichtbare Verhalten der Anwendung ändert. Tau hat einen Syntax-Check um die Formulierung des Modells zu prüfen inklusive eingebauter Simulation von Sequenzdiagrammen!

TAU gibt es in zwei Versionen mit UML 2.0:

TAU/Architect für die System- und Software Architektur

TAU/Developer für Echtzeit Systeme und Codegenerierung

Die beiden Tools lassen sich auch nahtlos in das firmeneigenene Tool Doors integrieren. Doors unterstützt das Anforderungsmanagement. In dieser Situation sind funktionierende Schnittstellen zwischen den einzelnen Tools unabdingbar. Diagramme für Use-Cases lassen sich beispielsweise elegant sowohl in Doors als auch in Tau integrieren.

Für den Architekturentwurf bietet Tau mit Architekturdiagrammen sehr weitreichende Unterstützung an: Die einfache Darstellung hilft, IT-Systeme zu gliedern, Komplexität zu reduzieren und so robuste Systemarchitekturen zu bauen. In den Diagrammen definieren Sie grafisch die Abhängigkeiten und Hierarchiebeziehungen zwischen Paketen oder Komponenten. Auf diese Weise kann man unter anderem vorgeben, welche Schnittstellen eines Paketes sich benutzen lassen und welche zu implementieren sind.

http://www.telelogic.com/resources/index.cfm

1.2.6 ObjectiF

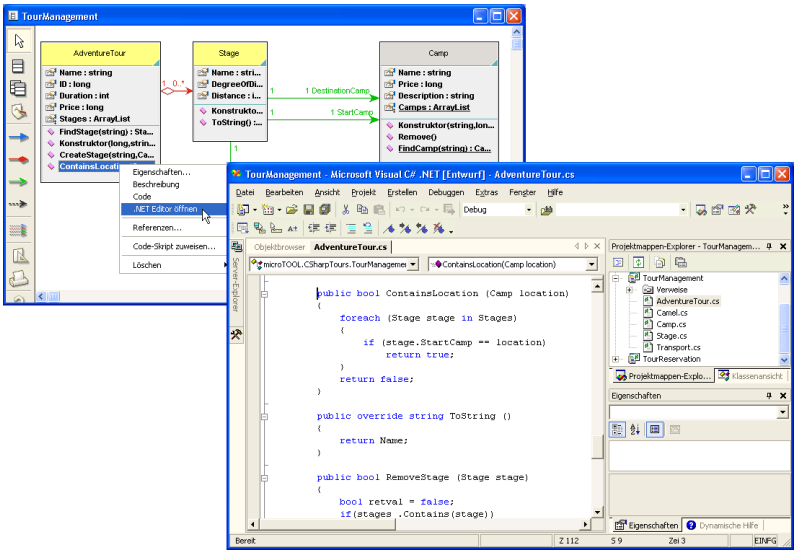

Abb. 1.6: Die microTool Suite für .NET

Für die unterschiedlichen Sprach- und Technologie-Präferenzen bietet microTOOL mit objectiF jetzt übrigens in zwei Tool-Varianten an: Die gerade freigegebene objectiF Visual Studio .NET Edition ist nahtlos mit Visual Studio .NET integriert und für C# und VB .NET optimiert. Sie ist damit speziell auf die .NET-Entwicklung zugeschnitten. Die "Vollversion" von objectiF – sie heißt Enterprise Edition -, wird in Kürze ebenfalls freigegeben (objectiF 5.0). Sie bietet alles, was objectiF Visual Studio .NET Edition mitbringt und unterstützt neben C# und VB .NET natürlich weiterhin auch Java, C++ und Delphi.

Zur Funktionalität der microTOOL Suite .NET gehört außerdem das Konfigurationsmanagement, also die Verwaltung aller im Projektverlauf entstehenden Ergebnisse. Für die agilen Techniken des Unit-Tests und der kontinuierlichen Integration wurden die Open Source Produkte NUnit und NAnt in die microTOOL Suite .NET eingebunden. Man findet - im Unterschied zu anderen Suites - in der microTOOL Suite .NET aber nicht nur optimal aufeinander abgestimmte Werkzeuge, sondern auch konkrete Unterstützung beim Vorgehen in Form von actiF, einem Prozess für die agile Entwicklung. actiF schafft die Grundlage für die maschinelle Planung und Steuerung von Projekten mit in-Step.

Apropos Java: Als nächstes wird sich in dieser Richtung etwas bei objectiF tun. Eine Integration mit Eclipse ist bereits in Arbeit. Sie wird einen ähnlich hohen Integrationsgrad wie objectiF Visual Studio .NET Edition haben.

http://www.microtool.de/

1.2.7 Innovator 8.1

Mit einer ausgefeilten Integration der IBM-WebSphere-Entwicklungsumgebung bietet Innovator 8.1 eine leistungsfähige Plattform für die Modellierung und Realisierung komplexer Anwendungen im Umfeld von EJB, Webservices und MDA. Auch von Eclipse aus können Sie nun nahtlos auf die Innovator-Modelle zugreifen.

Zusätzlich erweitert die Anbindung von PVCS Dimensions die Möglichkeiten im Konfigurations- und Versionsmanagement.

INNOVATOR 8.1 eignet sich hervorragend für die Definition und Erstellung von Anwendungen im Umfeld von WebServices und MDA. Die strikte Ausrichtung und Anpassung an Standards sichert auch in Projekten nach herkömmlichen Methoden und Verfahren getätigte Investitionen.

Abschließend kann man sagen, dass man Innovator eigentlich nicht gerecht wird, wenn man den Einsatz allein auf UML bezieht. Die verschiedenen Tools kommen am besten in einem Umfeld zum Einsatz, wo es eine Durchmischung aus strukturierter und OO-Entwicklung gibt. Wo immer es möglich ist unterstützt Innovator bestehende Industriestandards, was das Tool gerade auch für große und sage ich mal schwerfällige Projekte interessant macht.

Mit fünf Tools unterstützt Innovator sowohl objektorientierte sowie klassische, strukturierte Technologien (SA/SD und ERM/SERM) in Erstellung und Wartung komplizierter Softwaresysteme bis hin zur Geschäftsprozessmodellierung.

http://www.mid.de/de/news/profile/

1.3 Glossar

UML Profile

Ein UML-Modell wird mithilfe von spezifischen Erweiterungen (Stereotypen, Tagged Values, OCL) innerhalb einer formalen Designsprache (UML-Modellierungssprache) genauer beschrieben.

Stereotyp / Tagged Values

Stereotypen geben kommentierend die möglichen Verwendungszusammenhänge einer Klasse, einer Assoziation oder eines Paketes an. Stereotypen klassifizieren die möglichen Verwendungen eines Modellelementes. Ein Element, beispielsweise eine Klasse, kann beliebig viele Stereotypen aufweisen. Bsp.: «self», «import», «global», « # ».

Werte (Tagged Values)die in einem Modell konkret als Implementierungsanweisung zugewisen werden, wie: PersistenceType = Transient, Derived = True, InitialValue = Ord(`A`)

OCL

Die Object Constraint Language ist eine semiformulare Sprache zur Definition von Bedingungen/Einschränkungen. Sie definiert eine standardisierte Sprache zur Beschreibung von Zusicherungen innerhalb von Modellen.

Bsp.: {FIndex < FList.Count}, {salesman.knowledgeLevel >= 5}

MDA Generator

Der Parser des Generator Backend interpretiert ein UML-Profil oder ein Template, erzeugt den plattformspezifischen Code via FileStreaming, scannt und identifiziert die geschützen Bereiche bevor er die schon vorhandene Fachlogik in den neuen oder modifizierten Implementationsrahmen schreibt.

Max Kleiner, November 2003

[i] Statemate, www.ilogix.com/products/magnum/index.cfm

[ii] Struts: http://jakarta.apache.org/struts/