

Stand der Technik

Die Entwicklung verteilter Anwendungen im J2EE-Bereich erfordert das Schreiben einer vom J2EE-Framework vorgegebenen, verhältnismäßig großen Menge technischen Codes. Zum Einen erschwert dies die Fokussierung auf die zu implementierende Geschäftslogik, zum Anderen werden so Entwicklerkapazitäten für zwar notwendige, aber vergleichsweise anspruchslose Tätigkeiten gebunden. Daraus ergeben sich Nachteile in Zusammenhang mit time-to-market-Ansprüchen.

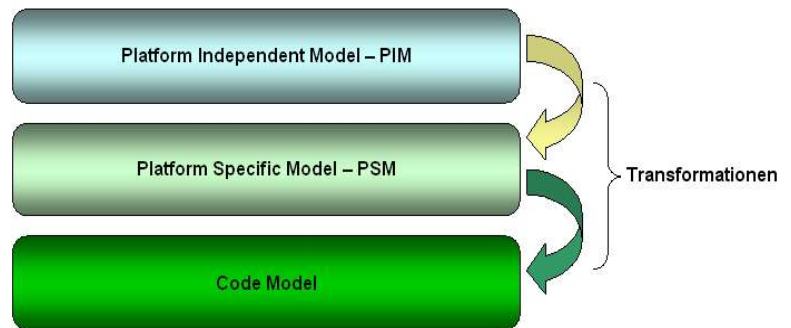
In nahezu allen Projekten ändern sich die Anforderungen im Laufe der Zeit. Parallel dazu veralten die in früheren Projektphasen verfassten Artefakte zusehends, wenn kein zusätzlicher Aufwand für ihre Aktualisierung eingeplant und auch betrieben wird.

Diesen und weiteren Nachteilen verspricht der Ansatz Model Driven Architecture/Model Driven Development entgegenzutreten.

Model Driven Architecture bezeichnet ein architektonisches Framework für Software-Entwicklung, welches von der Object Management Group (OMG) definiert wurde. Dieses Framework setzt (via Spezifikation) definierte Rahmenbedingungen für eine Software-Entwicklungsmethode, welche die Entwicklung einer gewünschten Geschäftslogik durch sukzessive Verfeinerung entsprechender abstrakter Modelle mittels Transformationen beschreibt. Der MDA-Software-Entwicklungsprozess ist ergo modellbasiert und generativ.

Ausgangspunkt ist das **Platform Independent Model (PIM)**, eine Beschreibung des softwaretechnisch zu realisierenden Anteils eines Geschäftsprozesses, d.h. von einem Modell, welches keine bestimmte hardware- oder softwaretechnische Realisierung voraussetzt.

Die plattformspezifischen Anteile zur Steuerung der nachfolgenden Programmcodegenerierung werden aus dem PIM auf Basis geeigneter Transformationsvorschriften und durch den Einsatz von Patterns ("Strickmuster" zur Steuerung der Programmgenerierung) erzeugt.



Die Trennung der Geschäftslogik von der späteren Zielsprache bzw. -plattform ermöglicht die Modellierung/ Spezifizierung und Verifizierung des PIM durch Stakeholder, die den Geschäftsprozess kennen oder sogar definiert haben. So wird die Erstellung eines Systems gefördert, das von Anfang an mit den Kundenanforderungen übereinstimmt.

Das PIM wird klassischerweise mit den Mitteln der **Unified Modeling Language (UML)** beschrieben.

Das Ergebnis dieses softwaretechnischen Prozesses wird als **Platform Specific Model (PSM)** bezeichnet.

Das PSM bildet eine Datenbasis für den letzten Schritt, die Generierung eines Grundgerüsts des finalen Modells der Geschäftslogik. Dieses Modell ist das plattformspezifische **Code Model**.

Der Programmcode zur Realisierung der Geschäftsabläufe wird beim aktuellen Stand der Technik in aller Regel von Hand an die passenden Stellen im generierten Code eingetragen.

Unsere Erfahrung

Auf der Suche nach einem effektiven und effizienten MDA-Werkzeug im Kontext J2EE ist K&A u.a. auf **OptimalJ** von



gestoßen und hat es im Rahmen von Evaluierungsprojekten untersucht. Verwendet wurde die Professional Edition 3.3 von OptimalJ, eine vollständige Entwicklungsumgebung für n-Tier-Applikationen, basierend auf dem **J2EE-Framework**.

Zur Evaluierung wurde der sog. „K&A-Bier-Shop“ entwickelt, eine Demo-Web-Anwendung. Mit einem Web-Client können „rein virtuell“ verschiedene Kölsch- & Alt-Biersorten bestellt werden.

OptimalJ generierte den bei weitem überwiegenden Anteil des Programm-Codes der 3 Tier (Web, Application, Database).

Lediglich in den Web-Komponenten musste detaillierter Code zur Implementierung der Geschäftslogik eingefügt werden. Er wurde in freien Blöcken des generierten Codes, welcher ansonsten grundsätzlich schreibgeschützt ist, platziert.

Das Web User Interface wurde mit Hilfe des in OptimalJ integrierten Web UI-Designers erstellt. Mit ihm lässt sich für eine Webseite ein Standard-Layout erstellen, welches einfach und schnell modifiziert werden kann.

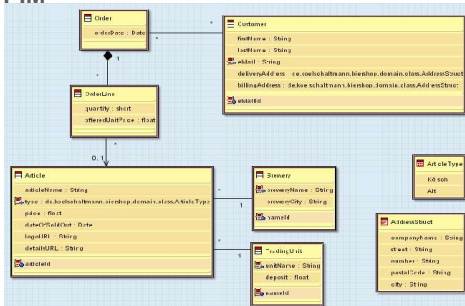
Die von OptimalJ generierten Maintenance Web Components bieten ein umfassendes Administrations-Interface. Mit ihm können z.B. Testdaten in die Datenbank gefüllt werden, um Startbedingungen für Tests der Applikation herzustellen.

Dies spart zusammen mit den automatisch generierten Skripts zum Erzeugen und Löschen des Datenbankschemas viel Aufwand.

Für Entwickler mit guten Kenntnissen in der J2EE-Plattform, J2EE Design Patterns, JSP und Struts ist der Einsatz von OptimalJ sehr zu empfehlen:

- Es entledigt den erfahrenen Entwickler von all dem Ballast, den die Entwicklung im J2EE-Umfeld mit sich bringt.
- Das Domain Model bleibt als High-Level-Dokumentation stets auf dem aktuellen Stand, im Gegensatz zu den früher separat erstellten UML-Diagrammen.
- Da die Modellierung der Geschäftslogik in den Vordergrund gerückt wird, entsteht eine Anwendung, die den Kundenanforderungen besser gerecht wird.
- Die Generierung eines Großteils des plattformspezifischen Codes führt zu einer deutlichen Steigerung der Produktivität, die wir im Bereich von 30-50% einschätzen.
- Gleichzeitig wird durch den hohen Generierungsanteil die Wartbarkeit des Codes erhöht.

PIM



Modellierung des Klassendiagramms im Domain Model

Code Model

```

/**
 * No data for information available.
 * @param genInfoNameCollection - return parameter of type java.util.Collection
 * @param altOrderConditionException thrown by the method if the pre-condition fails.
 * @param altOrderConditionException thrown by the method if the post-condition fails.
 */
public java.util.Collection getItems() {
    return com.komparee.alt1234567.application.altOrderConditionException.com.komparee.altOrderConditionException;

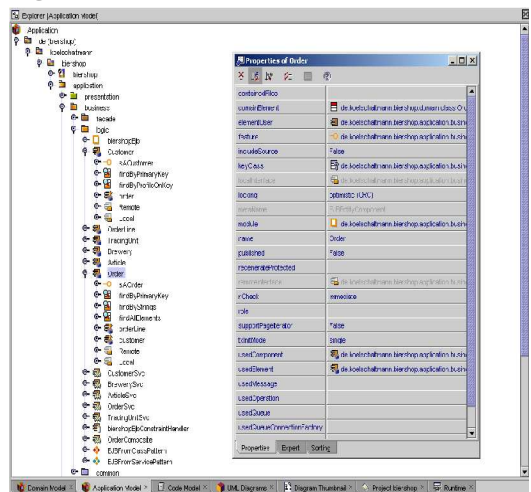
    Data.MallCollection returnName = null;

    // your code here

    // Example an bean mapped persisted data (e.g., aggregated data) need to be supplied to the application
    // In order to do so, call the following method (it is annotated with @default for performance purposes)
    // setDefaultValues();
    
```

Vervollständigung der Business-Methoden im generierten Code

PSM



Anpassungen im Application Model

Java, J2EE and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States or other countries. All other product names mentioned herein are trademarks of their respective owners.

Kölsch & Altmann
Software & Management
Consulting GmbH

Perlacher Straße 21
81539 München
Tel. 0 89/65 10 71-0
Fax 0 89/65 10 71-92
e-mail: info@ka-muc.de
Web: www.ka-muc.de

Stand: 07/2005