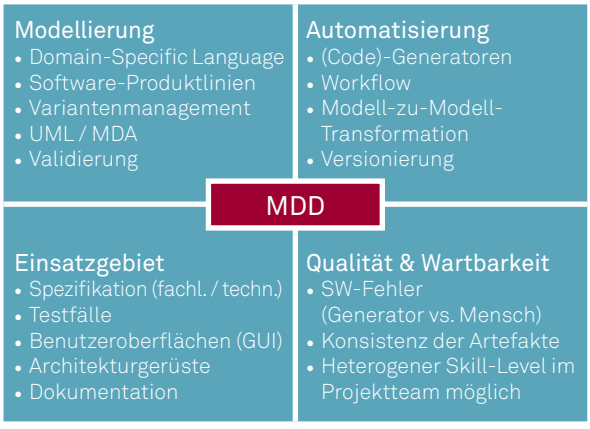
Model-Driven Development (MDD)

Domänenspezifische Sprachen und Generatoren

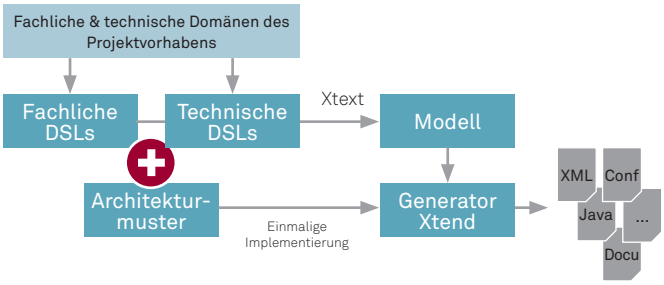
Modellgetriebene (Model-Driven) Entwicklung bedeutet in erster Linie, dass Modelle im Zentrum der Entwicklung stehen und aus ihnen Teile der zu erstellenden Software, automatisiert durch Codegeneratoren, erzeugt werden. Domänenspezifische Sprachen ermöglichen eine prägnante Modellierung der Fachlichkeit, helfen, deren Komplexität zu beherrschen, und tragen durch gute Integration mit Generatorwerkzeugen dazu bei, Software produktiver und in besserer Qualität zu erstellen.

Definition

Der Einsatz von modellgetriebenen Entwicklungsmethoden zielt auf mehr Produktivität durch Automatisierung, Reduktion von Komplexität durch Abstraktion sowie Verbesserung der Softwarequalität.



Modellierung abstrahiert von Details, die für das Verständnis der fachlichen und technischen Inhaltsbereiche (Domänen) eines Projektvorhabens unwichtig sind, beziehungsweise die von Generatoren im Nachgang automatisch hinzugefügt werden können. Die Automatisierung mit Modell-verarbeitenden Generatoren und Workflows wiederum wirkt sich positiv auf die Qualität aus, beispielsweise werden Architekturmuster mehrfach stringent durch Generatoren ausgerollt oder Fehler zentral im Generator behoben. Gleichzeitig sinkt, im Vergleich zu rein manueller Umsetzung, der Erstellungsaufwand der Software mit steigender Anzahl an Komponenten.

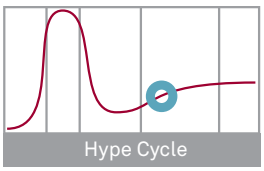


Die Kommunikation von Fachbereich und IT profitiert, wenn das spezifische Vokabular des Kunden oder seiner Domäne in den Modellsprachen Verwendung findet, weshalb sich Domain-Specific Languages (DSLs) für die Modellierung in vielfältigen Einsatzgebieten durchgesetzt haben. Diese individuellen, frei definierbaren Sprachen sind mit modernen DSL-Entwicklungswerkzeugen, wie zum Beispiel Xtext, effizient entwickelbar. Aufgrund ihrer einfachen Struktur (Meta-Modell) sind sie darüber hinaus, im Gegensatz zur komplexeren UML (Unified Modeling Language), beste Grundlage für Codegeneratoren.

Es gibt grafische und textuelle DSLs. Letztere haben den Vorteil, dass sie mit den gleichen Standard-Werkzeugen wie der Quellcode entwickelt und versioniert werden können – hier also kein Zusatzaufwand anfällt, wie das regelmäßig bei grafischer Modellierung der Fall ist. Grafische Visualisierungen von textuellen Modellen lassen sich zudem jederzeit bedarfsgerecht generieren.

Reifegrad

Model-Driven Development wird bereits seit vielen Jahren praktiziert. Nach einer gewissen



Ernüchterung bezüglich der wirtschaftlichen Anwendbarkeit der Model-Driven Architecture (MDA), insbesondere in Kombination mit der UML, haben Domain-Specific Languages das Feld weitläufig besetzt. Ein wesentlicher Erfolgsfaktor für die praktische Umsetzbarkeit dieser MDD-Variante im Projekt ist die Verfügbarkeit von mittlerweile sehr guten, integrierten MDD-Werkzeugen, die auch eine iterative Modellentwicklung in agilen Projekten auf leichtgewichtige Weise ermöglichen.

Marktübersicht

Es entstehen immer mehr textuelle DSL- und Generatorlösungen auf Basis des frei erhältlichen Xtext-Frameworks. Gründe hierfür sind dessen effiziente Werkzeugkette und eine große Community, so dass dieser Trend anhalten wird.

Ebenfalls textuelle DSLs sowie Generatorsupport bietet die Workbench MPS von JetBrains, die allerdings einen sehr hohen Einarbeitungsaufwand erfordert. Marktführer bei grafischer DSL-Erstellung ist die Firma MetaCase mit dem kommerziellen Produkt MetaEdit+.

Alternativen

Durch die Kombination DSL/Generator rücken Modellierung und Programmierung sehr nahe zusammen. Moderne Infrastrukturplattformen, wie JEE6, reduzieren den technisch notwendigen Code selbst auf ein Minimum und verringern so Generatoraufwände, die stattdessen für die MDD-Auswertung von Domänenübergreifenden Zusammenhängen, wie die Erzeugung von Testfällen und -daten, genutzt werden können.

Referenzszenario

MDD-Anwendungsszenarien ergeben sich immer dann, wenn aus einem Modellelement ein, oder besser mehrere Zielartefakte erzeugt werden können, die detailreicher sind als das Modellelement, die im Gesamtsystem öfters vorkommen oder die bei manueller Umsetzung regelmäßig fehlerträchtig sind.

Maskenorientierte, betriebliche Informationssysteme haben potenziell hohe MDD-Anteile auf der Frontend-, Backend- und Testseite.

Bei Software-Systemfamilien werden häufig im Zuge des Variantenmanagements die Unterschiede der Softwareprodukte mit sogenannten Featuremodellen beschrieben und daraus nach Featureselektion wesentliche Produktanteile generiert und assembliert.

Business Impact

Projektvorhaben mit MDD-Anteilen können insbesondere in einer Folgeleistungsstufe eine schnellere Umsetzung der Fachlichkeit in der Breite realisieren und damit die time-to-market reduzieren. Softwareprodukte, bei denen die Architektur mit MDD umgesetzt ist, sind wartungsfreundlicher und aufgrund der zentralen Generatorimplementierung einfacher zu erweitern.

Pro	Contra
Steigende Produktivität und Qualität	Ausbildungsaufwand für MDD-Know-how
Agile Vorgehensweisen und MDD sind gut kombinierbar	Zusätzliche Werkzeuge notwendig
Off-/Nearshoring-Aufwände für rein schematisches Coding entfallen durch Generatoren	Komplexerer Build-Prozess wegen zusätzlicher MDD-Strecke

msg systems ag
Robert-Bürkle-Straße 1 | 85737 Ismaning/München
Telefon: +49 89 96101-0 | Fax: +49 89 96101-1113
www.msg-systems.com | info@msg-systems.com

Stand: September 2012
http://www.msg-systems.com/techrefresh
XT-170-1209-1