Seminar Model Driven Integration Engineering

Based vs. Driven vs. Oriented

Patrick Jähnichen

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Das Forschungs- und Entwicklungsprojekt OrViA wird mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) gefördert, die innerhalb der zweiten Auswahlrunde der Forschungsoffensive „Softwareengineering 2006“ vergeben wurden, und vom Deutschen Zentrum für Luft- und Raumfahrt (DLR), Projektträger Informationstechnik/ Softwaresysteme betreut.

Agenda

1. Aufgabenstellung
2. Rückblick – gestellte Ziele
3. Einordnung des Themas
4. Oriented
5. Based
6. Driven
7. Fazit
8. Quellen

Aufgabenstellung

- Fragestellung, was steht wofür?
- Abgrenzung der Begrifflichkeiten voneinander
- Hauptfokus liegt auf Modellen
  - Welche Modelle gibt es
  - Welcher Art sind sie
  - Wofür werden sie verwendet
- Ziel
  - Dokument, nach dessen Lektüre die Gemeinsamkeiten und die Unterschiede dieser Begriffe klar sind
  - Besseres Verständnis für Begrifflichkeiten und Weitergabe des Wissens an andere Seminarteilnehmer

Rückblick – gestellte Ziele

- Weiterführende Recherchen zu Begrifflichkeiten
- Genaue Eingrenzung der Begriffe und ihrer Bedeutung
- Vor- und Nachteile der einzelnen Entwicklungsansätze
- Gemeinsamkeiten und Unterschiede (evtl. Synonyme?)
- Beispielszenarien zum besseren Verständnis

Einordnung

- Häufige Erwähnung der Begriffe Driven, Based und Oriented im Kontext des SE
- Verwandschaft bezüglich des Versuchs abstrakte Ideen konkret umzusetzen
- Grundsätzlich ist Unterscheidung anhand der „Strenge“ des Begriffs möglich
  - Oriented – bestehendes Wissen wird angewendet
  - Based – bestehende Informationen werden gebündelt und bei der Entwicklung einbezogen
  - Driven – bestehende Informationen werden gebündelt und dienen bei der Entwicklung nicht nur als Grundlage, sind Ergebnis der Entwicklung

Oriented

Oriented

- Hohe Abstraktion des Begriffs „orientiert“
- Es besteht kein eindeutiges Modell, eher eine grundlegende Idee
- Bei der Umsetzung stehen Entitäten, an denen sich orientiert wird, im Mittelpunkt
- Orientierung tritt fast immer im Zusammenhang mit getriebener oder basierter Entwicklung auf
- Beispiele: object-oriented, aspect-oriented

Oriented

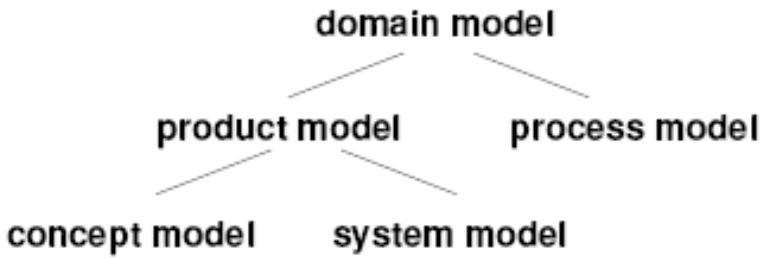
- object-oriented
  - Alles besteht aus Objekten
  - Im Mittelpunkt steht das Objekt, das Eigenschaften und Methoden aufweist
  - Orientiert sich die weitere Entwicklung nicht an diesem Paradigma, ist seine Umsetzung im System nicht durchführbar
- Aspect-oriented
  - Crosscutting Concerns (immer wieder auftretende Funktionalitäten, die nicht einem Modul zuzuordnen sind, z.B. Logging) werden gekapselt
  - Dazu stehen Programmiersprachkonstrukte zur Verfügung
  - Spezifizieren der Funktionalität und Join Points (Ausführungspunkte)
  - Zusammenführung durch ebenfalls vorhandenen AspectWeaver

Oriented

- Begriffszusammensetzungen mit „oriented“ könnten mit „das Paradigma befolgend“ übersetzt werden, sprich: symbolisieren Umsetzung eines Paradigmas
- Eine konkreter Entwicklungsprozess ist nicht erkennbar, entweder das Paradigma wird angewendet oder nicht (mit allen Konsequenzen)
- Bsp.: Objekt-Orientierte Entwicklung
  - Kein Entwicklungsprozess vorgeschrieben
  - Benutzung von Design Patterns möglich aber *nicht* notwendig

Based

- Informationen bündeln, d.h. das Sollverhalten wird in Modellen beschrieben
- Benutzung expliziter Modelle
  - um gesamten Entwicklungsprozess abzubilden (Struktur und Eingrenzung)
  - Um die Generation von Instanzen der Modelle (die Produkte des Prozesses) zu unterstützen
- Einbeziehen der Informationen, d.h. die Gesamtheit der Modelle schreibt den Entwicklungsprozess vor
- z.B. Model-Based Development
  - Kompletter Entwicklungsprozess wird anhand von Modellen dargestellt



Quelle: [5]

- Domänenmodell nicht explizit vorhanden, ergibt sich implizit aus der Gesamtheit der unterliegenden Modelle im Kontext der Domäne, in der sie gelten
- Domänenenmodell ergibt sich aus:
  - Prozeßmodell
    - Beinhaltet die einzelnen Entwicklungsschritte
    - Bezieht sich auf Koordination der Arbeitskraft
    - Hat Dokumentationscharakter

- Produktmodell
  - Beschreibt den logischen Aufbau des angestrebten Systems
  - Bezieht sich auf die Entwicklungsschritte, die in der Entwicklungsumgebung durchgeführt werden (Datenmodelle etc.)
  - Dient als Input für spätere Codegenerierung
  - Alle Prozesse des Prozessmodells haben entsprechende Entitäten im Produktmodell
  - Zerfällt noch in spezifischere Untermodelle:
  - Concept Model
    - Konzepte und Relationen (z.B. ERM)
  - Systemmodell
    - Semantisches Modell des Systems
    - Beschreibt das zu entwickelnde System
    - Nutzt dazu eine „Theorie“ wie z.B. Temporal Logic of Actions (TLA[6]) oder Higher Order Logic of Computable Functions (HOLCF[7])

- Benutzung der Modelle in verschieden Levels [5]
  - Process Level
    - Entitäten und Relationen beschreiben Entwicklungsprozess
    - Defnition fußt auf Concept Model
  - Conceptual Level
    - Entitäten und Relationen beschreiben Elemente, die zur Erstellung des Produkts benötigt werden
    - Visualisierung mittels verschiedener Beschreibungstechniken
  - System Level
    - Beschreibt die Bedeutung des Conceptual Model
    - Definiert Prozessaktivitäten, beschreibt und validiert ihre Eigenschaften
- Alle Phasen, Prozesse, Aktivitäten können im Modell eindeutig identifiziert werden
- Benefit: Zusammenspiel von Prozeß- und Produktmodell in Entwicklungsumgebungen
- Aber: oft sind Modelle zu Beginn der Implementierungsphase schon veraltet (durch undokumentierte Änderungen)

Driven

- Idee : Spezifikation unabhängig ihrer technischen Umsetzung
- Entwicklung läuft immer von abstrakt zu konkret ab
  - Aufeinanderfolgende Schritte (Entwicklungszyklen) bauen aufeinander auf, bedingen den vorhergehenden Schritt
- Modell ist Artefakt erster Klasse, möglichst hinreichend zur Generierung eines Systems
- Wechsel von deskriptiver zu präskriptiver Verwendung von Modellen
- Modellierungssprache wird zu Entwicklungssprache
- z.B. Model-Driven Architecture, Test-Driven Development

- Test-Driven Development
  - Anhand des Sollverhaltens werden Tests entwickelt, die dieses Sollverhalten prüfen
  - Eigentliche Entwicklung besteht darin, mit so wenig Aufwand wie möglich die Tests zu erfüllen
  - Relativ abstrakte Abbildung der Anforderungen (Input – Ouput)
  - Losgelöst von technischer Umsetzung

- Model-Driven Architecture
  - Im Mittelpunkt steht das Modell
  - Weiter Erläuterungen beziehen sich auf MDA

- Es bestehen formal eindeutige Modelle verschiedener Abstraktionsebenen (3 Kernmodelle) [10]
  - Computation-independent model (CIM)
  - Platform-independent model (PIM)
  - Platform-specific model (PSM)
- „platform“ immer relativ zu betrachten
- Strenge Trennung zwischen fachlichen und technischen Anteilen (Kommunikationsverbesserung zw. Fach- und Nichtfachleuten)

- Modelltransformationen (von höherer zu geringerer Abstraktion)
- Modell „geringster“ Abstraktion (PSM) dient meist als Input zur Codegenerierung (PIM → Code auch möglich [10])
- Modelltransformationen und Codegenerierung laufen idealerweise vollautomatisiert ab
  - Transformationstools, Arbeit besteht darin, Transformationsregeln festzulegen
- Änderungen im Design sollen nur im Modell vorgenommen werden → auch Nichtfachleute können auf Entwicklung Einfluß nehmen

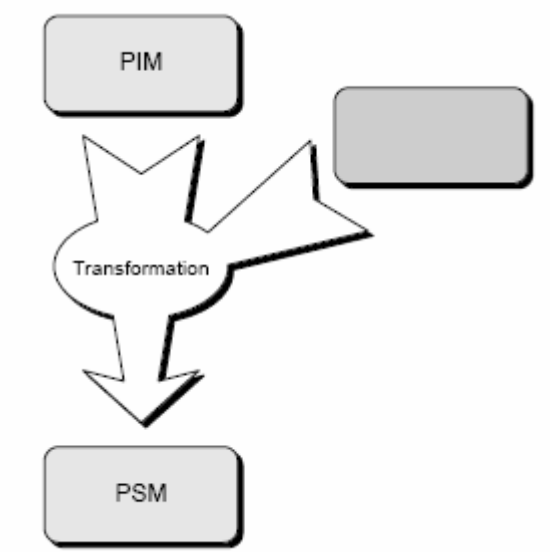
- Ziele der MDA [11]
  - Konservierung der Fachlichkeit
    - Nutzung der vorhandenen Kenntnisse über Geschäftsprozesse, durch plattform-unabhängige Modellierung dieser Prozesse
    - Schaffen von Ausgangspunkten für Pflege und Weiterentwicklung
    - Entgegenwirken der Dokumentationsdegeneration
  - Portierbarkeit
    - Durch systematische Abstraktion
  - Systemintegration und Interoperabilität
    - Trennung fachlicher Konzepte und der konkreten Repräsentation dieser Konzepte
    - Erleichterung der Bildung von Schnittstellen
    - Wiederverwendbarkeit von Komponenten (keine bzw. geringe Redundanzen)

- Domänen-Orientierung
  - Durch Erstellung domänenspezifischer Modelle und Wiederverwendung von Architekturmetamodellen
  - Wechsel von Technikzentrierung zu Konzentration auf fachliches Domänenwissen
  - Keine technologischen Vorgaben der IT an Geschäftsprozesse mehr
- Präzise Systembeschreibung
  - System so präzise durch Modell beschreiben, dass lauffähige Anwendung generiert werden kann

- CIM – Computation Independent Model
  - Sicht auf Gesamtsystem
  - Unabhängig von Implementierung
  - Beschreibung des Modells im Vokabular seiner Domäne
  - Betont Anforderungen an System und Umwelt
  - Synonyme: Business Model, Domain Model
- PIM – Platform Independent Model
  - Beschreibt formal Struktur und Funktionalität des Systems
  - Unabhängig von Implementierung
  - Abstrahiert von zugrundeliegender Plattform

- PSM – Platform Specific Model
  - Anreicherung des PIM mit plattformabhängigen Informationen
  - Wird zu Implementierung, falls es Informationen zur Konstruktion und zum Betrieb des Systems liefert
  - Auch direkte Ausführung des Modells möglich (Executable Models)

- Modelltransformation [11]:



Fazit

- Oriented fällt aus der Betrachtung heraus, es bezeichnet die Verwendung eines Paradigmas
- Based und Driven sind zwei sehr verschiedene Ansätze mit dem selben Ausgangspunkt: Modellierung des Systems in möglichst abstrakter Weise
- Wichtige Frage nach dem Anspruch der Entwicklungstechnik
  - Oriented bietet eine „große Idee“ an der sich orientiert wird, mehr nicht
  - Based hat Anspruch, den gesamten Entwicklungsprozess durch verschiedene Modelle abzubilden
  - Driven konzentriert sich auf eine Methodik, die den Entwicklungsprozess treibt, ihn voranbringt

Fazit – zeitliche Einordnung

- Oriented
  - Häufiges Auftreten des Begriffs seit Einführung der Objektorientierung (z.B. [8]) ca. 1993
- Based
  - Zusammenfassung der Anforderungen an den Software-Entwicklungsprozess (Mitte 1990er bis 2000)
  - Messbare (prüfbare) Methode (ISO 9000:2000 [9]) im Beispiel des Model-Based Development
    - Gesamter Projektprozess → Qualitätsmanagement
    - Dadurch viele Anwender dieser Methodik
    - Industriestandard
- Driven
  - Middleware tritt in den Vordergrund (2001 - ?)
  - Im Mittelpunkt steht das Entwickeln der Software in Art und Weise, die ermöglicht mehrere Plattformen zu bedienen
  - „Drumherum“ ist nicht mehr berücksichtigt

Fazit – Unterschiede

- Based:
  - Entwickelt weitere Modelle, um den Entwicklungsprozess zu beschreiben
  - Modelle sehr starr und unveränderlich in Ihrer Grundsätzlichkeit
  - Auf eine Plattform beschränkt (die für die entwickelt wird)
- Driven:
  - Entwicklungsprozess steht fest und soll größtenteils automatisch ablaufen(ideal: executable models)
  - Modelle lassen sich ineinander überführen und sind sehr flexibel (relative Verwendung des Begriffs „platform“)
  - Grundgedanke der Wiederverwendbarkeit von Modellen für verschiedene Plattformen

Quellen

- [1] A Component-Based Modeling Approach, Alois Stritzinger
  http://www.swe.uni-linz.ac.at/publications/pdf/TR-SE-96.08.pdf
  (zuletzt abgerufen: 7. Januar 2007)
- [2] Retrospective and Challenges for Model-Based Interface Development, Pedro Szekely
  http://www.idi.ntnu.no/emner/tdt12/szekely-retrospective-CADUI96.pdf
  (zuletzt abgerufen: 7. Januar 2007)
- [3] Model-Driven Development: A Metamodeling Foundation, Colin Atkinson und Thomas Kühne
  http://www.mm.informatik.tu-darmstadt.de/~kuehne/publications/papers/mda-foundation.pdf
  (zuletzt abgerufen: 7. Januar 2007)
- [4] An Aspect Oriented Model Driven Framework, Devon Simmonds, Arnor Solberg, Raghu Reddy, Robert France, Sudipto Ghosh
  http://www.cs.colostate.edu/puml/pub/EDOC05.pdf
  (zuletzt abgerufen: 7. Januar 2007)
- [5] Model-Based Development, B. Schaetz, A. Preschner, F. Huber, J. Philipps
  http://www.inf.ethz.ch/personal/pretscha/papers/TUM-I0402.pdf
  (zuletzt abgerufen: 7. Januar 2007)

- [6] L. Lamport. The Temporal Logic of Actions. ACM Transactions on Programming Languages and Systems, 16(3):872–923, May 1994
- [7] HOLCF: Higher Order Logic of Computable Functions in Higher Order Logic Theorem Proving and its Applications
  F. Regensburger, E. Schubert, P. Windley, and J. Alves-Foss,
  Volume 971 of Lecture Notes in Computer Science, pages 293–307
  Springer-Verlag, 1995
- [8] Object-Oriented Analysis and Design with Applications, Grady Booch
  Addison-Wesley, 1993
- [9] Examples of the ISO 9000 standards in use
  http://www.iso.org/iso/en/iso9000-14000/understand/selection_use/examplesofiso9000.html
  Example 6
- [10] MDA Guide Version 1.0.1, Object Management Group
  http://www.omg.org/docs/omg/03-06-01.pdf
- [11] MDA® Effektives Software-Engineering mit UML2® und Eclipse™, Gruhn, Pieper, Röttgers
  Springer-Verlag, 2006