Lernzettel

Modellgetriebene Entwicklung und modellbasierte Programmierung

Universität: Technische Universität Berlin

Kurs/Modul: Softwaretechnik und Programmierparadigmen
Erstellungsdatum: September 19, 2025

Lernzettel: Modellgetriebene Entwicklung und modellbasierte Programmierung

(1) Definition und Zielsetzung.
Modellgetriebene Entwicklung (MDE) bezeichnet einen Ansatz, bei dem Modelle die Primärartefakte der Softwareentwicklung darstellen. Aus Modellen leiten sich Artefakte wie Code, Tests oder Ausführungspläne ab. Ziele von MDE sind:

- Abstraktion und Trennung von Domänenkonzepten und Implementierung.
- Automatisierte Transformationen zwischen Modellen sowie in Code.
- Erhöhte Wiederverwendbarkeit, Konsistenz und Nachverfolgbarkeit.

(2) Kernkonzepte.

- Modelle, Metamodelle und Modeling Languages: Modelle sind Repräsentationen von Systemaspekten; Metamodelle definieren die Struktur der Modeling Language.
- DSMLs (Domain-Specific Modeling Languages): Domänennahe Sprachen zur Beschreibung von Konzepten einer Anwendungsdomäne.
- Modelltransformationen: Modell-zu-Modell (M2M) und Modell-zu-Text (M2T) Transformationen.
- Transformationstools und -sprachen: ATL, QVT etc. zur Definition von Transformationen.
- Codegenerierung und Templates: Aus Modellen wird automatisch Quellcode oder weitere Artefakte erzeugt.
- Verifikation und Validierung: Modelle prüfen, Spezifikationen validieren, Konsistenz sicherstellen.
- Round-Trip Engineering: Synchronisation zwischen Modellen und implementiertem Code.

(3) Architektur und Ebenen in MDE.

- M0 (Realwelt): Die physische und Geschäftswelt, aus der Anforderungen stammen.
- M1 (Modelle): Instanzen der Modeling Language, z. B. Domänenmodelle, Architekturmodelle.
- M2 (Metamodelle): Definition der Struktur einer Modeling Language (z. B. DSL-Metamodelle).
- M3 (Meta-Metamodel): Define die Grammatik der Metamodelle selbst (z. B. MOF/Ecore).

(4) Modelltransformationen und Codegenerierung.

- Modell-zu-Modell (M2M): Transformationen zwischen Modellen, z.B. Domain-Modelle zu Architekturmodellen.

- Modell-zu-Text (M2T): Generierung von Code, Tests oder Dokumentation aus Modellen.
- Transformationssprachen: ATL, QVT oder proprietäre DSL-Transformer.
- Generierungstemplates: Vorlagen, die Struktur und Codeformate definieren.
- Konsistenzmechanismen: Verifikation von Abhängigkeiten, regelmäßige Re-Synchronisation bei Änderungen.

(5) Vorgehensweise in typischen Projekten.

- Domänenanalyse und Domänenmodellierung: Welche Konzepte gibt es, welche Varianten?
- Entwurf eines DSML oder Anpassung eines bestehenden DSLs.
- Definition des Metamodels und der Modellierungsregeln.
- Implementierung von Modelltransformationen (M2M, M2T).
- Aufbau einer Codegenerierungslate mit Templates.
- Validierung der Modelle (Statischer Check, Konsistenzprüfungen) und Tests.
- Integration: Verknüpfung von generiertem Code mit bestehendem System.
- Round-Trip Engineering: Sicherstellen, dass Änderungen sowohl im Modell als auch im Code konsistent bleiben.

(6) Vorteile und Herausforderungen.
Vorteile:

- Höhere Abstraktion und klare Domänenabgrenzung.
- Automatisierung senkt manuelle Implementierungsaufwand.
- Verbesserte Nachverfolgbarkeit und Wiederverwendbarkeit von Modellen.
- Frühzeitige Validierung durch Simulation und Modelle.

Herausforderungen:

- Lernkurve und Toolunterstützung.
- Round-Trip Engineering und Modell- drift.
- Komplexität von Transformationslogik und Performance.
- Sicherstellung der Konsistenz zwischen Modellen und Code.

(7) Beispiel eines DSML-Snippets und Generierung.
Beispiel eines sehr einfachen DSML-Snippets zur Modellierung von Klassenstrukturen:

```
Klasse Person {
  String name;
  Integer alter;
}
```

Dieses Modell könnte durch eine M2T-Transformation in Java-Code übersetzt werden, z. B.:

```
public class Person {
  private String name;
  private Integer alter;
  // Getter/Setter
}
```

Zudem könnten dazu passende Unit-Tests generiert werden.

(8) Praxisanker.

- Modellgetriebene Ansätze unterstützen klare Architekturen und Trennung von Belangen.
- Sie ermöglichen Domain-Driven Design durch DSLs, die die Domäne direkt abbilden.
- Der Fokus liegt auf der Automatisierung von Wiederholungsaufgaben und der Verifikation von Modellen.

(9) Zusammenfassung.
Modellgetriebene Entwicklung verschiebt den Schwerpunkt der Softwareentwicklung von reinem Code auf Modelle. Durch Metamodelle, DSMLs und Transformationen entstehen automatisierte Pipelines von Modellen zu Code und Tests. Dadurch lassen sich Abstraktion, Wiederverwendbarkeit und Transparenz erhöhen, während gleichzeitig Herausforderungen wie Round-Trip Engineering und Toolunterstützung adressiert werden müssen.