|  |
| --- |
| Model driven software development mit dem tool swagger |
|  |
| Herval Bernice Nganya Nana1, Oussema Mzoughi2, Aliridha Haouari3, Christian Lange4 |
| 1,2,3,4Technische Hochschule Brandenburg Fachbereich Informatik und Medien Magdeburger Str. 50, 14770 Brandenburg an der Havel  1 nganyanana@th-brandenburg.de  2 oussema.mzoughi@th-brandenburg.de  3aliridha.haouari@gmail.com  4langchri@th-brandenburg.de |

Keywords:

Abriss: !!!!

Abstract: .

# Einleitung

* 1. **Motivation**

Text

* 1. **Ziel (Fachliches Ziel)**

Text

* 1. **Aufgabenstellung**

Text

* 1. **Abgrenzung**

Text

* 1. **Ergebnis**

Text

# Model-driven-development

# 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 [3].

# Dem gegenüber dienen Modelle als graphische Repräsentation, wie sie beispielsweise in der UML (Unified Modeling Language) notiert werden, im Softwareentwicklungsprozess hauptsächlich zur Spezifikation und Dokumentation. Nur partiell erfolgt eine Generierung von Programmcode aus solchen Modellen, der dann noch um wesentliche Teilemanuell zu ergänzen ist. In seltenen Fällen werden Änderungen auf der Quellcodeebene auch auf der Modellebene nachgeführt [2].

# Mit der Model Driven Architecture (MDA) wurde ein Standardisierungsvorschlag der Object Management Group (OMG)erstellt, der die Repräsentation von Software von der Programmcodeebene auf die Modellebene hebt. Um die Komplexität auf Modellebene zu reduzieren, werden Modelle und Plattformen verschiedener Abstraktionsebenen unterschieden, die aufeinander aufbauen

Dabei kann die Realisierung eines plattformunabhängigen Modells durch Wahl einer Plattform, also der konkreten technischen Umsetzung, teilweise oder vollständig automatisiert erfolgen [2].

Modelle werden durch die Auswahl von Plattformen in weniger abstrakte Modelle transformiert bis letztendlich ausführbarer Programmcode entsteht.

MDA soll die Portierbarkeit und Wiederverwendung von Modellen und dadurch der Software verbessern.

* 1. **Modele** (CIM / PIM/PSM)

In MDA wird hauptsächlich zwischen drei Typen von Modellen unterschieden: den computerunabhängigen (CIM, Computation Independent Model), plattformunabhängigen (PIM, Plattform Independent Model) und plattformspezifischen (PSM, Plattform Specific Model) Modellen [2].

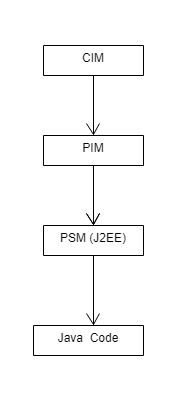
**Das Computation Independent Model (CIM):** beschreibt ein Softwaresystem auf fachlicher Ebene. Es liegt in einer Sprache vor, die für die fachlichen Anwender des Systems verständlich ist, und dient zum Diskurs zwischen Softwarearchitekten und Anwendern über Leistungsumfang und Anforderungen. Das CIM legt fest, was ein Softwaresystem leistet. Es definiert nicht, wie es dies leistet oder wie das System strukturiert ist [1].

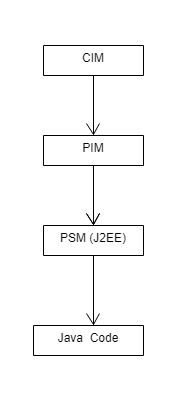
**Das Platform Independent Model (PIM):** modelliert die Funktionalität einer Komponente unabhängig von der gegebenen Plattform. Damit enthält ein PIM also den Teil eines Systems, der sich beschreiben lässt, ohne die endgültige Zielplattform zu kennen[1].

**Das Platform Specfic Model (PSM):** kennt eine spezielle Plattform und realisiert ein PIM, wofür die von der Plattform bereitgestellten Schnittstellen genutzt werden[1].

* 1. **Grundlegende Transformationsmöglichkeiten**

Die CIM-PIM und PSM Modelle sind die hauptschritt des MDA-Ansatzes. Jede dieser Modelle enthält Informationen, die zum Generieren des Codes der Anwendung benötigt werden. Der Code wird durch automatische Generierung aus dem PSM erhalten, Der PSM wird durch erfolgreich Transformationen von CIM-Modell erhalten zu PIM , und PIM-Modell zu PSM erhalten. Die Modelle Transformation ist ein wichtiger Schritt im MDA-Prozess Jede Transformation besteht aus der Anwendung von Transformationsregeln , sogenannten Mapping , die für automatische Ausführung formal definiert sind.





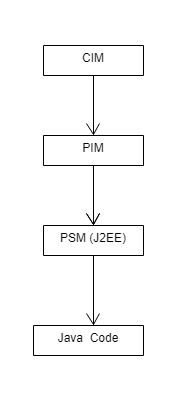


Abb3: Transformationen in MDA

# swagger (Rest service)

* 1. **Was ist Swagger**

Text

* 1. **Wie funktioniert Swagger**

Text

* 1. **Swagger mit PIM-Nutzung**

Text

* 1. **Dokumentation (Auto generated)**
     1. **Screen der Online-Dokumentation**
  2. **Buildvorgang (Von JSON-Datei zum generierten Projekt mit Swagger-2Editor)**

Text

# datenbank (Spring jpa / Hibernate)

1. Jede REST-API muss mit einer Datenbank verbunden sein. Daher muss Datenverbindung zwischen REST-API und Datenbasis konfigurieren. Für die Datenbasis setzen wir MySQL ein, der MySQL ist der bekannten Datenbank relationalen Datenbank. Mit Hilfe der Integration von [Spring Data](http://projects.spring.io/spring-data/) in unsere Spring Boot Anwendung, wird ein großer Teil der Standardimplementierung, wie z. B. eine Entität speichern, per REST freigeben, HTTP-Anfragen verarbeiten und vieles mehr, bereits automatisch bereitgestellt. Allerdinges Für die Generierter REST-API Code durch Swagger.io wird keine automatisch OR-Mapping bereitgestellt, deshalb setzen wir mit Hibernate ein. Hibernate ist als OR-Mapping-Framework seit Jahren fest in der Java-Welt, mit Hibernate dem Entwickler auch über die OR-Mapping-Thematik hinaus einfacher machen kann.

# Konzept/problemstellung

# Was soll gemacht werden?

# Persistenz (Datenbank, S3)

# Erreichbarkeit (EC2)

# aRCHITEKTUR

# SCHNITTSTELLENBESCHREIBUNG (dIAGRAMME)

# prototyp

* 1. **Funktionalität**

Text

* 1. **Schnittstellenerzeugung**

Text

* 1. **Erweiterung der Schnittstellen (mit Datenbankerstellung)**
     1. **Encryption-Möglichkeit**

Text

* 1. **Rebuild des Projektes nach Schnittstellenbearbeitung**
     1. **Paper lesen, eine Möglichkeit wählen und im Zusammenhang mit dem Thema beschreibung**

# Ergebnisse/Probleme

# Unübersichtlicher, generierter Code

# Projektstruktur (Packages, ...)

Text

# Zusammenfassung ausblick

Text

# Quellen

[1]Lexikon Erstellt von Dr. Martin Kempa, Zoltán Ádám Mann, sd&m AG software design & management

<https://gi.de/informatiklexikon/model-driven-architecture/>

# [2] ITwissen Online : MDSD (model driven software development)

<http://www.itwissen.info/MDSD-model-driven-software-development-Modellgetriebene-Software-Entwicklung.html>

[3] OMG: MDA, <http://www.omg.org/mda/>