docid: "" title: "Erstellung von grafischen Modellierungswerkzeugen mit Eclipse Sirius"
subtitle: "Seminar Modellgetriebene Softwareentwicklung"
author: "Denis Ayana"
institute: "FH Dortmund"
date: "03.06.2018"
lang: "de-DE"
...

# Einleitung

## Thema der Seminararbeit

Diese Seminararbeit handelt von Eclipse Sirius und dessen Einordnund in die modelgetriebe Softwareentwicklung, sowie die Konzepte, Realisierung und Verwendung.

## Ziel der Seminarbeit

Ziel der Seminararibeit ist es ein Verständnis für Eclipse Sirus zu entwickeln. Und Eclispe Sirius in die modelgetrieben Softwareentwicklung einzuordnen, sowie zu verstehen wie modelgetribene Softwareentwicklung mit Hilfe von Eclipse Sirius funnktioniert.

## Modelle in der Softwareentwicklung

In der Softwareentwicklung kommen Modelle in zwei verschiedenen Rollen zum Einsatz. Die erste Rolle von Modellen ist die Rolle von Modellen zur Dokumentation. Hierbei werden die Modelle eingestzt, um Softwaresysteme zu dokumentiern. Da diese Modelle von einem Programmierer zu meist erst durch die Implementierung von Code zu einer lauffähigen Anwendung werden, haben diese nur eine unterstzüzende Funktion. Oft werden diese Modelle, von Programmierern, als Overhead empfunden. Ein weiterer Nachteil diese Modelle ist es, dass sie oft an die dynamischen Softwaresysteme angepasst werden müssen. Da die Verbindung zwischen diesen Modellen und einer Anwendung nur gedanklicher Natur ist, müssen die Veränderung manuell von Programmieren nachgehalten werden. (Stahl/Völter MDSD s.10) Die Zweite Rolle von Modellen ist die Rolle von Modellen zur Entwicklung. Dabei sind die Modelle mit dem Code einer Anwendung gleichzusetzen. So sethen die Modelle und der Code in einer formalen Beziehung zueinander. Die Umsetztung der Modelle in Code erfolgt automatisiert. Für diese automatisierte Umsetzung werden bestimmte Compiler, Transformatoren oder Interpreter benutzt. Durch die enge verbindung zwischen Modellen und Code lässt sich die Qualität und Wartbarkeit von Softwaresystemen verbessern.(Stahl/Völter MDSD s.11)

## Modelgetriebene Softwareentwicklung

Bei der modelgetriebenen Softwareentwicklung geht es darum, durch eine domänenspezifische Abstraktion der Realität, formale Modelle zu erstellen. Dieser formalen Modelle werden dann mit Hilfe von Compilern, Transformatoren oder Interpretern zu lauffähigem Code automatisiert umgesetzt. Diese automatisierte Umsetztung kann durch zwei mögliche Wege geschehen. Zum einen kann durch ein Generator aus den Modellen Code generiert werden, welcher dann in den nachfolgen Build-Prozess einfließt. Zum anderen

können die Modelle durch einen entsprechenden Interpreter direkt zu einem lauffähigen Code interpretiert werden.

Oft werden die Modelle selbst durch Metamodelle beschrieben. Metamodelle werden wiederrum durch Metametamodelle definiert. Das heißt, dass ein Modell dem festgelegten Alphabet und der im Metamodell definierten Ordnung entsprechen muss. Metamodelle bescheiben durch eine abstracte Syntax und eine statische Semantik die formalen Modelle. So müssen die formalen Modelle sich strikt an die benannten Konstruke aus dem Metamodel halten. Die konkrete Syntax ist hierbei irrelevant. Auch die Sematik ist im Metamodell vorgegeben. Das Metamodell beschreibt somit, nach welchen Regeln das Modell als gültig bzw. als valide anzusehen ist. Das Metametamodell beschreibt auf einer noch abtracteren Ebene das Metamodell. Aber unter den selben Gegebenheiten, wie das Metamodell das Modell berschreibt. Somit kann gesagt werden, dass ein Metametamodel eine Instanz eines Metamodells ist.

- hier bild von metamodel und metametamodel ein tragen

Die Metaisierung von Modellen könnte theoretisch so weiter gesetzt werden.Jedoch macht dies wenig Sinn, da das Metametamodell das Fundament der Metamodelarchitektur darstellt. Es beschreibt sich somit selbst.

Zusammenfassen lässt sich die Metaisierung in einer 4-Ebenen-Metamodelarchitektur beschreiben. An oberster Stelle das Metametamodel, welches dei Grundlagen für die Metamodellarchitektur bildet und die Sprache des Metamodells beschreibt. Darunter liegt das Metamodell, welches eine Instanz des Metametamodells ist und die Sprache des formalen Modells beschreibt. An zweit unterster Stelle liegt das eigentliche formale Modell, welches eine domänenspezifische Abstarktion der Realität ist. Zuletzt, an unterster Stelle, das Benutzer-Objekt, welches keine abtraktion, sondern einer genaue Abbildung der Realität ist.

- hier-ebenen metamodelarchitektur einfügen
- Grafische oder Textuelle Darstellung
- Solution Space und Problem Space
    - General PurposeLanguages
    - Domain-specificLanguages

## Ziele modelgetribener Softwareentwicklung

Es gibt eine Hand voll Gründe warum bei der Entwicklung von Software modelgetriebene Verfahren zum Einsatz kommen sollten:

- Automation:

  Durch die Automation kann mit Hilfe von modelgetriebener Softwareentwicklung eine Verbesserung der Entwicklungsgeschwindigkeit erreicht werden. Aus formalen Modellen kann durch einen oder mehrere aufeinander folgende Transformationsschritte letztendlich lauffähiger Code erzeugt werden. (Stahl/Völter MDSD s.23)

- Wiederverwendbarkeit:

  Einmal definierte Architekturen, Modellierungssprachen und Transformationen können im Sinne einer Software-Produktionsstraße zur Herstellung diverser Softwaresysteme verwendet werden. Dies führt zu

einem höheren Grad der Wiederverwendung und macht Expertenwissen in Softwareform in der Breite verfügbar. (Stahl/Völter MDSD s.24)

- Handhabbarkeit:

  Ein weiteres wesentliches Potenzial ist die bessere Handhabbarkeit von Komplexität durch Abstraktion. Mit den Modellierungssprachen soll „Programmierung“ oder Konfiguration auf einer abstrakteren Ebene möglich werden. Die Modelle müssen dazu in einer möglichst problemorientierten Modellierungssprache ausgedrückt werden.

### Einordnung von Eclipse Sirius

Eclipse Sirius ist ein OpenSource Projekt der Eclipse Foundation. Es wurde 2013 von Obeo auf der Eclipse Con vorgestellt. Eclipse Sirius ist ein Framework zur Entwickelung von domänenspezifischen Modellierungswerkzeugen.

# Konzepte

### Konstrukte der Sprache

# Realisierung

### Verwendung

### Workflow

#### MDSD Workflow

#### MDA Workflow

### Beispiele

# Tooling

### Installation

### Tipps und Tricks

# Zusammenfassung

# Literatur