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.

- vier-ebenen metamodelarchitektur einfügen

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.

Domäne und domänenspezifische Sprache

Domäne

In der Softwareentwicklung beschreibt der Begriff der Domäne ein bestimmtes abzugenzendes Gebiet. Eine Domäne kann sowohl fachlicher, als auch technischer Natur sein. In der modelgetrieben Softwareentwicklung werden diese Domänen meist durch ein entsprechendes Metamodel beschrieben. Das Metamodel definiert die Konzepte der Domäne. Fachliche Domänen ist zum Beispiel ein Webshop. Die technische Domäne ist dann zum Beispiel die Architektur des Softwaresystems.

Domänenspezifische Sprache

Als domänenspezifische Sprache versteht sich eine formale Sprache oder auch Modellierungssprache der Softwareentwicklung. Diese Sprache beschreibt die Konstrukte der entsprechenden Domäne. Dabei ist sie einfacher und prägnanter als herkömliche Programmiersprachen. Um dies sicher zustellen nutzt die domänenspezifische Sprache das Vokabular der entsprechenden Domäne und eine Notation die Sachverhalte aus der Domäne in einer geeigneten Form darstellt. Die Notation der Sachverhalte einer Domäne erfolgt dabei in grafischer oder auch in textueller Darstellungart.

Eclipse Sirius

Auf der Eclipse Con 2013 wurde Eclipse Sirius erstmals, von Obeo, vorgestellt. Derzeit ist Eclipse Sirius ein Opensource Projekt der Eclipse Foundation. Eclipse Sirus ist ein Framework zur Entwickelung von domänenspezifischen Modellierungswerkzeugen. Dazu nutzt Eclipse Sirius das Eclipse Modeling Framwork(EMF) für das Erstellen und bearbeiten von Modellen, sowie das Grafical Modeling Framework(GMF) für die grafische Darstellung dieser Modelle. Durch die Verwendung von Eclipse Sirius ist es den Benutzern möglich verschiedene Modelle grafisch darzustellen. Grundlage dieser Modelle ist eine domänenspezifische Sprache(DSL). Diese domänespezifische Sparche ist in Form eines Metamodels definiert.

Konzept

Wie schon erwähnt ist die Grundlage für das Arbeiten mit Eclipse Sirius ein Metamodell. Durch die Erstellung des Metamodells ist es möglich Programmcode zu generieren. Dieser generierte Programmcode wird dann für die Ausführung einer neuen Entwicklungsumgebung genutzt. In der neuen Entwicklungsumgebung ist es den Benutzern möglich mit konkrete Modell zu arbeiten. Den Benutzern stehen dazu eine Menge von Eclipse-Editoren zur Verfügung um EMF Modelle erstellen, bearneiten und visualisieren zu können. MIt Hilfe von Eclipse Sirius können die Benutzer nicht nur Diagramme, sondern auch Tabellen, Matrizen oder Bäume zu Visualisierung der Modelle erstellen.

Diagramme

Die mit Eclispe Sirius erstellten Diagramme zeigen grafisch die Elemente der Modelle. Welche Elemente dargestellt werden und in welcher Art und Weise dies geschiet wird zuvor von den Benutzern angegeben. Die angegebenen Regeln definieren auch die Beziehungen zwischen den einzelnen Elementen der Modelle. Eine

Beziehung kann sowohl als Kante zwischen zwei Elementen oder als ein Element, welches sich in einem weiteren Element befindet, dargestellt werden. Wenn ein Diagramm mit einem EMF Modell synchronisiert ist, wird es automatisch angepasst, sobald eine Änderung an den Darstellungsregeln vorgenommen wird. Damit die Benutzer ein Modell direkt aus dem Diagramm heraus bearbeiten können, muss der Ersteller der Entwicklungsumgebung eine bestimmte Menga von Bearbeitungswerkzeugen bereitstellen. Diese Bearebitungswerkzeuge dienen dann dazu, dass die Benutzer neue Objekte oder Bezeihungen erstellen können. Des Weiteren kann auch das Verhalten bei Eingaben durch die Benutzer oder klassische Aktionen, wie das Feshalten oder Loslassen von Objekten, definiert werden. Um den Benutzern zu unterstützen, die potenzielle Komplexität ihrer Modelle zu meistern, bietet Eclipse Sirius mehrere Mechanismen, um sich auf relevante Elemente zu fokusieren.

- Bedingtes Aussehen:

  Die grafische Darstellung kann abhänig von den Eigenschaften eines Objekts geändert werden. Zum Beispiel kann ein Objekt propotional zu dem Wert eines Attributs sein Größe oder Farbe in dem Diagramm verändern.

- Schichten und Filter:

  Durch das Benutzen von Schichten oder Filtern kann eine bestimmte Menge von Objekten aus einem Diagramm, abhänig von einer angegebene Bedingung, angezeigt oder versteckt werden.

- Validierung und Schnell-Reperatur:

  Das Modell kann durch die Angabe von spezifizierten Regel validiert werden. Die Probleme, welche bei der Validierung auftauchen, werden in einer eigenen Übersicht aueglistet. Das dazugehörige Objekt des Diagramms wird dabei farblich markiert. Für einige Probleme wird eine Schnell-Reperatur vorgschlagen, um das Modell zu korrigieren.

Tabelen und Matrizen

Modelle könne auch als Tabelle und Matrix dargestellt werden. Die erste Spalte der Tabelle enthält dabei das Objekt. In den darauf folgen SPlaten werden die zum Objekt gehörenden Eigenschaften aufgelistet. In den Zellen stehen die entsprechenden Werte, welche für die Eigenschaften der Objekte angegeben werden. Die Benutzer können die Werte in den Zellen bearbeiten und neue Werte eintragen. Die Art der grafischen Darstellung kann auch, wie bei den Diagrammen schon beschrieben, von dem Ersteller der Entwicklungsumgebung definiert und geändert werden. Eine Matrix enthält zwei listen von Objekten. Die erste Liste wird in der ersten Spalte, die andere Liste in der ersten Zeile angebildet. Der Inhalt der Zellen beschreibt dabei das Zusammen spiel der beiden außenliegenden Objekte, einer jeden Liste.

Bäume

Die dritte und letzte Art der grafischen Darstellung sind Bäume. Dabei werden die Objekte eines Modells in hirarchischer Anordnung abgebildet. Wie bei den Diagrammen, Tabellen und Matrizen kann die Art der grafischen Darstellung von dem Ersteller der Entwicklungsumgebung definiert und geändert werden.

Viewpoints

Ein Viewpoint, zu Deutsch Aussichtspunkt, kapselt eine bestimmte Teil der Präsentation eines Modells. Um es den Beutzern einfacher zu machen sich auf einen bestimmten Teil einer Domäne zu konzentieren, werden

Viewpoints benutzt. Innerhalb des Viewpoint sind die Art und Weise der grafischen Dartstellung, sowie das Verhalten bei Benutzereingaben definiert und gespeichert. Ein Modell kann somit durch verschiedene Viewpoints visualisiert werden. Jeder dieser Viewpoints kann sich dabei auf einen von Ersteller definierten, abgegrenzten Bereich des Modells fokusieren.

Realisierung und Verwendung

Im nachfolgenden Teil der Seminararbeit wird die Verwendung von Eclipse Sirius und die Realisierung einer eigenen Entwicklungsumgebung, zum Erstellen von grafischen Modellen, beschrieben. Dazu wird an Hand eines konkreten Beispiels jeder Schritt einzeln erklärt. Am Anfang wird ein Metamodel erstellt, welches die Domäne abbildet. Aufbauend auf dem Metamodel wird in einer eigenen Entwicklungsumgebung ein konkretes Model erstellt. Zum Schluss wird noch die Erstellung eigener Modelierungswerkzeuge, zum Hinzufügen von Objekten und deren Beziehungen zueinander, erklärt.

Beispiele

Allgemin erklärt was die Domäne ist. was das metamodel beschreibt

Workflow

geneiren von code dann in neuer workbanch die konkreten projekte mit dem viewpoint projekt dann die objekte erstellen dann die grafischen aspekte erstellen dann die cration tools

MDSD Workflow

MDA Workflow

Tooling

Installation

obeodesigner

Tipps und Tricks

- bidirektionale ralationen kp wie abbilden
- bilder von den objekten ändern
- probleme mit der manifest datei beheben. durch preferences -> plug-in development -> target platform deactivate then activate problem neim anzeigen des namen- im square den lable expression in feature:bezeichnung ändern oder dass attribut der klasse was den namen speichert
-

Zusammenfassung

Literatur