Inhaltsverzeichnis

[1 F 2](#_Toc433364137)

[2 f 3](#_Toc433364138)

[3 Konzeption und Implementierung 4](#_Toc433364139)

[3.1 Genereller Überblick 4](#_Toc433364140)

[3.2 Konzept des GMF-Editors 5](#_Toc433364141)

[3.3 Implementierung des GMF-Editors 5](#_Toc433364142)

[3.4 Konzeption der XSL-Transformation 6](#_Toc433364143)

[3.5 XSL-Transformation Implementierung 9](#_Toc433364144)

[Literaturverzeichnis 10](#_Toc433364145)

[Glossar 11](#_Toc433364146)

[Abkürzungsverzeichnis 12](#_Toc433364147)

[Abbildungsverzeichnis 13](#_Toc433364148)

[Tabellenverzeichnis 14](#_Toc433364149)

[Quellcodeverzeichnis 15](#_Toc433364150)

[Inhalt der beigelegten DVD 16](#_Toc433364151)

# F

# f

# Konzeption und Implementierung

Dieses Kapitel beschreibt zunächst das Konzept des Gesamten Aufbaus dieser Abschlussarbeit. Darauffolgend wird die Realisierung der dafür benötigten Softwaremodule erläutert.

## Genereller Überblick

Wie in Abbildung x.1 zu sehen ist, besteht der Gesamte Aufbau aus mehreren Schichten, die aufeinander bauen. Auf der ersten Schicht befindet sich der FSM-Editor der mit Hilfe von GMF (Eclipse Foundatioin, 2011) aufgebaut wurde. Mit Diesen FSM-Editor steht dem Programmierer ein Werkzeug zum vereinfachten Entwurf eigener, graphischer Zustandsautomaten zur Verfügung. Das entstehende Zustandsdiagramm, was mit Hilfe vom FSM-Editor gezeichnet wurde, wird in einen XML File gespeichert.

Dieses XML-File enthält alle domänenspezifischen Aspekte sowie alle Informationen für die graphische Darstellung der erstellten FSM. Dadurch leicht zu parsen und auch von anderen Plug-ins oder Anwendungen zu verwenden. In diesem Fall werden diese XML-Informationen an der zweiten Schicht übergeben (siehe Abbildung 1).

Die zweite Schicht entspricht der XSL-Transformation (<http://www.w3.org/TR/xslt/>), die hier eingesetzt wird, um den XML-File in einen C++-File, der den generierten C++-Code der FSM beinhaltet, umzuwandeln.

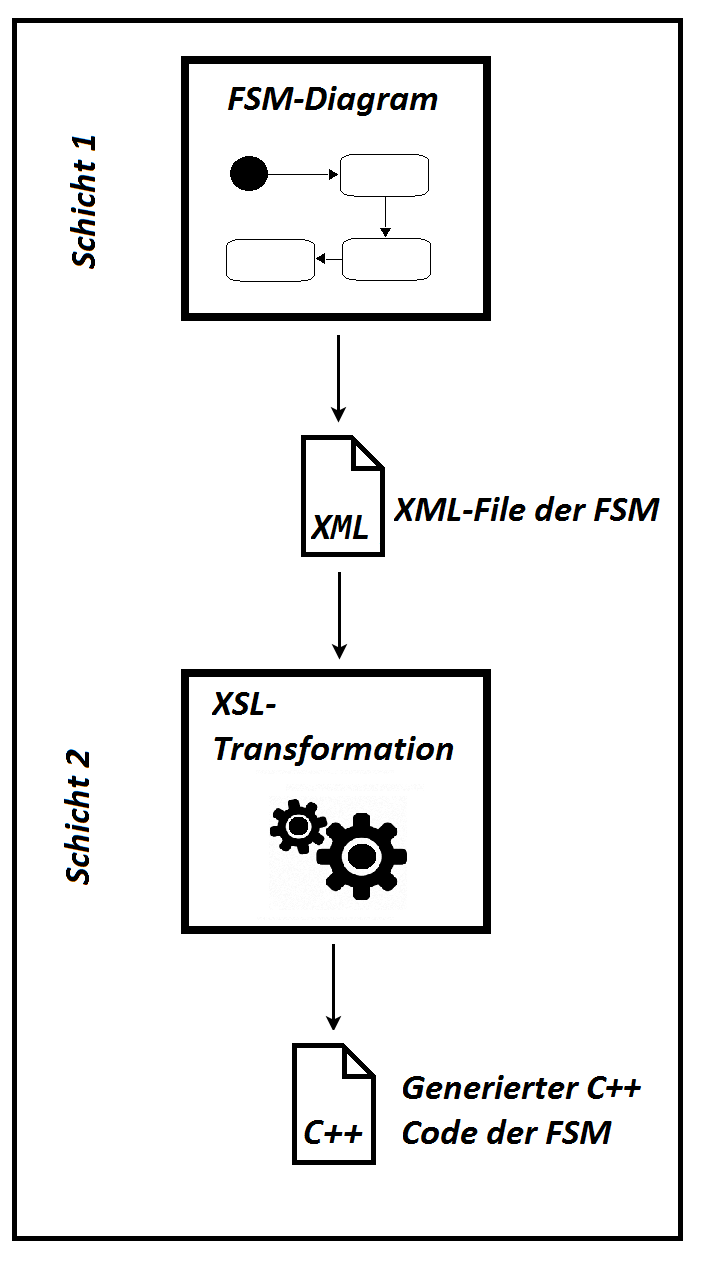


Abbildung 1: Ablauf des Übersetzungsschemas: Von dem Zustandsdiagramm zum generierten C++-Code

## Konzept des GMF-Editors

## Implementierung des GMF-Editors

## Konzeption der XSL-Transformation

In dieser Arbeit wird ein Verfahren benötigt, um aus dem [XML](#_Abkürzungsverzeichnis)-File, den der [FSM](#_Abkürzungsverzeichnis)-Editor bereitstellt (siehe Abbildung 1), den C++-Code zu generieren. Als verfahren wird die [XSL](#_Abkürzungsverzeichnis)-Transformation, kurz [XSLT](#_Abkürzungsverzeichnis), eingesetzt. (<http://www.w3.org/TR/xslt/>),

XSLT

„Die Abkürzung für XSLT steht für eXtendable Stylesheet Language Transformation. XSLT gehört zu den deklarativen Sprachen und beschreibt und steuert die Umwandlung (Transformation) von XML-Dokumenten in andere Formate wie HTML, XHTML, Text oder andere XML-Strukturen.“ (Bongers, 2004)

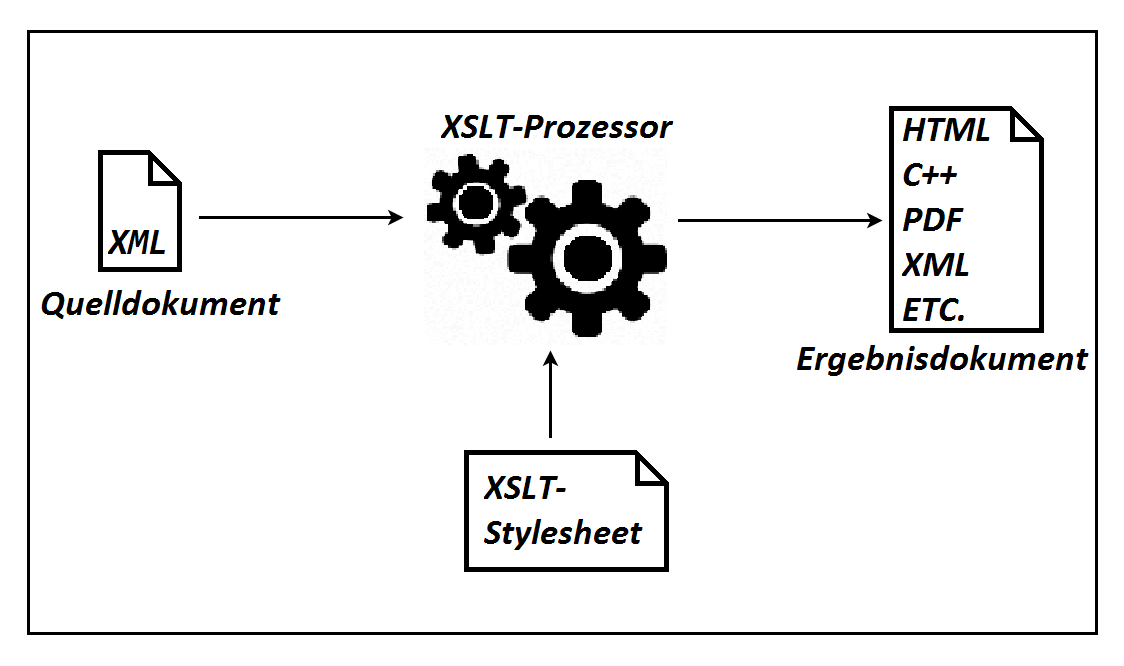


Abbildung 2: Schematische Darstellung einer Verarbeitung mit XSLT

Wie in Abbildung 2 zu sehen ist, beteiligen sich in jeder Transformation drei Dokumente.

1. **Quelldokument**

Das Quelldokument ist ein XML-Dokument, das alle Informationen umfasst und für die Ausführung einer XSL-Transformation notwendig ist (vgl. Bongers, 2004, S.26)

1. **XSLT-Stylesheet**

Dieses Dokument definiert die Umwandlungsregeln und ist für die Ausführung einer Transformation auch erforderlich. Das Stylesheets wird von dem [XSLT-Prozessor](https://de.wikipedia.org/wiki/XSLT-Prozessor) eingelesen und umwandelt anhand von den Umwandlungsregeln, ein oder mehrere XML-Dokumente in das gewünschte Ausgabeformat. (ebd., S. 28)

1. **Ergebnisdokument**

Das Ergebnisdokument lässt sich aus den Informationen des Quelldokumentes als auch die Umwandlungsregeln des XSLT-Stylesheets erzeugen (vgl. Bongers, 2004 S. 28, 29).

Einleitung der XSLT Elemente

Wie in Abschnitt [XSLT](#XSLT_Abschnitt) schon erwähnt, beinhaltet das XSLT-Stylesheet eine Menge von Umwandlungsregeln. Diese werden in der Stylesheet-Hierarchie nach drei unterschiedlichen Kategorien geordnet (ebd., S. 32):

* **Wurzelelemente:**

Die einzigen beiden XSLT-Elemente, die als Wurzelelement eingesetzt werden können, sind xsl:stylesheet oder xsl:transform, in denen die XSLT-Version festgelegt werden muss.

* **Toplevel-Elemente:**

Direkte Kinderelemente des Wurzelelementes werden Toplevel-Elemente genannt. Diese definieren den globalen Aufbau des Stylesheets (wie z.B. Funktionsköpfe, Schablonen (Templates), Formatierungsregeln usw.).

* **Instruktionen:**

Diese Elemente sind den Toplevel-Elementen untergeordnet (Schleifen, Bedingungen, Templatesaufrufe usw.).

Die ausführliche Beschreibung der XSLT-Funktionsweise, wird im nächsten Abschnitt explizit erklärt. Es wird gezeigt wie anhand von XSLT-Befehlen, der FSM-Code generiert wird.

## XSL-Transformation Implementierung

In diesem

(siehe Abschnitt 3.1 auf Seite 32)

# Literaturverzeichnis

**Bongers, Frank. 2004.** *XSLT 2.0.* Bonn : Galileo Press GmbH, 2004. ISBN 3-89842-361-1.

**Consortium, World Wide Web. 2007.** *XSL Transformations (XSLT) Version 2.0.* [Online] W3C Recommendation, 2007. [Zitat vom: 23. 10 2015.] http://www.w3.org/TR/xslt20/.

**Eclipse Foundatioin. 2011.** Graphical Modeling Framework. [Online] 2011. https://www.eclipse.org/gmf-tooling/.

# Glossar

# Abkürzungsverzeichnis

FSM Graphichal Modeling Framework

XSL EXtensible Stylesheet Language

XSLT Extensible Stylesheet Language Transformation

# Abbildungsverzeichnis

[Abbildung 1: Ablauf des Übersetzungsschemas: Von dem Zustandsdiagramm zum generierten C++-Code 5](#_Toc433290812)

[Abbildung 2: Funktionsweise der XSL-Transformation 6](#_Toc433290813)

# Tabellenverzeichnis

# Quellcodeverzeichnis

# Inhalt der beigelegten DVD