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 7](#_Toc433364144)

[Literaturverzeichnis 11](#_Toc433364145)

[Glossar 12](#_Toc433364146)

[Abkürzungsverzeichnis 13](#_Toc433364147)

[Abbildungsverzeichnis 14](#_Toc433364148)

[Tabellenverzeichnis 15](#_Toc433364149)

[Quellcodeverzeichnis 16](#_Toc433364150)

[Inhalt der beigelegten DVD 17](#_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 Aufbau des Übersetzungsschemas 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 (siehe Abschnitt 3.2). 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 wichtigen Informationen für die grafische Darstellung der erstellten FSM. Dadurch ist der FSM-Editor von anderen Plug-ins oder Anwendungen leicht 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 das XML-File in einem C++-File, der den generierten C++-Code der FSM beinhaltet, umzuwandeln. Der entstehende Code, entspricht die FSM\_Switch, die für uns OMNeT++ bereitstellt (siehe Abschnitt XX)

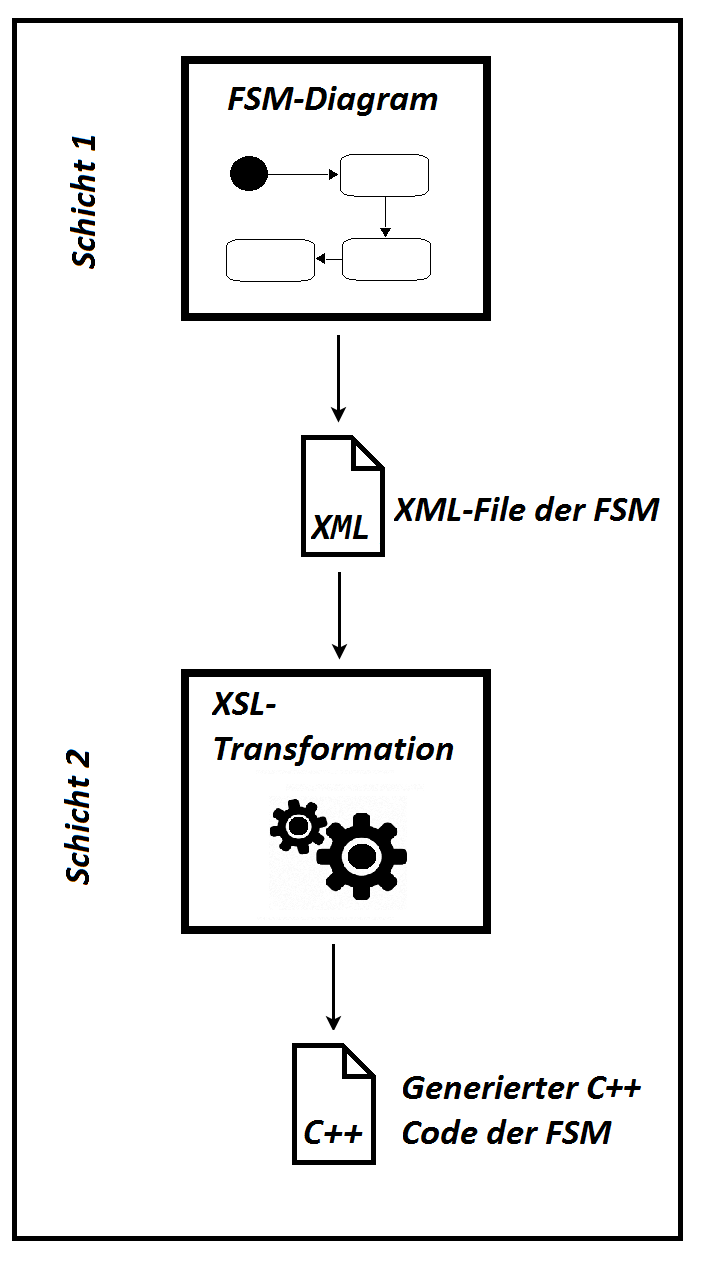


Abbildung 1: Ablauf des Übersetzungsschemas: Von dem Zustandsdiagramm zum generierten C++-Code

## Konzept des GMF-Editors

## Implementierung des GMF-Editors

## Konzeption der XSL-Transformation

Wie in Abbildung 1 zu sehen ist, wird in der zweiten Schicht ein verfahren benötigt, um aus dem [XML](#_Abkürzungsverzeichnis)-File, den der [FSM](#_Abkürzungsverzeichnis)-Editor bereitstellt, den C++-Code zu generieren. Für diesen Zweck wird die [XSL](#_Abkürzungsverzeichnis)-Transformation, kurz [XSLT](#_Abkürzungsverzeichnis), eingesetzt (World Wide Web Consortium, 2007).

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)

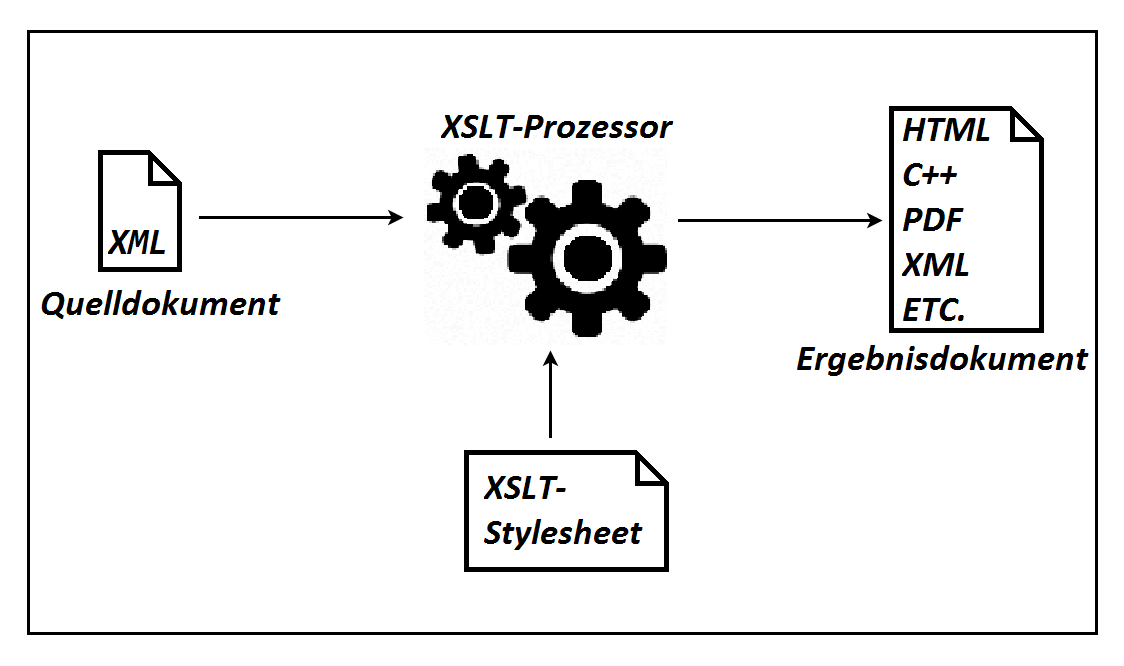


Abbildung 2: Schematische Darstellung einer Verarbeitung mit XSLT

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 den Oben genannten XSLT-Elementen, der C++-Code generiert wird.

## XSL-Transformation Implementierung

Nachdem die Implementierung des FSM-Editors beschrieben wurden ist (siehe [Abschnitt 3.3](#_Implementierung_des_GMF-Editors)), folgt die Beschreibung der Umsetzung der XSL-Transformation, die in der Lage ist einen XML-File einzulesen, um daraus den zugehörigen C++-Code zu generieren. Zuerst wird anhand eines FSM-Diagramms, der mit Hilfe vom FSM-Editor gezeichnet wurde (siehe Abbildung xx), gezeigt, wie so ein XML-File aussieht, gefolgt von einer Beschreibung des XSLT-Stylesheets, der für jede Übersetzung genutzt wird (siehe Abbildung 1). Der letzte Schritt ist die Darstellung und Begründung des resultierenden Files, der den generierten C++-Code enthält.

Beispiel

Als Beispiel betrachten wir ein Netzwerk, der aus zwei Knoten („Tic“ und „Toc“) besteht (siehe Abbildung 3). Diese sind mit einer bidirektionalen Kommunikationsverbindung miteinander verbunden. Einer der beiden Knoten „Tic“ erstellt ein Paket „transferredMsg“ und sendet dies an Toc weiter. Toc modifiziert das Paket und sendet das an Tic wieder zurück usw..

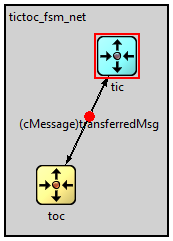


Abbildung 3: Beispiel:-TicToc-Netzwerk

FSM-Diagramm des FSM-Editors

Ab diesem Abschnitt wird nur noch der Knoten Tic betrachtet. Zuerst wird für dieser Knoten alle möglichen Zustanden mit Hilfe vom FSM-Editor gezeichnet.

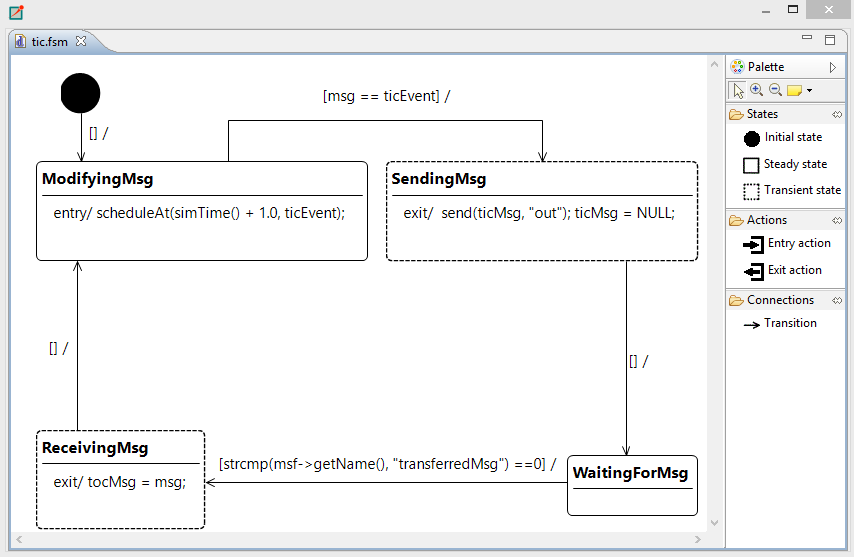


Abbildung 4: Tic-FSM mit dem FSM-Editor gezeichnet

Wie in Abbildung 4 zu sehen ist, kann sich Tic in fünf Zuständen befinden:

1. INIT (Initialstate)
2. ModifyingMsg (Steady State)
3. SendingMsg (Transient State)
4. WaitinForMsg (Steaty State)
5. ReceivingMsg (Transient State)

Beim Eintreten in den Startzustand „ModifyingMsg“, soll ein Timer „ticEvent“ gesetzt werden. Erst beim Ablauf dieses Timers wechselt die FSM in dem „SendingMsg“ Zustand, der ohne Bedingung (Guard) in den nächsten Zustand „WaitingForMsg“ wechselt und beim Verlassen, wird das Paket „transferredMsg“ an Toc versendet. Danach wartet die FSM in dem „WaitingForMsg“ Zustand so lange bis, das Paket „transferredMsg“ von Toc wieder ankommt. Erst dann wechselt die FSM in dem „ReceivingMsg“ Zustand. Nachdem das Paket erhalten wurde, wechselt die FSM wieder in den Startzustand und wiederholt den vorigen Ablauf nochmal.

XML-Code des FSM-Diagramms

Im vorigen Abschnitt wurde das FSM-Diagramm des Knotens Tic mit dem FSM-Editor gezeichnet. Dabei hinterlegt der FSM-Editor ein XML-File. In diesem Abschnitt wird der Aufbau eines solches XML-Files erläutert.



XSLT-Stylesheet

(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