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 14](#_Toc433364145)

[Glossar 15](#_Toc433364146)

[Abkürzungsverzeichnis 16](#_Toc433364147)

[Abbildungsverzeichnis 17](#_Toc433364148)

[Tabellenverzeichnis 18](#_Toc433364149)

[Quellcodeverzeichnis 19](#_Toc433364150)

[Inhalt der beigelegten DVD 20](#_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)

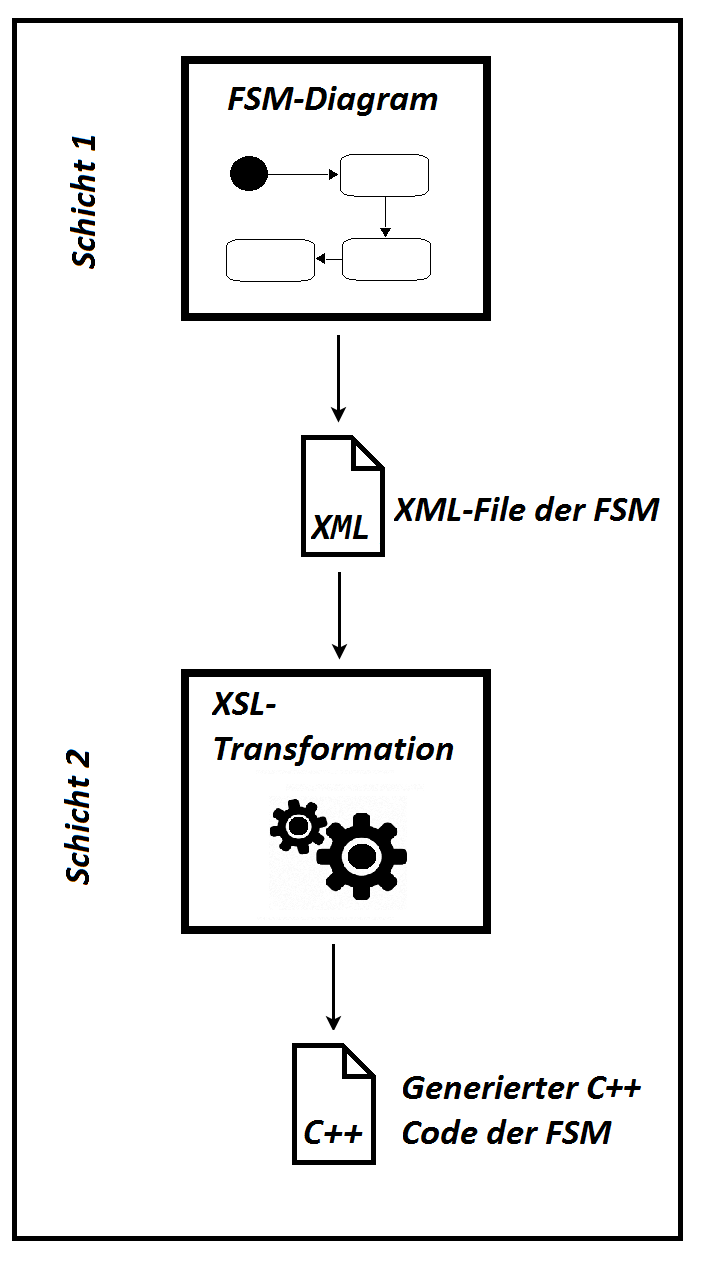


Abbildung 1: Ablauf des Übersetzungsschemas: Von dem Zustandsdiagramm zum generierten C++-Code

## 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 S. 26)

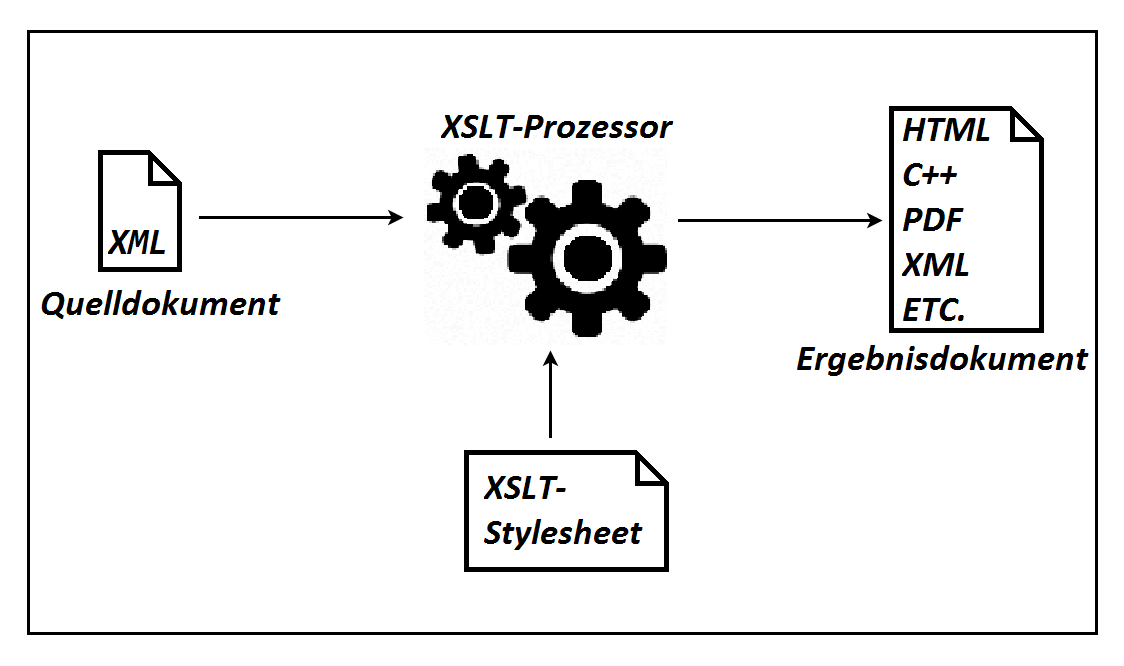


Abbildung 2: Schematische Darstellung einer Verarbeitung mit XSLT

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 Abschnitt (Die Transformation) 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 die 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 es an Tic wieder zurück usw..

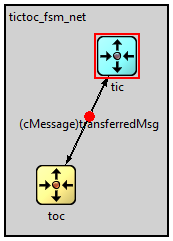


Abbildung 3: Beispiel:-TicToc-Netzwerk

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.

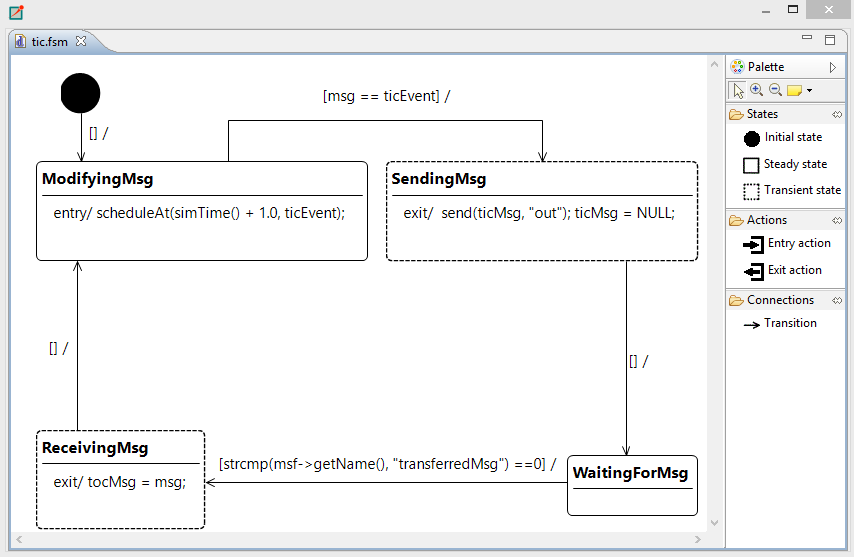


Abbildung 4: Tic-FSM mit dem FSM-Editor gezeichnet

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 dem Startzustand „ModifyingMsg“, soll ein Timer „ticEvent“ gesetzt werden. Beim Ablauf des Timers wechselt die FSM in dem „SendingMsg“ Zustand. Von dem „SendingMsg“ Zustand wechselt die FSM ohne eine Bedingung (Guard) in dem nächsten Zustand „WaitingForMsg“, dabei wird der Exit-Code ausgeführt, der 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 dem Startzustand und wiederholt den vorherigen Ablauf.

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, dessen Aufbau in diesem Abschnitt erläutert wird.



Abbildung 5: XML-Code des FSM-Diagramms "Tic"

Jedes XML-File beginnt mit einem Wurzelknoten. Wie in Abbildung 5 zu sehen ist, entspricht es in diesem Fall der <fsm:FSM> Knoten. Aus Sicht des Knotens <State> bzw. <initialState> ist der Knoten <fsm:FSM> der Elternknoten. Die Knoten <state> und <initalState> können wiederum untergeordnete Knoten (Attribute bzw. Elemente) haben. Da das XML-File auch Daten hält, die für die Transformation nicht relevant sind, werden die relevanten Daten in Abbildung 6 als eine Baumstruktur verständlicher dargestellt.

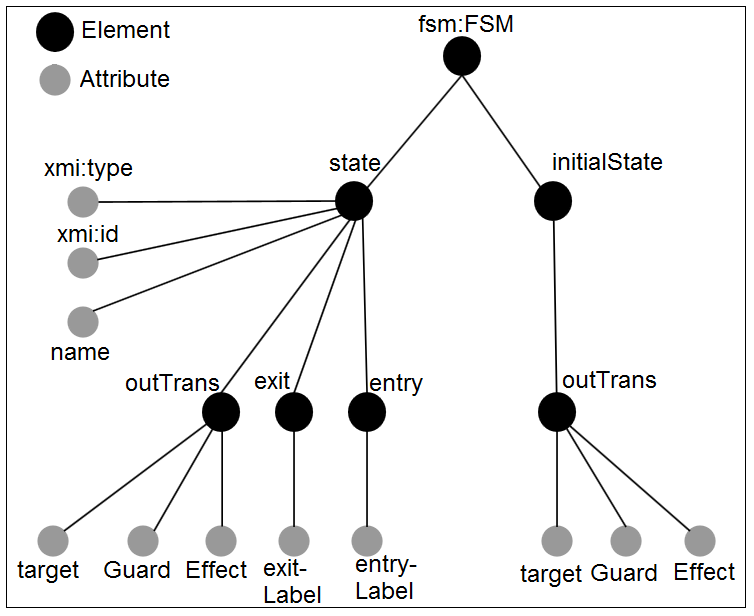


Abbildung 6: "Tic" XML-File als Baumstruktur

Der Knoten <state> hat sechs untergeordnete Knoten (Attribute bzw. Elemente), die für die Transformation relevant sind, nämlich:

* Attribut <xmi:type>: Um welchen Zustandstyp es sich handelt (Steady, Transient).
* Attribut <xmi:id>: Eine eindeutige generierte ID für jeden Zustand.
* Attribut <name>: Zustandsname.
* Element <outTrans> das wiederum drei relevante Attribute hat:
  + Attribut <target>: ID des Ziel-Zustandes.
  + Attribut <Effect>: Übergangseffekt.
  + Attribut <Guard>: Eine Transition kann nur durchlaufen werden, wenn der Wächterausdruck „Guard“ wahr ist.
* Element <entry> der wiederum ein relevantes Attribut hält:
  + Attribut <entryLabel>: Aktionen die beim Eintreten eines Zustandes stattfinden.
* Element <exit> der auch ein relevantes Attribut hält:
  + Attribut <exitLabel>: Aktionen die beim Verlassen eines Zustandes stattfinden.

Bei einem <intialState> Knoten sind die ausgehenden Transitionen und deren Attribute das einzig relevante für die XSL-Transformation, was auch auf Abbildung 6 zu sehen ist.

Die Transformation

Nachdem der Aufbau eines XML-Files, den der FSM-Editor hinterlegt, im vorigen Abschnitt erläutert wurde, wird jetzt eine kleine Transformation durchgeführt, um zu zeigen, wie der C++-Code anhand der Umwandlungsregeln des XSLT-Stylesheets, generiert wird.

Für die Transformation wird ein Codeausschnitt aus dem XSLT-Stylesheet verwendet (siehe Abbildung 7). Dieser generiert den C++-Code, der für die Initialisierung der FSM zuständig ist. dafür muss aus dem XML –File ausgelesen werden:

* Wie viele Zustände vorhanden sind
* Welche Typen diese Zustände haben

Wie in Abschnitt (Einleitung der XSLT Elemente) schon beschrieben wurde, enthält das XSLT-Stylesheet verschiedene Elemente. In Listing 1 (Zeile 1) wird das Element <xsl:template> verwendet. Nach Bongers wird das dies wie folgt beschrieben: „Die Deklaration xsl.template dient zur Definition einer Templateregel bzw. eines benannten Templates. Mit Hilfe des in diesem Template enthaltenen Sequenzkonstrukutors kann in Zusammenhang mit einem verarbeiteten Node ein beliebiger Output in einen Ergebnisbaum erzeugt werden.“ (Bongers, 2004 S. 863)

In diesem Fall wird mit der Anweisung <xsl:template match="fsm:FSM"> der Würzelknoten <fsm:FSM> (siehe Abbildung 6) durch das definierte Template verarbeitet.

Text innerhalb von Templates, wird vom XSLT-Prozessor in das Zieldokument übernommen, das heißt innerhalb des Templates <xsl:template match="fsm:FSM"> werden Zeile 2, 3, 4 und 11 in das Zieldokument übernommen (siehe Listing 2, Zeile A, B C und H).

Mit der Anweisung <xsl:for-each select="state[@xmi:type='fsm:SteadyState']"> (siehe Listing 1, Zeile 5) wird auf jeden „state“ Knoten, der vom Typ Steady ist, der Rumpf dieser for-each Schleife (Zeile 6) angewendet. Der Rumpf kopiert den Name und position des Zustandes

Abbildung 7: Codeausschnitt aus dem XSLT-Stylesheet

1. **<xsl:template match**=*"fsm:FSM"***>**
2. cFSM fsm;
3. enum {
4. INIT = 0,
5. **<xsl:for-each select**=*"state[@xmi:type='fsm:SteadyState']"***>**
6. **<xsl:value-of select**=*"@name"***/>** = FSM\_Steady(**<xsl:value-of select**=*"position()"***/>**),
7. **</xsl:for-each>**
8. **<xsl:for-each select**=*"state[@xmi:type='fsm:TransientState']"***>**
9. **<xsl:value-of select**=*"@name"***/>** = FSM\_Transient(**<xsl:value-of select**=*"position()"***/>**),
10. **</xsl:for-each>**
11. };
12. </xsl:template>

# 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](#_Toc433720855)

[Abbildung 2: Schematische Darstellung einer Verarbeitung mit XSLT 6](#_Toc433720856)

[Abbildung 3: Beispiel:-TicToc-Netzwerk 8](#_Toc433720857)

[Abbildung 4: Tic-FSM mit dem FSM-Editor gezeichnet 9](#_Toc433720858)

[Abbildung 5: XML-Code des FSM-Diagramms "Tic" 10](#_Toc433720859)

[Abbildung 6: "Tic" XML-File als Baumstruktur 11](#_Toc433720860)

# Tabellenverzeichnis

# Quellcodeverzeichnis

# Inhalt der beigelegten DVD