



FIESTA

Modul für TI- COACH

PFLICHTENHEFT

Bachelorpraktikum WS 05/06

Fachgebiet Knowledge Engineering

31. März 2006



Technische Universität
Darmstadt

TEAM WOLFSKIN

Stephan Henkel

Ingo Reimund

Gregor Karzelek

Änderungstabelle

01.11.2005	Wer: Wolfskin Betroffen: Vision	Was: Pflichtenheft
11.11.2005	Wer: Wolfskin Betroffen: Ist,Soll	Was: Pflichtenheft erweitert
18.11.2005	Wer: Tutor Betroffen: Vision, Ist, Soll	Was: Review
23.11.2005	Wer: Wolfskin Betroffen: Änderungstabelle, Featureliste, Rechte, Anhang	Was: Pflichtenheft erweitert
26.11.2005	Wer: Wolfskin Betroffen: Featureliste komplettiert	Was: Pflichtenheft erweitert
07.12.2005	Wer: Wolfskin Betroffen: Ist/Soll	Was: Pflichtenheft überarbeitet
09.12.2005	Wer: Wolfskin Betroffen: Pflichtenheft überarbeitet, Team hinzugefügt	Was: Review durch Gruppe
28.12.2005	Wer: Wolfskin Betroffen: Zeitplanung aktualisiert	Was: Pflichtenheft erweitert
30.12.2005	Wer: Wolfskin Betroffen: Vision, Team, Cover, Änderungstabelle	Was: Pflichtenheft überarbeitet
08.02.2006	Wer: Wolfskin Betroffen: Kapitel Qualitätssicherung erstellt	Was: Kapitel hinzugefügt
09.02.2006	Wer: Wolfskin Betroffen: Zeitplanung aktualisiert	Was: Pflichtenheft überarbeitet
10.02.2006	Wer: Wolfskin Betroffen: Qualitätssicherung	Was: Pflichtenheft überarbeitet
30.03.2006	Wer: Wolfskin Betroffen: Zeitplanung aktualisiert	Was: Pflichtenheft überarbeitet

Inhaltsverzeichnis

1	Das Team	1
2	Vision	2
3	Ist/Soll	3
3.1	Ist	3
3.2	Soll	4
4	Featureliste	5
4.1	Ausbaustufe 0	5
4.2	Ausbaustufe 1	8
5	Use-Cases	9
5.1	Ausbaustufe 0	9
5.2	Ausbaustufe 1	25
6	Prototyp	32
6.1	Oberfläche	32
6.2	Graphische Darstellung	33
6.3	Formale Darstellung	34
6.4	Playback	34
7	Zeitplanung	35
7.1	Terminplan	35
8	Qualitätssicherung	39
8.1	Ziele	39
8.2	Versionsverwaltung	39
8.3	Test-Driven Development	39
8.4	Rücksprache mit dem Auftraggeber	40
8.5	Coding Standards	40
8.6	Testverfahren	40
9	Rechte	42
A	Glossar	43
B	GNU General Public License	45

1 Das Team

Das Projekt, das in diesem Pflichtenheft beschrieben wird, ist im Rahmen des Bachelorpraktikums der Technischen Universität Darmstadt, im folgenden TUD, gestartet worden und wird vom Team „Wolfskin“ durchgeführt. Alle Mitglieder des Teams sind Studenten der TUD und werden im folgenden vorgestellt:

Stephan Henkel

Mailadresse: stephan_henkel@t-online.de
Aufgabenbereich: Verwaltung der Dokumente, UML-Diagramme

Ingo Reimund

Mailadresse: ingo@5id-lan.de
Aufgabenbereich: Organisation

Gregor Karzelek

Mailadresse: gregor@karzelek.com
Aufgabenbereich: GUI-Gestaltung

Trotz der Aufteilung der Teammitglieder in Aufgabenbereiche ist jedes Mitglied in allen Bereichen tätig. Diese Aufteilung zeigt nur, welches Mitglied für welchen Bereich die Hauptverantwortung übernimmt. Desweiteren gibt es noch eine Anzahl verschiedener Tätigkeiten, die jedes Gruppenmitglied in gleicher Weise wahrnimmt, wie z.b. die Erstellung dieses Pflichtenheftes oder die Überarbeitung desselben.

2 Vision

FieStA¹ soll ein Werkzeug für Studenten, Wissenschaftliche Mitarbeiter und Professoren sein, um endliche Automaten graphisch darstellen und die Funktion der dargestellten Automaten zu simulieren.

FieStA besteht aus einem Clientmodul und einem Servermodul. Das Clientmodul nimmt die Benutzereingaben entgegen und ist auch für die graphische Repräsentation zuständig, während das Servermodul eine umfassende Logfunktion beinhaltet.

Der Editor (das Clientmodul) stellt dem Benutzer einfache Werkzeuge zur Verfügung um Automaten auf intuitive Weise erstellen und verändern zu können, kann aber zusätzlich auch eine Datei einlesen und durcharbeiten ('batch') und ein Automaten auf Grund der dortigen Eingaben erstellen. Die Sprache für diese batch-Dateien ist 'absolut' mächtig, was bedeuten soll, dass jeder Aspekt des Automaten damit beschrieben und verändert werden kann. Der Editor kann auch die bisherigen Arbeitsschritte des Nutzers in eine Datei ausgeben, die dann bei Bedarf zu einem späterem Zeitpunkt als batch-Datei eingelesen werden kann. Jeder Handgriff des Benutzers wird mitprotokolliert, kann also sowohl rückgängig gemacht, als auch zu einem späteren Zeitpunkt nachverfolgt werden ('Playback'). Es ist aber auch möglich nur den Automaten ohne Bearbeitungsschritte abzuspeichern oder zu laden. Hat so eine Automatenbeschreibung keinerlei Informationen über die Layoutgestaltung des Automaten, erstellt der Client selbstständig eine einfache, ausreichende grafische Darstellung des Automaten.

Bei der Darstellung des Automaten kann der Benutzer zwischen einer graphischen Repräsentation und einer formalen, sowohl einer tupel- als auch eine tabellenorientierte Darstellung ist möglich, wählen.

Für die Server-Client-Kommunikation wird ein auf TCP/IP aufsetzendes Protokoll entwickelt und benutzt. Dadurch ist man bei der Entwicklung der einzelnen Komponenten nicht auf eine vorgegebene Sprache gebunden, sondern kann, bei Bedarf oder auf Wunsch, den Server oder den Client in einer anderen Sprache nachschreiben, ohne dass Änderungen auf der andern Komponente notwendig wären.

¹Finite State Automaton

3 Ist/Soll

Dieses Kapitel beschäftigt sich mit der Ausgangssituation des Auftraggebers sowie anderen Programmen die ergänzend bzw. alternativ benutzt werden können. Anschließend folgt eine Beschreibung von dem, was FieStA schließlich leisten soll, wobei auf die Details erst in der Featureliste eingegangen wird.

3.1 Ist

Der Auftraggeber verfügt über kein bestehendes System auf dem aufgebaut werden kann, deshalb muss FieStA gänzlich neu entwickelt werden. Es kann jedoch unterstützend auf *JGraph*, ein Framework zur Manipulation von Graphen, und auf eine Sammlung von graphischen Tools genannt *JFlap* zurück gegriffen werden.

JGraph ist ein open-source-Framework, zur graphischen Darstellung und Manipulation von Graphen. Es ermöglicht durch ein einfaches Toolbox ein intuitives Erstellen von Kanten und Knoten und die einfache Manipulation dieser in Größe und Farbe. Selbst das Zurücksetzen von Änderungen ist einfach, da *JGraph* schon eine funktionierende Undo und Redo Funktion mitbringt.

Zusätzlich zu den graphischen Eigenschaften, eignet sich *JGraph* aufgrund der Unabhängigkeit der Datenrepräsentation von der Darstellung, da es in weiten Teilen nach dem MVC²-Pattern entwickelt wurde.

JGraph wird unter der LGPL³ veröffentlicht, kann also für dieses Projekt kostenlos benutzt und verändert werden.

JFlap ist eine Sammlung von graphischen Tools, die sich mit Themen aus der “Formale Sprachen und Automaten”-Theorie befassen. Da diese Toolsammlung sehr umfangreich ist und weit über die Bedürfnisse hinausgeht, ist es nicht besonders geeignet, die gesamte Sammlung in FieStA einfließen zu lassen. Dennoch eignen sich einige Funktionen die es bietet, vorallem da es auf “Formale Sprachen und Automaten” ausgerichtet ist.

Nützlich für FieStA ist das einfache Erstellen von Automaten durch Klicks und Kontextmenüs, sowie das freie Bearbeiten des Layouts eines Automaten durch einfaches Drag’n’Drop. Ebenfalls existiert bereits ein Tool, das ein Wort abarbeitet und dabei jeden einzelnen Zustand des Automaten hervorhebt und so bestimmt ob das Wort Teil der Sprache ist, die der Automat beschreibt.

Zusätzlich ist es möglich in einem weiteren Fenster einen DFA aus einem bereits vorhandenen NFA zu konstruieren. Bei der anschließenden Überprüfung des konstruierten Systemes besteht die Möglichkeit Konstruktionstipps anzeigen zu lassen. Für akademischen Zwecken darf *JFlap* kostenlos benutzt werden.

²Model View Controller

³GNU Lesser General Public License

<http://www.opensource.org/licenses/lgpl-license.php>

3.2 Soll

Ziel von FieStA ist ein Editor für FSA⁴s der dann von dem Auftraggeber in ein größeres Projekt, TI- COACH , eingebettet wird. Der Schwerpunkt liegt deshalb auf der intuitiven Erstellung eines Automaten, was nicht nur ein einfaches Hinzufügen und Entfernen von Kanten und Knoten beinhaltet, sondern auch eine frei wählbare Beschriftung dieser und eine automatische Layoutfunktion. Mit der Layoutfunktion wird erst nach Wunsch des Benutzers der Automat formatiert, wobei darauf geachtet wird, dass es keine Überlappungen von Knoten gibt.

Neben der graphischen Repräsentation kann der Automat auch noch in formaler Tupel-schreibweise erstellt und bearbeitet werden. Dabei kann man zu jeder Zeit zwischen den beiden Ansichten hin und her wechseln.

Der Automat kann in eine Datei geschrieben und wieder aus dieser geladen werden. Dabei wird das Layout des Automaten zunächst automatisch formatiert, kann aber vom Benutzer nach dessen Belieben verändert werden.

Desweiteren kann man in FieStA beliebig viele Schritte des Herstellungsprozesses rückgängig machen.

FieStA ermöglicht auch die Überprüfung, ob ein Wort Teil der Sprache ist, die ein Automat beschreibt und stellt anschließend die Verarbeitung des Wortes von dem Automaten graphisch da, so dass der Benutzer dies leicht verfolgen kann.

⁴Finite State Automat

4 Featureliste

Im diesem Kapitel wird näher auf die einzelnen Features eingegangen, die dem Benutzer bei der Verwendung von FieStA zur Verfügung stehen. Bei der Beschreibung handelt es sich lediglich um eine generelle Vorstellung der einzelnen Funktionen, nicht jedoch um deren Funktionsweise oder exakte Anwendung. Diese wird im Kapitel Use-Cases näher erläutert.

Damit die Zuordnung von einem Feature zu einem Use-Case möglichst einfach ist, hat jedes Feature eine eindeutige Identität mit der auch der entsprechende Use-Case versehen ist. Diese Identität besteht aus einem Buchstaben G für graphisch oder P für das Programm selbst, sowie eine Zahl, die die Ausbaustufe wiedergibt in der dieses Feature hinzugekommen ist. Dem Folgt durch einen Punkt getrennt eine vierstelligen Nummerierung der Feature.

4.1 Ausbaustufe 0

G0.001 Hinzufügen eines Zustands

Priorität: hoch **Status:** implementiert

Der aktuelle Automat wird um einen Zustand erweitert. Hierbei kann der User entweder die Position im Graphen frei bestimmen, oder in der Formalen Darstellung des Automaten einen Zustand erstellen, der beim Wechsel in die graphische Darstellung automatisch positioniert wird.

G0.002 Hinzufügen einer Zustandsübergangs

Priorität: hoch **Status:** implementiert

Der Benutzer kann dem Automaten einen Zustandsübergang hinzufügen und gleichzeitig festlegen, von welchem Eingabezeichen dieser abhängt.

G0.003 Zustandstyp ändern

Priorität: hoch **Status:** implementiert

Der Zustand des Automaten wird als Start- oder Endzustand definiert und erhält im Graphen die entsprechende Representation. Die Auswahl kann ebenso wieder rückgängig gemacht werden.

G0.004 Zustand umbenennen

Priorität: mittel **Status:** implementiert

Das Label eines Zustandes wird abgeändert.

G0.005 Zustandsübergang umbenennen

Priorität: hoch **Status:** implementiert

Das Eingabezeichen eines Zustandsübergangs wird geändert und das Label an der Kante im Graphen angepasst.

G0.006 Entfernen eines Zustandsübergangs

Priorität: hoch **Status:** implementiert

Der Benutzer kann einen Zustandsübergang des Automaten löschen.

G0.007 Entfernen eines Zustands

Priorität: hoch **Status:** implementiert

Ein Zustand kann aus dem Automaten gelöscht werden. Dabei werden alle mit dem Zustand verbundenen Zustandsübergänge ebenfalls gelöscht.

G0.008 Wechsel des Anzeigemodus

Priorität: hoch **Status:** implementiert

Der Benutzer kann zwischen der graphischen und einer Formalen Darstellung des Automaten hin und her schalten. Beim Wechsel von der formalen in die graphische Darstellung werden neu erstellte Zustände automatisch in den Graphen ohne Überlappung eingefügt.

G0.009 Verschieben der Zustände

Priorität: hoch **Status:** implementiert

Der Benutzer kann mit Drag'n'Drop die Position jedes Zustandes verändert die Position eines Zustands und der dazugehörigen Zustandsübergänge.

G0.010 Vereinigen von Knoten

Priorität: hoch **Status:** implementiert

Der Benutzer kann per Drag'n'Drop in der Graphischen Darstellung des Automaten zwei Knoten miteinander vereinigen, indem er sie aufeinanderzieht.

P0.001 Automat laden

Priorität: hoch **Status:** implementiert

Es können bereits existierende Automaten in den Editor geladen werden, um diese anschließend zu bearbeiten.

P0.002 Automat speichern

Priorität: hoch **Status:** implementiert

Erstellte Automaten können in einem XML Baum in eine Datei gesichert werden.

P0.003 Automat erstellen

Priorität: hoch **Status:** implementiert

Im Editor wird ein neues Fenster geöffnet in dem dann ein neuer Automat erstellt werden kann.

P0.004 Änderungen rückgängig machen

Priorität: hoch **Status:** implementiert

Der Benutzer kann beliebig viele Veränderungen am Automaten rückgängig machen oder sie wiederherstellen.

P0.005 Grad der Protokollierung ändern

Priorität: hoch **Status:** implementiert

Der Benutzer ändert die Menge an Daten, die bei der Erstellung und beim weiteren Bearbeiten eines Automaten gespeichert und vom Server verarbeitet werden.

P0.006 Beenden des Programms

Priorität: hoch **Status:** implementiert

Das Programm wird beendet und nach Rückfrage ggf. die Änderungen gespeichert.

4.2 Ausbaustufe 1

G1.001 Autolayout

Priorität: hoch

Status: implementiert

Der Benutzer kann einen erstellten Automatisch neu ordnen lassen, sodaß keine Knoten überlappen

G1.002 Pfadpunkt

Priorität: niedrig

Status: implementiert

Der Benutzer kann einem Zustandsübergang Pfadpunkte hinzufügen. Dies geschieht bei der Erstellung des Zustandsübergangs oder bei späterer Bearbeitung.

G1.003 Playback

Priorität: hoch

Status: implementiert

Bereits erstellte Automaten können in einer Playback Funktion wiedergegeben werden.

G1.004 Wort parsen

Priorität: mittel

Status: implementiert

Der Benutzer kann eine Zeichenkette eingeben, welche dann im Automaten von allen Startzuständen verfolgt wird. So kann überprüft werden, ob ein Wort in der durch den Automaten dargestellten Grammatik enthalten ist.

G1.005 Synchronisierte parallele Ansicht

Priorität: niedrig

Status: implementiert

Der Benutzer kann sich während der Erstellung des Automaten in der graphischen Ansicht parallel die aktuelle formale Ansicht anzeigen lassen.

P1.001 Eingabezeile der formalen Ansicht

Priorität: niedrig

Status: implementiert

In der Formalen Ansicht steht dem Benutzer eine eingabezeile zur Verfügung, mit der alle essentiellen Funktionen zum Erweitern eines Automaten zur Verfügung stehen.

P1.002 Kopierbarer Automat

Priorität: niedrig

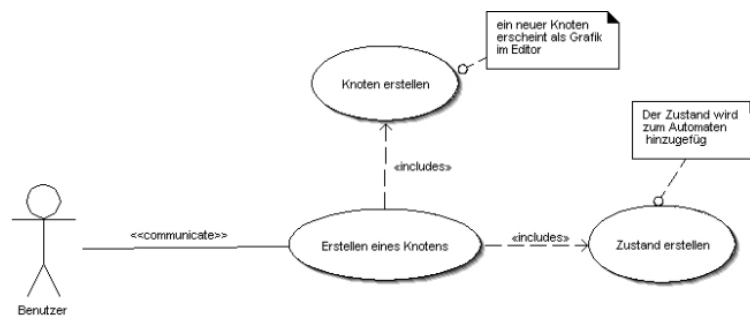
Status: implementiert

Die formale Ansicht eines erstellten Automaten kann komplett markiert werden und per Copy'n'Paste in andere Programme eingefügt werden.

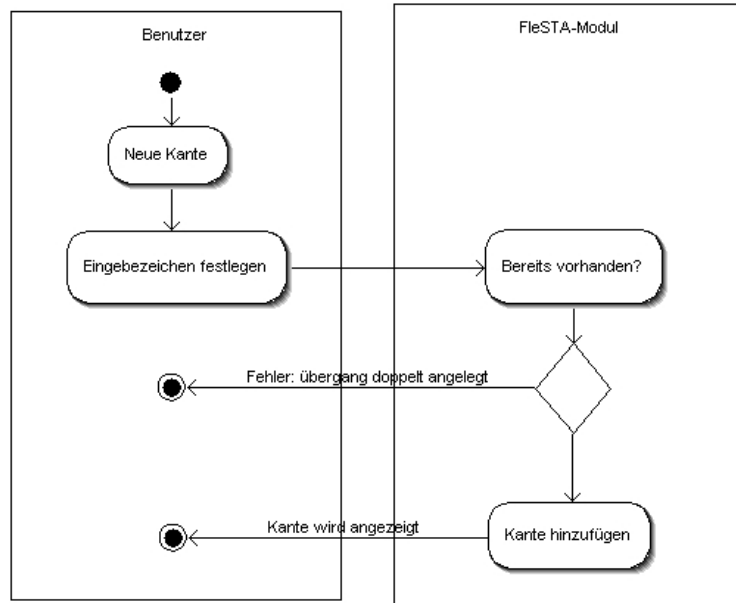
5 Use-Cases

In diesem Kapitel wird die genaue Benutzung der Features von FieStA erklärt. Das ganze ist, wie in der Featureliste beschrieben, mit einer Identität versehen und dadurch mit den einzelnen Features verbunden.

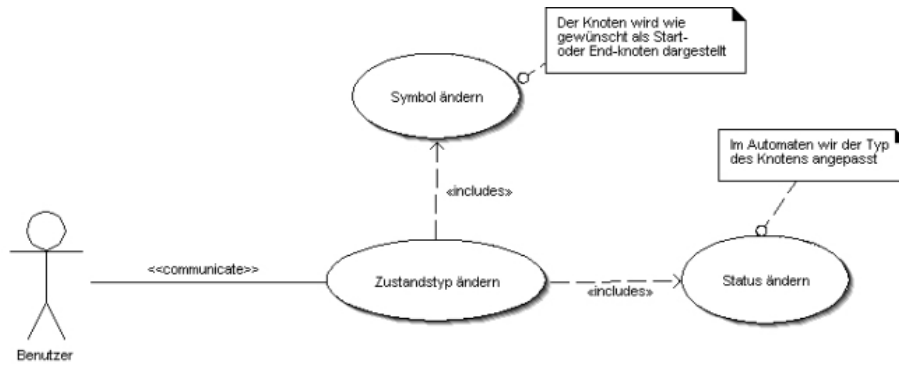
5.1 Ausbaustufe 0



TITEL: Erstellung eines Zustands			ID: G0.001
ZULETZT GEÄNDERT: 22.11.2005	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: niedrig
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Dem Automaten soll ein neuer Zustand hinzugefügt werden.			
VORBEDINGUNGEN: Der FSA-Editor ist geöffnet.			
NACHBEDINGUNGEN: Der aktuelle Automat wurde um einen Zustand erweitert.			
ABLAUF (AKTION): 1a. Der Benutzer aktiviert in der Toolbar die Option „neuer Zustand“ und klickt an die Position an der der Zustand erscheinen soll. 3. der Benutzer gibt eine Bezeichnung für den Zustand ein und bestätigt diese.		ABLAUF (REAKTION): 2. Es erscheint eine Abfrage für die Bezeichnung des Zustandes. 4. An der Stelle auf die der Benutzer geklickt hat erscheint ein neuer Knoten ohne jegliche Zustandsübergänge.	
ALTERNATIV (AKTION): 1b. Der Benutzer Öffnet ein Kontextmenü über einen Rechtsklick und wählt die Option „neuer Knoten“.		ALTERNATIV (REAKTION):	



TITEL: Hinzufügen eines Zustandsübergangs			ID: G0.002
ZULETZT GEÄNDERT: 26.11.2005	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: mittel
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Dem Automaten soll ein neuer Zustandsübergang hinzugefügt werden.			
VORBEDINGUNGEN: Der FSA-Editor ist geöffnet und mindestens 1 Zustand ist vorhanden.			
NACHBEDINGUNGEN: Ein Zustandsübergang wurde zum Automaten hinzugefügt.			
ABLAUF (AKTION): 1. Der Benutzer aktiviert in der Toolbar den Button „Neuer Zustandsübergang“ und klickt zunächst auf den Startzustand. 3. Der Benutzer klickt auf den Endzustand. 5. Der Benutzer gibt die Zeichen für den Zustandsübergang mit einem Trennzeichen getrennt ein		ABLAUF (REAKTION): 2. der Startzustand wird hervorgehoben. 4. Es öffnet sich eine Dialogbox in der die Zeichen des Zustandsübergangs eingegeben werden können. 6. Zwischen den gewählten Zuständen wird ein Zustandsübergang angelegt.	



TITEL: Zustandstyp ändern			ID: G0.003
ZULETZT GEÄNDERT: 26.11.2005	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: niedrig
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Ein Knoten wird als Start- oder Endknoten festgelegt und erh“alt die entsprechende graphische Darstellung			
VORBEDINGUNGEN: Mindestens 1 Zustand vorhanden.			
NACHBEDINGUNGEN: Der gewählte Zustand ist nun als Start- oder Endzustand definiert.			
ABLAUF (AKTION): 1. Der Benutzer klickt auf den Knoten mit der rechten Maustaste. 3. Der Benutzer wählt die gewünschte Option aus.		ABLAUF (REAKTION): 2. Es öffnet sich ein Kontextmenu. 4. Der Zustand verändert seine graphische Markierung nach der gewünschten Option.	

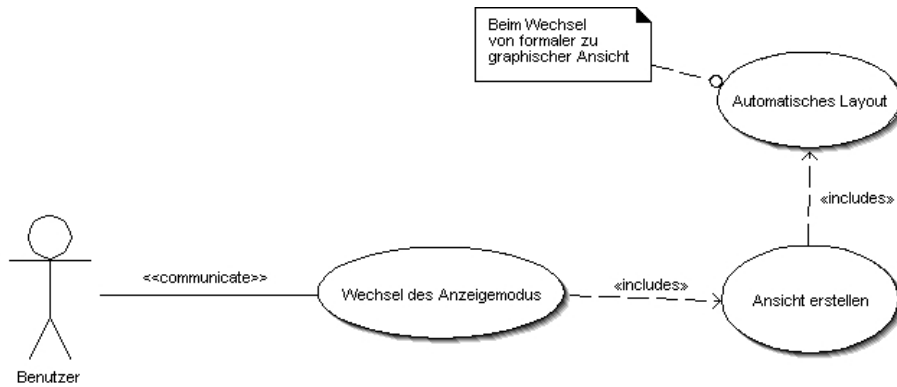


TITEL: Zustand umbenennen			ID: G0.004
ZULETZT GEÄNDERT: 27.11.2005	VERSION: 0.1	PRIORITÄT: mittel	KOMPLEXITÄT: niedrig
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Das Label eines Zustands im Automaten wird wie gewünscht abgeändert			
VORBEDINGUNGEN: Mindestens 1 Zustand vorhanden			
NACHBEDINGUNGEN: Der Name des gewählten Zustandes wurde verändert			
ABLAUF (AKTION): 1a. Der Benutzer klickt in der Toolbar auf den Button „Bearbeiten“ und anschließend auf den Zustand der bearbeitet werden soll. 3. Der Benutzer gibts einen Namen für den Zustand ein und schliesst seine Eingabe mit 'Enter' ab.		ABLAUF (REAKTION): 2. Es öffnet sich ein Dialog in der ein neuer Name eingegeben werden kann. 4. Der Zustand wird im Editor, mit dem neuen Namen bezeichnet, dargestellt.	
ALTERNATIV (AKTION): 1b. Der Benutzer öffnet mit einem Rechtsklick ein Kontextmenü auf dem gewünschten Zustand und wählt die Aktion „Bearbeiten“.		ALTERNATIV (REAKTION):	

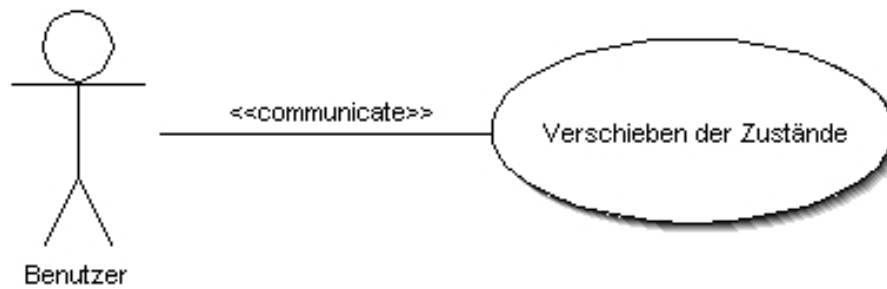
TITEL: Bezeichnung eines Zustandsübergangs ändern		ID: G0.005	
ZULETZT GEÄNDERT: 27.11.2005	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: niedrig
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Das Label einer Kante im Automaten wird ge“ändert.			
VORBEDINGUNGEN: Es existiert mindestens ein Zustandsübergang.			
NACHBEDINGUNGEN: Der Zustandsübergang wurde umbenannt.			
ABLAUF (AKTION): 1a. Der Benutzer klickt in der Toolbar auf den Button „Bearbeiten“ und anschließend auf den Zustandsübergang der bearbeitet werden soll. 3. Der Benutzer gibts einen Namen für den Zustandsübergang ein und schliesst seine Eingabe mit 'Enter' ab.		ABLAUF (REAKTION): 2. Es öffnet sich ein Dialog, in der ein neuer Name eingegeben werden kann. 4. Der Zustandsübergang wird im Editor, mit dem neuen Namen bezeichnet, dargestellt.	
ALTERNATIV (AKTION): 1b. Der Benutzer öffnet mit einem Rechtsklick ein Kontextmenü auf dem gewünschten Zustandsübergang und wählt die Aktion „Bearbeiten“.		ALTERNATIV (REAKTION):	

TITEL: Entfernen eines Zustandsübergangs			ID: G0.006
ZULETZT GEÄNDERT: 27.11.2005	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: niedrig
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Ein Zustandsübergang wird aus dem aktuellen Automaten entfernt, Die Kante wird aus dem Graph gelöscht.			
VORBEDINGUNGEN: Es existiert mindestens ein Zustandsübergang.			
NACHBEDINGUNGEN: Der ausgewählte Zustandsübergang wurde aus dem Automaten entfernt.			
ABLAUF (AKTION): 1a. Der Benutzer klickt in der Toolbar auf den Button „Löschen“ und anschließend auf den Zustandsübergang der entfernt werden soll. 2a. Der Benutzer klickt in der Toolbar auf den Button Löschen.		ABLAUF (REAKTION): 2. Der Zustandsübergang wurde entfernt. 3. Der ausgewählte Zustandsübergang wird aus dem Automaten entfernt.	
ALTERNATIV (AKTION): 2b. Der Benutzer öffnet per Rechtsklick ein Kontextmenü auf dem gewünschten Zustandsübergang und wählt die Aktion Löschen.		ALTERNATIV (REAKTION):	

TITEL: Entfernen eines Zustands			ID: G0.007
ZULETZT GEÄNDERT: 27.11.2005	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: niedrig
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Ein Zustand samt zugehöriger Zustandsübergänge wird aus dem Automaten entfernt.			
VORBEDINGUNGEN: Es existiert mindestens ein Zustand.			
NACHBEDINGUNGEN: Der ausgewählte Zustand samt zugehöriger Zustandsübergänge wurde entfernt.			
ABLAUF (AKTION): 1a. Der Benutzer klickt in der Toolbar auf den Button „Löschen“ und anschließend auf den zu entfernenden Zustand.		ABLAUF (REAKTION): 2. der Zustand mit seinen Zustandsübergängen wird aus dem Automaten entfernt.	
ALTERNATIV (AKTION): 1b. Der Benutzer öffnet per Rechtsklick ein Kontextmenü auf dem gewünschten Zustand und wählt die Aktion Löschen.		ALTERNATIV (REAKTION):	



TITEL: Wechsel des Anzeigemodus			ID: G0.008
ZULETZT GEÄNDERT: 27.11.2005	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: mittel
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Der Anzeigemodus(Tupelform/graphische Ansicht) wird gewechselt.			
VORBEDINGUNGEN: Der FSA-Editor ist geöffnet.			
NACHBEDINGUNGEN: Der Editor ist im gewünschten Modus geöffnet und der Automat kann dort bearbeitet werden.			
ABLAUF (AKTION): 1. Der Benutzer wählt im Menü den Eintrag "Ansicht" aus und wählt dort die gewünschte Ansicht aus.		ABLAUF (REAKTION): 2. Die Ansicht des Editors wird in die gewählte Ansicht gewechselt.	



TITEL: Verschieben der Zustände			ID: G0.009
ZULETZT GEÄNDERT: 27.11.2005	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: niedrig
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Der Benutzer verändert die Position eines Zustands und der dazugehörigen Zustandsübergänge.			
VORBEDINGUNGEN: Ein Automat besteht, der Editor befindet sich im graphischen Anzeigemodus.			
NACHBEDINGUNGEN: Die Position eines Zustandes im Graphen wurde geändert, alle verbundenen Kanten wurden entsprechend angepasst.			
ABLAUF (AKTION): 1. Der Benutzer klickt auf einen Zustand, hält die Maustaste gedrückt und bewegt die Maus. 3. Der Benutzer lässt die Taste los.		ABLAUF (REAKTION): 2. Der Zustand folgt der Mausbewegung mit einschließlich seinen Zustandsübergängen. 4. Der Zustand wird an der neuen Position abgelegt.	

TITEL: Vereinigen von Zuständen (graphisch)			ID: G0.010
ZULETZT GEÄNDERT: 27.11.2005	VERSION: 0.1	PRIORITÄT: mittel	KOMPLEXITÄT: mittel
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Zwei Knoten des Automaten werden verschmolzen			
VORBEDINGUNGEN: Es existieren mindestens 2 Zustände im Automaten, graphischer Bearbeitungsmodus ist angewählt			
NACHBEDINGUNGEN: Zwei Zustände des Automaten wurden zusammengelegt und die entsprechenden Zustandsübergänge angepasst.			
ABLAUF (AKTION): 1. Der Benutzer wählt in der Toolbar den Button „Zustände vereinigen“ aus und klickt anschließend einen der Zustände an die vereinigt werden soll. 3. Der Benutzer wählt nun den Zustand aus, mit dem er den bereits Markierten vereinigen möchte und klickt ebenfalls auf diesen.		ABLAUF (REAKTION): 2. der ausgewählte Zustand wird farblich hervorgehoben. 4. Die gewählten Zustände werden zu einem Zustand vereinigt und die Zustandsübergänge dementsprechend angepasst.	

TITEL: Automat laden			ID: P0.001
ZULETZT GEÄNDERT: 27.11.2005	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: mittel
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Es wird ein bereits vorhandener Automat aus einer Datei geladen.			
VORBEDINGUNGEN: keine			
NACHBEDINGUNGEN: Der gewählte Automat wurde geöffnet und kann nun im Editor weiter bearbeitet werden.			
ABLAUF (AKTION): 1a. Der Benutzer startet den Editor und öffnet das Menü Datei und wählt die Option öffnen. 3. Der Benutzer wählt im Dateibrowser die gewünschte Datei und bestätigt seine Wahl.		ABLAUF (REAKTION): 2. Es öffnet sich ein Dateibrowser. 4a. Der Automat wird geladen und im Editor dargestellt.	
ALTERNATIV (AKTION): 1b. Auf der Kommandozeile wird eine Eingabefile mit angeben.		ALTERNATIV (REAKTION): 4b. Der Editor wird mit dem in der Eingabefile gespeicherten Automaten geöffnet	

TITEL: Automat speichern			ID: P0.002
ZULETZT GEÄNDERT: 27.11.2005	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: niedrig
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Es wird ein erstellter Automat abgespeichert.			
VORBEDINGUNGEN: Ein Automat ist im Editor geöffnet.			
NACHBEDINGUNGEN: Der Aktuelle Automat wurde in einer Datei gespeichert			
ABLAUF (AKTION): 1. Der Benutzer öffnet das Menü Datei und wählt die Option Speichern. 3. Der Benutzer wählt im Dateibrowser die gewünschte Datei oder gibt einen Namen für eine neue Datei an.		ABLAUF (REAKTION): 2. Es öffnet sich ein Dateibrowser 4. Der Zuvor erstellte Automat wird gespeichert.	

TITEL: Automat erstellen			ID: P0.003
ZULETZT GEÄNDERT: 27.11.2005	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: niedrig
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Im Editor wird ein neuer leerer Automat erstellt.			
VORBEDINGUNGEN: Das Programm wird geöffnet.			
NACHBEDINGUNGEN: Ein leeres Editorfenster ist geöffnet.			
ABLAUF (AKTION): 1. Der Benutzer öffnet das Menü Datei und wählt die Option Neu		ABLAUF (REAKTION): 2. Es öffnet sich ein neues Editorfenster, in dem nun ein neuer Automat angelegt werden kann.	

TITEL: Änderungen rückgängig machen		ID: P0.004	
ZULETZT GEÄNDERT: 11.02.2006	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: niedrig
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Der letzte vom Benutzer durchgeführte "Änderungsschritt" (bezüglich des Automaten) wird rückgängig gemacht.			
VORBEDINGUNGEN: Automat ist geöffnet und mindestens eine Aktion auf dem Automaten wurde ausgeführt.			
NACHBEDINGUNGEN: Der letzte Änderungsschritt wurde rückgängig gemacht.			
ABLAUF (AKTION): 1a. Der Benutzer klickt in der Toolbar auf den Button „Rückgängig“.		ABLAUF (REAKTION): 2. Der letzte Änderungsschritt bei der Erstellung des Automaten wird rückgängig gemacht	
ALTERNATIV (AKTION): 1b. Der Benutzer wählt in dem Menü den Eintrag "Bearbeiten" aus und klickt dort auf Rückgängig		ALTERNATIV (REAKTION):	

TITEL: Grad der Protokollierung ändern		ID: P0.005	
ZULETZT GEÄNDERT: 11.02.2006	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: mittel
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Der Benutzer ändert die Menge an Daten die mitprotokolliert werden.			
VORBEDINGUNGEN: Das Programm ist geöffnet			
NACHBEDINGUNGEN: Nach der Änderung des Protokollierungsgrades werden nur noch die von dem neuem Grad betroffenen Daten mitprotokolliert.			
ABLAUF (AKTION): 1. Der Benutzer öffnet das Menü „Editor“ und wählt die Option „Protokollierungslevel ändern“. 3. Der Benutzer wählt den gewünschten Protokollierungsgrad aus einer Liste und bestätigt seine Eingabe.		ABLAUF (REAKTION): 2. Es erscheint ein Dialogfenster in dem der Protokollierungsgrad gewählt werden kann. 4. Die Änderungen beim Erstellen und erweitern des Automaten werden nur noch wie gewünscht protokolliert.	

TITEL: Beenden des Programms			ID: P0.006
ZULETZT GEÄNDERT: 27.11.2005	VERSION: 0.1	PRIORITÄT: hoch	KOMPLEXITÄT: niedrig
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Das Programm wird beendet.			
VORBEDINGUNGEN: Programm ist geöffnet.			
NACHBEDINGUNGEN: Das Programm wurde beendet.			
ABLAUF (AKTION): 1. Der Benutzer öffnet das Menü „TI- COACH “ und wählt die Option „Programm beenden“.		ABLAUF (REAKTION): 2. Es öffnet sich ein Dialogfenster, in dem der Benutzer bestätigt, dass das Programm geschlossen werden kann. Sollte ein Automat noch nicht gespeichert sein, so wird er Benutzer daraufhin aufmerksam gemacht. 3. Das Programm wird beendet und alle eventuell vorhandenen Fenster werden geschlossen.	

5.2 Ausbaustufe 1

TITEL: Autolayout			ID: G1.001
ZULETZT GEÄNDERT: 01.03.2006	VERSION: 0.9	PRIORITÄT: hoch	KOMPLEXITÄT: hoch
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Die Zustände werden nicht überschneidet angeordnet			
VORBEDINGUNGEN: Ein Editor ist geöffnet und es sind Zustände vorhanden.			
NACHBEDINGUNGEN: Die Zustände sind so angeordnet, dass sie sich nicht überschneiden.			
ABLAUF (AKTION): 1. Der Benutzer wählt im Menueintrag "Editor" den Eintrag "Autolayout".		ABLAUF (REAKTION): 2. Die vorhandnen Zustände werde so angeordnet, dass es keine Überschneidungen gibt.	

TITEL: Pfadpunkte		ID: G1.002	
ZULETZT GEÄNDERT: 01.03.2006	VERSION: 0.9	PRIORITÄT: niedrig	KOMPLEXITÄT: mittel
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Einem Zustandsübergang werden Pfadpunkte hinzugefügt			
VORBEDINGUNGEN: Es ist mindestens ein Zustandsübergang vorhanden oder es wird ein Zustandsübergang neu erstellt.			
NACHBEDINGUNGEN: Der Zustandsübergang hat nun einen Pfadpunkt.			
ABLAUF (AKTION): 1a. Der Benutzer öffnet das Kontext über einem Zustandsübergang und wählt dort den Eintrag "Pfadpunkt hinzufügen".		ABLAUF (REAKTION): 2a. Dem Zustandsübergang wird ein Pfadpunkt hinzugefügt.	
ALTERNATIV (AKTION): 1b. Der Benutzer erstellt wie in dem Use-Case "Hinzufügen eines Zustandsübergangs" einen Zustandsübergang und klickt bevor er den Endzustand markiert an die Stelle, an der der Pfadpunkt hinzugefügt werden soll und abschließend auf den Endzustand des Zustandsübergangs.		ALTERNATIV (REAKTION): 2b. Ein neuer Zustandsübergang wird erstellt und enthält bereits den hinzugefügten Pfadpunkt.	

TITEL: Playback			ID: G1.003
ZULETZT GEÄNDERT: 01.03.2006	VERSION: 0.9	PRIORITÄT: hoch	KOMPLEXITÄT: mittel
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Das Erstellen des Automaten kann abgespielt werden.			
VORBEDINGUNGEN: Es wurde mindestens eine Aktion auf dem Editor ausgeführt.			
NACHBEDINGUNGEN: Der Benutzer kann sich das Erstellen des Automaten ansehen.			
ABLAUF (AKTION): 1. Der Benutzer wählt in dem Menüeintrag "Editor" den Eintrag "Playback". 3. Der Benutzer klickt nun auf den "Play" Button in der Toolbar.		ABLAUF (REAKTION): 2. Es öffnet sich im Editor der Playbackmodus. 4. Das Playback wird automatisch abgespielt.	

TITEL: Wort parsen			ID: G1.004
ZULETZT GEÄNDERT: 01.03.2006	VERSION: 0.9	PRIORITÄT: mittel	KOMPLEXITÄT: mittel
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Es kann überprüft werden, ob der Automat ein bestimmtes Wort akzeptiert.			
VORBEDINGUNGEN: Ein Automat der ein Wort akzeptiert.			
NACHBEDINGUNGEN: Ein Ablauf wie der Automat das Wort akzeptiert.			
ABLAUF (AKTION): 1. Der Benutzer wählt in dem Menüeintrag "Editor" den Eintrag "Wort parsen". 3. Der Benutzer kann in der Eingabezeile die Zeichen des Wortes getrennt durch ein Trennzeichen eingeben. Anschließend klickt der Benutzer auf den Button "Übernehmen" und startet das abspielen durch den Button "Play"		ABLAUF (REAKTION): 2. Es öffnet sich im Editor das Wortparsen-Modus. 4. Es wird abgespielt wie der Automat das Wort akzeptiert.	

TITEL: Synchronisierte parallele Ansicht			ID: G1.005
ZULETZT GEÄNDERT: 01.03.2006	VERSION: 0.9	PRIORITÄT: niedrig	KOMPLEXITÄT: niedrig
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Graphische und formale Ansicht wird gleichzeitig angezeigt.			
VORBEDINGUNGEN: Der Editor ist geöffnet.			
NACHBEDINGUNGEN: Ein Automat wird in graphischer und formaler Ansicht angezeigt.			
ABLAUF (AKTION): 1. Der Benutzer wählt in dem Menüeintrag "Editor" den Eintrag "Parallel".		ABLAUF (REAKTION): 2. Der Automat wird in der formalen und der graphischen Ansicht angezeigt.	

TITEL: Eingabezeile der formalen Ansicht			ID: P1.001
ZULETZT GEÄNDERT: 01.03.2006	VERSION: 0.9	PRIORITÄT: niedrig	KOMPLEXITÄT: mittel
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Der Benutzer kann den Automaten in der formalen Ansicht bearbeiten.			
VORBEDINGUNGEN: Die formale Ansicht ist geöffnet.			
NACHBEDINGUNGEN: Der Automat wurde bearbeitet.			
ABLAUF (AKTION): 1. Der Benutzer gibt einen Befehl in die Eingabezeile ein.		ABLAUF (REAKTION): 2. Der Automat zeigt die Änderung direkt an.	

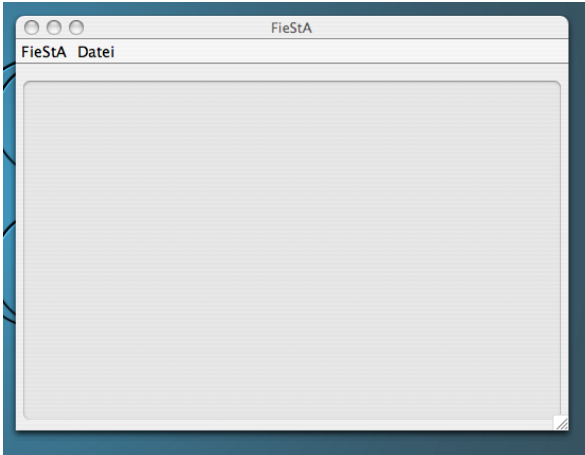
TITEL: Kopierbarer Automat			ID: G1.002
ZULETZT GEÄNDERT: 01.03.2006	VERSION: 0.9	PRIORITÄT: hoch	KOMPLEXITÄT: mittel
AKTEURE: Benutzer			
KURZBESCHREIBUNG: Der Automat kann in einen externen Editor eingebunden werden.			
VORBEDINGUNGEN: Die formale Ansicht ist geöffnet.			
NACHBEDINGUNGEN: Der Text ist in einem externen Editor eingefügt.			
ABLAUF (AKTION): 1. Der Benutzer klickt in die formale Ansicht und hält die Maustaste bei der Bewegung gedrückt. 3. Der Benutzer lässt die Maustaste los und beendet das selektieren. 5. Der Benutzer wählt den externen Editor und fügt den Text ein.		ABLAUF (REAKTION): 2. Der Text in der formalen Ansicht wird markiert. 4. Der Text wird in die zwischenablage des Betriebssystem kopiert. 6. Der Text erscheint in dem externen Editor.	

6 Prototyp

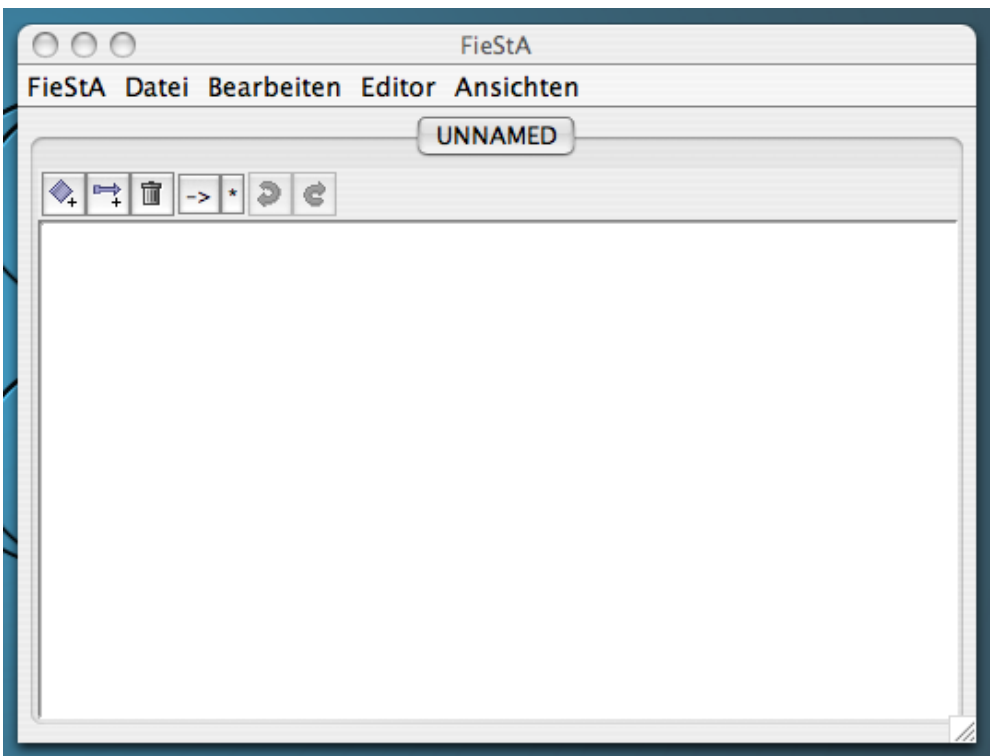
In diesem Kapitel stellen wir den Prototyp des Editor, der im Rahmes des Bachelorprojekts erstellt werden soll, vor.

6.1 Oberfläche

Beim starten des Programm wird zunächst die reine Applikation angezeigt.

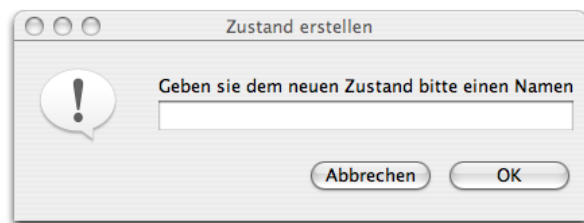


In die Application wird anschließend durch das Erstellen eines neuen Editor oder durch das Laden eines Automaten ein Editorfenster geöffnet.

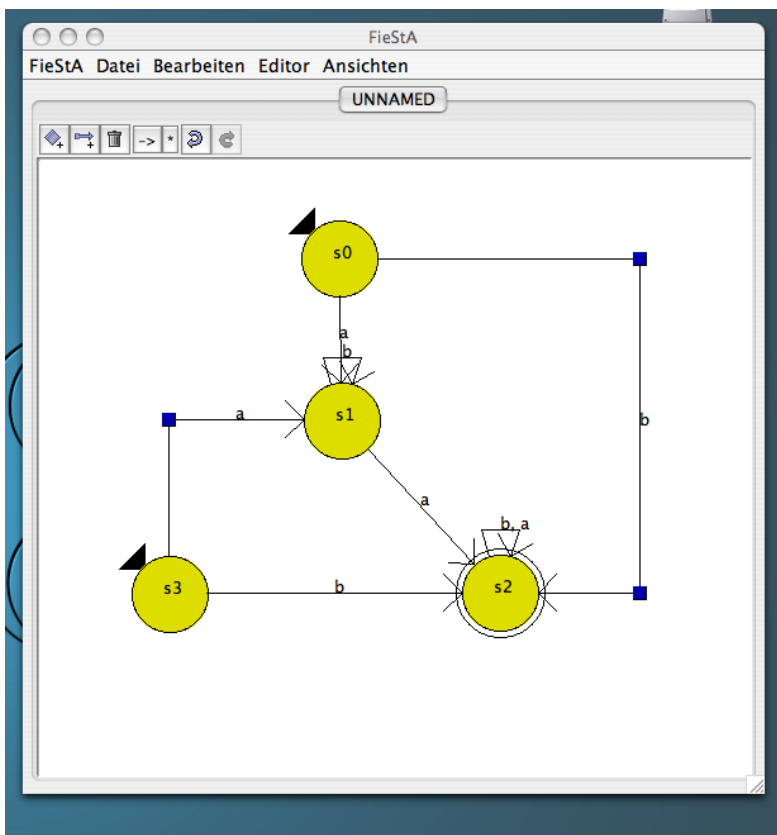


6.2 Graphische Darstellung

In dem nun geöffneten Editor können nun Zustände hinzugefügt werden. Damit keine ungewünschten Aktionen ausgeführt werden, kann der Benutzer die Option Zustand hinzufügen aktivieren. Nun erzeugt jeder Klick im Editor einen neuen Zustand, für den ein Dialogfenster zur Eingabe des Names geöffnet wird.

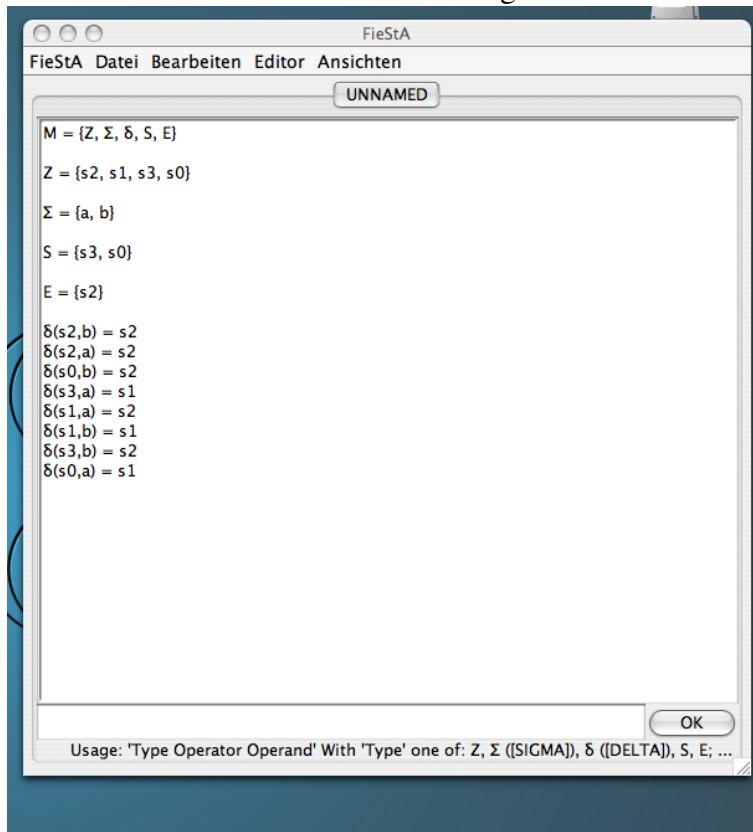


Auf gleich weise können auch Transitions hinzugefügt werden, wodurch schnell komplexere Automaten erstellt werden können. Bereits erstellte Zustände können auch nachträglich manipuliert werden, so daß der Namen verändert werden kann, oder der Typ des Zustandes in einen Start oder Endzustand geändert werden kann. Ebensoleicht können die Zustandsübergänge um weitere Zeichen erweitert werden.



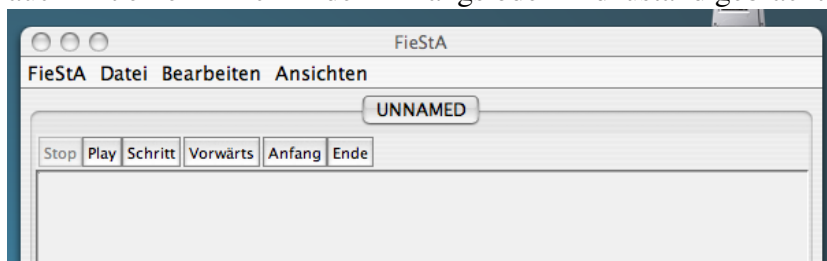
6.3 Formale Darstellung

Die Ansicht kann bei Bedarf auf eine formale Darstellung umgestellt werden. Diese bietet dem Benutzer ein bekanntes Bild, da sie sich stark an der formalen Beschreibung von NFAs, wie sie an der TU–Darmstadt gelehrt werden orientiert. Es soll auch möglich sein den Autoamten mit der Hilfe einer Eingabezeile zu bearbeiten.



6.4 Playback

Ein erstellter Automat kann in eine Playbackansicht gebracht werden. In dieser Ansicht soll es möglich sein, die Erstellungsschritte des Automaten schrittweise durch einfache clicks abzuspielen. Alternativ wird ein Autoplayback ausführbar sein, das die gesamte Erstellung in einem nachvollziehbaren Tempo abspielt. Auf Wunsch kann der Automat auch mit einem Klick in den Anfangs oder Endzustand gebracht werden.



7 Zeitplanung

Dieses Kapitel beschäftigt sich mit der Zeitplanung der Gruppe Wolfskin. Darin zu finden ist der Terminplan mit erfüllten und verfehlten Termin.

7.1 Terminplan

Im Folgenden wird die Terminplanung der Gruppe behandelt. Diese beinhaltet alle festen Termine, die durch die Projektbegleitung vorgegeben ist, so wie die von der Gruppe selbst gesetzten Vorgaben und Ziele.

Zu jedem Termin ist der Terminggeber sowie die Aufgabe vermerkt. Dazu kommt der aktuelle Status, der zwischen „erfüllt“ und „verfehlt“ unterscheidet, und eine Notiz über Probleme bzw. Gründe für das Verfehlen der Aufgabe. Der zusätzlich Status „wartend“ beschreibt einen noch ausstehenden Termin.

16.11.2005

Vorgabe von: Wolfskin	Status: verfehlt
Ziel: Fertigstellung der Abgabe 0 mit Vision, Ist/Soll und Featureliste	Notiz: Ist/Soll deckt noch nicht das ganze Kernsystem ab

18.11.2005

Vorgabe von: Projektbegleitung	Status: erfüllt
Ziel: Abgabe der 0 Version des Pflichtenheftes mit Vision, Ist/Soll und Featureliste	Notiz: Die Featureliste sollte eigenständig sein, ist aber mit in das Ist/Soll eingeflossen

30.11.2005

Vorgabe von: Wolfskin	Status: verfehlt
Ziel: Fertigstellung der Usecases für die Abgabe 1	Notiz: Es fehlen noch einpaar Usecases

05.12.2005

Vorgabe von: Wolfskin	Status: verfehlt
Ziel: Fertigstellung der 1 Version des Pflichtenheftes	Notiz: Featureliste und Usecases müssen noch überarbeitet werden, da diese noch nicht genügend Informationen beinhalten.

09.12.2005

Vorgabe von: Projektbegleitung	Status: erfüllt
Ziel: Abgabe der 1 Version des Pflichtenheftes mit der Abdeckung des ganzen Kernsystems	Notiz: Es wurden nur einpaar Use-Case-Diagramme eingebunden

13.12.2005

Vorgabe von: Wolfskin	Status: erfüllt
Ziel: Fertigstellung der Präsentation für die HDA-Schulung	Notiz: keine Probleme

15.12.2005

Vorgabe von: Wolfskin	Status: erfüllt
Ziel: Testpräsentation vor der ganzen Gruppe	Notiz: Präsentation wurde verfeinert und von Fehler befreit

16.12.2005

Vorgabe von: Projektbegleitung	Status: erfüllt
Ziel: Review des Pflichtenheftes durch den Auftraggeber	Notiz: Präsentation war erfolgreich

16.12.2005

Vorgabe von: Projektbegleitung	Status: erfüllt
Ziel: 2 Block Projektbegleitung Design	Notiz: keine Probleme

09.01.2006

Vorgabe von: Wolfskin	Status: verfehlt
Ziel: Fertigstellung der Design-Dokumentation des Kernsystems	Notiz: UML-Diagramme erstellt

12.01.2006

Vorgabe von: Projektbegleitung	Status: erfüllt
Ziel: Präsentationstraining durch die HDA	Notiz: Ursprungstermin: 16.12.2005. Der Termin wurde wegen Überschneidung mit der Projekt-Review auf den 12.01.2006 verlegt.

13.01.2006

Vorgabe von: Projektbegleitung	Status: erfüllt
Ziel: Abgabe der Architekturbeschreibung und der Design–Dokumentation des Kernsystems	Notiz: keine Probleme

14.01.2006

Vorgabe von: Wolfskin	Status: erfüllt
Ziel: Begin mit der eigentlichen Programmierung	Notiz: keine Probleme

16.01.2006

Vorgabe von: Wolfskin	Status: verfehlt
Ziel: Fertigstellung der Präsentation für die Review des Design des Kernsystems	Notiz: Verzögerung beim Erstellen der UML–Diagramme.

24.01.2006

Vorgabe von: Wolfskin	Status: erfüllt
Ziel: Präsentation des Design des Kernsystems vor dem Tutor	Notiz: Ursprungstermin: 17.01.2006. Wegen Verlegung der Review mit dem Auftraggeber wurde auch dieser Termin verlegt

25.01.2006

Vorgabe von: Projektbegleitung	Status: erfüllt
Ziel: Review des Design von dem Kernsystem	Notiz: Ursprungstermin: 18.01.2006. Der Termin wurde um eine Woche verlegt.

27.01.2006

Vorgabe von: Projektbegleitung	Status: erfüllt
Ziel: 3 Block Projektbegleitung Qualitätssicherung	Notiz: keine Probleme

06.02.2005

Vorgabe von: Wolfskin	Status: verfehlt
Ziel: Fertigstellung der Qualitätssicherungs Dokumentation des Kernsystems	Notiz: Probleme beim erstellen der Tests

10.02.2006

Vorgabe von: Projektbegleitung	Status: erfüllt
Ziel: Abgabe der Qualitätssicherungs- Dokumentation des Kernsystems	Notiz: keine Probleme

01.03.2006

Vorgabe von: Wolfskin	Status: verfehlt
Ziel: Fertigstellung der Endversionen des Produkts und der Dokumentation	Notiz: Verzögerungen durch Doku- mentation

30.03.2006

Vorgabe von: Projektbegleitung	Status: wartend
Ziel: Abnahme des Produkts und Ab- gabe der Endversion der gesamten Do- kumentation	Notiz: keine Probleme

8 Qualitätssicherung

Um die Qualität von FieStA schon in der Entwicklung zu gewährleisten werden vom Team einige Grundsätze und Standards formuliert und eingehalten. Diese Vorgehensweisen werden in diesem Kapitel erläutert und ausformuliert.

8.1 Ziele

Bei der Entwicklung von FieStA wurden gruppeninterne Ziele definiert, die die Qualität des Produktes sicherstellen. Ziel hiervon ist die semantische Korrektheit jeder Funktion der einzelnen Klassen zu gewährleisten. Zu diesem Zweck setzen wir JUnit-Tests ein, anhand derer überprüft wird ob die Ausführung der tatsächlichen Methoden den Vorgaben entsprechen, die an sie gestellt werden. Mithilfe von JUnit-Test wollen wir eine 80%ige Codeabdeckung von eigenem, nicht trivialen Code erreichen. In diesem Zusammenhang werden automatisch oder manuell generierte getter und setter oft nicht getestet, da diese ihre Funktion zuverlässig erfüllen und auch nur über das jeweilige Gegenstück getestet werden können. Ein nicht korrekter Getter würde beispielsweise bei den JUnit-Tests eines Setters falsche Ergebnisse liefern, auch wenn der Setter ordnungsgemäß funktioniert. Zudem wollen wir über das Setzen verschiedener Vorbedingungen bei den Tests eine Zweigabdeckung von 80% erreichen. Durch die Entwicklung der Tests anhand der Vor- und Nachbedingungen der im Pflichtenheft aufgeführten Usecases, werden alle elementaren Funktionen des Editors über JUnit-Tests überprüft.

8.2 Versionsverwaltung

Da in der Softwareentwicklung oft Probleme auftauchen, die sich durch die Wiederherstellung einer älteren lauffähigen Version schneller lösen lassen als durch die Manipulation von teilweise falschem Code, haben wir uns für die Verwendung eines CVS-Servers entschieden. Mithilfe des CVS-Servers können Entwicklungsschritte in eine falsche Richtung problemlos rückgängig gemacht werden. Alle Teammitglieder laden wichtige Entwicklungsschritte, wie das Hinzufügen neuer Methoden ins CVS und kommentieren diese, was zu einer guten Übersicht bei den einzelnen Versionen führt. Ein weiterer Vorteil bietet sich durch die Möglichkeit mehrere Änderungen an verschiedenen Methoden in derselben Klasse zu übernehmen. Die Möglichkeit alle Änderungen in einer gemeinsamen Datei auf dem Server zusammenzufügen erleichtert das verteilte Arbeiten und beugt Zugriffsberechtigungsproblemen vor.

8.3 Test-Driven Development

Um sicherzugehen, daß alle implementierten Methoden den Anforderungen gerecht werden, hat das Team Wolfskin besonders bei der Erstellung der Datenstruktur auf Pre- und Postconditions geachtet. Die JUnit-Tests, die die einzelnen Methoden der Klassen der Datenstruktur überprüfen, wurden aus diesem Grund auch vor dem eigentlichen Code auf-

gestellt. Auf diese Art und Weise konnte sichergestellt werden, daß Aktionen, die auf der Datenstruktur ausgeführt werden auch den Konventionen, die durch die Schnittstelle definiert sind, gerecht werden. Auch bei der Anbindung der UI an den Rest des Systems werden die jeweiligen Tests anhand der angeführten Usecases erstellt, um so die Korrektheit aller Funktionen sicherzustellen.

8.4 Rücksprache mit dem Auftraggeber

Nach Fertigstellung der minimalen Ausbaustufe wird die weitere Entwicklung von FieStA stets mit dem Auftraggeber abgesprochen. So können neue Features des Programms auf dessen Wunsch angepasst und bei Bedarf zusätzliche Ideen beider Seiten realisiert werden, ohne daß neue Features ungewünschte Funktionalitäten mit sich führen. Fehlerhafte Funktionen oder Mißverständnisse können auf diese Art effizient und schnell behoben werden, ohne größere Teile des Codes dadurch in Mitleidenschaft zu ziehen.

8.5 Coding Standards

Da alle Mitglieder des Teams Wolfskin jederzeit Einsicht in den Code der anderen haben und FieStA zudem von weiteren Entwicklungsteams bearbeitet werden soll, ist es von Nöten, daß gewisse Richtlinien bei der Programmierung eingehalten werden. Hierzu verwenden wir Checkstyle. Dieses gibt, da es bereits standardmäßig in Eclipse unterstützt wird, allen Programmierern die am Code Arbeiten die Möglichkeit ein gewohntes Codebild vorzufinden und dieses ebenfalls weiterzuführen. Die verwendeten Einstellungen wurden hierbei von allem Teammitgliedern mit dem gleichen Template geladen und können auch bei Bedarf angepasst werden.

8.6 Testverfahren

Während der Entwicklung und nach der Fertigstellung von neuen Codeteilen werden immer wieder JUnit-Tests durchgeführt, die Aufschluß über die Semantische Korrektheit des Codes geben. Bei der Fertigstellung einzelner Komponenten für die erste und alle weiteren Ausbaustufen hält sich der Testaufwand durch das häufige Testen bei der Erstellung neuen Codes und bei Änderungen an bestehenden Proqramnteilen dadurch in Grenzen.

Allgemeine Tests

Jede einzelne Klasse wird vom Programmierer selbst getestet, indem er ausführliche White-Box-Tests durchführt. Erst wenn eine Klasse ausgiebig mit Hilfe der White-Box-Tests auf ihre Richtigkeit geprüft wurde und somit nachgewiesen ist, daß alle Aktionen, die auf ihr ausgeführt werden können korrekt laufen können andere Komponenten erstellt und in FieStA eingebunden werden. Diese neuen Pakete werden wiederum soweit möglich mit White-Box-Tests überprüft und können parallel entwickelt werden.

Integrationstests

Sind zwei Komponenten, die voneinander abhängig sind fertiggestellt, wird die eigentliche Anbindung aneinander überprüft. Da, wie im vorhergehenden Kapitel beschrieben, alle Klassen auf ihre semantische Korrektheit überprüft wurden, kann davon ausgegangen werden, daß alle Methoden genau das bewirken, was in der Beschreibung der Methoden angegeben ist. Auf diese Weise ist es möglich mit Black-Box-Tests zu überprüfen, ob die Anbindung korrekt vorgenommen wurde.

Systemtests

Als letzten Schritt in der Entwicklung von FieStA werden alle essentiellen Funktionen in Systemtests im gesamten Kontext überprüft. Hierbei wird die Funktionsfähigkeit des gesamten Programms evaluiert und evtl. gefundene Bugs behoben. Die Tests in diesem letzten Schritt werden nicht nur von Mitgliedern des Teams durchgeführt, sondern auch von anderen Studenten, die mit dem Praktikum nichts zu tun haben und mit dem Code nicht vertraut sind. Alle Tester werden angehalten alle Funktionen, die im Pflichtenheft in Form der Usecases versprochen wurden zu testen. Des weiteren kann so Feedback zur Benutzbarkeit des Tools eingeholt werden.

9 Rechte

Die Software, die dieses Pflichtenheft beschreibt ist unter der GNU General Public License (GPL) veröffentlicht und kann nur unter den Bedingungen dieser Lizenz weitervertrieben oder verändert werden. Der ganze Text der Lizenz ist dem Anhang dieses Dokumentes zu entnehmen.

Die Rechte an diesem Dokument, an der Software, sowie an dem Athene-Symbol liegen bei der TU-Darmstadt.

A Glossar

B

batch

Der Begriff batch (deutsch: Stapel) stammt aus der Frühzeit der Computer-Geschichte. Alte Computer haben damals nur ein Programm nach dem anderen abarbeiten können, nicht, wie es häutige können, mehrere Programme gleichzeitig. Um sich die Arbeit zu erleichtern wurden mehrer Programme hintereinander weg als Stapel ausgeführt. Der Begriff der batch-Verarbeitung ist dadurch geprägt und bis heute noch ab und an anzutreffen. Unter MS DOS[©] und MS Windows[©] gibt es bis heute batch-Dateien mit dem Suffix *.bat*. batch-Verarbeitung bedeutet also, dass eine Liste von Befehlen nacheinander weg ausgeführt wird.

Weiter Informationen:

<http://de.wikipedia.org/wiki/batch>

F

Framework

Unter einem Framework (deutsch: Rahmenkonstrukt) versteht man ein Programm, das eine Art Rahmen bereitstellt, in das man andere Programmbestandteile reinladen kann. Diese Programmbestandteile bieten die eigentliche Funktionalität des Programmes. Jedes Programmbestandteil kann auf die gleichen Funktionen innerhalb des Frameworks zugreifen und bietet nach außen dadurch eine ähnliche oder gleiche Funktionsweise. Ein bekanntes Beispiel ist das Eclipse Framework.

Weiter Informationen:

<http://de.wikipedia.org/wiki/framework>

H

HTML

HTML (Hypertext Markup Language, deutsch: Hypertextbeschreibungssprache) ist die Sprache, in der das Aussehen der WWW-Seiten beschrieben ist. HTML selbst beschreibt dabei die Struktur während CSS das Layout und Design der Seite bestimmt. Der HTML-Standard ist vom W3C-Konsortium definiert.

Weiter Informationen:

<http://de.wikipedia.org/wiki/html>

J

JUnit

JUnit ist ein Test Framework, mit dessen Hilfe das Testen von Teilkomponenten eines Systems ermöglicht wird. Ist ein Test mit JUnit geschrieben und läuft dieser durch, so kann an den zutestenden Klassen gearbeitet und diese immer wieder auf ihre Funktionalität überprüft werden.

Weiter Informationen:

<http://www.junit.org>

M

Modul

Ein Modul ist im Kontext dieses Pflichtenheftes als ein Programmbestandteil zu verstehen, dass in ein Framework geladen wird. Das Modul bietet dem Nutzer die letztendliche Funktionalität, das Framework sorgt für die Rahmenbedingungen, innerhalb deren das Modul läuft.

P

Protokoll

Ein Protokoll ist die Beschreibung der Kommunikation zwischen zwei Kommunikationspartnern um ein bestimmtes Ziel zu erfüllen. In der Netzwerkprogrammierung ist man auf Protokolle angewiesen, damit eine Kommunikation zwischen zwei Computern geregelt ist und z.b. Dateien ausgetauscht werden können. Für die verschiedenen Schichten des ISO OSI-Schichtenmodells gibt es verschiedene Protokolle, die die Kommunikation zwischen Computer verschiedener Architekturen und Betriebssystem vereinheitlichen und somit ermöglichen.

S

Suffix

Ein Suffix (deutsch: Endung) ist eine Endung eines Wortes. Im Bereich der Computer versteht man darunter häufig die, durch einen Punkt vom eigentlichen Namen einer Datei oder eines Ordners getrennte, Endung, die den Typ der Datei angibt.

B GNU General Public License

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND
MODIFICATION

- 0** This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

- 1.** You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

- 2.** You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
- a)** You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
 - b)** You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
 - c)** If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these

conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
 - a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is

normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole

purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF

THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

- 12.** IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS