- Systemanalyse -

Lastenheft für „Lernsoftware zum Verstehen und Programmieren von Turing-Maschinen“

Version: 1.0

Projektbezeichnung	tmSim	
Projektleiter	Tobias Lettner	
Verantwortlich	Team A	
Erstellt am	24.03.2022	
Zuletzt geändert	04.05.2022 20:57	
Bearbeitungszustand		in Bearbeitung
	X	vorgelegt
		fertig gestellt
Dokumentablage	In Git-Branch main	

Änderungsverzeichnis

Änderung			Geänderte Kapitel	Beschreibung der Änderung	Autor	Zustand
Nr.	Datum	Version				
1		1.0	Alle	Initiale Produkterstellung		

Prüfverzeichnis

Die folgende Tabelle zeigt einen Überblick über alle Prüfungen – sowohl Eigenprüfungen wie auch Prüfungen durch eigenständige Qualitätssicherung – des vorliegenden Dokumentes.

Datum	Geprüfte Version	Anmerkungen	Prüfer	Neuer Produkt-zustand
26.04.2022	0.3	Rechtschreibfehler ausgebessert und mit Gruppe Verbesserungen abgesprochen	Adrian Rall	0.4
04.05.2022	0.4	Prüfung nach Prüfprotokoll	Team A	1.0

Inhalt

1 Einleitung

Dieses Dokument enthält alle an das zu entwickelnde System gestellten Anforderungen. Die Gliederung orientiert sich am Aufbau des V-Modell-XT®[1]-Produkts „Anforderungen (Lastenheft)“, ist jedoch zur Verwendung für die Veranstaltung **„Software-Projekte“** in Informatik-Curricula der **OTH-Amberg-Weiden** angepasst worden (und nicht konform zum V-Modell-XT): Teilnehmer dieser Veranstaltung erhalten von ihrem „Auftraggeber“ lediglich einen Überblick über das gewünschte System, was ungefähr dem Thema „Ausgangssituation und Zielsetzung“ in diesem Dokument entspricht; die Anforderungen müssen die Teilnehmer dann in enger Abstimmung mit ihrem „Auftraggeber“ selbst erarbeiten und in diesem Dokument niederlegen. Dadurch sollen sie Gelegenheit erhalten, auch Tätigkeiten der System-Analyse intensiver zu üben. Die „Auftraggeberseite“ liefert also nicht – wie im V-Modell-XT vorgesehen - das komplette Lastenheft, aus dem die „Auftragnehmerseite“ ein separates Pflichtenheft ableitet; stattdessen wird das hier vorliegende Dokument vom studentischen Entwicklerteam zur Dokumentation der Analyse-Ergebnisse erstellt und zugleich als Ersatz für die im V-Modell-XT vorgesehenen Dokumente Lasten- und Pflichtenheft verwendet.

Kern dieses Dokuments sind die funktionalen und nicht-funktionalen Anforderungen an das System, sowie eine Skizze des Gesamtsystementwurfs. Der Entwurf berücksichtigt die zukünftige Umgebung und Infrastruktur, in der das System später betrieben wird, und gibt Richtlinien für Technologieentscheidungen. Ebenfalls Teil der Anforderungen ist die Festlegung von Lieferbedingungen und Abnahmekriterien.

Die funktionalen und nicht-funktionalen Anforderungen dienen nicht nur als Vorgaben für die Entwicklung, sondern sind zusätzlich Grundlage der Anforderungsverfolgung und des Änderungsmanagements. Die Anforderungen sollten so aufbereitet sein, dass die Verfolgbarkeit (Traceability) sowie ein geeignetes Änderungsmanagement für den gesamten Lebenszyklus eines Systems möglich sind.

Im Allgemeinen sollten keine technischen Lösungen vorgegeben werden, um Softwarearchitekten und -entwickler bei der Suche nach optimalen technischen Lösungen nicht einzuschränken.

2 Ausgangssituation und Zielsetzung

Viele im Internet auffindbare Webseiten zur eigenen Programmierung und Simulation von Turingmaschinen erlauben entweder nur die Auswahl vorgefertigter Programme, mit denen eigene Eingaben überprüft werden können oder die eigene Programmierung wird nur durch viele Zeilen komplizierter Syntax möglich. Keine dieser beiden Varianten stellt Benutzer zufrieden, die sich erst in das Thema einarbeiten wollen und durch die Syntax abgeschreckt werden oder Benutzer mit Vorwissen zu Turingmaschinen, die durch die Syntax ausgebremst werden.

Das Ziel dieses Projektes ist es daher, eine Lernsoftware zu erstellen, mit der man Turingmaschinen verstehen und selbst programmieren kann, von einfachen hin zu sehr Komplexen. Hierfür soll der Benutzer – durch das System unterstützt – lernen, wie Turingmaschinen funktionieren, um dann immer komplexere Varianten selbst schreiben zu können, ohne dass komplizierte Syntax in seinen Weg gerät.

Die wichtigsten Programmfunktionen sind:

- Der Benutzer kann eine Turingmaschine in das Programm einlesen oder selbst eine erstellen, diese bearbeiten und abspeichern, um sie später wieder zu benutzen.
- Der Benutzer kann sich die Turingmaschine als Tabelle oder als Zustandsdiagramm darstellen lassen, wobei eine freie Wahl zwischen den beiden Darstellungsarten jederzeit möglich ist.
- Das Programm stellt sicher, dass die Turingmaschine gültig ist und weist den Benutzer bei Ungültigkeit auf die Fehlerquelle(n) hin (z.B. im aktuellen Zustand existieren mehrere Übergangsregeln für ein gleiches Symbol).

[1] V-Modell® ist eine geschützte Marke der Bundesrepublik Deutschland.

- Der Benutzer hat ein endlos langes Band vor sich, welches in einzelne Felder unterteilt ist. Auf das Band wird das vom Benutzer eingetragene Eingabewort übernommen, falls alle Symbole des Eingabeworts im Eingabealphabet der Turingmaschine vorhanden sind.

- Das Programm erlaubt es dem Benutzer zur besseren Nachvollziehbarkeit des Programmablaufs, diesen in geeigneter Weise anzupassen. Hierfür soll es folgende Möglichkeiten geben:

    - Simulationsdurchlauf in Einzelschritten

    - Abspielen in auswählbarer Geschwindigkeit

    - Sprung zum Programmanfang (Abbruch der Simulation)

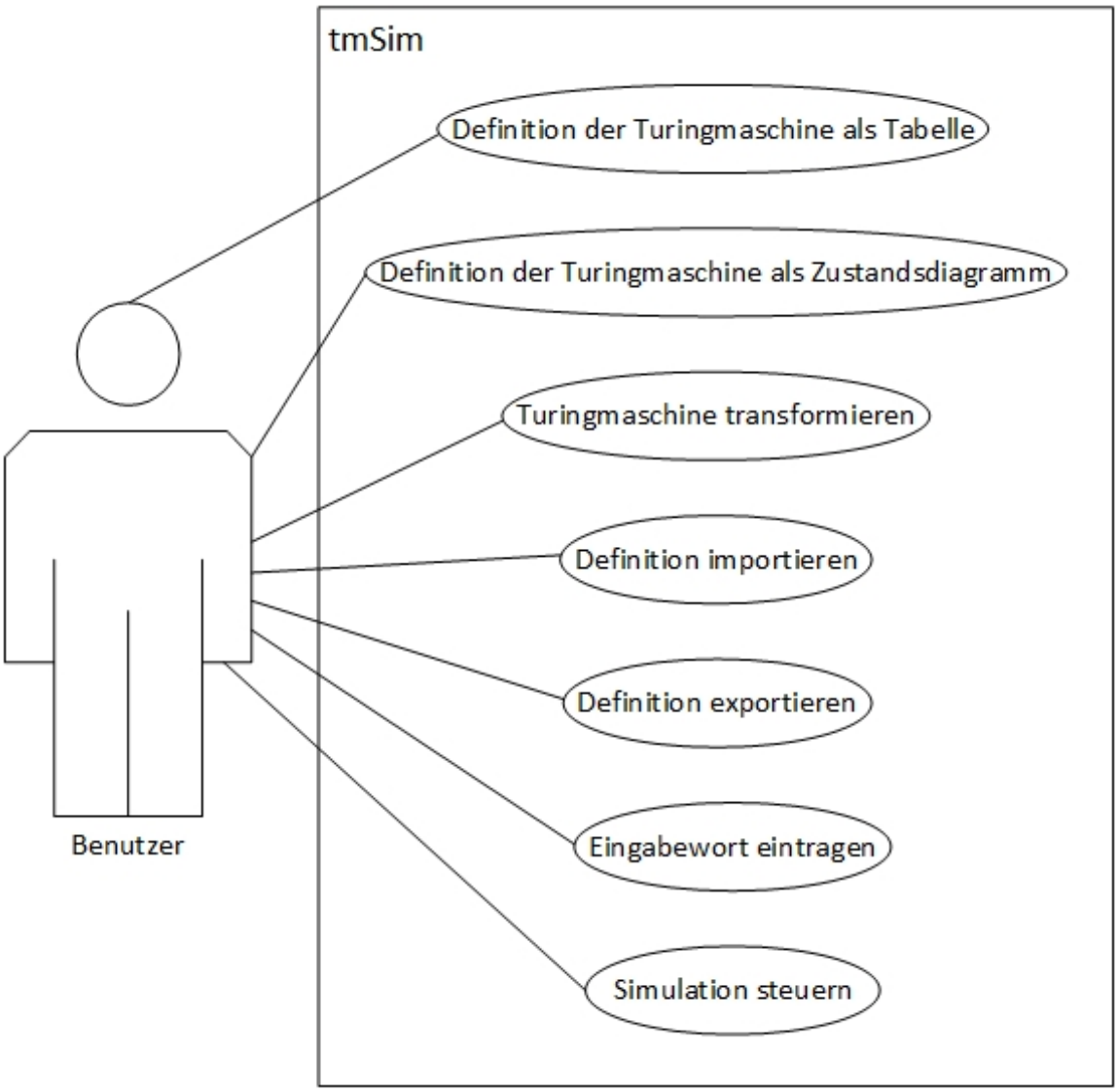
    - Pausieren und Fortsetzen des Ablaufs

    - Aktuellen Zustandsübergang in Tabelle und Zustandsdiagramm markieren

- Das Programm informiert den Benutzer am Ende des Programmdurchlaufs darüber, ob die Turingmaschine in einem akzeptierenden Zustand gestoppt ist oder nicht.

3 Funktionale Anforderungen

3.1 Use-Case Übersicht



3.2 Use-Case Beschreibungen

Definition der Turingmaschine als Tabelle	
Kennung	UC-1
Priorität	Hoch
Kurzbeschreibung:	
Der Benutzer definiert: • Kommentare • Zustände • Eingabealphabet • Bandalphabet • Zustandsübergänge • Zustandsbenennung in einer Tabelle. Dabei kann entweder eine neue Tabelle erstellt oder eine Bestehende bearbeitet werden.	
Vorbedingung(en):	
• Die Simulation ist beendet.	
Nachbedingung(en):	
• Die Definition der Turingmaschine ist gültig.	
Normaler Ablauf:	
	1. Der Anwendungsfall beginnt, wenn der Benutzer die Definition über Tabellenansicht auswählt. 2. Der Benutzer hat die Möglichkeit, die Tabelle zu bearbeiten. 3. Der Benutzer bestätigt seine Eingabe. 4. Das System prüft, ob die Tabelle gültig ist. 5. Das System übernimmt die Tabelle als Definition der Turingmaschine. **Ende.**
Ablauf-Varianten:	
2a	Der Benutzer fügt eine neue Spalte (Zeichen) hinzu.
	1. Das System fügt eine leere Spalte hinzu. **Rückkehr nach**: 2
2b	Der Benutzer löscht eine existierende Spalte (Zeichen).
	1. Das System löscht die markierte Spalte. **Rückkehr nach**: 2
2c	Der Benutzer fügt eine neue Zeile (Zustand) hinzu.
	1. Das System fügt eine leere Zeile hinzu. **Rückkehr nach**: 2
2d	Der Benutzer löscht eine existierende Zeile (Zustand).
	1. Das System löscht die markierte Zeile. **Rückkehr nach**: 2
2e	Der Benutzer füllt die Tabelle aus.
	1. Der Benutzer hat die Möglichkeit, folgende Eigenschaften festzulegen: • Kommentare • Startzustand • Akzeptierende Zustände • Zwischenzustände • Eingabealphabet • Bandalphabet • Zustandsübergänge • Zustandsbenennung **Rückkehr nach**: 2
4a	Ungültige Tabelle
	1. Das System weist den Benutzer auf die gefundenen Fehler hin. **Ende.**
Spezielle Anforderungen:	
Zu klärende Punkte:	

Definition der Turingmaschine als Zustandsdiagramm	
Kennung	UC-2
Priorität	Mittel

Kurzbeschreibung:

Der Benutzer definiert:

- Kommentare
- Zustände
- Eingabealphabet
- Bandalphabet
- Zustandsübergänge
- Zustandsbenennung

in einem Zustandsdiagramm. Dabei kann entweder ein neues Zustandsdiagramm erstellt oder ein Bestehendes bearbeitet werden.

Vorbedingung(en):

- Die Simulation ist beendet.

Nachbedingung(en):

- Die Definition der Turingmaschine ist gültig.

Normaler Ablauf:

1. Der Anwendungsfall beginnt, wenn der Benutzer die Definition über Zustandsdiagramm auswählt
2. Der Benutzer hat die Möglichkeit, das Zustandsdiagramm zu bearbeiten.
3. Der Benutzer bestätigt seine Eingabe.
4. Das System prüft, ob das Zustandsdiagramm gültig ist.
5. Das System übernimmt das Zustandsdiagramm als Definition der Turingmaschine.

Ende.

Ablauf-Varianten:

2a	Der Benutzer fügt einen neuen Knoten (Zustand) hinzu.
	1. Das System fügt einen neuen Knoten hinzu. *Bem.: Der Knoten erhält hier automatisch eine Benennung, welche später bearbeitet werden kann.* **Rückkehr nach**: 2
2b	Der Benutzer löscht einen existierenden Knoten (Zustand).
	1. Das System löscht den markierten Knoten. 2. Das System löscht alle ein- und ausgehenden Pfeile. **Rückkehr nach**: 2
2c	Der Benutzer bearbeitet einen existierenden Knoten (Zustand).
	1. Der Benutzer hat die Möglichkeit, folgende Eigenschaften festzulegen: • Startzustand • Akzeptierender Zustand • Zwischenzustände • Zustandsbenennung **Rückkehr nach**: 2
2d	Der Benutzer fügt einen neuen Pfeil (Zustandsübergang) hinzu.
	1. Der Benutzer wählt einen Start- und Zielknoten aus. 2. Das System fügt einen neuen Pfeil zwischen den gewählten Knoten hinzu. *Bem.: Start- und Zielknoten können identisch sein.* **Rückkehr nach**: 2
2e	Der Benutzer löscht einen existierenden Pfeil (Zustandsübergang).
	1. Das System löscht den markierten Pfeil. **Rückkehr nach**: 2
2f	Der Benutzer bearbeitet einen existierenden Pfeil (Zustandsübergang).
	1. Der Benutzer hat die Möglichkeit, folgende Eigenschaften festzulegen: • Kommentar • Eingabesymbol (gelesenes Zeichen) • LSK Bewegung (links oder rechts) **Rückkehr nach**: 2
2g	Der Benutzer bestimmt das Eingabe- und Bandalphabet.
	Rückkehr nach: 2
4a	Ungültiges Zustandsdiagramm

	1. Das System weist den Benutzer auf die gefundenen Fehler hin. **Rückkehr nach:** 2
Spezielle Anforderungen:	
Zu klärende Punkte:	

Turingmaschine transformieren	
Kennung	UC-3
Priorität	Mittel
Kurzbeschreibung:	
Der Benutzer hat die Möglichkeit, eine Turingmaschine in eine äquivalente Turingmaschine zu transformieren. Dem Benutzer werden hierfür verschiedene Transformationsmöglichkeiten angeboten, welche er auswählen und durchführen kann: 1. Kein Übergang führt in den Startzustand und keiner der Übergänge beginnt in einem akzeptierenden Zustand. 2. Es gibt keine Zustandsübergänge mit unbewegtem Lese-/Schreibkopf. 3. Es wird niemals das Leerzeichen geschrieben. 4. Die Zustandsmenge ist in eine rechte und linke Hälfte geteilt. 5. Es gibt genau einen akzeptierenden Zustand.	
Vorbedingung(en):	
• Es liegt eine vollständig definierte Turingmaschine vor. • Die Simulation ist beendet.	
Nachbedingung(en):	
• Die transformierte Turingmaschine ist gültig. • Die transformierte Turingmaschine akzeptiert die gleiche Sprache.	
Normaler Ablauf:	
	1. Dieser Anwendungsfall beginnt, wenn der Benutzer eine Transformation auswählt. 2. Das System prüft die Bedingung der Transformation gemäß untenstehender Tabelle. 3. Das System führt die Transformation aus. **Ende**.
Ablauf-Varianten:	
2a	Die Bedingung für die ausgewählte Transformation ist nicht erfüllt
	1. Das System gibt dem Benutzer eine entsprechende Rückmeldung. **Ende.**
3a	Transformation 1 wurde ausgewählt
	1. Das System stellt sicher, dass in der Definition der TM kein Zustandsübergang hinterlegt ist, dessen Zielzustand der Startzustand ist. 2. Das System stellt sicher, dass kein Übergang beginnend von einem akzeptierenden Zustand existiert. **Ende.**
3b	Transformation 2 wurde ausgewählt
	1. Das System überprüft, ob es Zustandsübergänge gibt, bei dem der LSK stehen bleibt. **Ende.**
3c	Transformation 3 wurde ausgewählt
	1. Der Benutzer wählt ein Zeichen, welches das Leerzeichen ersetzen soll. 2. Das System überprüft das Leerzeichen auf Gültigkeit. 3. Das System ersetzt bei allen entsprechenden Zustandsübergängen das Leerzeichen durch das gewählte Zeichen. **Ende.**

3d	Transformation 4 wurde ausgewählt
	1. Das System teilt die Zustandsmenge in eine linke und eine rechte Teilmenge auf. In der linken Teilmenge befinden sich die Zustände, welche den LSK nach links bewegen. In der rechten Teilmenge befinden sich die Zustände, welche den LSK nach rechts bewegen. **Ende.**
3e	Transformation 5 wurde ausgewählt
	1. Das System stellt sicher, dass es genau einen akzeptierenden Zustand gibt. **Ende.**

Bedingungen für die Zulässigkeit der einzelnen Transformationen

Kommando	Bedingung
Transformation 1	---
Transformation 2	---
Transformation 3	---
Transformation 4	- Der Startzustand ist kein Endzustand (Transformation 1) - LSK ist niemals unbewegt (Transformation 2)
Transformation 5	- Es gibt mindestens einen akzeptierenden Zustand

Spezielle Anforderungen:

Zu klärende Punkte:

Definition importieren	
Kennung	UC-4
Priorität	Mittel

Kurzbeschreibung:

Der Benutzer lädt eine Definition für eine Turingmaschine in das System. Das System prüft, ob diese Definition gültig ist.

Vorbedingung(en):

- Die Simulation ist beendet.

Nachbedingung(en):

- Die Definition der Turingmaschine ist gültig.

Normaler Ablauf:

1. Dieser Anwendungsfall beginnt, wenn der Benutzer eine Text-Datei zum Importieren auswählt.
2. Das System überprüft, ob die Text-Datei eine syntaktisch gültige Definition einer Turingmaschine darstellt.
3. Das System erstellt aus der Definition in der Text-Datei eine Turingmaschine.

Ende.

Ablauf-Varianten:

2a	Die Text-Datei enthält eine ungültige Definition
	1. Das System weist dem Benutzer auf die ungültige Text-Datei hin. **Ende.**

Spezielle Anforderungen:

Zu klärende Punkte:

Definition exportieren	
Kennung	UC-5
Priorität	Niedrig
Kurzbeschreibung:	
Der Benutzer gibt den Befehl, die aktuelle Definition der Turingmaschine zu exportieren. Das System speichert diese in menschenlesbarer Form als Text-Datei an dem gewünschten Ablageort.	
Vorbedingung(en):	
• Die Definition der Turingmaschine ist gültig.	
Nachbedingung(en):	
Normaler Ablauf:	
	1. Dieser Anwendungsfall beginnt, wenn der Benutzer den Befehl zum Exportieren sendet. 2. Das System fordert den Benutzer auf, einen Speicherort und einen Dateinamen zu wählen. 3. Der Benutzer wählt einen Speicherort und Dateinamen. 4. Das System exportiert die Definition der Turingmaschine in menschenlesbarer Form als Text-Datei. **Ende**.
Ablauf-Varianten:	
4a	Speichervorgang kann nicht ausgeführt werden (z.B. nicht genügend Speicher vorhanden, keine Berechtigung)
	1. Das System weist den Benutzer auf das Problem hin. **Ende.**
Spezielle Anforderungen:	
Zu klärende Punkte:	

Eingabewort eintragen	
Kennung	UC-6
Priorität	Hoch
Kurzbeschreibung:	
Der Benutzer trägt ein Eingabewort auf der Benutzeroberfläche ein. Das System überprüft, ob alle Symbole des Eingabeworts im Eingabealphabet enthalten sind.	
Vorbedingung(en):	
• Es liegt eine vollständig definierte Turingmaschine vor. • Die Simulation ist beendet.	
Nachbedingung(en):	
• Auf dem Band der Turingmaschine befindet sich ein gültiges Eingabewort.	
Normaler Ablauf:	
	1. Dieser Anwendungsfall beginnt, wenn der Benutzer das Eingabewort einträgt. 2. Der Benutzer bestätigt seine Eingabe. 3. Das System überprüft anhand des Eingabealphabets das Eingabewort auf Gültigkeit. 4. Das System schreibt das Eingabewort auf das Band. **Ende**.
Ablauf-Varianten:	
3a	Ungültiges Eingabewort
	1. Das System signalisiert, dass das Eingabewort ungültig ist. **Ende.**
Spezielle Anforderungen:	
Zu klärende Punkte:	

Simulation steuern

Kennung	UC-7
Priorität	Hoch

Kurzbeschreibung:

Dem Benutzer stehen zur Steuerung der Simulation verschiedene Möglichkeiten zur Verfügung.
Dabei kann der Benutzer die:

- Simulation starten.
- Simulation abbrechen.
- Simulation pausieren.
- Simulation fortsetzen.
- Simulationsverzögerung einstellen.
- Simulation in Einzelschritte durchlaufen.
- Option Zustandsübergänge markieren an-/abwählen.
- Definitionsansicht wählen

Vorbedingung(en):

- Die Definition der Turingmaschine ist gültig.
- Auf dem Band der Turingmaschine befindet sich ein gültiges Eingabewort.

Nachbedingung(en):

Normaler Ablauf:

	1. Dieser Anwendungsfall beginnt, wenn der Benutzer einen Befehl zur Steuerung der Simulation sendet. **Ende**.

Ablauf-Varianten:

1a	Befehl **Starten**
	1. Das System startet die Simulation im automatischen Ablauf. *Bem.: Automatischer Ablauf bedeutet Einzelschrittausführung mit eingestellter Verzögerung, bis das Ende der TM erreicht ist.*
1b	Befehl **Abbruch**
	1. Das System beendet die Simulation. *Bem.: Der aktuelle Zustand der Simulation wird wieder auf den Startzustand zurückgesetzt.*
1c	Befehl **Pausieren**
	1. Das System pausiert die Simulation an der aktuellen Stelle.
1d	Befehl **Fortsetzen**
	1. Das System setzt den automatischen Ablauf an der aktuellen Stelle fort. *Bem.: Automatischer Ablauf wie bei Variante '1a' erläutert.*
1e	Befehl **Verzögerungsanpassung**
	1. Das System verzögert anhand des eingestellten Werts die zeitlichen Abstände zwischen dem Wechsel zweier Zustände.
1f	Befehl **Einzelschritt**
	1. Das System führt einen einzelnen Simulationsschritt aus. 2. Das System prüft, ob das Ende des Programmdurchlaufs der TM erreicht wurde. 3. Das System benachrichtigt den Benutzer, ob sich die TM in einem akzeptierenden Endzustand befindet.
1g	Befehl **Zustandsübergänge markieren an-/abwählen**
	1. Das System hebt den aktuellen Zustandsübergang farblich hervor.
1h	Befehl **Definitionsansicht wählen**
	1. Das System zeigt je nach Auswahl das Zustandsdiagramm und/oder die Tabelle an.

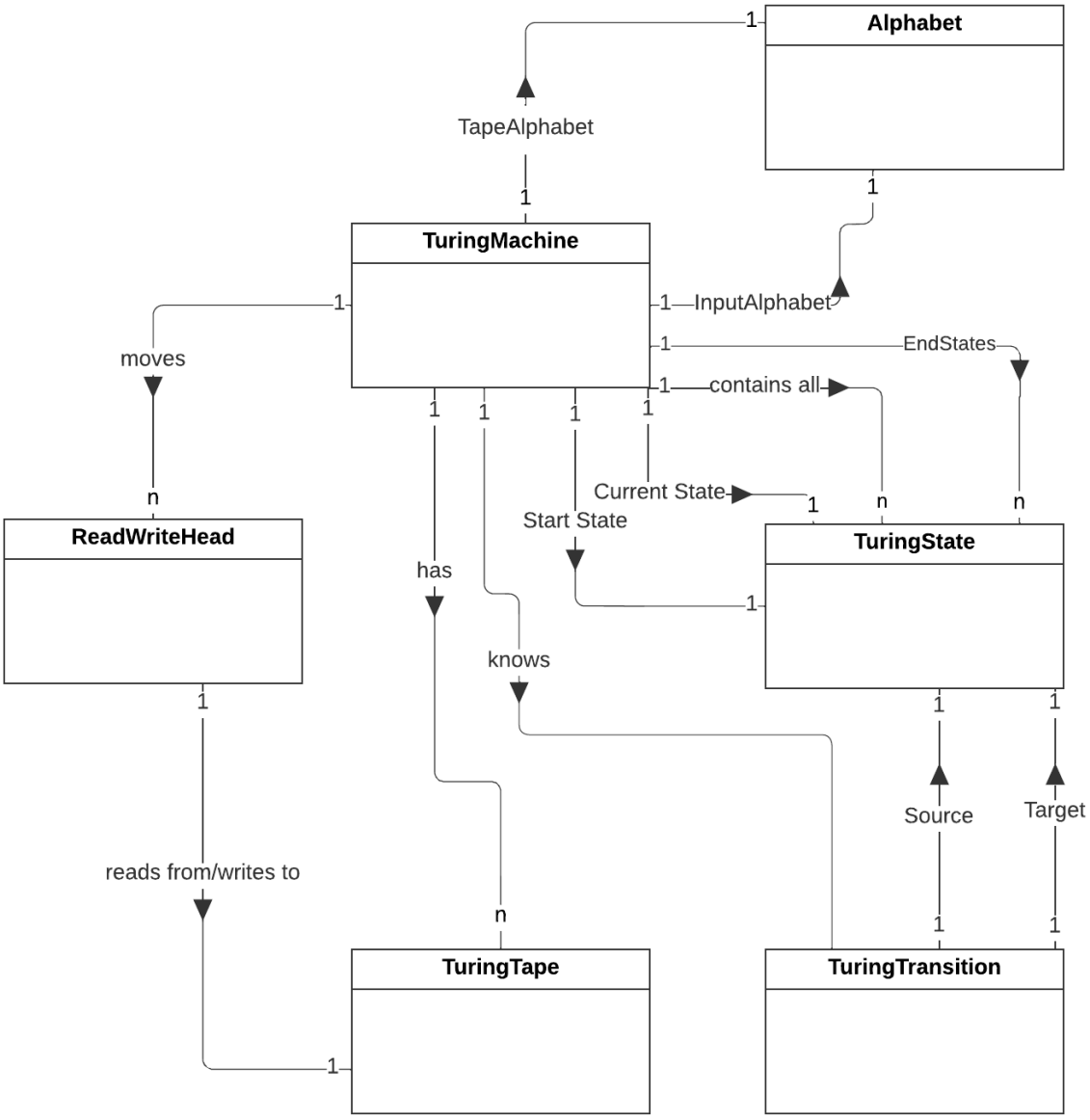
Spezielle Anforderungen:

Zu klärende Punkte:

(Sonstige) Funktionalität

ID	Beschreibung	Querverweise

3.3 Modell des Problembereichs (Konzeptionelles Datenmodell)



4 Nicht-Funktionale Anforderungen

4.1 Benutzbarkeit (Usability)

ID	Beschreibung	Querverweise
UR-001	Das System soll über eine grafische Benutzerschnittstelle bedienbar sein.	
UR-002	Die Simulation kann über die Tastatur gesteuert werden.	
UR-003	Die exportierte Definition der Turingmaschine sind in menschenlesbarer Form und können im Texteditor bearbeitet werden.	
UR-004	Die Benutzeroberfläche unterstützt Mehrsprachigkeit.	

4.2 Zuverlässigkeit (Reliability)

ID	Beschreibung	Querverweise
RR-001	Das System speichert kontinuierlich die aktuelle Definition der Turingmaschine in einer Text-Datei ab, um im Falle eines Absturzes zum vorherigen Zustand zurückkehren zu können.	

4.3 Leistung (Performance)

ID	Beschreibung	Querverweise
PR-001	Das System soll nicht länger als drei Sekunden benötigen, um auf Benutzereingaben zu reagieren.	
PR-002	Das System soll nicht länger als zwanzig Sekunden benötigen, um die Definition einer Turingmaschine mit maximal 100 Übergängen einzulesen oder abzuspeichern.	

4.4 Unterstützbarkeit (Supportability)

ID	Beschreibung	Querverweise
SR-001	Das System soll aus Komponenten zusammengesetzt sein, die eine geringe Kopplung untereinander aufweisen und auf eine Aufgabe spezialisiert sind.	

4.5 Sonstige Einschränkungen

4.5.1 Schnittstellen

4.5.2 Implementierung

- Zielumgebung:
  Das Programm soll in der im Software-Projektlabor vorliegenden Zielumgebung ausführbar, der Quellcode mit den dort installierten Versionen von Visual-Studio 2019 bzw. Eclipse analysierbar sein.

- Verwendete Programmiersprache:
  Das Projektteam hat sich darauf geeignet, dass die Programmiersprache C# für das Projekt verwendet wird.

4.5.3 Entwurf

ID	Beschreibung	Querverweise
DR-001	Der Entwurf soll den Grundsatz der Trennung zwischen Model und View gemäß dem „Model-View-ViewModel“-Entwurfsmuster strikt einhalten.	UR-001
DR-002	Die Systemarchitektur soll so gestaltet werden, dass das System leicht um weitere Bänder ergänzt werden kann.	

5 Risikoakzeptanz

Entfällt hier.

6 Skizze der Gesamtsystemarchitektur

Entfällt hier.

7 Lieferumfang

Die folgende Tabelle enthält alle Arbeitsergebnisse, die in der Veranstaltung „Software-Projekte“ zu dem vom Team zu liefernden „End-Produkt“ gehören – für die individuell von jedem Projektteilnehmer zu liefernden Ergebnissen lesen Sie bitte im Projektleitfaden bzw. im Projektkalender nach. Die Benotung erfolgt nicht nur auf Grundlage des lauffähigen Programms, sondern bezieht die Qualität der Analyse, des Entwurfs und des Systemtests mit ein.

Lfd. Nr.	Was?	Art des Dokuments	Bemerkungen
Ergebnis der System-Analyse			

1	Das Dokument „Systemanalyse“ (also dieses Dokument) mit funktionalen, nichtfunktionalen Anforderungen und konzeptionellem Datenmodell.	• Siehe Vorlage. • Wird bei Projektbeginn mit einem Überblick gebenden Systembeschreibung an das Team ausgegeben. Das Dokument ist vom Team weiterzuführen und wieder abzugeben.	• Bitte auf Abgabetermin während des Semesters achten (s. Projektkalender). • Rechtzeitig vor Abgabe auf Qualitätssicherung achten (Review)
Dokumentation des Systementwurfs			
2	Das Dokument „Systementwurf“.	• Siehe Vorlage.	• Bitte auf Abgabetermin während des Semesters achten (s. Projektkalender) • Rechtzeitig vor Abgabe auf Qualitätssicherung achten (Review)
Implementierung			
3	Lauffähiger und getesteter Quellcode		Abgabe am Semesterende
Test			
4	Testspezifikation Systemtest	• Siehe Vorlage	• Endgültige Abgabe am Semesterende; zur Vorbereitung des Abnahmetests ist die Aufstellung der in den Abnahmetest einbezogenen Testfälle früher vorzulegen (Termin im Projektkalender)

8 Abnahmekriterien

In der Veranstaltung „Software-Projekt“ werden vom „Auftraggeber“ (in Absprache mit den Teilnehmern) rechtzeitig vor Semesterende Systemtestfälle ausgewählt, die das System dann am Tag der Abnahme ohne Beanstandung „überstehen“ muss.

9 Glossar

Begriff	Erklärung
Eingabealphabet	Es ist eine endliche Teilmenge des Bandalphabets, welche alle Zeichen, die der Benutzer eingeben kann, beinhaltet.
Bandalphabet	Menge der Zeichen, die eine Turingmaschine auf ihrem Band verarbeitet. Zu diesen Zeichen zählt das Eingabealphabet und das Leerzeichen.
Eingabewort	Die Zeichenkette, welche vom Benutzer auf das Band der Turingmaschine geschrieben werden soll. Dieses Wort besteht nur aus dem Eingabealphabet.
Gültige Definition	Die Turingmaschine enthält genau einen Startzustand, mindestens einen Endzustand und Zustandsübergänge bestehend aus: • Mindestens einem Zielzustand • Dem neuen Zeichen, welches ggf. anstelle des alten geschrieben werden soll. • Der Richtung in die sich der LSK als nächstes bewegen soll.
Zielzustand	Der Zustand, welcher nach einem Zustandsübergang erreicht wird.
Leerzeichen	Ein besonderes Zeichen des Bandalphabets, welches nicht im Eingabealphabet vorkommt und dazu dient ein Feld auf dem Band der Turingmaschine als leer zu kennzeichnen.

10 Abkürzungsverzeichnis

Abkürzung	Erklärung
TM	Turingmaschine
LSK	Lese-/Schreibkopf

11 Literaturverzeichnis

[Lar] Larman Craig, *Applying UML And Patterns. An Introduction to Object-Oriented Analysis And Design,* Prentice Hall, 2nd ed., 2002

12 Abbildungsverzeichnis