HINWEIS: Blauer Text kann gelöscht werden, beziehungsweise soll ersetzt werden

- Systemanalyse -

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

Version: 1.0

|  |  |  |
| --- | --- | --- |
| Projektbezeichnung | PRODUKT.PROJEKTBEZEICHNUNG | |
| Projektleiter |  | |
| Verantwortlich | Anforderungsanalytiker (AG) | |
| Erstellt am |  | |
| Zuletzt geändert | 15.04.2022 08:49 | |
| Bearbeitungszustand | X | in Bearbeitung |
|  | vorgelegt |
|  | fertig gestellt |
| Dokumentablage |  | |

Änderungsverzeichnis

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Änderung | | | Geänderte Kapitel | Beschreibung der Änderung | Autor | Zustand |
| Nr. | Datum | Version |
| 1 |  | 1.1 | 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 Produktzustand |
|  |  |  |  |  |

Inhalt

[1 Einleitung 4](#_Toc34239361)

[2 Ausgangssituation und Zielsetzung 4](#_Toc34239362)

[3 Funktionale Anforderungen 5](#_Toc34239363)

[3.1 Use-Case Übersicht 5](#_Toc34239364)

[3.2 Use-Beschreibungen 5](#_Toc34239365)

[3.3 (Sonstige) Funktionalität 7](#_Toc34239366)

[3.4 Modell des Problembereichs (Konzeptionelles Datenmodell) 7](#_Toc34239367)

[4 Nicht-Funktionale Anforderungen 7](#_Toc34239368)

[4.1 Benutzbarkeit (Usability) 7](#_Toc34239369)

[4.2 Zuverlässigkeit (Reliability) 8](#_Toc34239370)

[4.3 Leistung (Performance) 8](#_Toc34239371)

[4.4 Unterstützbarkeit (Supportability) 9](#_Toc34239372)

[4.5 Sonstige Einschränkungen 9](#_Toc34239373)

[5 Risikoakzeptanz 10](#_Toc34239374)

[6 Skizze der Gesamtsystemarchitektur 10](#_Toc34239375)

[7 Lieferumfang 11](#_Toc34239376)

[8 Abnahmekriterien 11](#_Toc34239377)

[9 Glossar 11](#_Toc34239378)

[10 Abkürzungsverzeichnis 12](#_Toc34239379)

[11 Literaturverzeichnis 12](#_Toc34239380)

[12 Abbildungsverzeichnis 12](#_Toc34239381)

# Einleitung

Dieses Dokument enthält alle an das zu entwickelnde System gestellten Anforderungen. Die Gliederung orientiert sich am Aufbau des V-Modell-XT®[[1]](#footnote-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 Architekten und Entwickler bei der Suche nach optimalen technischen Lösungen nicht einzuschränken.

# Ausgangssituation und Zielsetzung

In diesem Thema werden die Ausgangssituation und der Anlass zur Durchführung des Projekts anschaulich dargestellt. Es wird beschrieben, welche Defizite bzw. Probleme existierender Systeme oder auch der aktuellen Situation zur Entscheidung geführt haben, das Projekt durchzuführen, und welche Vorteile durch den Einsatz des neuen Systems erwartet werden.

Es werden zusätzlich alle relevanten Stakeholder des Projekts benannt und die technische und fachliche Einbettung des zu entwickelnden Systems in seine Umgebung skizziert. Zusätzlich werden erste Rahmenbedingungen für die Entwicklung identifiziert und beschrieben. Rahmenbedingungen können beispielsweise technische Vorgaben oder Vorgaben zur Sicherheit sein.

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 zu schreiben, ohne das komplizierte Syntax in seinen Weg gerät.

Die wichtigsten Programmfunktionen sind:

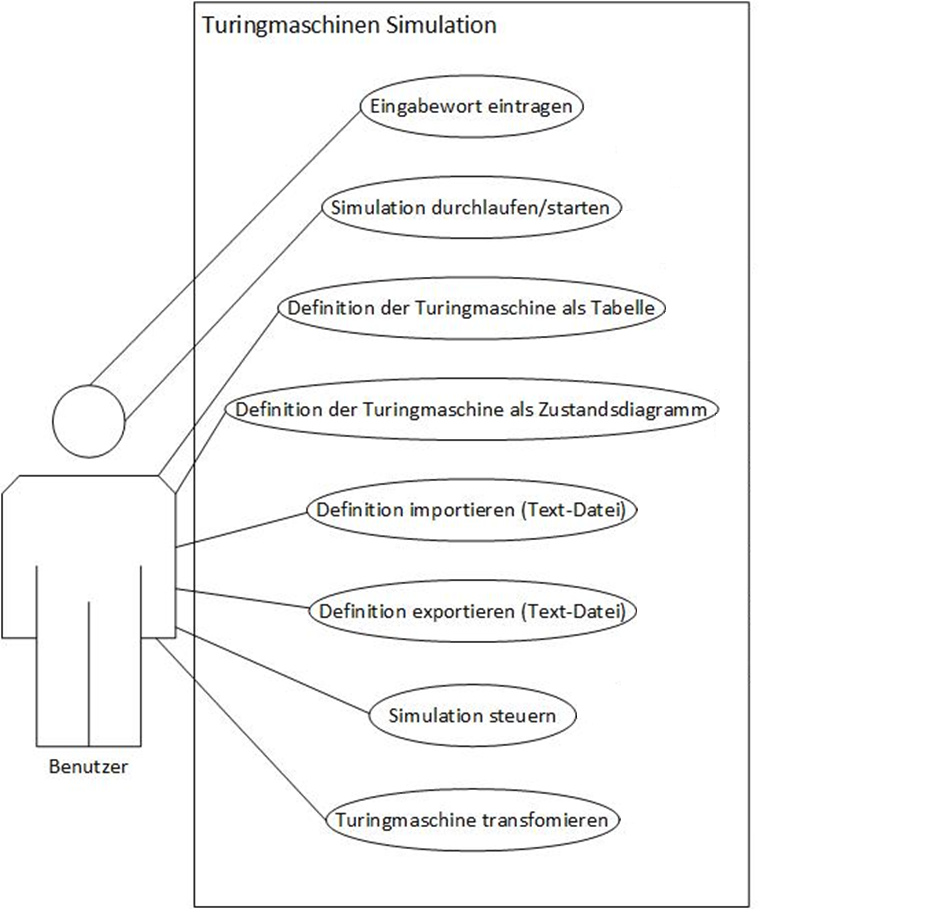
* Der Benutzer kann eine Turingmaschine in das Programm einlesen oder selbst eine erstellen und diese abspeichern, um sie später wieder zu benutzen.
* Der Benutzer kann sich die Turingmaschine als Tabelle oder als Zustandsdiagramm darstellen lassen, wobei eine freie Wahl zwischen den beiden Darstellungsarten 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. dasselbe Symbol besitzt mehrere Übergänge)
* Der Benutzer hat ein Band, aufgeteilt in einzelne Kästchen, vor sich und kann mittels geeigneter Schnittstelle auf dieses schreiben. Hierbei erlaubt das Programm nur die Eingabe von Zeichen, die im Eingabealphabet der Turingmaschine vorkommen.
* Der Benutzer kann weitere Bänder hinzufügen oder diese wieder entfernen, um Mehrband-Turingmaschinen zu simulieren. Das System stellt sicher, dass immer mindestens ein Band vorhanden ist. Werden beschriebene Bänder entfernt, so schreibt das Programm deren Inhalt auf das darüberliegende Band.
* Das Programm erlaubt es dem Benutzer zur besseren Nachvollziehbarkeit des Programmablaufs diesen in geeigneter Weise anzupassen. Hierfür soll es folgende Möglichkeiten geben:
  + In Einzelschritten (vorwärts und rückwärts)
  + Abspielen in auswählbarer Geschwindigkeit
  + Sprung zum Programmanfang
  + Pausieren des Ablaufs
  + Aktuellen Zustandsübergang in Tabelle und Zustandsdiagramm farblich hervorheben
* Das Programm informiert den Benutzer am Ende des Programmdurchlaufs darüber, ob die Turingmaschine in einem akzeptierenden Zustand gestoppt ist oder nicht.
* Funktionale Anforderungen

Funktionale Anforderungen beschreiben die Fähigkeiten eines Systems, die ein Anwender erwartet, um mit Hilfe des Systems ein fachliches Problem zu lösen. Die Anforderungen werden aus den zu unterstützenden Geschäftsprozessen und den Ablaufbeschreibungen zur Nutzung des Systems abgeleitet.

Die Beschreibung der funktionalen Anforderungen erfolgt beispielsweise in Form von Anwendungsfällen (Use Cases). Ein Anwendungsfall beschreibt dabei einen konkreten, fachlich in sich geschlossenen Teilvorgang. Die Gesamtheit der Anwendungsfälle definiert das Systemverhalten. Ein Anwendungsfall kann in einfachem Textformat beschrieben werden, häufig stehen jedoch organisationsspezifische Muster zur Beschreibung zur Verfügung. Für datenzentrierte Systeme wird im Rahmen der funktionalen Anforderungen ein erstes fachliches Datenmodell erstellt, das als Grundlage des späteren Datenbankentwurfs dient. Das fachliche Datenmodell des Systems wird aus den Entitäten des Domänenmodells abgeleitet.

Die funktionalen Anforderungen sind die zentralen Vorgaben für die Systementwicklung.

## Use-Case Übersicht



## Use-Beschreibungen

|  |  |  |
| --- | --- | --- |
| **Eingabewort eintragen** | | |
| **Kennung** | | UC-1 |
| **Priorität** | | Hoch |
| **Kurzbeschreibung:** | | |
| Der Benutzer trägt ein Eingabewort in das dafür vorgesehene Eingabefeld auf der Oberfläche 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.~~ (In welchem Programm-Zustand soll der Benutzer die Möglichkeit haben den Use-Case auszuführen?) | | |
| **Nachbedingung(en):** | | |
| * Auf dem Band der Turingmaschine befindet sich ein gültiges Eingabewort. | | |
| **Normaler Ablauf:** | | |
|  | 1. Der Anwendungsfall beginnt, wenn der Benutzer das gewünschte Eingabewort in das dafür vorgesehene Eingabefeld einträgt. 2. Der Benutzer bestätigt seine Eingabe. 3. Das System prüft, ob eine Simulation bereits gestartet wurde. *Bem.: Ist der aktuelle Zustand q0?* 4. Das System überprüft anhand des Eingabealphabets das Eingabewort auf Gültigkeit. 5. Das Eingabewort wird auf das Band geschrieben.   **Ende**. | |
| **Ablauf-Varianten:** | | |
| 3a | Simulation bereits gestartet | |
|  | 1. Das System weist den Benutzer darauf hin, dass eine Simulation bereits ausgeführt wird. 2. Das System fragt den Benutzer, ob er fortsetzen **möchte**. Dadurch wird die laufende Simulation zurückgesetzt. 3. Der Benutzer wählt zwischen Fortsetzen und Abbrechen.   **Rückkehr nach**: 4 ***oder* Ende.** | |
| 4a | Ungültiges Eingabewort | |
|  | 1. Das System signalisiert, dass ein oder mehrere Symbole des Eingabeworts ungültig sind. 2. Der Benutzer hat die Möglichkeit das Eingabewort zu bearbeiten.   **Rückkehr nach**: 2 | |
| **Spezielle Anforderungen:** | | |
|  | | |
| **Zu klärende Punkte:** | | |
| Laut dieser Beschreibung kann das Eingabewort zu jeder Zeit geändert werden; während der Definition als auch im Simulationsmodus | | |

Drei Zustände, die unser Programm einnehmen kann:

Die letzten Beiden ggf. zusammenfassen

Bei der Einzelschrittausführung kann nicht so leicht festgestellt werden, wann die Simulation beendet ist!

|  |  |  |
| --- | --- | --- |
| **Simulation Einzelschritt durchführen** | | |
| **Kennung** | | UC-2 |
| **Priorität** | | Hoch |
| **Kurzbeschreibung:** | | |
| Der Benutzer gibt dem System den Startbefehl. Das System durchläuft die Simulation der gültigen Definition. Das System signalisiert, ob das Eingabewort zu einem akzeptierenden Endzustand geführt hat. | | |
| **Vorbedingung(en):** | | |
| * Das Programm befindet sich im Simulationsmodus. * Die Definition der Turingmaschine ist gültig. * Auf dem Band der Turingmaschine befindet sich ein gültiges Eingabewort. | | |
| **Nachbedingung(en):** | | |
| * Die Definition der Turingmaschine befindet sich im gleichen Zustand wie in der Vorbedingung. | | |
| **Normaler Ablauf:** | | |
|  | 1. Der Anwendungsfall beginnt, wenn der Benutzer den Befehl **Einzelschritt** auswählt. *Bem.: Anstelle des Benutzers kann auch das System dieses Ereignis beim automatischen Ablauf auslösen. (UC-7)* 2. Das System liest das Symbol unter dem LSK. 3. Das System sucht abhängig vom aktuellen Zustand und vom eingelesenen Symbol eine passende Übergangsregel. 4. Das System schreibt ein Symbol auf das Band gemäß gefundener Übergangsregel. 5. Das System verschiebt den LSK gemäß gefundenen Übergangsregel. *Bem.: Eine Position nach links oder rechts; auch keine Bewegung erlaubt.* 6. Das System prüft, ob sich die TM in einem akzeptierenden Endzustand befindet.   **Ende**. | |
| **Ablauf-Varianten:** | | |
| 3a | Der Benutzer hat die Option "Zustandsübergang markieren" aktiviert | |
|  | 1. Das System hebt nur den aktuellen Zustandsübergang farblich hervor. *Bem.: Vorherige Hervorhebungen werden hierbei entfernt.*   **Rückkehr nach**: 4 | |
| 3b | Das System hat keine passende Übergangsregel gefunden | |
|  | 1. Das System benachrichtigt den Benutzer, dass die Maschine nicht weiterarbeiten kann. 2. Das System beendet den automatischen Ablauf der Simulation, falls dieser aktiviert ist.   **Ende.** | |
| 6a | Die Turing-Maschine befindet sich in einem akzeptierenden Endzustand | |
|  | 1. Das System benachrichtigt den Benutzer, dass sich die Turing-Maschine in einem akzeptierenden Endzustand befindet. 2. Das System beendet den automatischen Ablauf der Simulation, falls dieser aktiviert ist.   **Ende.** | |
| **Spezielle Anforderungen:** | | |
| Nebenläufigkeit der Use-Cases erforderlich; hier **UC-7** für die Steuerung | | |
| **Zu klärende Punkte:** | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Simulation steuern** | | |
| **Kennung** | | UC-7 |
| **Priorität** | | Hoch |
| **Kurzbeschreibung:** | | |
| Dem Benutzer stehen zur Steuerung der Simulation verschiedene Steuerungsmöglichkeiten zur Verfügung. Das System reagiert auf die entsprechende Auswahl. | | |
| **Vorbedingung(en):** | | |
| * Das Programm befindet sich im Simulationsmodus. * Die Definition der Turingmaschine ist gültig. * Auf dem Band der Turingmaschine befindet sich ein gültiges Eingabewort. | | |
| **Nachbedingung(en):** | | |
| * Die Turingmaschine befindet sich in einem gültigen Zustand. * Die Definition der Turingmaschine ist gültig | | |
| **Normaler Ablauf:** | | |
|  | 1. Der Anwendungsfall startet, wenn der Benutzer einen Befehl zur Steuerung der Simulation absendet. | |
| **Ablauf-Varianten:** | | |
| 1a | Befehl **Zurücksetzen** („Abbruch“ ist nicht ganz so genau…) | |
|  | 1. Das System stoppt den automatischen Ablauf, falls dieser aktiviert ist. 2. Das System leert das Band und schreibt das (original) Eingabewort auf das Band. 3. Das System setzt den LSK über das erste Symbol des Eingabeworts auf dem Band. 4. Das System versetzt die TM in den Startzustand q0.   **Ende.** | |
| 1b | Befehl **Automatisch abspielen** (Hierunter fällt auch fortsetzen) | |
|  | 1. Das System startet den automatischen Ablauf der Simulation (von aktueller Position und Zustand) *Bem.: Nach Aufruf des Befehls wird sofort der in* ***UC-2*** *beschriebene Ablauf durchgeführt. Anschließend automatisch in der vom Benutzer eingestellten Geschwindigkeit wiederholt.*   **Ende.** | |
| 1c | Befehl **Pausieren** (für automatisches Abspielen) | |
|  | 1. Das System unterbricht den automatischen Ablauf der Simulation.   **Ende.** | |
| 1d | Befehl **Geschwindigkeitsanpassung** (für automatisches Abspielen) | |
|  | 1. Das System übernimmt für den automatischen Ablauf der Simulation die vom Benutzer eingestellte Geschwindigkeit.   **Ende.** | |
| 1e | Befehl **Einzelschritt durchführen** | |
|  | Befehl separat in **UC-2** beschrieben. | |
| 1f | Befehl **Zustandsübergang markieren** (aktivieren / deaktivieren) | |
|  | 1. Das System speichert die vom Benutzer gewünschte Einstellung. *Bem.: Wird in* ***UC-2*** *verwendet.*   **Ende.** | |
| **Spezielle Anforderungen:** | | |
|  | | |
| **Zu klärende Punkte:** | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Programmmodus-Wechsel** | | |
| **Kennung** | | UC-8 |
| **Priorität** | | Mittel |
| **Kurzbeschreibung:** | | |
| Der Benutzer hat die Möglichkeit das Programm in zwei Zustände zu schalten. Den Editiermodus, in welchem die Definition der Turing-Maschine vorgenommen wird. Außerdem den Simulationsmodus, welcher für den Ablauf der Turing-Maschine verwendet wird. | | |
| **Vorbedingung(en):** | | |
|  | | |
| **Nachbedingung(en):** | | |
|  | | |
| **Normaler Ablauf:** | | |
|  | 1. Der Anwendungsfall beginnt, indem der Benutzer den gewünschten Modus auswählt. | |
| **Ablauf-Varianten:** | | |
| 1a | Der Benutzer wählt: **Editiermodus** | |
|  | 1. Das System stoppt die ggf. laufende Simulation 2. Das System gibt die Bearbeitungsmöglichkeiten der Definition der TM frei.   **Ende** | |
| 1b | Der Benutzer wählt: **Simulationsmodus** | |
|  | 1. Das System sperrt die Bearbeitungsmöglichkeiten der Definition der TM. 2. Das System setzt den LSK über das erste Symbol des Eingabeworts auf dem Band. 3. Das System versetzt die TM in den Startzustand q0.   **Ende** | |
| **Spezielle Anforderungen:** | | |
|  | | |
| **Zu klärende Punkte:** | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Definition der Turingmaschine als Tabelle** | | |
| **Kennung** | | UC-3 |
| **Priorität** | | Hoch |
| **Kurzbeschreibung:** | | |
| Der Benutzer definiert alle Zustände, Zustandsübergänge sowie Anfangszustand und die akzeptierenden Zustände in einer Tabelle. Es kann dabei entweder eine neue Tabelle erstellt oder eine bestehende bearbeitet werden. Das System ermittelt aus der Tabelle das Bandalphabet und das Eingabealphabet. | | |
| **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 durch Betätigung der zugehörigen Schaltfläche. 4. Das System prüft, ob die Tabelle gültig ist. 5. Das System ermittelt aus der Tabelle das Bandalphabet und das Eingabealphabet.   **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. | |
|  | * Der Benutzer hat die Möglichkeit, folgende Eigenschaften festzulegen: * Kommentar * Eingabesymbol (gelesenes Zeichen) * LSK Bewegung (links oder rechts) * Startzustand * Akzeptierender Zustand   **Rückkehr nach**: 2 | |
| 4a | Ungültige Tabelle | |
|  | 1. Das System weist den Benutzer auf die gefundenen Fehler hin.   **Rückkehr nach**: 2 | |
| **Spezielle Anforderungen:** | | |
|  | | |
| **Zu klärende Punkte:** | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Definition der Turingmaschine als Zustandsdiagramm** | | |
| **Kennung** | | UC-4 |
| **Priorität** | | Mittel |
| **Kurzbeschreibung:** | | |
| Der Benutzer definiert alle Zustände, Zustandsübergänge sowie Anfangszustand und die akzeptierenden Zustände in einem Zustandsdiagramm. Es kann dabei entweder ein neues Zustandsdiagramm erstellt oder ein bestehendes bearbeitet werden. Das System ermittelt aus dem Zustandsdiagramm das Bandalphabet und das Eingabealphabet. | | |
| **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 durch Betätigung der zugehörigen Schaltfläche 4. Das System prüft, ob das Zustandsdiagramm gültig ist 5. Das System ermittelt aus dem Zustandsdiagramm das Bandalphabet und das Eingabealphabet   **Ende**. | |
| **Ablauf-Varianten:** | | |
| 2a | Der Benutzer fügt einen neuen Knoten (Zustand) hinzu. | |
|  | 1. Das System fügt einen neuen Knoten hinzu.   **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   **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 | |
| 4a | Ungültiges Zustandsdiagramm | |
|  | 1. Das System weist den Benutzer auf die gefundenen Fehler hin.   **Rückkehr nach:** 2 | |
| **Spezielle Anforderungen:** | | |
|  | | |
| **Zu klärende Punkte:** | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Definition importieren (Text-Datei)** | | |
| **Kennung** | | UC-5 |
| **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. Der Anwendungsfall beginnt, wenn der Benutzer auf die Schaltfläche „Import“ klickt.  2. Das System zeigt einen Dateiauswahldialog, in dem nur Text-Dateien gezeigt werden.  3. Der Benutzer wählt die Text-Datei aus.  4. Das System überprüft, ob die Text-Datei eine syntaktisch gültige Definition einer Turingmaschine darstellt.  5. Das System erstellt aus der Definition in der Text-Datei eine Turingmaschine.  **Ende**. | |
| **Ablauf-Varianten:** | | |
| 3a | Der Benutzer beendet den Auswahldialog | |
|  | **Der Use-Case endet** | |
| 4a | Datei enthält eine ungültige Definition | |
|  | 1. Das System erstellt keine Turingmaschine.  2. Das System fordert den Benutzer auf, eine andere Text-Datei zu wählen.  **Rückkehr nach**: 2 | |
|  |  | |
|  | 1. … 2. …   **Rückkehr nach**: | |
|  |  | |
|  | 1. … 2. …   **Rückkehr nach**: | |
|  |  | |
|  | 1. … 2. …   **Ende.** | |
|  |  | |
|  | 1. … 2. …   **Ende.** | |
|  |  | |
|  | 1. … 2. …   **Ende.** | |
| **Spezielle Anforderungen:** | | |
|  | | |
| **Zu klärende Punkte:** | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Definition exportieren (Text-Datei)** | | |
| **Kennung** | | UC-6 |
| **Priorität** | | Mittel |
| **Kurzbeschreibung:** | | |
| Der Benutzer gibt den Befehl die aktuelle Definition der Turingmaschine zu exportieren. Das System speichert diese in menschenlesbarer Form als Textdatei an dem gewünschten Ablageort. | | |
| **Vorbedingung(en):** | | |
| * Die Definition der Turingmaschine ist gültig. | | |
| **Nachbedingung(en):** | | |
|  | | |
| **Normaler Ablauf:** | | |
|  | 1. Der Anwendungsfall beginnt, wenn der Benutzer auf die Schaltfläche „Export“ klickt.  2. Das System zeigt einen Dateiauswahldialog, in dem alle Dateien gezeigt werden.  3. Der Benutzer benennt seine Text-Datei und speichert diese am gewünschten Ablageort als Text-Datei.  **Ende**. | |
| **Ablauf-Varianten:** | | |
| 3a | Speichervorgang kann nicht ausgeführt werden (z.B. Nicht genügend Speicher vorhanden, keine Berechtigung) | |
|  | 1. Speicherdialog verweigert das Speichern.   **Rückkehr nach**: 2, der Dialog bleibt bestehen | |
| 3b | Der Benutzer beendet den Auswahldialog | |
|  | **Der Use-Case endet.** | |
|  |  | |
|  | 1. … 2. …   **Rückkehr nach**: | |
|  |  | |
|  | 1. … 2. …   **Rückkehr nach**: | |
|  |  | |
|  | 1. … 2. …   **Ende.** | |
|  |  | |
|  | 1. … 2. …   **Ende.** | |
|  |  | |
|  | 1. … 2. …   **Ende.** | |
|  |  | |
|  | 1. … 2. …   **Ende.** | |
| **Spezielle Anforderungen:** | | |
|  | | |
| **Zu klärende Punkte:** | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Turingmaschine transformieren** | | |
| **Kennung** | | UC-8 |
| **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 q0 ist. Wird ein solcher Zustandsübergang gefunden, wird ein neuer Startzustand hinzugefügt, welcher die Bedingung erfüllt und die Funktion der TM unverändert lässt. 2. Das System stellt sicher, dass kein Übergang beginnend von einem akzeptierenden Zustand existiert. Wird ein solcher Übergang gefunden, wird ein neuer Zustand hinzugefügt, welcher die Bedingung erfüllt und die Funktion der TM unverändert lässt.   **Ende.** | |
| 3b | Transformation 2 wurde ausgewählt | |
|  | 1. Das System überprüft, ob es Zustandsübergänge gibt, bei dem der LSK stehen bleibt. Wird ein solcher Zustandsübergang gefunden, werden ein oder mehrere Zwischenzustände hinzugefügt. Der ursprüngliche Zustandsübergang wird durch den Zustandsübergang in einen Zwischenzustand ersetzt, sodass der LSK nach links oder rechts bewegt wird. Im Anschluss wird der LSK in den ursprünglichen Zustand zurückbewegt. Die Funktion der TM bleibt dabei unverändert.   **Ende.** | |
| 3c | Transformation 3 wurde ausgewählt | |
|  | 1. Das System überprüft, ob der LSK ein Leerzeichen schreiben möchte. Sollte dies der Fall sein, wird anstelle des Leerzeichens eine Raute (#) geschrieben.   **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. Für den Fall, dass man in einen Zustand mit einer Rechts- und einer Linksbewegung des LSK kommen kann, wird dieser Zustand zweigeteilt.   *Bem.: Falls ein akzeptierender Zustand zweigeteilt werden muss, werden beiden Teile als akzeptierender Zustand markiert.*   1. Die beiden Teilmengen werden farblich gekennzeichnet.   **Ende.** | |
| 3e | Transformation 5 wurde ausgewählt | |
|  | 1. Das System stellt sicher, dass es genau einen akzeptierenden Zustand gibt. Für den Fall, dass es mehrere akzeptierende Zustände gibt, wird ein neuer akzeptierender Zustand hinzugefügt, alle bisherigen akzeptierenden Zustände werden zu normalen Zuständen umgewandelt.   **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:** | | |
|  | | |

## (Sonstige) Funktionalität

Hier werden funktionale Anforderungen erfasst, die sich nicht vernünftig durch einen einzelnen Use-Case beschreiben lassen, weil sie z.B. „Querschnitt“-Funktionalität betreffen, die sich über viele Use-Cases erstreckt. Beispiele könnten sein:

|  |  |  |
| --- | --- | --- |
| **ID** | **Beschreibung** | **Querverweise** |
| FR-001 | Das System soll jede Fehlersituation dauerhaft protokollieren |  |
| FR-002 | Nutzung des Systems ist generell nur nach vorheriger Authentifizierung möglich |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Modell des Problembereichs (Konzeptionelles Datenmodell)

Hier wird ein Modell des Problembereichs („konzeptuelles Datenmodell“) in Form eines oder mehrerer UML-Klassendiagramme eingefügt. Das konzeptuelle Datenmodell wird oft nicht streng getrennt von den Anforderungen erarbeitet, weil ein genaues Verständnis z.B. der Use-Cases nicht ohne Verständnis der „Konzepte“ aus dem Problembereich möglich ist. Aus diesem Grund wird das konzeptionelle Modell an dieser Stelle zusammen mit den Use-Cases dokumentiert.

# Nicht-Funktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben Anforderungen an das System, die nicht-fachlicher Natur sind, jedoch entscheidend zur Anwendbarkeit des Systems beitragen. Sie definieren beispielsweise Qualitätsanforderungen, Sicherheitsanforderungen oder Performanceanforderungen.

Nicht-funktionale Anforderungen definieren grundlegende Eigenschaften eines Systems, die im Architekturentwurf berücksichtigt werden müssen. Sie können zur Abschätzung der Entwicklungskosten herangezogen werden und sollten, soweit möglich, messbar beschrieben sein.

Zur einfachen Strukturierung der Anforderungen werden diejenigen Anforderungen, die nicht eindeutig zu den funktionalen Anforderungen gehören, den nicht-funktionalen Anforderungen zugeordnet.

Die hier verwendete Einteilung unterscheidet verschiedene Arten von Anforderungen nach dem „FURPS“-Schema (**F**unctionality, **U**sability, **R**eliability, **P**erformance, **S**upportability), das auf Hewlett-Packard zurückgeht. Das FURPS-Schema ist hier noch um die Kategorie „Sonstige Einschränkungen“ erweitert. Anforderungen zur Funktionalität sind bereits im Kapitel 3 dokumentiert, in diesem Kapitel folgen lediglich alle restlichen Anforderungen.

Bei Bedarf kann dieses Schema zur Einteilung der Anforderungen auch durch ein anderes Schema (z.B. nach DIN ISO) ersetzt werden – wichtig ist nur, dass bei der Erfassung der Anforderungen überhaupt ein erprobtes Schema verwendet wird. Dies soll eine bessere Übersicht bieten und dazu beitragen, dass keine Anforderungen vergessen werden.

## Benutzbarkeit (Usability)

Hier werden Anforderungen erfasst, die die Benutzbarkeit („usability“ = Benutzbarkeit / Benutzerfreundlichkeit / Gebrauchstauglichkeit) des Systems betreffen. Hierzu zählen insbesondere Anforderungen zur (Software/Hardware)-Ergonomie („human factors“).

|  |  |  |
| --- | --- | --- |
| **ID** | **Beschreibung** | **Querverweise** |
| UR-001 | Das System soll über eine grafische Benutzerschnittstelle bedienbar sein. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Zuverlässigkeit (Reliability)

Hier werden Anforderungen erfasst, die die Zuverlässigkeit („reliability“ = Zuverlässigkeit) des Systems betreffen. Hierunter fallen insbesondere Anforderungen an die Wiederherstellbarkeit („recoverability“) und die Verfügbarkeit („availabilty“) des Systems. Die Wiederherstellbarkeit betrifft die Fähigkeit, bei Ausfall oder Störung das Leistungsniveau wieder zu erreichen und betroffene Daten wieder zu gewinnen. Eine Kenngröße in diesem Zusammenhang ist MTTR („**m**ean **t**ime **t**o **r**epair“), die die mittlere Zeit bis zur Wiederinbetriebnahme des Systems bei einem Ausfall angibt. Die Verfügbarkeit kann als Verhältnis zwischen der Zeit, in der das System funktionsfähig ist, und der Gesamtzeit angegeben werden:

V=MTBF / (MTBF+MTTR)

MTTR: mean time to repair (s.o.)

MTBF: **m**ean **t**ime **b**etween **f**ailures (also die mittlere Zeit zwischen zwei Ausfällen)

|  |  |  |
| --- | --- | --- |
| **ID** | **Beschreibung** | **Querverweise** |
| RR-001 | Das System speichert im Falle eines Absturzes die aktuelle Turingmaschine in einer Text-Datei ab. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Leistung (Performance)

Hierunter fallen Anforderungen an die Leistung („performance“) des Systems. Die Anforderungen beziehen sich insbesondere auf das Zeitverhalten (Ausführungsgeschwindigkeit, Antwortzeiten, Durchsatz) sowie auf das Verbrauchsverhalten (Anzahl der belegten Betriebsmittel und Dauer der Betriebsmittelbelegung).

|  |  |  |
| --- | --- | --- |
| **ID** | **Beschreibung** | **Querverweise** |
| PR-001 | Das System soll nicht länger als eine Sekunde benötigen, um auf Benutzereingaben zu reagieren. |  |
| PR-002 | Das System soll nicht länger als fünf Sekunden benötigen, um die Definition einer Turingmaschine einzulesen oder abzuspeichern. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Unterstützbarkeit (Supportability)

Hierunter fallen Anforderungen, die Bereiche wie Anpassbarkeit („adaptability“), Testbarkeit („testability“), Wartbarkeit („maintainability“), Erweiterbarkeit („extensibility“), Lokalisierbarkeit („localizability“ = Anpassbarkeit an verschiedene Sprach- und Kulturräume) betreffen.

|  |  |  |
| --- | --- | --- |
| **ID** | **Beschreibung** | **Querverweise** |
| SR-001 | Der angezeigte Text soll aus einer Datei entnommen werden, die äquivalente Formulierungen für andere Sprachen bereitstellt. |  |
| SR-002 | Das System soll aus Komponenten zusammengesetzt sein, die eine geringe Kopplung untereinander aufweisen und auf eine Aufgabe spezialisiert sind. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Sonstige Einschränkungen

In manchen Fällen können von vorneherein Einschränkungen („constraints“) für Entwurf, Implementierung, Schnittstellen und Hardware des geplanten Systems bestehen, die ebenfalls als Anforderungen zu berücksichtigen sind und das bisherige „FURPS“-Schema zu „FURPS+“ (vgl. [Lar], S. 88) erweitern.

### Schnittstellen

### Implementierung

* Verwendbare Programmiersprachen:
* C#
* Java
* Zielumgebung:
* Visual Studio Code 2019
* Eclipse

### Entwurf

|  |  |  |
| --- | --- | --- |
| **ID** | **Beschreibung** | **Querverweise** |
| DR-001 | Der Entwurf soll den Grundsatz von der Trennung zwischen model und view gemäß dem „Model-View-ViewModel“-Entwurfsmuster strikt einhalten. | UR-001 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Risikoakzeptanz

Für sicherheitskritische Systeme werden in diesem Thema Vorgaben für die Behandlung der Systemsicherheit festgelegt. Es wird aufgezeigt, welche Risiken im Rahmen des Systembetriebs bestehen, welche Schäden, oder auch welche Klassen von Schäden, mit welcher Wahrscheinlichkeit auftreten können und inwieweit das Eintreten eines Schadensfalls toleriert wird bzw. nicht mehr akzeptabel ist.

Die Risikoakzeptanz für die identifizierten möglichen Schadensfälle wird beispielsweise in Form einer Risikoakzeptanzmatrix dokumentiert. Die Matrix ist eine Vorgabe des Auftraggebers, in der er festlegt, bei welcher Schadensklasse und welcher Eintrittswahrscheinlichkeit er welche Risikoklasse akzeptiert.

Bei vielen Aufgabenstellungen in der Veranstaltung „Software-Projekte“ kann dieses Thema weggelassen werden.

Beim Abspeichern der Turingmaschine als Text-Datei ist es Aufgabe des Betriebssystems den Benutzer daran zu hindern, diese an Orten auf der Festplatte abzuspeichern, für die er keine Berechtigung besitzt. Das Programm übernimmt hierfür keine eigenständige Kontrolle und die Entwickler sind nicht für mögliche Folgeschäden belangbar, die durch das Abspeichern der Text-Datei an nicht erlaubten Orten entstehen. Weiterhin wird durch das Programm bei der Benennung der Text-Datei beim Abspeichern nicht überprüft, ob dieser Name möglicherweise mit wichtigen Dateien des Betriebssystems am selben Speicherort übereinstimmt und dieses hierdurch in seiner Ausführung gestört wird.

# Skizze der Gesamtsystemarchitektur

Das reine Aufstellen von Anwenderanforderungen ohne Überlegungen zu möglichen Lösungsräumen birgt die große Gefahr, unrealistische Anwenderanforderungen zu definieren. Für die Einordnung, Systematisierung, Kategorisierung und auch Priorisierung von Anwenderanforderungen ist ein Koordinierungsrahmen hilfreich, um die Visualisierung der Anwenderanforderungen zu erleichtern.

Diese Aufgabe kann eine Gesamtsystemarchitektur leisten, die die Sichtweise des Anwenders repräsentiert und nicht die technische Sichtweise des Systemanalytikers beziehungsweise des Systemarchitekten. Das heißt, es ist eine funktionale Systemarchitektur mit Einbettung in die funktionalen Abläufe von Nachbarsystemen zu erstellen. Eine technische Systemarchitektur ist in dieser frühen Phase kaum möglich.

Des Weiteren sind die Besonderheiten der Einsatzumgebung des neuen Systems zu beschreiben, um vor allem die Anforderungen an die Systemsicherheit berücksichtigen zu können.

…hier Ihren Text einfügen…

# 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 Ergebnisse 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(XYZ)“ (also dieses Dokument) mit funktionalen, nicht-funktionalen Anforderungen und konzeptionellem Datenmodell. | * Siehe Vorlage. * Wird bei Projektbeginn mit einer 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(XYZ)“. | * 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) |

# Abnahmekriterien

In der Veranstaltung „Software-Projekte“ 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.

# Glossar

|  |  |
| --- | --- |
| Begriff | Erklärung |
| Eingabealphabet | Besteht aus den einzelnen Zeichen, die vom Benutzer in die Tabelle oder das Zustandsdiagramm eingetragen wurden und dem leeren Zeichen. |
| Eingabewort | Die Zeichenkette, welche vom Benutzer auf das Band der Turingmaschine geschrieben werden soll. Dieses Wort besteht nur aus dem Eingabealphabet, aber ohne das leere Zeichen. |
| Gültige Definition | Die Turingmaschine enthält genau einen Anfangszustand, mindestens einen Endzustand und jedes Zeichen des Eingabealphabets enthält einen Zustandsübergang bestehend aus:   * Dem nächsten Zustand, in den übergeganen werden soll. * Dem neuen Zeichen, welches anstelle des alten geschrieben werden soll. * Der Richtung in die sich der LSK als nächstes bewegen soll. |
| Gültige Turningmaschine |  |
| Zielzustand | Kommt die Turingmaschine über einen Zustandübergang in diesen Zustand, so beendet diese die Programmausführung. |
|  |  |

# Abkürzungsverzeichnis

|  |  |
| --- | --- |
| Abkürzung | Erklärung |
| TM | Turingmaschine |
| LSK | Lese-/Schreibkopf |

# Literaturverzeichnis

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

# Abbildungsverzeichnis

1. V-Modell® ist eine geschützte Marke der Bundesrepublik Deutschland. [↑](#footnote-ref-1)