

# Spezifikation zum Projekt

## Hierarchische Petrinetze – Komposition

### 1. Funktionsspezifikation

#### 1.1 Gesamtfunktion:

Mit dem Programm soll die Erstellung und Komposition von hierarchischen Petrinetzen ermöglicht werden. Zum Einen kann mit Hilfe eines Editors ein solches Netz erstellt bzw. bereits vorhandene Netze können editiert werden. Zum Anderen soll auch die Hierarchie des Netzes verdeutlicht werden indem man die einzelnen Ebenen des Netzes getrennt betrachten kann oder aber das gesamte Netz als ein einzelnes darstellt. Zu guter Letzt sollen die erstellten bzw. modifizierten Netze auch persistent gespeichert werden können. Dies geschieht durch Konvertierung des Netzes in eine XML-Struktur.

#### 1.2 Funktionen:

( *betreffen den Editor* )

/F10/

**Geschäftsprozess:** Stelle Einfügen : Von Einfügen bis Attribute festlegen

**Ziel:** Dem hierarchischen Petrinetz wurde eine Stelle hinzugefügt

**Kategorie:** primär

**Vorbedingung:** das Programm befindet sich im Editiermodus

**Nachbedingung Erfolg:** hierarchisches Petrinetz enthält neue Stelle

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht eine neue Stelle einzufügen

**Beschreibung:**

1. neue Stelle in das hierarchische Petrinetz einfügen
2. Attributeditor der Stelle aufrufen
3. eingegebene Attribute der Stelle speichern
4. aktualisiertes hierarchisches Petrinetz anzeigen

**Erweiterung:** --

**Alternative:** --

/F20/

**Geschäftsprozess:** verfeinerbare Stelle einfügen: Von Einfügen bis Verfeinerungeditor öffnen

**Ziel:** Dem hierarchischen Petrinetz wurde eine verfeinerbare Stelle hinzugefügt

**Kategorie:** primär

**Vorbedingung:** das Programm befindet sich im Editiermodus

**Nachbedingung Erfolg:** hierarchisches Petrinetz enthält neue verfeinerbare Stelle

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:**

**Beschreibung:**

1. neue verfeinerbare Stelle in das hierarchische Petrinetz einfügen
2. Attributeditor der verfeinerbare Stelle öffnen
3. eingegebene Attribute der Stelle speichern
4. neuen Petrinetzeditor für die Verfeinerung öffnen
5. aktualisiertes hierarchisches Petrinetz anzeigen

**Erweiterung:**

**Alternative:**

4 a eine neue Instanz eines bestehenden hierarchischen Petrinetzes laden

/F30/

**Geschäftsprozess:** verfeinerbare Transition einfügen: Von Einfügen bis Verfeinerungseditor öffnen

**Ziel:** Dem hierarchischen Petrinetz wurde eine verfeinerbare Transition hinzugefügt

**Kategorie:** primär

**Vorbedingung:** das Programm befindet sich im Editiermodus

**Nachbedingung Erfolg:** hierarchisches Petrinetz enthält neue verfeinerbare Transition

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:**

**Beschreibung:**

1. neue Verfeinerbare Transition in das hierarchische Petrinetz einfügen
2. Attributeditor der verfeinerbare Transition öffnen
3. eingegebene Attribute der Transition speichern
4. neuen Petrinetzeditor für die Verfeinerung öffnen
5. aktualisiertes hierarchisches Petrinetz anzeigen

**Erweiterung:**

**Alternative:**

4 a eine neue Instanz eines bestehenden hierarchischen Petrinetzes laden

/F40/

**Geschäftsprozess:** Transition Einfügen : Von Einfügen bis Attribute festlegen

**Ziel:** Dem hierarchischen Petrinetz soll eine Transition hinzugefügt werden

**Kategorie:** primär

**Vorbedingung:** das Programm befindet sich im Editiermodus

**Nachbedingung Erfolg:** hierarchisches Petrinetz enthält neue Transition

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht eine neue Transition einzufügen

**Beschreibung:**

1. neue Transition in das hierarchische Petrinetz einfügen
2. Attributeditor der Transition aufrufen
3. eingegebene Attribute der Transition speichern
4. aktualisiertes hierarchisches Petrinetz anzeigen

**Erweiterung:** --

**Alternative:** --

/F50/

**Geschäftsprozess:** Kante Einfügen : Von Einfügen bis Attribute festlegen

**Ziel:** Dem hierarchischen Petrinetz wurde eine Kante hinzugefügt

**Kategorie:** primär

**Vorbedingung:** das Programm befindet sich im Editiermodus

**Nachbedingung Erfolg:** hierarchisches Petrinetz enthält neue Kante

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht eine neue Kante einzufügen

**Beschreibung:**

1. Auswahl einer Transition oder Stelle von der die Kante ausgeht
2. Auswahl einer Transition oder Stelle zu der die Kante hinführt
3. Attributeditor für die Kante öffnen
4. eingegebene Attribute der Kante speichern
5. aktualisiertes Petrinetz anzeigen

**Erweiterung:** --

**Alternative:** --

/F60/

**Geschäftsprozess:** Marken Einfügen : Von Stelle auswählen bis Markenanzahl festlegen

**Ziel:** Einer Stelle des hierarchischen Petrinetz werden Marken hinzugefügt

**Kategorie:** primär

**Vorbedingung:** das Programm befindet sich im Editiermodus

**Nachbedingung Erfolg:** Stelle hat Marken erhalten

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht Marken einzufügen

**Beschreibung:**

1. Auswahl einer Stelle
2. Angabe der Markenanzahl
3. Speichern der Markenanzahl
4. Anzeigen des aktualisierten hierarchischen Petrinetzes

**Erweiterung:** --

**Alternative:** --

/F70/

**Geschäftsprozess:** Objekt Löschen: Von Auswahl bis entfernen aus dem hierarchischen Petrinetz

**Ziel:** ein Objekt ( Stelle, Transition, Kante, verfeinerte Stelle oder verfeinerte Transition) aus dem Petrinetz entfernen

**Kategorie:** primär

**Vorbedingung:** das Programm befindet sich im Editiermodus

**Nachbedingung Erfolg:** ausgewähltes Objekt aus dem hierarchischen Petrinetz entfernt

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht ein Objekt aus dem hierarchischen Petrinetz zu entfernen

**Beschreibung:**

1. Auswahl des zu löschenden Objekts
2. Feststellen der Abhängigkeiten
3. Entfernen des Objekts
4. Aktualisierung der Abhängigkeiten
5. Anzeigen des aktualisierten hierarchischen Petrinetzes

**Erweiterung:** --

**Alternative:** --

/F80/

**Geschäftsprozess:** Komposition des hierarchischen Petrinetzes Von Auswahl der zu erweiternden Transition oder Stelle bis zur Anzeige der Komposition

**Ziel:** eine verfeinerte Stelle oder Transition auflösen

**Kategorie:** primär

**Vorbedingung:** das Programm befindet sich im Editiermodus

**Nachbedingung Erfolg:** ausgewählte Stelle oder Transition ist aufgelöst im entsprechenden Level angezeigt

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht eine ausgewählte verfeinerte Stelle oder Transition aufzulösen

**Beschreibung:**

1. Auswahl der Stelle oder Transition deren Komposition ausgeführt werden soll
2. Einsetzen in Vaterknoten
3. Anzeigen des hierarchischen Petrinetzes nach der Komposition der Stelle oder Transition

**Erweiterung:** --

**Alternative:** --

/F90/

**Geschäftsprozess:** Speichern des hierarchischen Petrinetzes Von URL Angabe bis persistente Speicherung

**Ziel:** ein entworfenes oder editiertes hierarchisches Petrinetz auf Festplatte gespeichert

**Kategorie:** primär

**Vorbedingung:** das Programm befindet sich im Editiermodus

**Nachbedingung Erfolg:** hierarchisches Petrinetz als Datei gespeichert

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht ein entworfenes oder editiertes hierarchisches Petrinetz auf Festplatte zu speichern

**Beschreibung:**

1. Auswahl eines Verzeichnisses
2. Eingabe eines Dateinamens

3. Speichern des hierarchischen Petrinetzes

**Erweiterung:** --

**Alternative:** --

/F100/

**Geschäftsprozess:** Laden eines hierarchischen Petrinetzes Von URL Angabe bis Anzeigen

**Ziel:** ein entworfenes oder editiertes hierarchisches Petrinetz auf Festplatte laden

**Kategorie:** primär

**Vorbedingung:** --

**Nachbedingung Erfolg:** gewünschtes hierarchisches Petrinetz wird angezeigt

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht ein entworfenes oder zu editiertes hierarchisches Petrinetz von Festplatte zu laden

**Beschreibung:**

1. Auswahl eines Verzeichnisses
2. Eingabe/Auswahl eines Dateinamens
3. Laden des hierarchischen Petrinetzes
4. Anzeigen des gewünschten hierarchischen Petrinetzes

**Erweiterung:** --

**Alternative:** --

/F110/

**Geschäftsprozess:** Editieren von Attributen einzelner Stellen, Transitionen oder Kanten Von Auswahl bis Änderung der Attributwerte

**Ziel:** Änderung einzelner Attributwerte

**Kategorie:** primär

**Vorbedingung:** das Programm befindet sich im Editiermodus

**Nachbedingung Erfolg:** ausgewählte Attribute sind verändert

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung und Attribute behalten alte Werte

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht Attribute einzelner Objekte zu verändern

**Beschreibung:**

1. Auswahl des gewünschten Objekts
2. Auswahl des zu editierenden Attributs
3. Ändern des Attributwertes
4. Anzeigen des aktualisierten hierarchischen Petrinetzes

**Erweiterung:** --

**Alternative:** --

/F120/

**Geschäftsprozess:** Komposition des gesamten hierarchischen Petrinetzes

**Ziel:** Komposition eines hierarchisches Netz bis auf Level 0

**Kategorie:** primär

**Vorbedingung:** das Programm befindet sich im Editiermodus

**Nachbedingung Erfolg:** alle verfeinerten Stellen und Transitionen sind bis auf Level 0 aufgelöst

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht alle verfeinerten Stellen und Transitionen aufzulösen

**Beschreibung:**

1. Berechnung des Level 0 Petrinetzes
2. Anzeigen des hierarchischen Petrinetzes nach der Komposition aller Stellen und Transitionen

**Erweiterung:** --

**Alternative:** --

( Simulation des Schaltverhaltens )

/F130/

**Geschäftsprozess:** Moduswechsel in Simulationsmodus Von Editiermodus zum Simulationsmodus

**Ziel:** Programm befindet sich im Simulationsmodus

**Kategorie:** primär

**Vorbedingung:** Programm befindet sich im Editiermodus

**Nachbedingung Erfolg:** Programm befindet sich im Simulationsmodus

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht in den Simulationsmodus zu gehen

**Beschreibung:**

1. Wechsel in den Simulationsmodus
2. Markierung der feuerbereiten Transitionen

**Erweiterung:** --

**Alternative:** --

/F140/

**Geschäftsprozess:** Transition feuern Von Transitionsauswahl bis Schritt ausführen

**Ziel:** Transition feuern lassen

**Kategorie:** primär

**Vorbedingung:** Programm befindet sich im Simulationsmodus, Transition ist feuerbar

**Nachbedingung Erfolg:** Transition hat gefeuert

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht eine Transition zu feuern

**Beschreibung:**

1. Ausführen des Feuern der Transition
2. Aktualisieren der Attribute der Vor- und Nachstellen
3. Anzeigen des aktualisierten hierarchischen Petrinetzes

**Erweiterung:** --

**Alternative:**

- 1 a wenn es sich um eine verfeinerte Transition handelt → Aufruf des entsprechenden Unternetzes
- 2 a wenn es sich um eine verfeinerte Stelle handelt → Aufruf des entsprechenden Unternetzes

/F150/

**Geschäftsprozess:** Moduswechsel in Editiermodus Von Simulationsmodus zum Editiermodus

**Ziel:** Programm befindet sich im Simulationsmodus

**Kategorie:** primär

**Vorbedingung:** Programm befindet sich im Simulationsmodus

**Nachbedingung Erfolg:** Programm befindet sich im Editiermodus

**Nachbedingung Fehlschlag:** Akteur erhält eine Fehlermeldung

**Akteur:** Angestellter des Instituts für Informatik Lehrstuhl Automaten und Sprachen der Universität Leipzig

**Auslösendes Ereignis:** der Akteur wünscht in den Editiermodus zu gehen

**Beschreibung:**

1. Wechsel in den Editiermodus
2. Anzeigen des hierarchischen Petrinetzes

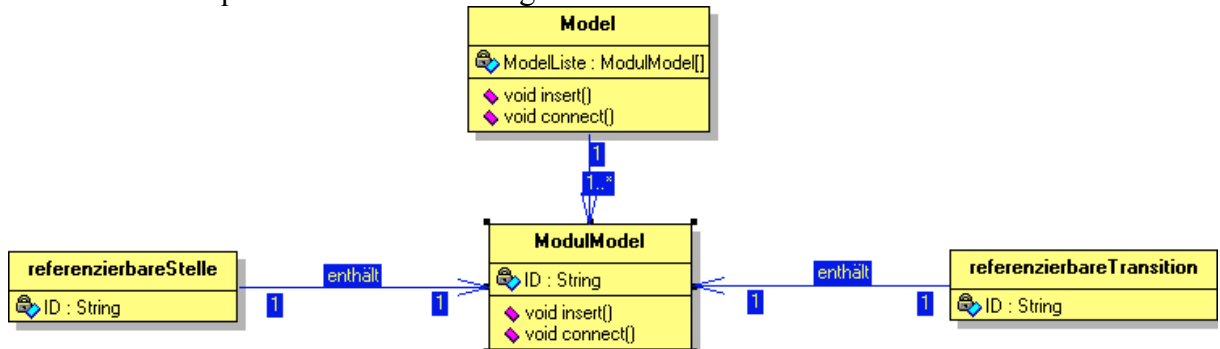
**Erweiterung:** --

**Alternative:** --

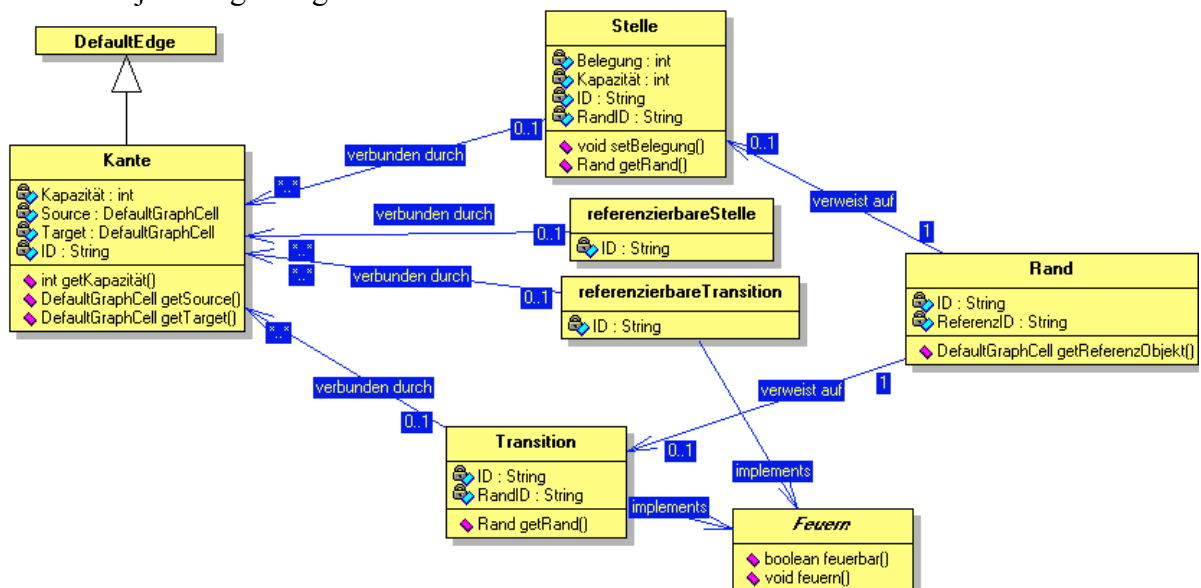
## 2. Datenspezifikation

- Verwendung von ModelViewControl (MVC)
- Petrinetz wird in einem hierarchischen Model gehalten
- Model enthält Array von ModulModels welche jeweils zu einer referenzierbaren Stelle oder Transition gehören und das entsprechende Subnetz repräsentieren
- sämtliche im View ausgelösten Editierfunktionen direkt auf dem Model ausgeführt (MVC)
- transiente Daten im Model
- StellenView mit Renderer gehört zum View → Darstellung

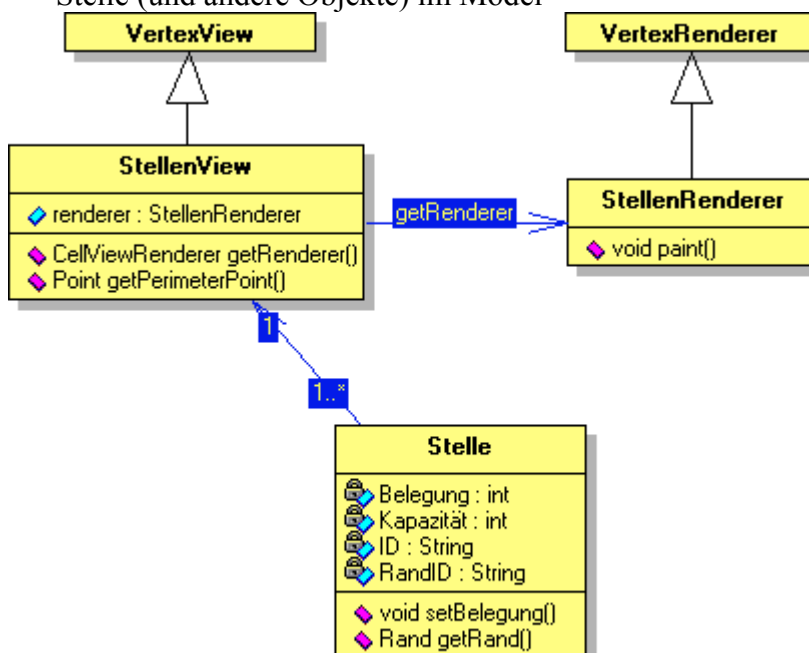
- Aufruf die äquivalente Funktion im gesuchten ModulModel



- alle Objekte implementieren Interface PetriObjekt → ID-Vergabe
- Stelle/Transition, referenzierteStelle/Transition und Rand erben von DefaultGraphCell
- die Transitionen implementieren Interface Feuern → Schaltmodus
- Kanten zwischen den Ebenen über Rand geleitet
- Randobjekt zeigt auf genau eine Stelle/Transition im Unternetz



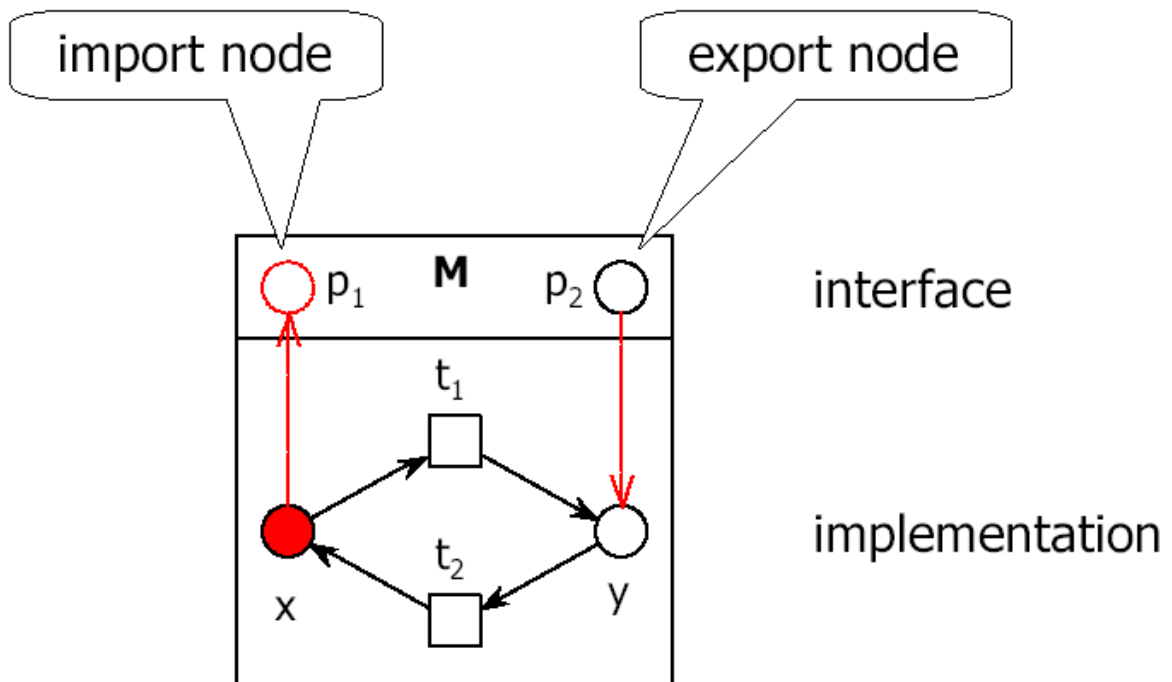
- Stelle (und andere Objekte) im Model



### 3. Ein-/ Ausgangs-Spezifikation

#### 3.1 Ein- und Ausgabeformate

- Speicherung der Netze in modular Petri Net Markup Language (modular PNML)
- auf XML basiertes Format



- import/export node jeweils ein Randobjekt
- Teilnetz entspricht einem ModulModel
- beim Laden Erzeugung von Model aus XML-Datei
- beim Speichern schreiben von XML-Datei aus Model

#### 3.2 Druck/Präsentationskonzept

- aktuell angezeigtes Teilnetz druckbar

### 4. Nutzerschnittstelle

#### 4.1 Sprache

- deutsch

#### 4.2 Bedienung

- Eingabe Maus und Tastatur
- Funktionen im Menü und als Button in der Toolbar
- deaktiviert falls Verwendung zum gegenwärtigen Zeitpunkt nicht gestattet
- Rechtsklick im Editorfenster → Editiermenü für gewähltes Objekt
- Drag and Drop für Stellen und Transitionen
- Kanten durch ziehen zum Zielobjekt erzeugen
- Unternetze durch Doppelklick auf referenzierbare Stelle/Transition oder entsprechende Funktion aufrufbar
- Ebenenwechsel per Button oder Menüeintrag

### 4.3 Oberfläche

- auf linker Bildschirmseite Fenster zum Editieren der Attribute eines Objektes
- Index sämtliche Objekte auswählbar
- Anzeige Hauptfenster eine Ebene der Hierarchie
- Anzeige eines Unternetz bei Auswahl
- Wechsel zwischen Editier- und Simulationsmodus
- im Simulationsmodus Markierung aller feuerbaren Transitionen
- Feuern einer referenzierbare Transition → Darstellung des Unternetzes
- manueller Wechsel auf eine andere Ebene möglich

### 4.4 Hilfefunktion

- umfassende HTML Hilfe über Menüeintrag

## 5. Dokumentationsspezifikation

### 5.1 Dokumentationsformat

- HTML-Dokument

### 5.2 Primär- und Sekundärdokumente

- Hilfedatei in HTML
- javadoc

### 5.3 Datendokumentation

- dokumentierte DTD

### 5.4 Protokollkonzept

- nicht vorgesehen

### 5.5 Verantwortlichkeiten:

Tätigkeit	Berger	Dassow	Sosnicki
Pflichtenheft		X	
Recherche ( fachlich )	X	X	
Recherche ( technisch )		X	X
Programmspezifikation	X		X
Implementierung	X ( GUI's + Komposition )	X ( pnml Datenhaltung )	X ( GUI's + Simulation )
Dokumentation	X		X

## 6. Umgebungsspezifikation

### 6.1 Betriebssystem

- plattformunabhängig

### 6.2 Hardwareanforderungen

- nicht sehr rechenaufwendig
- Maus und Tastatur
- grafische Benutzeroberfläche

### 6.3 Standardsoftware

- Java Virtual Machine (JVM)
- JDK ab Version 1.4.1

### 6.4 Bibliotheken

- JGraph ab Version 3

### 6.5 Gesetzregelung:

This program is free software; you can redistribute it and/or modify it. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

## 7. Qualitätsanforderungen

Produktqualität	Sehr gut	Gut	Normal	nicht relevant
<b>Funktionalität</b>				
Angemessenheit		x		
Richtigkeit	X			
Interoperabilität			x	
Ordnungsmäßigkeit	X			
Sicherheit			x	
<b>Zuverlässigkeit</b>				
Reife			x	
Fehlertoleranz			x	
Wiederherstellbarkeit			x	
<b>Benutzbarkeit</b>				
Verständlichkeit		x		
Erlernbarkeit		x		
Bedienbarkeit		x		
<b>Effizienz</b>				
Zeitverhalten			x	
Verbrauchsverhalten			x	
<b>Änderbarkeit</b>				
Analysierbarkeit		x		
Modifizierbarkeit			x	
Stabilität			x	
Prüfbarkeit			x	
<b>Übertragbarkeit</b>				
Anpassbarkeit			x	
Installierbarkeit			x	
Konformität			x	
Austauschbarkeit				x

## 8. Testspezifikation

### 8.1 Validierungskonzept

- Einhaltung der Regeln des Petrinetzes, speziell bei Aufbau und Simulation
- nur exakte hierarchische Petrinetze
- Simulation nach Schaltregeln

### 8.2 Funktionstest

- jede neu implementierte Funktion sogleich ausgiebig getestet
- fertiges Programm ausführliche Tests mit verschiedene Testfälle
- bei Auftreten von Fehlern umgehende Behebung und erneut testen

### 8.3 Datentest und Ein- /Ausgabetest

- Speichern/Laden mehrmals mit verschiedenen
- Ursprungnetz muss unverändert erscheinen
- bei Änderungen am Model erneutes Testen

## **9. Einsatzspezifikation**

### **9.1 Einsatzumgebung**

- speziell für Vorlesung Petrinetze an der Universität Leipzig

### **9.2 Änderungsdienst**

- nicht vorgesehen
- dokumentierte DTD für gespeichert Netze

### **9.3 Einsatzbeispiele**

- mit einigen Beispielnetzen
- Beispiele frei modifizierbar
- zur Verdeutlichung der Funktionalitäten
- sollen Lehrenden beim Unterrichten
- können von anderen Applikationen verwendet werden bei Übernahme der verwendeten Datenhaltung

### **9.4 Einsatzorganisation**

- als JAR-Archiv
- Beispiele fester Bestandteil und immer verfügbar
- Archiv entpacken und Programm von der Kommandozeile starten

## **10. Ausnahmespezifikation**

### **10.1 Bedienfehler**

- Dialogfenster mit der Fehlermeldung
- Funktionen nur durchführbar, wenn Verwendung zum gegenwärtigen Augenblick rechtens
- durch Deaktivieren des entsprechenden Buttons

### **10.2 Datenfehler**

- falls Fehler beim Speichern/Laden → Vorgang auf Userwunsch wiederholen
- Ursache wird Benutzer mitgeteilt

### **10.3 Programm- und Laufzeitfehler**

- sollten durch ausführliches Testen vermieden werden
- kein totaler Programmausfall

### **10.4 Fehlerprotokolle**

- -nicht vorgesehen