

Pflichtenheft

„Hierarchisches Petrinetz - Komposition“

1. Projekteinführung

Thema :

Petri-Netze im allgemeinen ermöglichen die Modellierung, Analyse und Simulation nebenläufiger Systeme. In Abhängigkeit des zu modellierenden Systems können verschieden mächtige Klassen von Petri-Netzen eingesetzt werden, auf die hier nicht weiter eingegangen werden soll. Jeden Netztyp kann man als hierarchisches Petri-Netz strukturieren, wobei wir uns auf *Stellen-Transitions-Netze* beschränken werden. Der Vorteil hierarchischer Petri-Netze liegt in der Abstraktion von umfangreichen, kleinen und evtl. unübersichtlichen Strukturen hin zu einfacheren Netzen in verschiedenen Ebenen.

Bearbeiter :

- Thorsten Berger <mail@thorsten-berger.net>
- Björn Dassow <mai99lfj@studserv.uni-leipzig.de>
- Stefan Sosnicki <stefan@sosnicki.de>

Veranlassung :

In der Vorlesung „Petrinetze 1“ wurde den Studenten die Möglichkeit eingeräumt durch die Bearbeitung eines Projektthemas eine prüfungsrelevante Leistung für eben diese Vorlesung zu erbringen.

Zielsetzung :

Herr Professor Siegmar Gerber und der Lehrstuhl Automaten und Sprachen sollen mit dem Programm „Hierarchisches Petrinetz“ in die Lage versetzt werden hierarchische Petrinetze zu entwerfen, ihr Schaltverhalten zu simulieren und eine Komposition eines solchen durchzuführen.

Musskriterien

- Erstellen von hierarchischen Petrinetzen
- Editieren von hierarchischen Petrinetzen
- Simulieren des Schaltverhaltens von hierarchischen Petrinetzen
- Persistente Speicherung von hierarchischen Petrinetzen
- Komposition von hierarchischen Petrinetzen

Wunschskriterien

- Bis jetzt sind noch keine bekannt.

Projektumfeld :**Produkteinsatz :**

Das Produkt dient zur Visualisierung des Lehrinhalts hierarchische Petrinetze im Vorlesungsbetrieb des Instituts für Informatik, Lehrstuhl Automaten und Sprachen der Universität Leipzig.

Anwendungsbereich :

Vorlesungsbetrieb des Lehrstuhls Automaten und Sprachen.

Zielgruppe :

Die Angestellten des Instituts theoretische Informatik.

Meilensteine :

- Anfertigen der ersten Pflichtenheftversion
(Abgabe per Email am Fr., den 21.11.03, Vorstellung Mo., der 24.11.03)
- Anfertigen der zweiten Pflichtenheftversion
(Abgabe per Email am Do., den 27.11.03)
- Anfertigung der Programmspezifikation
(Abgabe 1./2. Woche Januar 04)
- Anfertigen der Dokumentation und der Beispielnetze
- Implementieren
(Programmabgabe spätestens Ende des Wintersemesters 2003/04)

Eckdaten:

- grundsätzliche Implentierung in Java, als Applikation sprich ausführbares „jar“-Archiv
- Datenhaltung in pnml (Xml-Spezifikation)
- GUI in SWING
- Implementierung verschiedener Algorithmen, um die unter Zielsetzung genannten Dinge zu gewährleisten.
- Ein Hilfesystem im Programm (Html-basiert)
- Abgabe von o.g. Jar-Archiv, Beispiel Netze im pnml Format und die Projektdokumentation

2. Aufgabenstellung**Kurzbeschreibung :**

Unsere Gruppe soll im Rahmen des Projektbetriebes der Vorlesung „Petrinetze 1“ ein Programm entwerfen und implementieren, welches das Entwerfen, Editieren und die Simulation des Schaltverhaltens von hierarchischen Petrinetzen ermöglicht. Außerdem soll die Komposition einzelner Bestandteile oder aber ganzer hierarchischer Petrinetze möglich sein.

Gliederung :

- Ein Editor für hierarchische Petrinetze
- Ein Simulator für das Schaltverhalten
- Algorithmus für die Komposition

Ablauf :

Grundsätzlich muss zunächst ein für die Programmanforderungen hinreichendes Modell für hierarchische Petrinetze gefunden werden. Auf der Basis dieser Definition werden dann folgende Schritte in dieser Reihenfolge ausgeführt

- Modell für die Abbildung von hierarchischen Petrinetzen auf Java Klassen finden
- Erstellen eines GUI zum Entwerfen und Editieren
- Erstellen eines GUI zur Simulation des Schaltverhaltens
- Modell für die Abbildung der Java Klassen in pnml Dateien finden
- Datenhaltung in pnml Dateien realisieren

Datenformat :

Unser Programm wird alle Beispielnetze, sowie die Selbsterstellten sofern nicht explizit anders gewünscht immer in ein gesondertes Unterverzeichnis seines Stammverzeichnisses speichern und auch versuchen sie zunächst von dort zu öffnen. Alle Daten der hierarchischen Petrinetze sollen aus Kompatibilitätsgründen in pnml (Petri Net Markup Language). einem XML Dialekt gespeichert werden. Dies ermöglicht eine wesentlich erhöhte Kompatibilität zu anderen Petrinetz Tools.

Beschreibungssprache :

pnml (Petri Net Markup Language)

3. Schnittstellenbeschreibung**Einsatzbedingungen :**

Zur Nutzung dieses Programms muss grundsätzlich nur eine Java Virtual Machine auf dem Computer installiert sein. Alle eventuell verwendeten packages, die nicht Teil des java_1_4_2 releases sind, sind open source packages und im Lieferumfang des Programms enthalten. Natürlich muss der Computer über angeschlossene Eingabegeräte wie Tastatur und Mouse sowie Ausgabegeräte wie Monitor oder Videobeamer verfügen.

Hard & Software :

Der Rechner muss nur eine JVM ausführen können und über eine graphische Oberfläche verfügen.

Produktfunktionen :**Geschäftsprozesse**

(*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 Verfeinerungseditor 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. eingegeben 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 hierarchischen 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: --

Daten :

Die Daten des hierarchischen Petrinetzes werden in pnml Dateien gespeichert.

4. Qualitätsanforderungen :

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

5, Projektabwicklung

Organisation : siehe Gliederung / Ablauf

Zeitraumen : das Wintersemester 2003/04

Durchführung / Verantwortlichkeiten :

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