Modellierung, Simulation und Analyse mit dem Petri-Netz-Tool POSEIDON

Kurt Lautenbach, Jörg Müller und Stephan Philippi

Universität in Koblenz, Universitätsstr. 1, 56075 Koblenz
email{laut,georgm,philippi}@uni-koblenz.de
http://www.uni-koblenz.de/∼ag-pn

Abstract: Gegenstand des vorliegenden Artikels ist die Vorstellung des aktuellen Entwicklungstandes von POSEIDON, einem Werkzeug zur Modellierung, Simulation und Analyse von Petri-Netzen. Die breite Palette verfügbarer Simulations- und Analysealgorithmen sowie die konsequente Ausrichtung der Bedienkonzepte an die Bedürfnisse akademischer Anwender, prädestinieren POSEIDON für die Nutzung im wissenschaftlichen Bereich. Die modulare und offene Konzeption sowie die plattformunabhängige Implementierung erlauben hierbei den Einsatz in nahezu beliebigen Anwendungsgebieten.

1 Einleitung

Das Petri-Netz-Tool POSEIDON wird seit mehreren Jahren an der Universität in Koblenz am Institut für Softwaretechnik entwickelt. Es stellt eine Plattform für Studenten und Mitarbeiter der Universität dar, mit deren Hilfe die Entwicklung von und das experimentieren mit neuen Analyse- und Simulationsalgorithmen aus dem Bereich der Petri-Netz-Theorie ermöglicht wird. Eines der Hauptziele von POSEIDON ist somit die Schaffung eines Rahmenwerkes, mit dessen Hilfe Studien- und Diplomarbeiten sowie Dissertationen aus dem Bereich der Petri-Netze zu einem umfassenden und leistungsfähigen Tool integriert werden können. Aufgrund der konsequenten Ausrichtung der Konzeption des Werkzeuges auf dieses Ziel, kann eine nahezu beliebige Bandbreite von Themen mit POSEIDON abgedeckt werden, z.B. die Diagnostik mit Petri-Netzen ([Kru01]) oder die Generierung von Steuerungen aus Petri-Netzen ([Rau02]).

Hauptanliegen dieses Artikels ist es, POSEIDON als Werkzeug für die praktische Arbeit mit Petri-Netzen vorzustellen. Vor diesem Hintergrund wird der Beschreibung der Editor-, der Simulator- und der Analysekomponente im nächsten Abschnitt breiter Raum gewidmet. Im dritten Kapitel wird daraufhin der Einsatz von POSEIDON zur Fehlerfallsimulation auf der Basis von Petri-Netzen anhand eines konkreten Beispiels aus der Automobilindustrie umrissen. Im Anschluß wird daraufhin kurz der Entwurfsprozess und die Implementierung des Werkzeuges beschrieben, bevor abschließend ein Ausblick auf weitere Entwicklungen gegeben wird.

2 Arbeiten mit Poseidon

Modelle realer Systeme bestehen im allgemeinen aus mehreren Netzen. Derart zusammengehörige Teilnetze eines Systems werden in POSEIDON innerhalb von Projekten organisiert (siehe Abbildung 1).

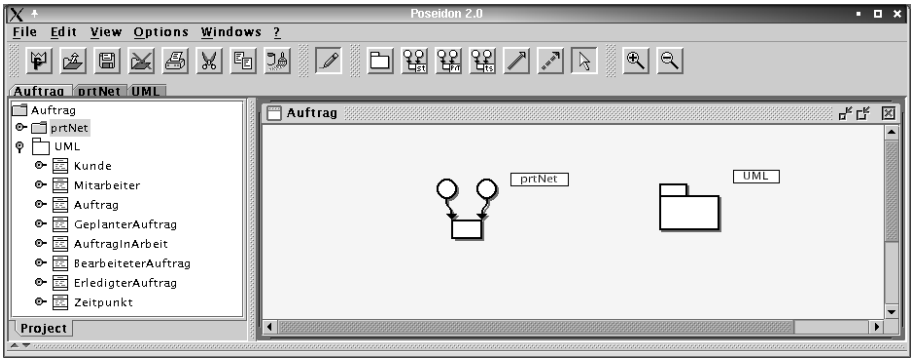


Abbildung 1 stellt die oberste Ebene des Projektes “Auftrag” dar. Dieses Projekt besteht aus einem Pr/T-Netz und einem UML-Klassendiagramm. Der im linken Bereich gegebene Baum zur Projektnavigation beinhaltet Repräsentanten für die verschiedenen Klassen der statischen Projektsicht. In der Toolbar werden Werkzeuge zur Verfügung gestellt, die u.a. die Möglichkeit bieten innerhalb des betrachteten Projektes weitere UML-Klassendiagramme und Netzmodelle anzulegen.

2.1 Die Editorkomponente

In POSEIDON stehen für verschiedene Klassen von Petri-Netzen wie S/T-Netze, Pr/T-Netze und Zeitstempelnetze (siehe [Lau99])) graphische Editoren zur Verfügung, mit denen Modelle anschaulich erzeugt und manipuliert werden können (siehe Abbildung 2). Die verschiedenen Editoren sind hierbei syntaxgesteuert, so daß es dem Nutzer nicht möglich ist a) eine Struktur einzugeben, die kein Netz des gewählten Typs darstellt und b) eine Modifikation an einem gegebenen Modell vorzunehmen, die zu einem solchen Resultat führt. Für alle Netzklassen ist dabei das Erstellen von hierarchischen Modellen mit Hilfe von *Supertransitionen* möglich (siehe [HJS90]). Da auf intuitive und übereinstimmende Bedienkonzepte bei den verschiedenen Editoren Wert gelegt wurde, läßt sich mit diesen auf einfache und einheitliche Weise arbeiten.

Der hier beschriebene Teil von POSEIDON wurde in der Vergangenheit vorwiegend im akademischen Bereich eingesetzt. In diesem Umfeld werden neue Erkenntnisse häufig durch das Studium (abstrakter) Netze mit sehr spezifischen Eigenschaften gewonnen. Hierbei wird das zu inspizierende Netz oftmals in Abhängigkeit bestimmter Analyseergebnisse

modifiziert. Um dieser Vorgehensweise Rechnung zu tragen, besteht in POSEIDON die Möglichkeit, Analyseergebnisse auf einfache Art und Weise während des editierens sichtbar zu halten. Wurde beispielsweise ein Deadlock berechnet und dieser soll durch eine entsprechend markierte Stelle geschützt werden, so müssen der Deadlock selbst, sein Defekt und die Anfangsmarkierung bekannt sein (siehe [LR96]). In POSEIDON können daher die benötigten Informationen aus der vorherigen Analyse übernommen und während des editierens sichtbar gehalten werden.

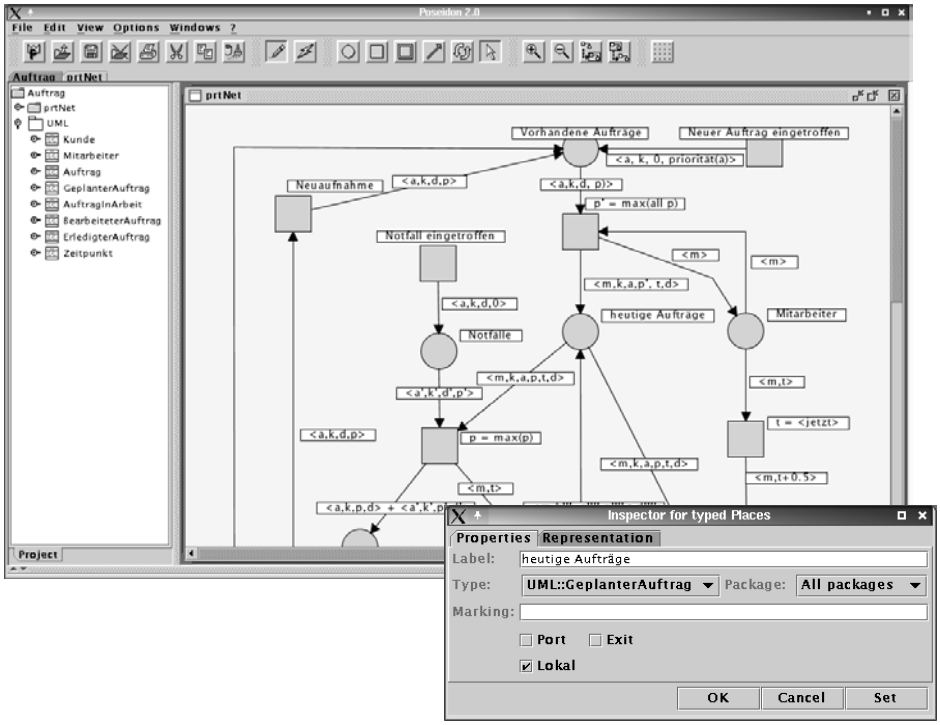


Abbildung 2 zeigt die Editorkomponente von POSEIDON. Im Hauptfenster ist ein Ausschnitt eines Pr/T-Netz-Modells zur Auftragsbearbeitung im Kundendienst dargestellt (hier: POSEIDON in der Anwendung zur Spezifikation von Logistik-Management-Systemen in der Arbeitsgruppe “Unternehmensmodellierung” der Universität in Koblenz (siehe [JF01])). In der Toolbar der betrachteten Komponente sind nun Editor-spezifische Bedienelemente eingeblendet. Ebenso ist der Inspektor zur Stelle “heutige Aufträge” dargestellt. Dieser zeigt an, daß Objekte auf dieser Stelle vom Typ “GeplanterAuftrag” sind.

Sind die durch ein System fließenden Informationen komplexer, strukturierter Natur, so sind die Stellen in einem Modell entsprechend zu typisieren. Dies wird mit Hilfe eines

Editors für UML-Klassendiagramme ermöglicht - die Typisierung der Stellen in einem Petri-Netz erfolgt somit durch UML-Klassen.

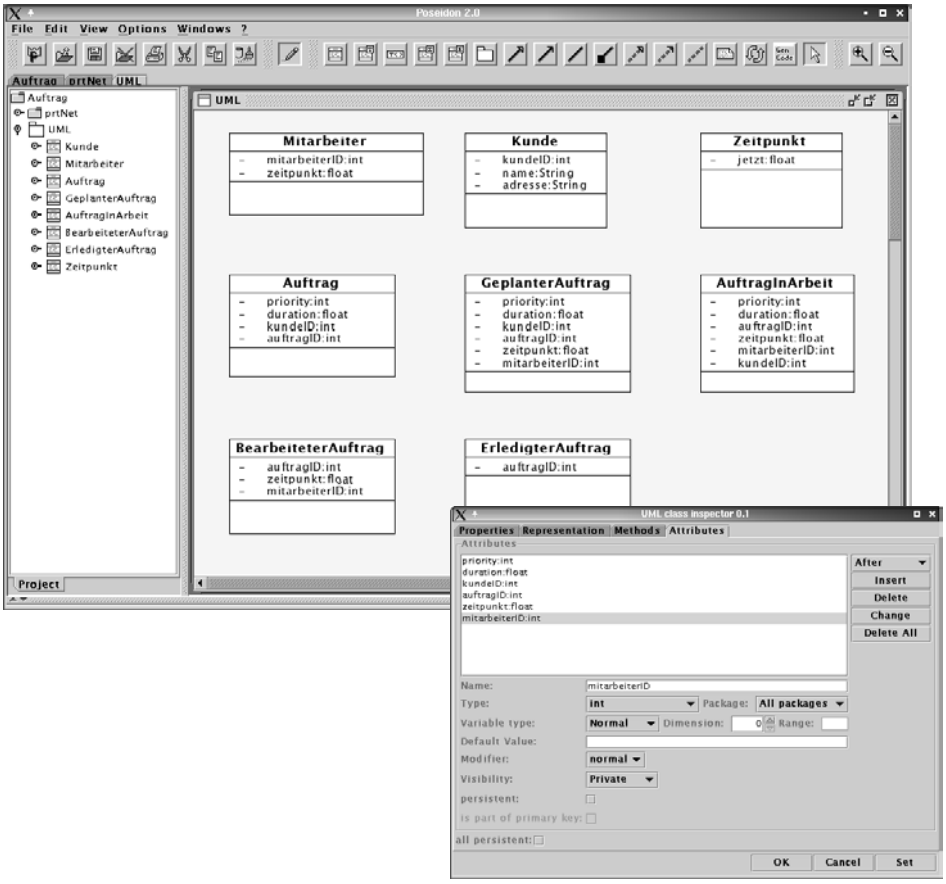


Abbildung 3 gibt einen Überblick über die Typisierung der Stellen im betrachteten Anwendungsbeispiel der Auftragsbearbeitung. Die Typdefinition erfolgt unter Verwendung eines Editors für UML-Klassendiagramme, wobei die spezifischen Eigenschaften der einzelnen Klassen mit Hilfe von Inspektoren editiert werden.

2.2 Die Simulatorkomponente

Für höhere Petri-Netze mit komplexen Markenstrukturen stehen formale Analysemöglichkeiten nur in sehr begrenztem Umfang zur Verfügung ([MV85]). Mit der Simulation von

Modellen können jedoch bei durchdachter Anlage der Simulationsexperimente punktuell Fehler im modellierten System aufgespürt werden.

In POSEIDON stehen u.a. Simulatoren für S/T-Netze, Pr/T-Netze und Zeitstempelnetze zur Verfügung (siehe Abbildung 4). In einem gegebenen Netz werden hierbei ggf. Anfangsparameter (z.B. Anfangsmarkierung, Kapazitäten, Zeitbewertungen) gesetzt und Funktionen zu Transitionen spezifiziert. Anschließend können mögliche Abläufe visualisiert werden. Der Markenfluß läßt sich dann im Netzmodell verfolgen, da aktivierte Transitionen hervorgehoben werden. Ebenso ist eine Simulation entgegen der Flußrichtung möglich. Durch eine solche "Rückwärts-Simulation" kann nach Zuständen gesucht werden, von denen ausgehend ein bestimmter (unliebsamer) Zustand erreichbar ist. In diesem Zusammenhang kann auch von einer Vorstufe zur Diagnostik mit Petri-Netzen gesprochen werden (siehe [Kru01]). Während der Simulation auftretende Konflikte können sowohl interaktiv als auch automatisiert gelöst werden.

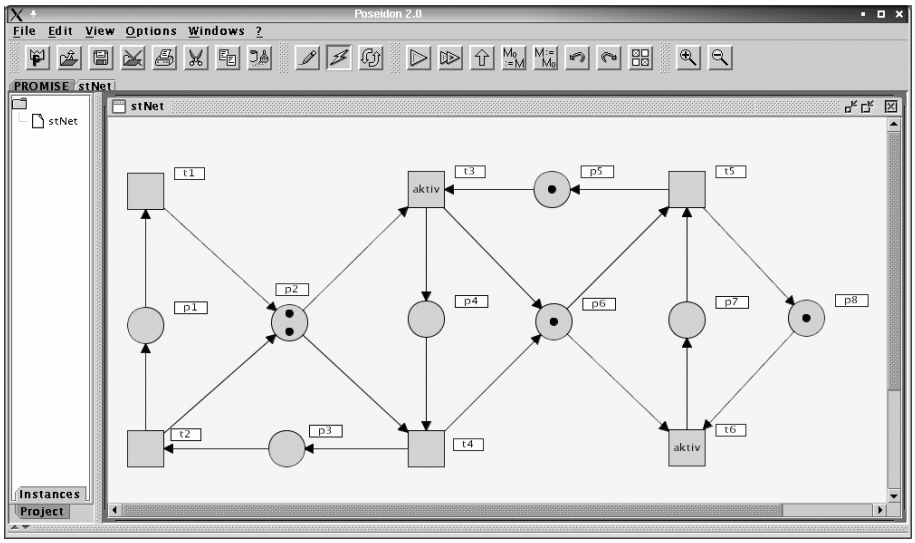


Abbildung 4 zeigt den Simulator für S/T-Netze. Unter der dargestellten Markierung sind die Transitionen t_3 und t_6 aktiviert. Der Simulator kann sowohl über die Tastatur als auch mit den Simulator-spezifischen Bedienelementen in der Toolbar gesteuert werden.

2.3 Die Analysekomponente

Petri-Netze unterscheiden sich durch ihre formale Basis wesentlich von einer Vielzahl anderer Modellierungssprachen. So können beispielsweise mit Hilfe mathematischer Verfahren aus der Struktur von Modellen Aussagen über deren Dynamik abgeleitet werden.

Da gerade die strukturelle Analyse von Petri-Netzen einen Schwerpunkt der Forschung an der Universität in Koblenz darstellt, stehen derartige Verfahren auch im Mittelpunkt der im weiteren zu betrachtenden Analysekomponente von POSEIDON (siehe Abbildung 5).

Strukturelle Analysemethoden sind fast ausschließlich für S/T-Netze bekannt (die wenigen Ausnahmen für höhere Netze werden weiter unten behandelt). Die in POSEIDON implementierten Algorithmen für S/T-Netze lassen sich in Petri-Netz-spezifische Algorithmen und allgemeine Graphenalgorithmen differenzieren. Letztere werden z.B. zur Identifizierung besonderer Klassen von S/T-Netzen verwendet. Betrachten wir jedoch zunächst die Petri-Netz-spezifischen Algorithmen: Für die gängigen Analyseverfahren allgemeiner S/T-Netze (Deadlocks, Traps, D- und T-Systeme, Invarianten, u.a.) berechnet POSEIDON minimale, ganzzahlige und nicht negative Erzeugendensysteme (siehe [Jax85]). Neben diesen grundlegenden Verfahren sind in POSEIDON weitere implementiert - so beispielsweise zwei Ansätze zum Testen des Schutzes von Deadlocks. Mit Hilfe der Deadlock-Trap-Eigenschaft können Deadlocks bekanntlich "von innen" mit Hilfe von Traps geschützt werden (siehe [Com72]). Durch kontrollierende Invarianten können Deadlocks dagegen "von aussen" durch Invarianten geschützt werden (siehe [LR96]). Oftmals wird auch nach speziellen Vektoren gesucht, z.B. bestimmten Bedingungen hinsichtlich der Schalthäufigkeit von Transitionen gerecht werdende T-Invarianten. In POSEIDON stehen zum Lösen solcher Probleme mit *Cutting Plane* und *Branch and Bound* zwei verschiedene Verfahren zur Verfügung. Die mit diesen Algorithmen gefundenen Lösungen des (bedingten) inhomogenen Gleichungssystems werden automatisch mit dem Erzeugendensystem des entsprechenden homogenen Systems kombiniert. Dadurch wird es möglich, alle Lösungen eines (bedingten) inhomogenen Gleichungssystems anzugeben.

Erfüllt ein S/T-Netz bestimmte graphenspezifische Anforderungen, so lassen sich zusammen mit den oben genannten Analyseverfahren weitergehende Aussagen über die Dynamik des modellierten Systems treffen. Aus diesem Grund sind in POSEIDON neben den Petri-Netz-spezifischen Algorithmen auch allgemeine Graphenalgorithmen implementiert. Hierbei handelt es sich zum einen um die Möglichkeit, die Klasse eines gegebenen Netzes zu bestimmen - hierbei wird auf rein strukturelle Charakteristika einer Reihe von speziellen Klassen getestet (z.B. (extended)free-choice, (extended)synchronization-graphs, ...). Zum anderen stehen Algorithmen zur Erkennung von Teilgraphen mit bestimmten Eigenschaften zur Verfügung - so z.B. Verfahren zum Finden von Kreisen oder (starken) Zusammenhangskomponenten.

Die in POSEIDON implementierten Algorithmen für höhere Netze teilen sich ebenfalls in zwei Bereiche. Zum einen stehen die Algorithmen für S/T-Netze auch für höhere Netze zur Verfügung. Durch anonymisieren von Tokens und ausschließlicher Beachtung der Kardinalität von Kantenanschriften – wir bezeichnen ein solches aus einem höheren Netz entstandenes S/T-Netz als *schwarzes* Netz – entsteht ein S/T-Netz, das in gewöhnlicher S/T-Manier analysiert werden kann. Die auf diesem Weg ermittelten Ergebnisse sind i.a. notwendige Bedingungen für die entsprechende Eigenschaft im initialen (höheren) Netz. Beispielsweise ist die Existenz einer S/T-Invarianten in einem schwarzen Netz notwendige Voraussetzung für die Existenz einer höheren Invariante im ursprünglichen Netz.

Den zweiten Bereich von Analysemethoden in höheren Netzen stellen spezielle Algorithmen für diese dar. Hier sind in der Theorie nur wenige rein strukturelle Verfahren zu finden. Implementiert sind aus diesem Bereich daher lediglich zwei Algorithmen, mit deren Hilfe zwei Arten von strukturellen Invarianten in höheren Netzen gefunden werden können (siehe [MV85]).

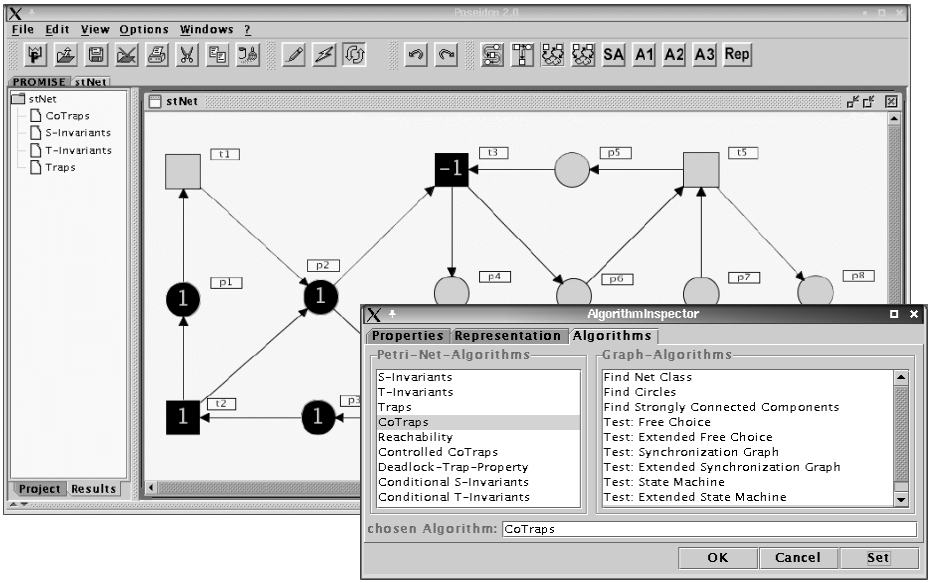
Die von POSEIDON bei einer Analyse gelieferten Ergebnisse sind unmittelbar im Netz sichtbar. Dabei werden die Einträge der Ergebnisvektoren in den entsprechenden Knoten angezeigt - beispielsweise werden die Werte einer S-Invarianten in den zugehörigen Stellen dargestellt. Auch solche Darstellungen von Ergebnissen innerhalb eines Netzes lassen sich in einer Datei für die weitere Verarbeitung speichern. Die Navigation durch eine Ergebnisliste erfolgt auf einfache Weise mit Hilfe entsprechender Bedienelemente.

Oftmals soll nur ein Teil der Ergebnismenge einer Analyse näher untersucht werden, da die anderen Ergebnisse aus Sicht des Anwenders weniger “interessant“ oder gar trivial sind. Wird beispielsweise ein Netz mit einer großen Zahl von Regulationskreisen untersucht, so findet eine Analyse schon aufgrund dieses Sachverhaltes eine Menge von S-Invarianten, die i.a. jedoch nicht weiter betrachtet werden. POSEIDON trägt diesem Umstand Rechnung, indem Ergebnisse interaktiv durch Auswahl aus der Ergebnisliste ausgeblendet werden können. Diese Organisation der Ergebnisse durch den Anwender erlaubt es diesem, sich leichter auf die aus seiner Sicht “interessanten“ und weiter zu untersuchenden Aspekte des Netzes zu konzentrieren.

Ebenso werden mit einem Projekt alle zu diesem berechneten Ergebnisse abgespeichert, so daß sie beim nächsten laden sofort zur Verfügung stehen und nicht abermals berechnet werden müssen. Durch die exponentielle Laufzeit einiger der verfügbaren Algorithmen sind die Vorzüge der Ergebnisspeicherung innerhalb eines Projektes leicht erkennbar.

Ähnlich wie die Analyseergebnisse lassen sich auch die durch POSEIDON bereitgestellten Algorithmen organisieren. Die Anwendung des Werkzeuges sowohl im akademischen als auch im anwendungsorientierten Bereich hat gezeigt, daß die Menge der häufig zugegriffen Algorithmen stark variiert. Um einerseits nun diese Algorithmen über spezielle Bedienelemente direkt zugreifbar zu machen, andererseits jedoch die graphische Oberfläche nicht mit einem Element für jede Funktion zu überladen, wurden mehrere frei konfigurierbare Schaltflächen in POSEIDON eingeführt. Dem Anwender stehen somit zusätzlich den Bedienelementen zur Auswahl der Berechnung von S- und T-Invarianten, Deadlocks und Traps auch mehrere frei konfigurierbare Schaltflächen zur Verfügung, deren konkrete Verknüpfung mit einem Algorithmus durch Tooltips anzeigt wird.

Läßt man POSEIDON zu einem gegebenen Netzsystem einen sog. *report* erstellen, so wendet das Werkzeug alle zuvor ausgewählten Algorithmen automatisch an und speichert die berechneten Ergebnisse (auf Wunsch mit dem dazugehörenden Netz) in einer separaten und editierbaren Datei ab.



In Abbildung 5 ist der Deadlock bestehend aus den Stellen p_1, p_2 und p_3 markiert (die Stellen sind schwarz unterlegt und mit '1' gekennzeichnet). Die Werte innerhalb der Transitionen t_2 und t_3, '1' und '-1' entsprechend, geben den Defekt dieser Transitionen bezüglich des angezeigten Deadlocks an. Mit Hilfe des Inspektors für Algorithmen können entweder Berechnungen direkt angestoßen oder diese den konfigurierbaren Schaltflächen der Toolbar zugeordnet werden.

3 Fallbeispiel: Elektronisch-mechanische Bremse

In diesem Kapitel wird eine der vielfältigen Anwendungsmöglichkeit von POSEIDON vorgestellt. Konkret handelt es sich mit dem Modell einer elektronisch-mechanische Bremse um ein Beispiel aus der modernen Kraftfahrzeugtechnik. Im folgenden wird das Anwendungssystem sowie die der Modellierung zugrunde liegende Netzklasse grob beschrieben, nähere Informationen hierzu sind zu finden in [MPS01].

3.1 Einordnung des Projektes

In der modernen Automobiltechnik werden zunehmend mechanische Komponenten traditioneller Bauart durch elektronisch-mechanische Fahrzeugsysteme ohne direkte mechanische Verbindung zwischen Bedienelementen und Aktoren ersetzt. Für diese 'x-by-wire' Technologie gibt es im Fahrzeugbau eine Reihe von Anwendungsmöglichkeiten, z.B. elektronisch gesteuerte Fahrwerke mit den Teilkomponenten Bremse, Lenkung und Federung.

Im einzelnen werden hierbei die Kommandos des Fahrers wie 'Bremsen' oder 'Räder einschlagen' nicht wie in der traditionellen Fahrzeugtechnik üblich über mechanische Komponenten zu den entsprechenden Aktoren weitergegeben, sondern ausschließlich über kabelgebundenen Datentransport. In diesem Kontext unterstützt POSEIDON (prototypisch) den Architekturentwurf und die zeitbewertete Simulation von 'x-by-wire' Systemen. Im weiteren wird der Aspekt der Architekturmodellierung mit POSEIDON in diesem Bereich anhand eines elektronisch-mechanischen Bremssystems (EMB) skizziert.

3.2 Das Szenario

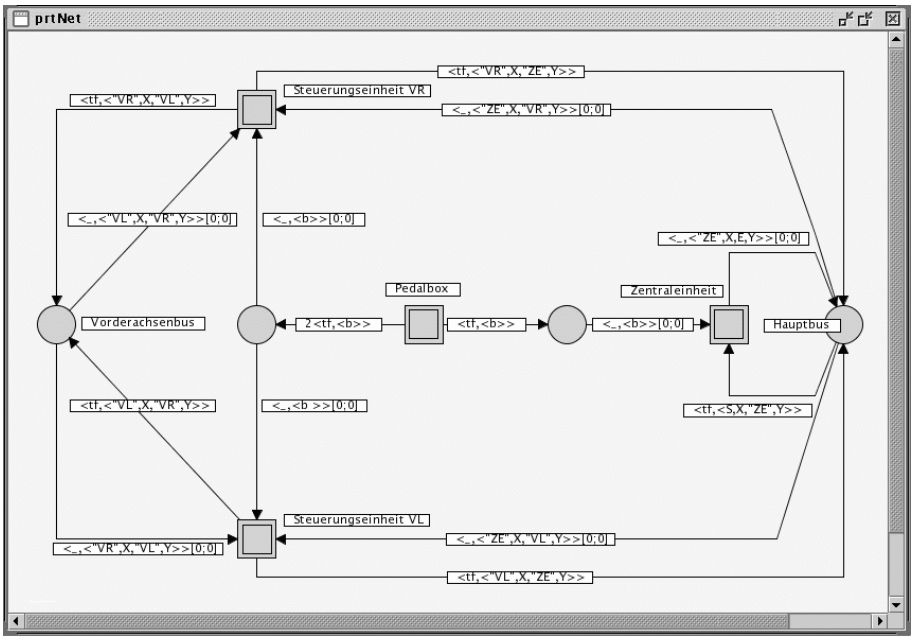
Aufgrund der zentralen sicherheitstechnischen Bedeutung des Bremssystems in einem Kraftfahrzeug müssen elektronisch-mechanisch Komponenten in diesem Bereich offensichtlich höchsten Anforderungen bzgl. des Echtzeitverhaltens, der Fehlertoleranz und der Robustheit genügen. Idealerweise werden diese Eigenschaften bereits auf der Architekturebene mit Hilfe von Modellen des Systems überprüft. Eine geeignete Vorgehensweise zu einer derartigen Designabsicherung ist eine Fehlerfallsimulation. Mit POSEIDON wird dem Entwickler von 'x-by-wire' Systemen die Möglichkeit gegeben, die Auswirkungen verschiedener Einzel- und Kombinationsfehler auf den jeweiligen Entwurf bereits im Vorfeld der Systemerstellung auf der Basis von Architekturmodellen zu untersuchen. Die konkrete Funktionalität der in einem Entwurf enthaltenen Komponenten wird hierbei auf der abstrakten Architekturebene nicht berücksichtigt. Von zentraler Bedeutung für die korrekte Funktionalität einer Komponente in einem verteilten Echtzeitsystem wie der EMB ist die Verfügbarkeit relevanter Informationen. Offensichtlich arbeiten Komponenten in einem derartigen System am zuverlässigsten, wenn alle lokal benötigten Informationen (z.B. Sensordaten) rechtzeitig zur Verfügung stehen. Die aktuelle Verfügbarkeit der für eine Komponente relevanten Informationen zur optimalen Entscheidungsfindung bezeichnen wir im weiteren als *Informationshorizont* dieser Komponente. In einem ungestörten System liegt der Informationshorizont einer jeden Komponente bei 100%. Im Falle von Fehlern, wie durchtrennten Kabeln, defekten Steckerverbindungen oder dem Teilausfall von Sensoren, liegt dieser Wert für die betroffenen Komponenten entsprechend niedriger. Da die Informationen für die Entscheidungsfindung in den verschiedenen Teilsystemen unterschiedlich starke Bedeutungen haben, sind die in die einzelnen Komponenten einfließenden Informationen mit einem sich aus der konkreten Anwendung ergebenden Gewicht zu versehen, das eine realistische Berechnung des Informationshorizontes erlaubt.

3.3 Das Simulationsmodell

Mit einem auf dem beschriebenen Szenario aufbauenden Simulationsmodell können für die verschiedenen Fehlerfallkombinationen die Auswirkungen auf den Informationshorizont der einzelnen Komponenten eines Architekturentwurfes überprüft werden. Wird im Rahmen der Fehlerfallsimulation beispielsweise virtuell ein Kabel durchtrennt, so lassen sich die Auswirkungen auf den Informationshorizont der verteilten Komponenten über das

System hinweg verfolgen. Im Ergebnis kann somit der Grad der Robustheit und der Fehlertoleranz eines ’x-by-wire’ Systems bereits im Vorfeld der Realisierung überprüft werden.

Als Basis für die Umsetzung der beschriebenen Ideen zur Erstellung von Architekturmodellen für verteilte Echtzeitsysteme im allgemeinen und ’x-by-wire’ Systeme im speziellen, werden Zeitstemplenetze (vgl. [Lau99]) verwendet.



In Abbildung 6 ist die oberste Hierarchieebene eines Teilmodells der EMB mit den aktiven Komponenten der Vorderachse, den Bremspedalsensoren (Pedalbox) sowie der zentralen Steuereinheit inkl. der Datenverbindungen zwischen diesen dargestellt.

Auf dieser Ebene werden die Spezifika der einzelnen Komponenten durch Supertransitionen verborgen und die verschiedenen Bussysteme durch Stellen repräsentiert. Die durch ein solches Modell fließenden Token sind hierbei mit Sender- und Empfängerinformationen versehen, so dass auf den Busstellen liegende Daten entsprechend an die adressierten Komponenten weitergeleitet werden.

Der hierarchische Abstieg in die Komponente zur Steuerung des Aktors an der rechten Seite der Vorderachse (VR) resultiert in dem in Abbildung 7 dargestellten Teilmodell. Während die termingerechte Versendung von Informationen in der Verantwortung der Sendeeinheit einer Komponente liegt, nimmt die Empfangseinheit einer solchen die an diese adressierten Nachrichten entgegen. Die Verbindung mit den anderen Modellkomponenten erfolgt hierbei über fusionierte Stellen [HJS90], die die verschiedenen Bussysteme nachbilden (hier beispielsweise über die Stellen “Hauptbus” und “Vorderachsenbus”).

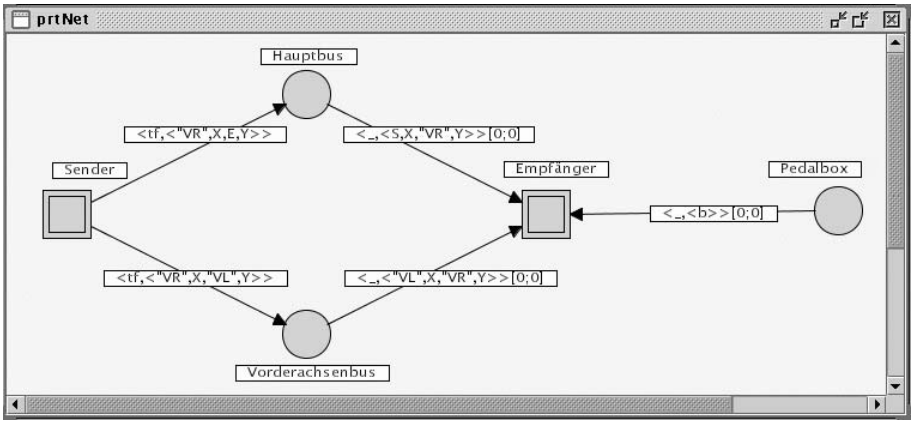


Abbildung 7 zeigt die abstrakte Sicht der Steuerungseinheit VR. Auf dieser Abstraktionsebene werden innerhalb einer verteilten Komponente Sende- und Empfangseinheiten unterschieden, die in der graphischen Darstellung wiederum durch Supertransitionen repräsentiert sind.

4 Design und Implementierung

In einem benutzerorientierten Entwurfsprozeß wird laufend versucht den ergonomischen Bedürfnissen der Benutzer von POSEIDON Rechnung zu tragen. Dabei setzt sich die Gruppe der Benutzer nicht ausschließlich aus Wissenschaftlern und Studenten aus höheren Semestern zusammen, sondern auch aus Anwendern aus wirtschaftlichen und ingenieurwissenschaftlichen Bereichen. Daher bestimmen sowohl Bedürfnisse aus dem akademischen als auch dem anwendungsorientierten Bereich maßgeblich die Entwicklung des Tools. In regelmäßig stattfindenden Arbeitsgruppentreffen wird erläutert, welches Systemverhalten die Beteiligten in bestimmten Situationen erwarten und welche Features das Arbeiten mit dem Tool angenehmer gestalten könnten. So wurde gerade im Hinblick auf den Bedienkomfort eine Reihe von Verbesserungen erzielt (konfigurierbare Toolbars, Verwaltung von Ergebnissen, komfortables Editieren, etc.).

Die Implementation des Systems basiert auf einer Softwarearchitektur mit zwei Hauptbestandteilen: einem in Java implementierten, plattformunabhängigen und zustandslosen *Front-End* und einem in C++ implementierten *Back-End*. Letzeres setzt sich aus einer Reihe von unabhängigen Komponenten (zur Zeit: Editor-, Simulator- und Analysekomponente) und einer zentralen Datenstruktur zusammen, in der mit Hilfe von Graphen alle in POSEIDON bearbeiteten Objekte (also Netze und UML-Diagramme) repräsentiert werden. Durch diese Architektur konnte sowohl die Plattformunabhängigkeit als auch die Offenheit des Werkzeugs gewährleistet werden. Durch die Möglichkeit von POSEIDON mit externen Mathematikprogrammen transparent zu kommunizieren, kann auf die gesamte Leistungsfähigkeit solcher Software zurückgegriffen werden. Zur Zeit wird POSEIDON (optional) mit einem MATHEMATICA-Kern (siehe [Wol96]) verlinkt.

5 Zusammenfassung und Ausblick

Im Rahmen dieses Artikels wurde gezeigt, daß POSEIDON durch die breite Auswahl zur Verfügung stehender Algorithmen und die Ausrichtung der Bedienkonzepte an die Bedürfnisse des wissenschaftlichen arbeitens für vielfältige Anwendungen geeignet ist. Dies wird unterstrichen durch die gegebene Plattformunabhängigkeit, die relativ leichte Erweiterbarkeit und die Möglichkeit zur Anbindung externer Programmpakete.

Dem in diesem Artikel als Schwerpunkt thematisierten Bereich der strukturellen Analyse wird bei zukünftigen Entwicklungen besondere Aufmerksamkeit zuteil werden. Insbesondere werden weitere Verfahren zur Behandlung von höheren Netzen und ggf. eingeschränkter Teilklassen derer, wie z.B. unäre Pr/T-Netze, entwickelt und in POSEIDON integriert.

Literatur

[Com72] F. Commoner. 'Deadlocks in Petri Nets.'. *Applied Data Research, Inc., Wakefield, Massachusetts, Report CA-7206-2311*, 1972.

[HJS90] Peter Huber, Kurt Jensen, and Robert M. Shapiro. 'Hierarchies in Coloured Petri Nets'. In G. Rozenberg, editor, *'Advances in Petri Nets 1990'*, LNCS 483. Springer-Verlag, 1990.

[Jax85] M. Jaxy. 'Analyse linearer diophantischer Ungleichungs- und Gleichungssysteme im Hinblick auf Anwendungen in der Theorie der Petri-Netze'. *Diplomarbeit, Universität in Koblenz*, 1985.

[JF01] J. Jung and U. Frank. 'Flottenmanagementsystems im Kundendienst'. In T. Grünert and H.-J. Sebastian, editor, *'Logistik Management - Supply Chain Management and e-Business'*. Teubner, 2001.

[Kru01] R. Kruse. 'Dualität bei Petri-Netzen – Anwendungen für Netze mit Stellen- und Transitionsmarken'. *Dissertation, Universität Koblenz*, 2001.

[Lau99] K. Lautenbach. 'Erweiterte Zeitstempelnetze zur Modellierung hybrider Systeme'. Fachbericht 3–99, Universität in Koblenz, 1999.

[LR96] Kurt Lautenbach and Hanno Ridder. 'Die Lineare Algebra der Verklemmungsvermeidung – Ein Petri-Netz-Ansatz.'. *Fachbericht 25–96, Universität in Koblenz*, 1996.

[MPS01] J. Müller, S. Philippi, and M. Seidel. 'Modellierung verteilter Echtzeitsysteme am Beispiel drive-by-wire'. In *'Proceedings des 8. Workshop Algorithmen und Werkzeuge für Petri-Netze, Universität Eichstätt'*, Eichstätt, 2001.

[MV85] G. Memmi and J. Vautherin. 'Computation of Flows for Unary-Predicates/Transition Nets'. In Rozenberg, G., editor, *'Advances in Petri Nets 1984'*. LNCS 188, Springer-Verlag, 1985.

[Rau02] H. Raue. 'Generierung von Steuerungen aus Petri-Netzen'. Diplomarbeit, Universität in Koblenz, erscheint 2002.

[Wol96] S. Wolfram. 'The Mathematica Book'. *Wolfram Research*, 1996.