Konsistenzprüfung von ProC/B-Modellen zur Vorbereitung einer simulativen Analyse

Jan Kriege*
Universität Dortmund
jan.kriege@udo.edu

Zusammenfassung

Dieser Beitrag stellt Verfahren zur Konsistenzprüfung von Prozesskettenmodellen, die im Rahmen des Sonderforschungsbereichs 559 "Große Netze in der Logistik" zur Modellierung von logistischen Netzwerken verwendet werden, vor. Die Verfahren dienen zur Vorbereitung einer simulativen Analyse des Modells und wurden in das im SFB 559 verwendete Toolset integriert. Unter anderem wird eine Methode zur Erkennung nicht-stationärer Modelle vorgestellt.

1 Einführung

Im Rahmen des SFB559 "Große Netze in der Logistik" ([Son]) dient das Prozesskette-Paradigma nach Kuhn ([Kuh95, Kuh99]) der Darstellung von logistischen Systemabläufen. Auf Basis dieses Paradigmas wurde der ProC/B-Modellformalismus ([BBT+99]) entwickelt. Der ProC/B-Modellformalismus verfügt über eine präzise definierte Semantik und schließt so die Lücke zwischen den häufig informalen Prozesskettenbeschreibungen und formalen Modellen, die die Nutzung von Analysetechniken ermöglichen. Zur Erzeugung der ProC/B-Modelle wurde ein Toolset ([BBF+02]) erschaffen, dass die grafische Spezifikation dieser Modelle erlaubt.
In diesem Beitrag soll eine Erweiterung des ProC/B-Toolsets vorgestellt werden, um eine Konsistenzprüfung von ProC/B-Modellen durchzuführen. Die Verfahren zur Konsistenzprüfung dienen dabei zur Vorbereitung einer simulativen Analyse und wurden im Rahmen von [Kri06] in das Toolset integriert. Hierbei werden zwei Ansätze verfolgt: Zum Einen sollen infolge einer syntaktischen Untersuchung formale Unzulänglichkeiten des Modells aufgezeigt werden, zum Anderen soll dem Modellierer mit Hilfe einer strukturbasierten Untersuchung des Modells die Möglichkeit gegeben werden, bereits vor der Simulation nicht-stationäre Modelle erkennen zu können. Die strukturbasierte Untersuchung wird auf Basis von Petri-Netzen durchgeführt und kombiniert so die Vorteile von Prozesskettenmodellen bei der Modellierung von logistischen Systemen mit Analyseverfahren von Petri-Netzen. Petri-Netze verfügen über eine formale Semantik, die die Überprüfung diverser

*Lehrstuhl Informatik 4, Universität Dortmund, D-44221 Dortmund, Germany. Diese Arbeit wurde durch die Deutsche Forschungsgemeinschaft im Rahmen des Sonderforschungsbereichs 559 "Modellierung großer Netze in der Logistik" unterstützt.

Eigenschaften eines Petri-Netz-Modells erlauben ([Mur89]) und werden in unterschiedlichen Anwendungsgebieten wie z.B. der Produktion ([MSP98]) eingesetzt. Der Nutzen einer Kombination von verschiedenen Formalismen zur Modellierung ist hinlänglich bekannt und wurde z.B. beim Entwurf des Möbius Frameworks ([CCD+01]) berücksichtigt. Das hier vorgestellte Verfahren weicht allerdings von dem Ansatz, der für Möbius verfolgt wurde ab: Während Möbius die Spezifikation von Modellen in verschiedenen Formalismen erlaubt, werden Petri-Netze im Folgenden nur für Untersuchungen der ProC/B-Modelle verwendet, da Prozesskettenmodelle in der Logistik verbreiteter sind und eine verständlichere Beschreibung von logistischen Netzwerken als Petri-Netze erlauben. Ein ähnlicher Ansatz wird in [vdA98] zur Formalisierung von Verifikation von Geschäftsprozessen vorgestellt. Dem Autor sind keine Simulationstools bekannt, die derartige Voruntersuchungen zur Erkennung unerwünschter Modelleigenschaften wie Nicht-Stationarität anbieten, insofern wird hier ein neuer Ansatz präsentiert. Verbreitete Simulationswerkzeuge wie Arena ([KSS04]), AutoMod ([Ban04]) oder Extend ([Kra02]) unterstützen den Modellierer lediglich während der Simulation, z.B. durch die Erstellung von Traces oder Debugging-Funktionen wie Breakpoints.

1.1 ProC/B-Modellformalismus

ProC/B-Modelle sind in der Regel hierarchisch aufgebaut. Sogenannte Funktionseinheiten (FEs) repräsentieren dabei Unternehmen oder Unternehmensteile. FEs bieten Dienste an, die von ihrer Umgebung genutzt werden können. Gleichzeitig können FEs weitere Funktionseinheiten enthalten und deren Dienste nutzen. Den Abschluss der hierarchischen Beschreibung bilden Standard-Funktionseinheiten wie Server und Counter. Server können für bestimmte Zeitspannen angefordert werden und eignen sich beispielweise für die Modellierung von Maschinen. Ein Counter kann zur Erfassung eines Lagers oder einer begrenzten Anzahl von Betriebsmitteln genutzt werden. Das Verhalten der FEs wird durch Prozessketten (PKs) dargestellt. Eine PK beschreibt dabei das Verhalten eines Typs von Prozessen. Die Erzeugung dieser Prozesse wird durch Quellen und Senken gesteuert. Die Aktivitäten der Prozesse (z.B. Dienstaufrufe oder Wartezeiten) werden durch Prozessketten-Elemente (PKEs) dargestellt. Die Elemente sind durch gerichtete Kanten verbunden, so dass ein Prozess die Prozesskette von der Quelle über einzelne PKEs bis zur Senke durchläuft. Zusätzlich existieren Kontrollstrukturen, um Prozesse in alternative oder parallele Zweige aufteilen zu können, Prozesse zu synchronisieren oder Schleifen zu realisieren, und Variablen. Bei den Variablen wird zwischen lokalen Variablen, die nur innerhalb eines Prozesses, und globalen Variablen, die innerhalb einer FE und den in ihr enthaltenen FEs sichtbar sind, unterschieden. Die Elemente eines ProC/B-Modells verfügen dabei über Attribute, mit denen das Verhalten des jeweiligen Elements genau festgelegt werden kann. Abbildung 1 zeigt einen Ausschnitt aus dem Modell eines Güterverkehrszentrums ([DV03]). Im oberen Bereich sind zwei Prozessketten dargestellt, die Dienste einer Funktionseinheit darstellen. Die beiden Prozessketten bestehen im Wesentlichen aus PKEs, die die Aktivitäten der Dienste beschreiben. Die zweite Prozessketten wird zusätzlich durch einen Konnektor in alternative Zweige aufgeteilt. Diese Zweige werden durch einen weiteren Konnektor später wieder zusammengeführt. An den Prozess-IDs (links in der Abbildung) werden lokale Variablen

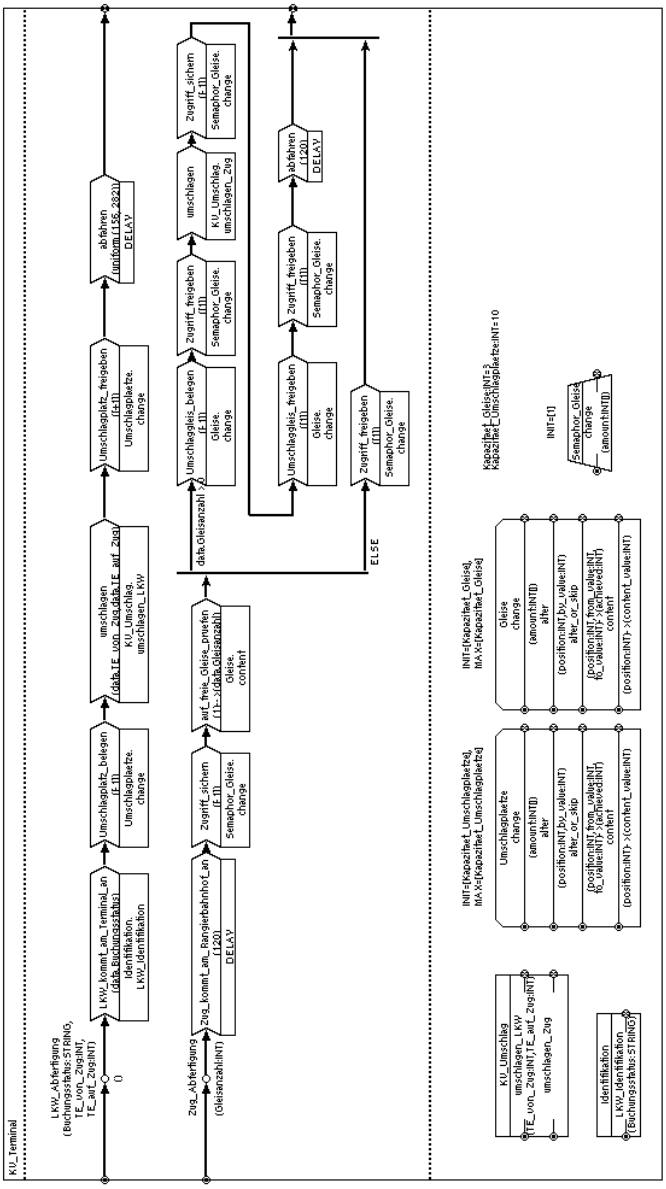

Abbildung 1: Ausschnitt aus einem ProC/B-Modell

deklariert. Weitere globale Variablen werden im unteren Bereich zusammen mit enthaltenen Funktionseinheiten und Standard-FEs abgebildet.

1.2 Motivation

ProC/B-Modelle werden zunächst im ProC/B-Editor grafisch spezifiziert und können danach in ein Austauschformat übersetzt werden. Dieses Austauschformat ermöglicht die Anbindung an verschiedene Analysetools. So sind Abbildungen auf Warteschlangenetze oder Petri-Netze möglich ([BBF+02]). Komplexe ProC/B-Modelle werden üblicherweise mit simulativen Methoden analysiert und hierzu in die Eingabesprache des Simulationstools HIT ([BMW94]) übersetzt. Hierbei können zwei Arten von Problemen auftreten: Zum einen sind dies syntaktische Fehler im Modell, die vor der Simulation beseitigt werden müssen. Hierbei kann es sich zum Beispiel um unvollständige Prozessketten mit fehlenden Verbindungen zwischen Elementen oder fehlerhaft geschachtelte Konnektoren handeln. Weitere potentielle Fehlerquellen sind die Attribute der Modellelemente, in denen der Nutzer teilweise Ausdrücke oder sogar vollständigen Code in der Syntax der Eingabesprache des Simulationstools eingeben kann. Erschwert wird die Beseitigung dieser Fehler durch die

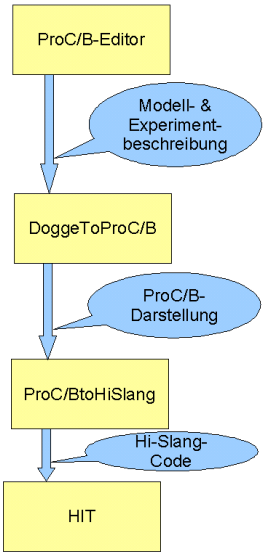

Abbildung 2: Aufbau des ProC/B-Toolsets

Struktur des ProC/B-Toolsets (s. Abbildung 2). Das im ProC/B-Editor erzeugte Modell wird durch zwei Konverter zunächst in ein Austauschformat und dann in die Eingabesprache von HIT umgewandelt. Dabei werden Bezeichner von Elementen unter Umständen umbenannt, um Namenskonventionen einzuhalten. Da Fehler in dem Modell oft erst durch HIT erkannt werden, ist es für den Modellierer schwierig, das betroffene Element in dem

Prozesskettenmodell zu identifizieren.
Ein weiteres Problem ergibt sich bei der Simulation, wenn das Modell über unerwünschte Eigenschaften verfügt, die die Simulationsergebnisse unbrauchbar machen können. Eine typische Situation in Modellen von logistischen Netzwerken sind Umladevorgänge, bei denen mehrere Prozessketten synchronisiert werden. In ProC/B-Modellen kann dies beispielsweise durch Prozessketten-Konnektoren oder durch einen Zugriff auf eine gemeinsam benutzte Ressource erfolgen. Derartige Modelle reagieren empfindlich auf Änderungen der Ankunftsraten der Prozesse und erreichen unter Umständen keine stationäre Phase. Die Analyse derartiger Modelle resultiert in unbrauchbaren Simulationsergebnissen und je nachdem welche Größen bei der Analyse untersucht werden, gelingt die Erkennung derartiger Situationen gar nicht oder erfordert lange Simulationsläufe.
Im Folgenden sollen nun einige Verfahren zur Konsistenzprüfung, die im Rahmen von [Kri06] in das ProC/B-Toolset integriert wurden, vorgestellt werden.

2 Überprüfung der Syntax

Ziel der syntaktischen Überprüfung des Modells ist es, Modellierungsfehler bereits im ProC/B-Editor vor der Umwandlung nach HIT zu erkennen und den Nutzer bei der Beseitigung dieser Fehler zu unterstützen. Da die Struktur von ProC/B-Modellen dem Aufbau von Programmen ähnelt, bietet es sich an, Techniken aus dem Übersetzerbau ([ASU88])zu verwenden. Funktionseinheiten entsprechen hierbei Klassen, die eine bestimmte Funktionalität kapseln. Prozessketten bzw. Dienste sind mit Methoden vergleichbar: Sie verfügen über Ein- und Ausgabeparameter und beschreiben bestimmte durchzuführende Aktionen. Zusätzlich existieren in ProC/B-Modellen auch Kontrollstrukturen sowie globale und lokale Variablen.
Prinzipiell sind bei der Syntaxüberprüfung zwei Teilaufgaben zu bewältigen: Bei der Untersuchung der Struktur des Modells sollten z.B. Fehler in der Schachtelung der Konnektoren oder unvollständige Prozessketten untersucht werden. Zusätzlich müssen die Attribute der Modellelemente überprüft werden, die in der Syntax der Simulationssprache vorliegen.
Die beiden Teilaspekte können jeweils von einem Scanner und einem Parser bewältigt werden. Zusätzlich sollten Symboltabellen für Variablen und Dienste gepflegt werden, in denen für Variablen Datentyp, Dimension, Initialwert, Sichtbarkeitsbereich und eine Zugriffsliste und für Dienste die Signatur (Anzahl, Typ und Dimension der Ein- und Ausgabeparameter) und eine Zugriffsliste abgelegt werden. Mit Hilfe dieser Symboltabellen lassen sich Zugriffe auf nicht deklarierte Variablen oder ungenutzte Dienste und Variablen erkennen. Zusätzlich erlauben sie eine semantische Analyse, bei der arithmetische Ausdrücke daraufhin überprüft werden können, ob die Operanden zu dem jeweiligen Operator passen.
Da ProC/B-Modelle modular aufgebaut sind, bietet es sich an, auch die Analyse von Modellteilen zu ermöglichen. Für die Untersuchung einer FE ist deren Umgebung nicht relevant. Funktionseinheiten bieten nach außen lediglich Dienste an, die in der umgebenden FE genutzt werden können. Ein Prozess durchläuft diesen Dienst von der Quelle bis zur Senke und verlässt die FE danach wieder. FEs können also unabhängig von ihrer Umgebung untersucht werden. Von in einer Funktionseinheit enthaltenen weiteren FEs müssen lediglich

die Signaturen der Dienste bekannt sein, um eventuelle Dienstanbindungen überprüfen zu können. Auch hier ergeben sich wieder Parallelen zu Programmen und dem Übersetzerbau: Einzelne Klassen können unabhängig vom Rest des Programms übersetzt werden, lediglich die Signaturen von Diensten müssen (z.B. durch Header-Dateien) bekannt gemacht werden. Eine Überprüfung der Syntax ist also auch für einzelne FEs oder Teilhierarchien des Modells problemlos möglich.

3 Strukturbasierte Überprüfung auf Basis von Petri-Netzen

In [Bau03] wurde ein Verfahren für Petri-Netze vorgestellt, mit dem GSPNs auf E-Sensitivität überprüft werden können. Bei einem GSPN handelt es sich um ein 4-tupel $GSPN = (PN, T_1, T_2, W)$, wobei $PN = (P, T, I^-, I^+, M_0)$ ein Stellen-Transitionen-Netz mit einer Menge von Stellen P, einer Menge von Transitionen T, den Inzidenzfunktionen I^- und I^+ und der Anfangsmarkierung M_0 ist. $T_1 \subset T$ und $T_2 \subset T$, $T_1 \cap T_2 = \emptyset, T_1 \cup T_2 = T$, teilen die Transitionen in zeitbehaftete und zeitlose Transitionen auf. Der Vektor $W = (w_1, ..., w_{|T|})$ gibt für zeitbehaftete Transitionen eine Feuerungsverzögerung und für zeitlose Transitionen eine Gewichtung an.
E-Sensitivität impliziert in der Regel Nicht-Stationarität. Der Test auf E-Sensitivität basiert auf einer Untersuchung der Struktur des Modells. Da das Verfahren für Petri-Netze formuliert wurde, bietet es sich an, das ProC/B-Modell zunächst in ein Petri-Netz umzuwandeln. Dabei wird Hierarchie des ProC/B-Modells durch ein hierarchisches Petri-Netz abgebildet. Die einzelnen Prozesse werden durch Marken repräsentiert. Die Elemente des ProC/B-Modells können einzeln in Petri-Netz-Konstrukte übersetzt werden. Diese Umwandlung unterliegt allerdings einigen Einschränkungen: Variablen und somit auch arithmetische Ausdrücke oder Vergleiche lassen sich nur schlecht durch ein Petri-Netz abbilden. Deshalb werden sie bei der Umwandlung ignoriert, so dass das erzeugte Petri-Netz im Wesentlichen nur die Struktur des ProC/B-Modells wiedergibt, während das Verhalten nur annähernd nachempfunden wird.
Das erzeugte Petri-Netz kann außerdem im APNN-Format ([BKK94]), einer textuellen Notation für Petri-Netze, ausgegeben werden, um so die Nutzung weiterer Analyseverfahren zu ermöglichen, die die APNN-Toolbox ([BFKT01]) zur Verfügung stellt.
Falls das Petri-Netz e-sensitiv ist, werden durch den Test aus [Bau03] Transitionen identifiziert, die das e-sensitive Verhalten des Netzes verursachen. Um diese Ergebnisse auf das ProC/B-Modell übertragen zu können, ist es notwendig, bereits während der Konvertierung zu jedem Element des Prozesskettenmodells die Stellen und Transitionen zu speichern, die bei der Umwandlung erzeugt wurden. Auf diese Weise ist es möglich, auch in dem ProC/B-Modell kritische Modellteile zu erkennen, die nicht-stationäres Verhalten verursachen können.
In ProC/B-Modellen entsteht nicht-stationäres Verhalten häufig durch Synchronisationseffekte an PK-Konnektoren oder durch gemeinsam genutzte Ressourcen. Ein stark vereinfachtes Beispiel zeigt Abbildung 3. Das Modell besteht lediglich aus zwei Prozessketten, die auf ein gemeinsames Lager zugreifen, wobei die obere Prozesskette Ein- und die untere

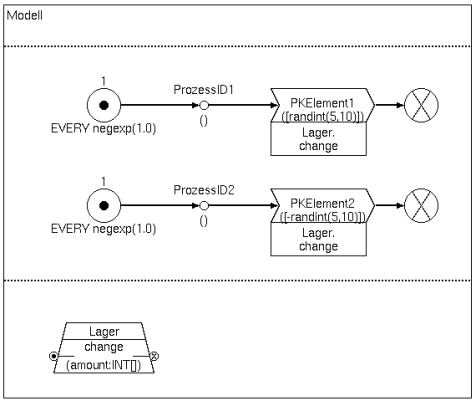

Abbildung 3: Ein einfaches ProC/B-Modell mit zwei Prozessketten

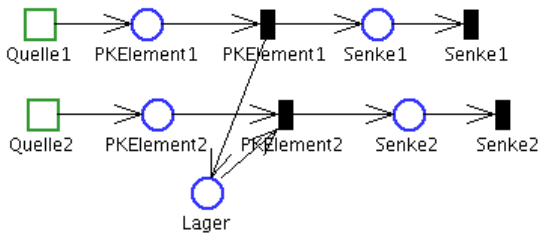

Abbildung 4: Petri-Netz zu dem Modell aus Abbildung 3

PK Auslagerungen vornimmt. Abbildung 4 zeigt das Petri-Netz, das von dem Verfahren erzeugt wird. Offensichtlich ist die untere Prozesskette von der oberen abhängig, da Auslagerungen nur möglich sind, wenn vorher eine entsprechende Menge eingelagert wurde. Bei kleinen Abweichungen oder Schwankungen der Ankunftsraten werden die Prozesse der einen Prozesskette blockiert und stauen sich bei dem Zugriff auf das Lager. Entsprechend wird das Modell von dem Verfahren als e-sensitiv (und damit nicht-stationär) erkannt.
Das hier beschriebene Verfahren ist als Untersuchung vor der eigentlichen simulativen Analyse vorgesehen, um bereits im Vorfeld mögliche Probleme identifizieren zu können. Ebenso wie bei der Syntaxüberprüfung kann sowohl die Konvertierung in ein Petri-Netz als auch der Test auf Nicht-Stationarität für das gesamte Modell oder Modellteile durchgeführt werden.

4 Integration der Verfahren in das ProC/B-Toolset

Die zuvor beschriebenen Verfahren wurden vollständig in den ProC/B-Editor integriert, um dem Modellierer so eine einfache Möglichkeit zur Erkennung und Beseitigung der Fehler zur Verfügung zu stellen. Die Auswahl der durchzuführenden Tests erfolgt über einen Auswahldialog im ProC/B-Editor (s. Abbildung 5). Anschließend werden die gewählten Konsistenzprüfungen durchgeführt und die Ergebnisse im Editor dargestellt (s. Abbildung 7). Zusätzlich nutzen die Verfahren auch Ergebnisse der anderen Verfahren. So wird vor der Umwandlung in ein Petri-Netz geprüft, ob die Struktur des ProC/B-Modells fehlerfrei ist. Das erzeugte Petri-Netz wird wiederum von dem Test auf Nicht-Stationarität verwendet. Über den Auswahldialog (s. Abbildung 5) lässt sich das Verhalten der durchzuführenden

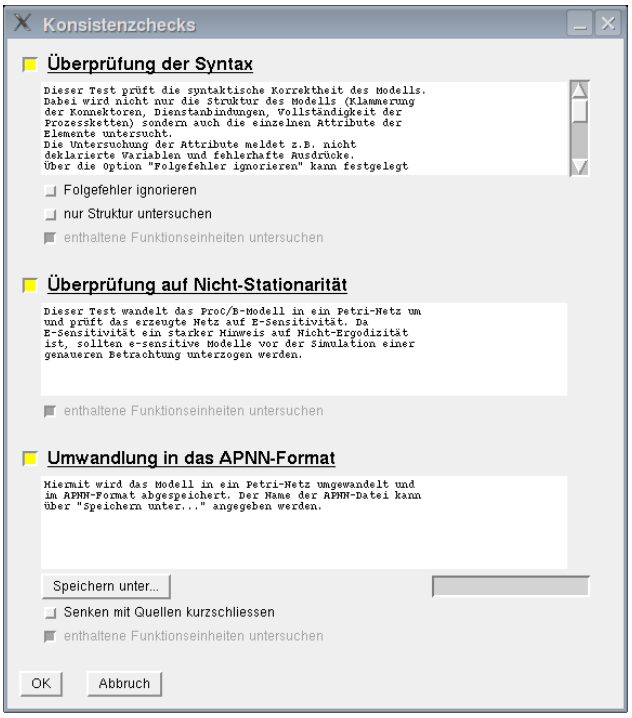

Abbildung 5: Auswahldialog für die Konsistenzprüfungen ([Kri06])

Verfahren zur Konsistenzprüfung durch einige Optionen genauer festlegen. So kann für alle Verfahren festgelegt werden, ob das gesamte Modell oder nur eine Teilhierarchie untersucht werden soll. Für die Überprüfung der Syntax kann außerdem z.B. angegeben werden, dass pro Prozesskette nur der erste Fehler in der Struktur ausgegeben wird und eventuelle Folgefehler unterdrückt werden. Für die erzeugten Petri-Netze im APNN-Format kann angegeben werden, dass Senken und Quellen miteinander verbunden werden sollen, um so

den Zustandsraum des Netzes zu beschränken.
Das Ergebnisfenster (s. Abbildung 7) besteht aus drei Teilen. Der linke untere Bereich dient zur Sortierung und Filterung der Ergebnisse. Eine Sortierung ist nach Funktionseinheiten oder Konsistenzprüfungen möglich. Entsprechend enthält die Auswahlliste (oben links) alle Funktionseinheiten bzw. Konsistenzprüfungen für die Ergebnisse vorliegen. Zusätzlich lassen sich bestimmte Meldungen (z.B. Warnungen) ausblenden. Im rechten Bereich des Fensters werden die Ergebnisse der durchgeführten Verfahren angezeigt. Die einzelnen Meldungen bestehen dabei aus einem Typ (z.B. Fehler oder Warnung), der farbig hervorgehoben wird, einem Text mit einer genauen Beschreibung und den betroffenen Elementen des Modells. Die Elementbezeichner können dabei ähnlich wie Links in HTML-Dokumenten angeklickt werden, um das Element in einem Arbeitsfenster zu öffnen und zu markieren. Auf diese Art und Weise können die fehlerhaften Elemente schnell gefunden werden, um den Fehler zu beseitigen.

5 Anwendungsbeispiele

Im Folgenden soll der Test auf Nicht-Stationarität an einem Beispielmodell aus dem SFB 559 vorgeführt werden. Bei dem Modell handelt es sich um ein Güterverkehrszentrum, das in [DV03] beschrieben wird. Abbildung 6 zeigt die Hierarchie des Modells. Das Modell besteht im Prinzip aus drei Teilen: In der FE *GVZ_Generator* wird die Erzeugung von LKWs und Zügen beschrieben. In der Stückgutumschlaghalle (FE *SUH* und enthaltene FEs) werden palettierte Güter und Kisten von einem LKW auf einen anderen umgeschlagen. In der FE *KV_Terminal* und den in ihr enthaltenen FEs findet ein Umschlag von Gütern von LKWs auf Züge und umgekehrt statt. Wie aus der Darstellung des Hierarchiebaums bereits ersichtlich ist, handelt es sich hierbei um ein sehr umfangreiches und komplexes Modell, was die Untersuchung einzelner Modellteile nahelegt. Von besonderem Interesse ist hier das KV-Terminal mit enthaltenen Funktionseinheiten, da hier ein Umschlag von Gütern zwischen LKWs und Zügen stattfindet und dafür verschiedene Prozessketten synchronisiert werden. Abbildung 7 zeigt die Ergebnisse des Tests auf Nicht-Stationarität, der mehrere einzelne Elemente identifiziert hat. Diese Elemente sind in Abbildung 8 grau markiert. Durch die Synchronisation handelt es sich hier um einen Modellteil, der empfindlich auf Veränderungen der Ankunftsraten der Prozesse reagiert und bei dessen Modellierung und Analyse besonders sorgfältig vorgegangen werden sollte.
Wie bereits erwähnt, gehen bei der Umwandlung eines ProC/B-Modells in ein Petri-Netz Informationen verloren, da z.B. Variablen, arithmetische Ausdrücke oder Vergleiche nicht in das Petri-Netz übertragen werden. Dies kann dazu führen, dass Probleme, die bei der Analyse des Petri-Netzes erkannt werden, in dem ProC/B-Modell nicht existieren. In dem Beispiel des GVZ kann beispielsweise durch Bedingungen an Konnektoren dafür gesorgt werden, dass Prozesse den als kritisch identifizierten Bereich des Modells nicht erreichen, falls keine Synchronisation möglich ist. Aber auch in diesen Fällen liefert der Test auf Nicht-Stationarität hilfreiche Hinweise, da bei der Modellierung derartiger Bedingungen zur Verhinderung von nicht-stationärem Verhalten natürlich besonders genau vorgegangen werden sollte.

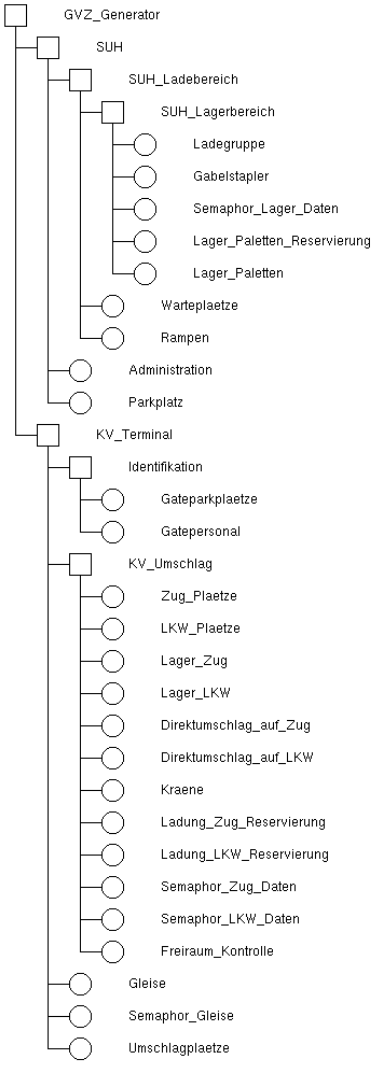

Abbildung 6: Hierarchiedarstellung des Modells eines GVZ ([Kri06])

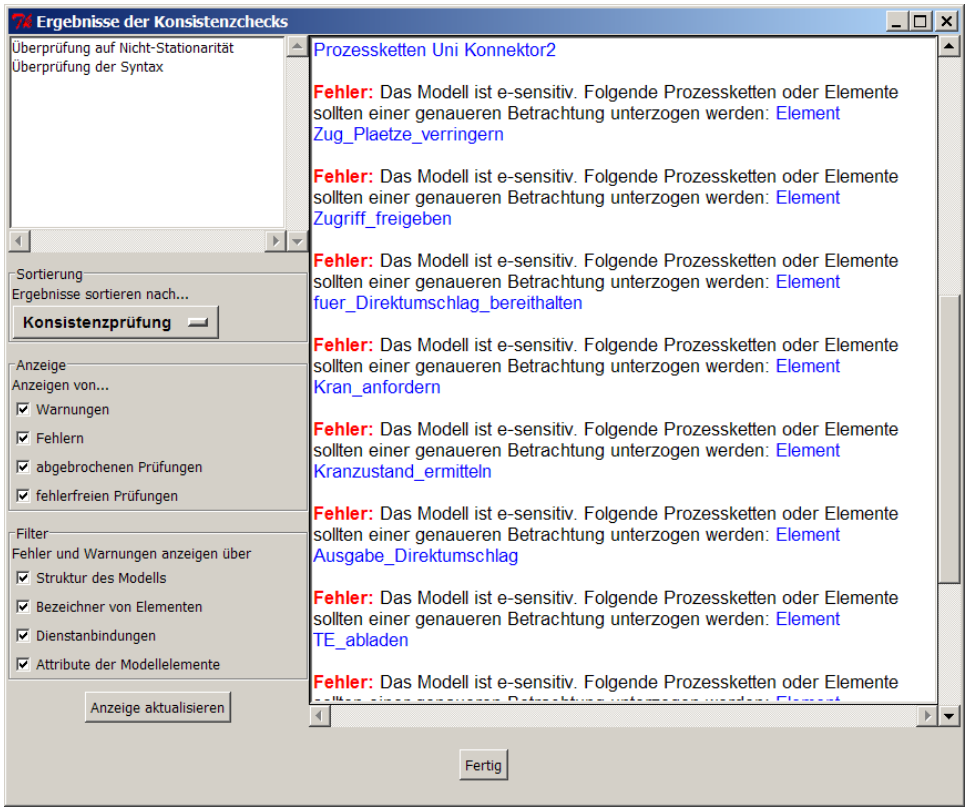

Abbildung 7: Ergebnisse der Konsistenzprüfung ([Kri06])

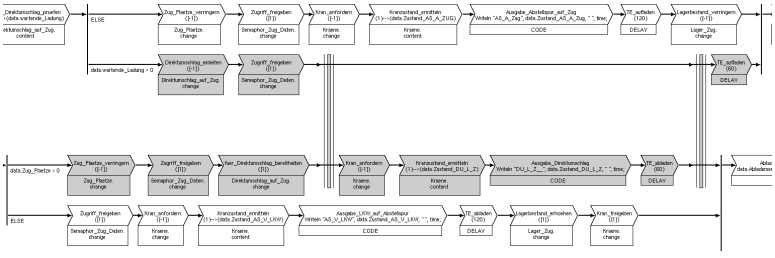

Abbildung 8: Kritischer Bereich des Modells eines GVZ ([Kri06])

6 Fazit

In diesem Beitrag wurde eine Erweiterung des ProC/B-Toolsets vorgestellt, mit der sich durch nicht-simulative Verfahren Modellierungsfehler beseitigen und kritische Modellteile identifizieren lassen. Diese Verfahren können als Voruntersuchungen zur Simulation eingesetzt werden. Die Funktionalität dieser Verfahren wurde anhand eines konkreten Modells aus dem SFB 559 gezeigt.

Literatur

[ASU88] AHO, A. V., R. SETHI und J. D. ULLMAN: *Compilerbau, Teil 1*. Addison-Wesley, 1988. ISBN 3-89319-150-X.

[Ban04] BANKS, J.: *Getting Started With AutoMod*. Brooks Automation, Inc., 2. Auflage, 2004. ISBN 0-9729100-3-4.

[Bau03] BAUSE, F.: *On Non-Ergodic Infinite-State Stochastic Petri Nets*. In: *Proceedings of the 10th International Workshop on Petri Nets and Performance Models (PNPM 2003)*, Seiten 84–92. IEEE Society Press, 2003.

[BBF+02] BAUSE, F., H. BEILNER, M. FISCHER, P. KEMPER und M. VÖLKER: *The ProC/B-Toolset for the Modelling and Analysis of Process Chains*. In: FIELD, T., P.G. HARRISON, J. BRADLEY und U. HARDER (Herausgeber): *Computer Performance Evaluation, Modelling Techniques and Tools, 12th International Conference, TOOLS 2002, London (UK)*, Nummer 2324 in *Lecture Notes in Computer Science*, Seiten 51–70. Springer, 2002.

[BBT+99] BEILNER, H., F. BAUSE, H. TATLITÜRK, A. VAN ALMSICK und M. VÖLKER: *Zum B-Modellformalismus - Version B1 - zur Vorbereitung automatisierter Analysen von Modellen logistischer Systeme hinsichtlich technischer, ökonomischer und ökologischer Ziele*. Technischer Bericht 99002, Sonderforschungsbereich 559 Modellierung großer Netze in der Logistik, 1999.

[BFKT01] BUCHHOLZ, P., M. FISCHER, P. KEMPER und C. TEPPER: *New features in the APNN toolbox*. In: KEMPER, P. (Herausgeber): *Tools of Aachen 2001 Int. Multiconference on Measurement, Modeling and Evaluation of Computer-Communication Systems*, Seiten 62–68, 2001. Universität Dortmund, Fachbereich Informatik, Forschungsbericht Nr. 760.

[BKK94] BAUSE, F., P. KEMPER und P.S. KRITZINGER: *Abstract Petri Net Notation*. In: *Forschungsbericht Nr. 563 des Fachbereichs Informatik der Universität Dortmund (Germany)*. 1994.

[BMW94] BEILNER, H., J. MÄTER und C. WYSOCKI: *The Hierarchical Evaluation Tool HIT*. In: *Short Papers and Tool Descriptions of the 7th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, 1994.

[CCD+01] CLARK, G., T. COURTNEY, D. DALY, D. DEAVOURS, S. DERISAVI, J. DOYLE, W. SANDERS und P. WEBSTER: *The Moebius Modeling Tool*. In: *Proceedings of 9th International Workshop on Petri Nets and Performance Models, PNPM'01 Aachen, Sept. 11-14, 2001, Reinhard German and Boudewijn Haverkort (eds.), IEEE*, Seiten 241–250, 2001.

[DV03] DILLING, C. und M. VÖLKER: *Beispielmodellierung eines Güterverkehrszentrums im ProC/B-Paradigma*. Technischer Bericht 03016, Sonderforschungsbereich 559 Modellierung großer Netze in der Logistik, 2003. ISSN 1612-1376.

[Kra02] KRAHL, D.: *The Extend Simulation Environment*. In: *Proceedings of the 2002 Winter Simulation Conference*, 2002.

[Kri06] KRIEGE, J.: *Konsistenzprüfung von ProC/B-Modellen zur Vorbereitung einer simulativen Analyse*. Diplomarbeit, Universität Dortmund, Lehrstuhl Informatik IV, 2006.

[KSS04] KELTON, W.D., R.P. SADOWSKI und D.T. STURROCK: *Simulation with Arena*. McGraw-Hill, 3. Auflage, 2004. ISBN 0-07-291981-7.

[Kuh95] KUHN, A.: *Prozessketten in der Logistik - Entwicklungstrends und Umsetzungsstrategien*. Verlag Praxiswissen, Dortmund, 1995.

[Kuh99] KUHN, A.: *Prozesskettenmanagement - Erfolgsbeispiele aus der Praxis*. Verlag Praxiswissen, Dortmund, 1999.

[MSP98] M. SILVA, E. TERUEL, R. VALETTE und H. PINGAUD: *Petri nets and production systems*. In: REISIG, W. und G. ROZENBERG (Herausgeber): *Lectures on Petri Nets II: Applications*, Band 1492/1998 der Reihe *Lecture Notes in Computer Science*, Seiten 85–124. Springer-Verlag, 1998.

[Mur89] MURATA, T.: *Petri Nets: Properties, Analysis and Applications*. Proceedings of the IEEE, 77(4):541–580, April 1989.

[Son] SONDERFORSCHUNGSBEREICH 559: *Modellierung großer Netze in der Logistik*. http://www.sfb559.uni-dortmund.de.

[vdA98] AALST, W.M.P. VAN DER: *Formalization and Verification of Event-driven Process Chains*. Computing Science Reports 98/01, Eindhoven University of Technology, Eindhoven, 1998.