OOPr/T-Modelle - ein Pr/T-Netz basierter Ansatz zur objektorientierten Modellierung

Stephan Philippi

Universität Koblenz-Landau
Institut für Softwaretechnik
Rheinau 1, 56075 Koblenz

eMail: philippi@uni-koblenz.de

Zusammenfassung

Aufgrund der potentiellen Vorzüge einer Synthese von Petri-Netzen und objektorientierten Konzepten in den Bereichen der Modularisierung von Petri-Netzen und der formal basierten objektorientierten Modellierung, sind seit Mitte der achtziger Jahre eine Vielzahl von Ansätzen entstanden, von denen sich jedoch auf breiter Front keiner hat durchsetzen können. Dieser Artikel beleuchtet kurz die Probleme existierender Arbeiten und gibt einen Überblick über Kriterien, denen ein Ansatz in diesem Bereich idealerweise genügt. Mit OOPr/T-Modellen wird im weiteren ein neuartiger Ansatz zur Synthese von Petri-Netzen und objektorientierten Konzepten vorgestellt, der im Hinblick auf diesen Kriterienkatalog entwickelt wurde und der darüber hinaus die Generierung nebenläufiger (Java-) Programme erlaubt.

1 Einleitung

Die Motivation zur Entwicklung eines Ansatzes zur Synthese von Petri-Netzen und objektorientierten Konzepten ergibt sich aus der Beobachtung, daß einerseits existierende objektorientierte Modellierungssprachen geeignete Modularisierungskonzepte zur Handhabung komplexer Systeme bieten, diese Sprachen jedoch entweder nicht formal basiert sind, wie z.B. UML [Rati97], oder aber im entgegengesetzten Fall zumeist keine graphische Repräsentation besitzen und es an Möglichkeiten zur Simulation erstellter Modelle sowie zur Gestaltung nebenläufiger Systeme mangelt[1]. Mit Petri-Netzen steht andererseits ein Formalismus zur Verfügung, der eine graphische Repräsentation besitzt sowie Möglichkeiten zur Simulation erstellter Modelle und zum Entwurf nebenläufiger Systeme bietet. Als problematisch im Zusammenhang mit dem Einsatz von (high-level) Petri-Netzen zur Modellierung komplexer Systeme erweist sich, daß diese auch bei Verwendung hierarchischer Erweiterungen [HuJeSh90] keine adäquaten Modularisierungskonzepte zur Verfügung stellen.

Ein Vergleich der beschriebenen Vor- und Nachteile von bisher praktizierter objektorientierter Modellierung und der Verwendung von Petri-Netzen ergibt, daß die Vorteile der einen Technik gerade den Nachteilen der jeweils anderen entsprechen. Aus dieser Sicht wird deutlich, daß die Synthese von Petri-Netzen und objektorientierten Konzepten ein vielversprechendes Potential zur Lösung der beschriebenen Probleme in beiden Gebieten aufweist. Ein Ansatz in diesem Gebiet, der sowohl die objektorientierte Strukturierung

[1] Eine vergleichende Untersuchung derartiger Ansätze ist zu finden in [LanHau94].

von Petri-Netzen als auch die formal basierte objektorientierte Modellierung ermöglicht, vereint hierbei idealerweise ausschließlich die Vorzüge der Ausgangskomponenten, nicht jedoch deren individuelle Nachteile.

Vor diesem Hintergrund wird im zweiten Abschnitt des Artikels beschrieben, welchen Kriterien ein Ansatz zur Synthese von Petri-Netzen und objektorientierten Konzepten genügen sollte. Im dritten Abschnitt wird mit OOPr/T-Modellen ein neuartiger Ansatz in diesem Gebiet vorgestellt, bevor der vierte Abschnitt diese kurz evaluiert und einen Ausblick auf zukünftige Entwicklungen gibt.

2 Anforderungen an objektorientierte Petri-Netze

Die aus der bisherigen Beschreibung hervorgehende Attraktivität einer Synthese von Petri-Netzen und objektorientierten Konzepten hat Mitte der achtziger Jahre erste Arbeiten in diesem Gebiet entstehen lassen. Obgleich der Vielzahl der bis heute entwickelten Ansätze, herrscht über die Kriterien, die ein solcher Ansatz idealerweise erfüllen sollte, nach wie vor Unklarheit. Grund hierfür sind die verschiedensten Zielrichtungen sowie die Vielzahl möglicher Ansatzpunkte einer Synthese und das sich hieraus ergebende uneinheitliche, stark inhomogene Feld nicht erkennbar aufeinander aufbauender Arbeiten.

Die Hauptproblemfelder der existierenden Arbeiten zur Synthese von Petri-Netzen und objektorientierten Konzepten liegen in der 'Vollständigkeit aus Sicht der Objektorientierung' und der 'Ergonomie der Notation'. Diese Punkte werden im weiteren überblicksartig als Teil eines Katalogs von Kriterien beschrieben, denen ein Ansatz zur Synthese von Petri-Netzen und objektorientierten Konzepten genügen sollte, um dem zuvor beschriebenen Ideal möglichst nahe zu kommen[2].

- Vollständigkeit aus Sicht der Petri-Netze: Ein Ansatz zur Synthese von Petri-Netzen und objektorientierten Konzepten erhält idealerweise die positiven Eigenschaften der Petri-Netze, d.h. sowohl die formale Basis als auch die Möglichkeit zur Simulation gegebener Modelle und deren nebenläufige Gestaltung sollten sich bei objektorientierten Erweiterungen von Petri-Netzen wiederfinden.

- Vollständigkeit aus Sicht der Objektorientierung: Neben dem Erhalt der positiven Eigenschaften der Petri-Netze, sollte eine Synthese derer mit objektorientierten Konzepten auch dahingehend vollständig sein, daß die relevanten objektorientierten Konzepte bei der Integration berücksichtigt wurden. Im einzelnen werden hierbei komplexe Objekte, Klassen, Kapselung, Vererbung, Überschreiben und Polymorphismus/dynamisches Binden als relevant erachtet.

- Ergonomie der Notation/Verwendbarkeit: Der Hautverwendungszweck einer Modellierungssprache ist die Vermittlung kognitiver Modelle zwischen verschiedenen Personen. Neben den bisher aufgeführten, aus theoretischer Sicht wünschenswerten Eigenschaften, sollte die aus einer Synthese von Petri-Netzen und objektorientierten Konzepten entstehende Sprache somit auch ergonomischen Aspekte genügen, d.h. eine Synthese ist dahingehend zu gestalten, daß die entstehende Modellierungssprache selbst möglichst wenig Aufmerksamkeit auf sich zieht, um ihre Benutzer bei der inhärent anspruchsvollen Tätigkeit des Modellierens bestmöglich zu unterstützen.

[2]Eine ausführliche Untersuchung der existierenden Ansätze sowie eine detailliertere Beschreibung des Kriterienkatalogs findet sich in [Phil99].

Auf Grundlage dieser Kriterien wird im weiteren mit OOPr/T-Modellen ein neuartiger Ansatz zur Synthese von Petri-Netzen und objektorientierten Konzepten vorgestellt, in den die Erfahrungen der existierenden Arbeiten in diesem Gebiet miteinfließen.

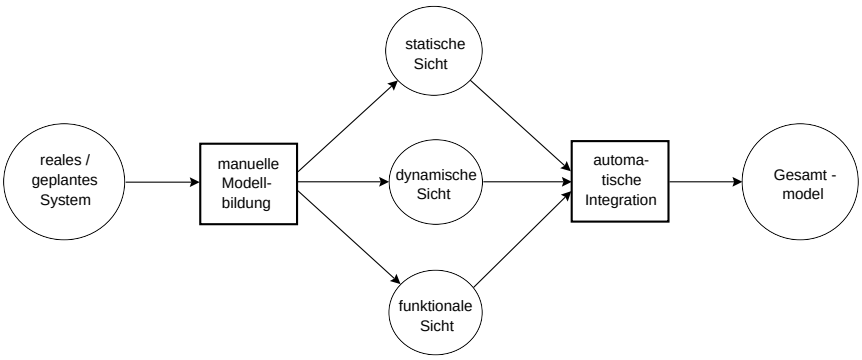

Abb. 1: Szenario der OOPr/T-Modellierung

3 OOPr/T-Modelle

Das prinzipielle Vorgehen bei der OOPr/T-Modellierung gestaltet sich wie in Abb. 1 skizziert. Ausgehend von dem zu modellierenden realen/geplanten System ist es die Aufgabe des Modellierers, unter Berücksichtigung des gegebenen Anwendungskontextes und der gewünschten Abstraktionsebene, die Teilmodelle für die relevanten Sichten zu gestalten. Zum Einsatz kommen hierbei (eingeschränkte) UML-Klassendiagramme für die statische Sicht, während dynamische und funktionale Aspekte durch Petri-Netze beschrieben werden. Die formale Semantik der verschiedenen Sichten sowie der formale Zusammenhang zwischen diesen wird durch Transformationsregeln definiert [Phil99], die eine automatische Integration der verschiedenen Sichten zu einem Gesamtmodell in Form eines Prädikat/Transitions-Netzes (Pr/T-Netz) [GenLau81] ermöglichen. Dieses Gesamtmodell stellt die formale Grundlage eines OOPr/T-Modells dar und sollte aus Sicht des Modellierers transparent sein, d.h. dieser sollte nur mit den von ihm gestalteten Sichten und nicht mit dem diese zusammenfassenden, komplexen Gesamtmodell konfrontiert werden.

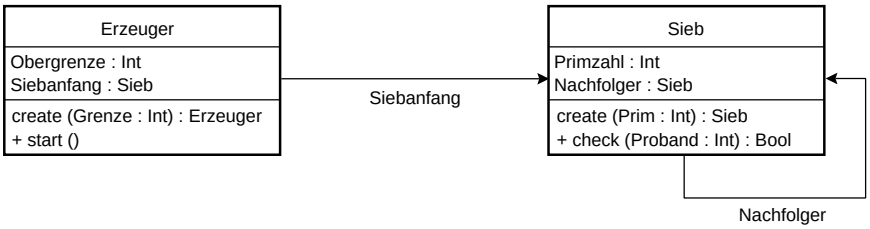

Abb. 2: Klassendiagramm für das 'Sieb des Eratosthenes'

Zur Veranschaulichung dieser Vorgehensweise wird im weiteren ein objektorientiertes Modell eines Systems zur Berechnung von Primzahlen nach der Methode 'Sieb des Eratosthenes' auf einer niedrigen Abstraktionsebene beschrieben. Die Berechnung der Primzahlen nach diesem Verfahren erfolgt (in einer für die Implementierung optimierten Variante) durch eine sequentielle Suche über $\mathbb{N}^+$ bis hin zu einer gegebenen Obergrenze. Jede Zahl wird dahingehend überprüft, ob sie durch ein Element der Liste der bereits ermit-

telten Primzahlen ohne Rest geteilt werden kann. Ist dies nicht möglich, so ist eine neue Primzahl gefunden, die in die Liste der bereits gefundenen Primzahlen aufgenommen wird.

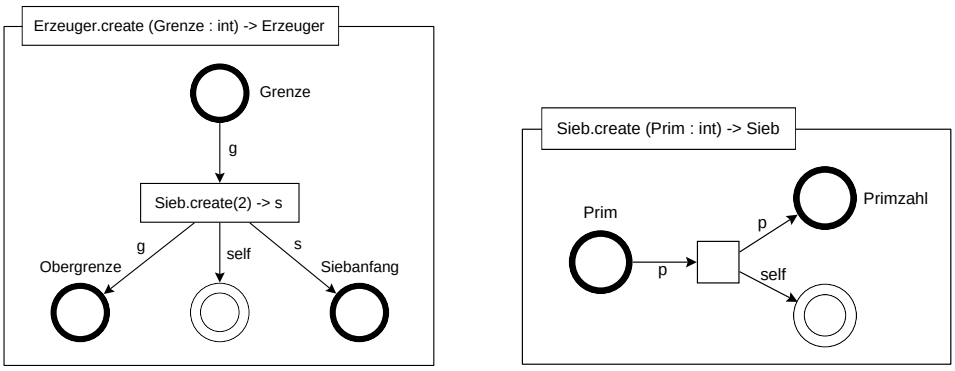

Abb. 3: funktionale Modelle der Konstruktoren

Der erste Schritt der Gestaltung von OOPr/T-Modellen ist die Erstellung eines Klassendiagramms unter Verwendung der eingeschränkten UML-Notation. Das Klassendiagramm für das Beispiel (Abb. 2) besteht einzig aus den Klassen 'Erzeuger' und 'Sieb', die der Erzeugung von Primzahlkandidaten bzw. der Überprüfung dieser Kandidaten auf die Primzahleigenschaft dienen. Die Klasse 'Erzeuger' enthält die Attribute 'Obergrenze' und 'Siebanfang', in denen die Obergrenze des Berechnungsverfahrens gespeichert bzw. die Assoziation zum initialen Siebobjekt realisiert wird. Darüber hinaus enthält die Klasse 'Erzeuger' einen Konstruktor sowie eine 'start'-Methode, mit der das Berechnungsverfahren gestartet wird. Die Klasse 'Sieb' enthält die Attribute 'Primzahl' und 'Nachfolger', in denen sowohl die durch ein Siebobjekt repräsentierte Primzahl als auch die ggf. vorhandene Referenz auf ein Nachfolgeobjekt hinterlegt wird. Neben einem Konstruktor besitzt die Klasse 'Sieb' eine 'check'-Methode, in der die Überprüfung eines Kandidaten auf die Primzahleigenschaft erfolgt.

Der zweite Schritt der OOPr/T-Modellierung besteht in der Zuordnung von dynamischen Modellen zu den einzelnen Klassen des UML-Diagramms. Mit Hilfe von Petri-Netzen werden hierbei Bedingungen für die (nebenläufige) Aktivierung von Methoden spezifiziert. Aufgrund des rein sequentiellen Charakters des Beispiels, in dem keine zusätzlichen Aktivierungsbedingunen für Methoden enthalten sind, wird auf die Darstellung der in diesem Fall trivialen dynamischen Modelle für die Klassen 'Erzeuger' und 'Sieb' verzichtet.

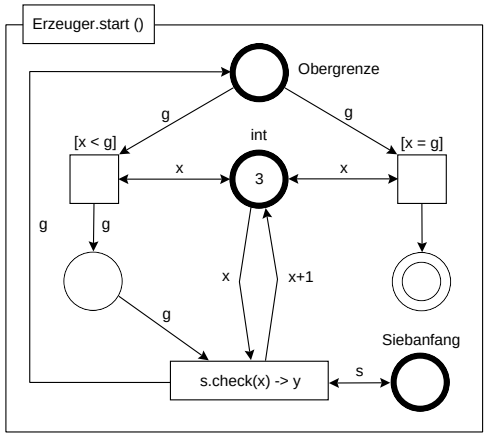

Abb. 4: Modell der 'start'-Methode

Der dritte Schritt der OOPr/T-Modellierung umfaßt die funktionale Spezifikation der Methoden durch erweiterte Pr/T-Netze. Abbildung 3 zeigt dies für die Konstruktoren der Klassen 'Erzeuger' und 'Sieb'. Da eine Methode einmalig für alle Instanzen einer Klasse spezifiziert wird, kann das die Funktionalität einer Methode beschreibende Netz keine Anfangsmarkierung im herkömmlichen Sinne beinhalten. Vielmehr wird eine objektabhängige Anfangsmarkierung benötigt, die es erlaubt, Methodenparameter und aktuelle Attributwerte in eine Methodenspezifikati-

on zu importieren. Für diese Zwecke wurden in funktionalen OOPr/T-Netzen sogenannte 'preload places' eingeführt, die sich rein äußerlich durch einen dickeren Rand von herkömmlichen Stellen unterscheiden und die mit dem Bezeichner eines Methodenparameters bzw. eines Attributes zu beschriften sind. Wird eine Methode aktiviert, so stehen auf den 'preload places' automatisch die aktuellen Werte gemäß ihrer Beschriftung bereit. Nachrichten zur Aktivierung anderer Methoden werden mit Hilfe sog. 'message transitions' versendet, die im Vergleich zu herkömmlichen Transitionen zusätzlich die Spezifikation der zu sendenden Nachricht beinhalten. In Abb. 3 wird somit der Konstruktor der Klasse 'Sieb' mit dem Argument '2' als initialer Primzahl aktiviert. Die als Ergebnis dieser Aktivierung an den Konstruktor der Klasse 'Erzeuger' zurückgegebene OID des instanziierten Siebobjektes wird nachfolgend durch die Verwendung eines ebenfalls mit einem dickeren Rand gezeichneten 'postsave place' im Attribut 'Siebanfang' hinterlegt - zur Unterscheidung von 'preload places' besitzen 'postsave places' per Definition nur einlaufende Kanten. Analog hierzu wird innerhalb des Konstruktors der Klasse 'Erzeuger' die diesem als Argument übergebene Berechnungsgrenze im Attribut 'Obergrenze' durch Verwendung eines weiteren 'postsave place' gespeichert. Die Terminierung einer Methode wird durch einen sog. 'exit place' modelliert, der durch einen doppelten Rand gekennzeichnet wird. Gibt eine Methode ein Ergebnis zurück, so ist dies der Wert des Tokens, das die Methodenausführung unter Verwendung eines 'exit place' beendete.

Innerhalb der 'start'-Methode (Abb. 4) der Klasse 'Erzeuger' stehen bei Aktivierung durch die Verwendung von 'preload places' die aktuellen Werte der Attribute 'Obergrenze' und 'Siebanfang' sowie eine mit dem Wert '3' initialisierte lokale Variable vom Typ 'int' als Anfangsmarkierung bereit. In jedem Durchlauf der in dieser Methode modellierten Schleife wird diese Zählervariable erhöht und der somit erzeugte Primzahlkandidat der 'check'-Methode des initialen Siebobjektes übergeben. Erreicht der Wert des Zähler die Obergrenze des Berechnungsverfahrens, so terminiert die Methode.

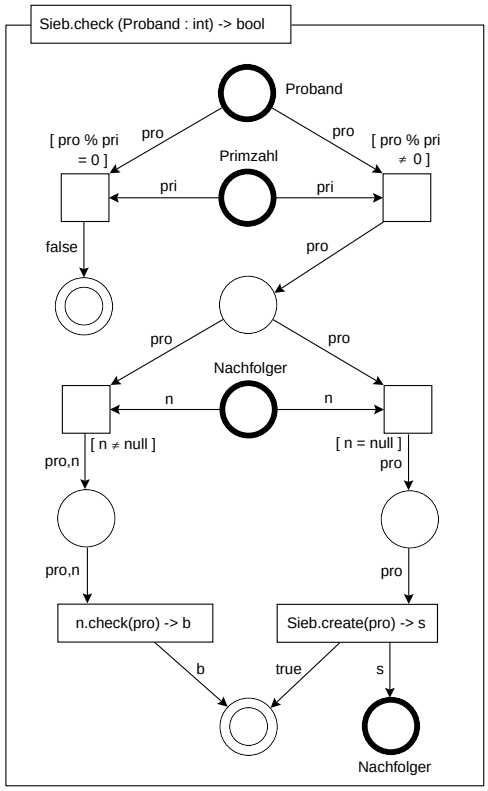

Abb. 5: Modell der 'check'-Methode

Innerhalb der 'check'-Methode der Klasse 'Sieb' (Abb. 5) erfolgt die Überprüfung der übergebenen Kandidaten im Hinblick auf die Primzahleigenschaft. Durch die Verwendung von 'preload places' stehen bei der Methodenaktivierung das übergebene Argument sowie die aktuellen Werte der Attribute 'Primzahl' und 'Siebanfang' als Anfangsmarkierung zur Verfügung. Kann der übergebene Kandidat ohne Rest durch die vom betrachteten Siebobjekt repräsentierte Primzahl geteilt werden, so liegt per Definition keine neue Primzahl vor und die Methode terminiert. Existiert andernfalls für das betrachtete Siebobjekt bereits ein Nachfolger, so ist mit Hilfe der 'check'-Methode des Nachfolgers zu überprüfen, ob die Nachfolgeprimzahl den Kandidaten teilt. Wurde hingegen bisher kein Nachfolger für das betrachtete Siebobjekt bestimmt, so ist der Kan-

didat selbst eine Primzahl und ein zusätzliches Siebobjekt zu dessen Repräsentation ist zu instanziieren. Die Referenz auf dieses Siebobjekt wird durch Verwendung des mit dem Bezeichner 'Nachfolger' beschrifteten 'postsave place' im gleichnamigen Attribut für weitere Überprüfungen hinterlegt. Das Ergebnis der Berechnung ist somit eine verkettete Liste von Siebobjekten, die in aufsteigender Reihenfolge die ermittelten Primzahlen bis hin zur Berechnungsobergrenze repräsentieren.

4 Zusammenfassung und Ausblick

Aufgrund der Ausrichtung der Entwicklung der OOPr/T-Modelle auf den im zweiten Abschnitt vorgestellten Kriterienkatalog, erlauben diese sowohl die objektorientierte Strukturierung von Petri-Netzen als auch die formal basierte Modellierung nebenläufiger objektorientierter Systeme. Im Vergleich zu existierenden Ansätzen, bieten OOPr/T-Modelle darüber hinaus ein deutlich höheres Maß an Benutzerfreundlichkeit, wenngleich im Zusammenhang mit diesem Kriterium weitere Untersuchungen erforderlich sind, um den Grad der Akzeptanz gerade auch bei nicht informationstechnisch vorgebildeten Experten anderer Fachrichtungen zu evaluieren. Als erster Schritt in dieser Richtung wurde zur Unterstützung der OOPr/T-Modellierung ein prototypisches Werkzeug entwickelt [George99], daß zusätzlich zur Gestaltung der einzelnen Sichten insbesondere auch die Generierung von nebenläufigem Java-Programmcode aus OOPr/T-Modellen ermöglicht. Neben der Durchführung von Fallstudien, werden zukünftig Konzepte zur Handhabung persistenter und verteilter Objekte entwickelt. Ebenso soll ein GUI-Builder in das Werkzeug integriert werden, um Systeme mit graphischer Benutzeroberfläche entwerfen zu können.

Fernziel der durch diese Erweiterungsmöglichkeiten aufgezeigten Entwicklungsrichtung ist die Bereitstellung einer integrierten Entwicklungsumgebung für nebenläufige/verteilte objektorientierte Systeme auf der Basis von Petri-Netzen.

Literatur

[GenLau81] **H. J. Genrich und K. Lautenbach**. *'System Modelling with High-Level Petri Nets'*. Theoretical Computer Science, 13(1), 1981.

[George99] **T. George**. *'OOPr/T-Modeller : Ein Werkzeug zur Modellierung nebenläufiger objektorientierter Systeme auf der Basis von UML und Petri-Netzen'*. Diplomarbeit, Universität Koblenz-Landau, 1999.

[HuJeSh90] **Peter Huber, Kurt Jensen und Robert M. Shapiro**. *'Hierarchies in Coloured Petri Nets'*. G. Rozenberg, *'Advances in Petri Nets 1990'*, LNCS 483. Springer-Verlag, 1990.

[LanHau94] **K. Lano und H. Haughton**. *'A Comparative Description of Object-Oriented Specification Languages'*. K. Lano und H. Haughton, *'Object-Oriented Specification Case Studies'*. Prentice Hall International, 1994.

[Phil99] **S. Philippi**. *'Synthese von Petri-Netzen und objektorientierten Konzepten'*. Dissertation, Universität Koblenz-Landau, 1999.

[Rati97] **Rational Software Corporation**. *'UML-Documentation V1.1'*. erhältlich über 'www.rational.com/uml', 1997.