Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association Suisse des Electriciens, de l'Association des Entreprises électriques suisses

Band: 79 (1988)

Heft: 17

Artikel: Petri-Netze : Netze zur Modellierung verteilter Systeme

Autor: Gisiger, H. P. / Kündig, A.

DOI: https://doi.org/10.5169/seals-904071

Petri-Netze: Netze zur Modellierung verteilter Systeme

H.P. Gisiger, A. Kündig

Unter Petri-Netz versteht man ein Modell, welches sich besonders gut zur Beschreibung verteilter Systeme – im wesentlichen kooperierender Prozesse – in der Informationstechnik eignet. Der Prozessfortschritt wird in einem Petri-Netz durch das bedingte Feuern sogenannter Transitionen dargestellt. Da ein Petri-Netz nur aus wenigen, einfach visualisierbaren Elementen besteht, anderseits aber auf einem soliden theoretischen Fundament beruht, kann es als guter Kompromiss für ein zukunftsträchtiges Beschreibungs- und Entwurfsverfahren betrachtet werden.

Par réseau de Pétri, on entend un modèle qui se prête particulièrement bien à la description de systèmes répartis – pour l'essentiel des processus coopérants – dans les techniques de l'information. L'avance du processus est représentée dans un réseau de Pétri par la production conditionnelle de dites transitions. Etant donné qu'un réseau de Pétri ne se compose que d'un petit nombre d'éléments faciles à visualiser mais que d'autre part il repose sur de solides fondements, il peut être considéré comme un bon compromis pour une méthode de description et de projet prometteuse.

Adresse der Autoren

Hans Peter Gisiger, Dipl. El.-Ing. ETH und Prof. Dr. *Albert Kündig* Institut für Elektronik, ETH-Zentrum, 8092 Zürich.

Die Verwendung von Modellen besitzt in den verschiedensten Bereichen der Ingenieurtätigkeit seit Jahrzehnten eine zentrale Bedeutung. So bildet z.B. der Wasserbauer ein komplexes System von Kanälen, Schleusen, Pumpen und Speicherbecken in einem verkleinerten Modell nach. Experimente mit dem Modell erlauben ihm, Rückschlüsse auf den Wasserhaushalt des geplanten Systems zu ziehen. Auf diese Weise hofft er, Fehler in der Gestaltung und Dimensionierung eliminieren zu können, bevor in teure Bauwerke Geld investiert wird. Auch der Architekt oder Stadtplaner lässt seine Vision einer Grossüberbauung zuerst in ein Modell einfliessen. Die am Vorhaben Beteiligten und Betroffenen können sich frühzeitig dank dem Modell eine Vorstellung von der Wirklichkeit machen, und Korrekturen an den Plänen lassen sich praktisch ohne Aufwand anschaulich darstellen.

An die Seite dieser altvertrauten Modelle treten immer mehr rechnergestützte Verfahren. Computer Aided Design (CAD) - rechnergestützte Entwurfsverfahren - gehören heute zum Rüstzeug des Maschinenbauers, des Verfahrensingenieurs, des Regelungstechnikers und vieler anderer Berufsleute. Wesentlich scheint uns auch die Tatsache zu sein, dass es in vielen Fällen nicht nur um eine wirtschaftlichere Gestaltung der Entwurfsarbeit geht: Bei grossen technischen Systemen kann eine optimale Gesamtlösung unter Umständen gar nicht mehr innert nützlicher Frist manuell und mit heuristischen Verfahren gefunden werden, da die Komplexität jenseits des vom einzelnen noch Überschaubaren liegt.

Eigenartigerweise wird aber gerade auf dem Gebiet der Computertechnik, welche andere technische Disziplinen seit Jahren revolutioniert, vom Einsatz von Rechnern in der Entwurfsphase von Programmen oder Computersystemen noch überraschend wenig Gebrauch gemacht. Damit möchten wir natürlich nicht die grossartigen Leistungen herabmindern, welche die Informatik in den letzten 30 Jahren erst auf den heutigen Stand gebracht haben. Stellvertretend erwähnen wir den Einsatz von höheren Programmiersprachen, welche den effizienten und sicheren Bau von System- und Anwendungsprogrammen erlauben. Wie ein Beitrag in dieser Zeitschrift kürzlich aufzeigte [1], gestattet die Strukturierung in *Module* in derartigen modernen Programmiersprachen eine saubere Gliederung eines Programmsystems in Einheiten, welche einen natürlichen inneren Zusammenhang aufweisen und als Ansatz für eine sauber abgegrenzte arbeitsteilige Entwicklung oder die baukastenartige Erweiterung dienen können. Erst der Einsatz des Computers für die Aufgabe der Übersetzung von Hochsprachprogrammen (Compilierung) in maschinenspezifische Befehlsfolgen verhalf dieser Technik zum Durchbruch.

Im vorliegenden Beitrag soll nun ein Verfahren vorgestellt werden, welches sich für die Modellierung und den Entwurf von Systemen auf einer höheren Ebene eignet. Wir denken dabei vor allem an Systeme, welche aus mehreren kooperierenden Rechnern bestehen, und die im Sinne der Steuerung oder Regelung in Echtzeit mit vorgegebenen physikalischen, chemischen und anderen Prozessen zusammenarbeiten. Als Schlagwort für die Bezeichnung entsprechender Entwurfsverfahren ist ab und zu schon der Begriff *Programming in the Large* verwendet worden.

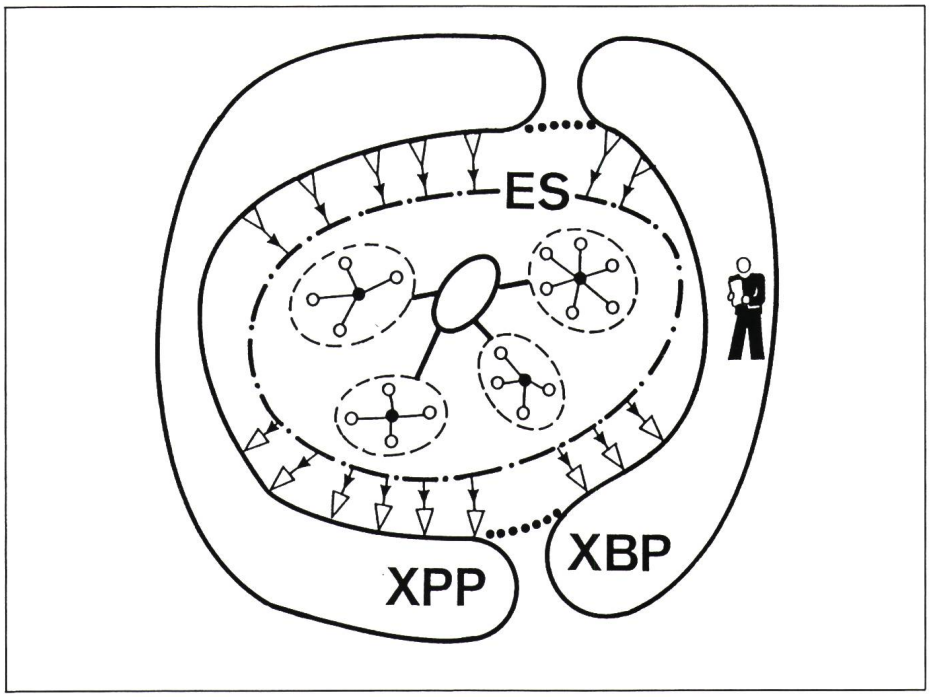
Welche Arten von Systemen sollen modelliert werden?

Der Begriff System kann hier in einem sehr weiten Sinne aufgefasst werden: Er umfasst nicht nur elektro-

nische und rechnerintegrierte Anlagen, sondern Systeme aller Art, in denen Flüsse von Gegenständen und Informationen auftreten - also auch organisatorische oder logistische Systeme. Besonders interessieren uns aber doch sogenannte *eingebettete Systeme*, wie sie in Figur 1 auf einer recht abstrakten Ebene dargestellt sind. Diese Illustration impliziert folgende Charakteristiken:

- Das betrachtete System ist in eine Umgebung eingebettet, mit welcher es über geeignete *Sensoren* und *Aktoren* kooperiert. An den Interaktionspunkten werden sowohl einzelne, zu beliebigen Zeitpunkten auftretende *Meldungen* oder *Befehle* ausgetauscht wie auch kontinuierliche Datenströme.
- Mit den Anforderungen an das zu entwickelnde eingebettete System (Pflichtenheft, Requirements Specification) werden unter anderem bestimmte logische und zeitliche Zusammenhänge zwischen den Eingabe- und Ausgabesignalen vorgeschrieben.
- Bei umfangreicheren Anlagen wird das eingebettete System zweckmässigerweise in mehrere *Subsysteme* zerlegt (ein Vorgang, welcher rekursiv im Sinne der schrittweisen Verfeinerung auch wieder auf ein Subsystem angewendet werden kann). Für eine solche Zerlegung können einerseits die räumliche Ausdehnung sprechen, anderseits aber auch Funktionen, welche nur mit besonders leistungsfähigen, aufgabenspezifischen Teilsystemen verwirklicht werden können. Eine vernünftige Aufteilung in Untersysteme kann auch gute Voraussetzungen für die arbeitsteilige Entwicklung schaffen und wird später im Betrieb die Lösung von Problemen wie Eingrenzung von Fehlern und Reparatur von Anlageteilen erleichtern.

Wer praktische Beispiele sucht, welche mit dieser Beschreibung in Übereinstimmung gebracht werden können, wird ein breites Spektrum von denkbaren Anwendungen finden, angefangen von vergleichsweise einfachen Geräten wie einer Telefonstation oder einem Fahrkartenautomaten bis hin zu komplexen Anlagen, zum Beispiel zur Steuerung von chemischen Produktionsprozessen oder Flugzeugen. Diese Beispiele zeigen auch, dass die zeitlichen Anforderungen viele Grössenordnungen überstreichen können. Während bei den erstgenannten Geräten menschliche Erwartungen Antwortzeiten im Bereich von etwa 0,1...1 s vorschreiben, wird bei kritischen technischen Prozessen zuweilen eine Reaktionszeit unterhalb einer Millisekunde gefordert.



Figur 1 Eingebettetes System (Embedded System)

ES Eingebettetes System
..... mit Sensoren u. Aktoren
--·-- ohne Sensoren u. Aktoren

XPP externer physischer Prozess

XBP externer Bedienungsprozess

Sensor

Aktor

Prozessor

Prozessor-Gruppe (Processor Cluster)

Intergruppen-Kommunikationssystem

Anforderungen an Modelle

Bei den von uns betrachteten Systemen wird eine geordnete Entwicklung und Produktion immer mehrere Phasen durchlaufen. Eine denkbare Aufteilung in verschiedene Abschnitte zeigt die Figur 2, welche aber keineswegs implizieren soll, dass die verschiedenen Phasen zeitlich streng nacheinander abgewickelt werden. Vielmehr wird das Erkennen von Fehlern oder das Zurückkommen auf frühere Entwurfsentscheide oft zu einem iterativen Vorgehen zwingen, bei welchem die gezeigten Phasen unter Umständen mehrmals durchlaufen werden (dementsprechend wird neuerdings von einem der «Altmeister» des Software Engineering - *B.W. Boehm* - ein Spiralmodell für die Softwareentwicklung und -erweiterung vorgeschlagen [2]). Viel wichtiger im Zusammenhang mit den von uns gesuchten Modellen ist die Interpretation von Figur 2 als Darstellung verschiedener Abstraktionsstufen für das gleiche System. Während die Anforderungen (das Pflichtenheft des Auftraggebers einer Entwicklung) eine Beschreibung auf sehr hoher Abstraktionsstufe in der Form der Interaktionen zwischen eingebettetem System und Umgebung umfassen, werden die Detailspezifikationen für ein Hardware- oder Softwaremodul schliesslich auf der Ebene digitaler Signalfolgen oder mit den Elementen einer Programmiersprache formuliert werden. Allen Ebenen ist nun aber gemeinsam, dass ein wesentlicher Teil der Beschreibung von Schnittstellen die *Spezifikation* und die *Kooperation* von *Prozessen* betrifft. Wir verstehen unter Prozess für den Moment eine in elementare Schritte auflösbare Aktionsfolge, getragen von einem aktiven Objekt (einem Prozessor zum Beispiel oder allgemein einem Subsystem). Durch den Austausch von *Meldungen* soll die Zusammenarbeit von Prozessen sichergestellt werden (Tab. I).

Bei jedem einigermassen realistischen System kann weder davon ausgegangen werden, dass alle seine Komponenten streng synchronisiert

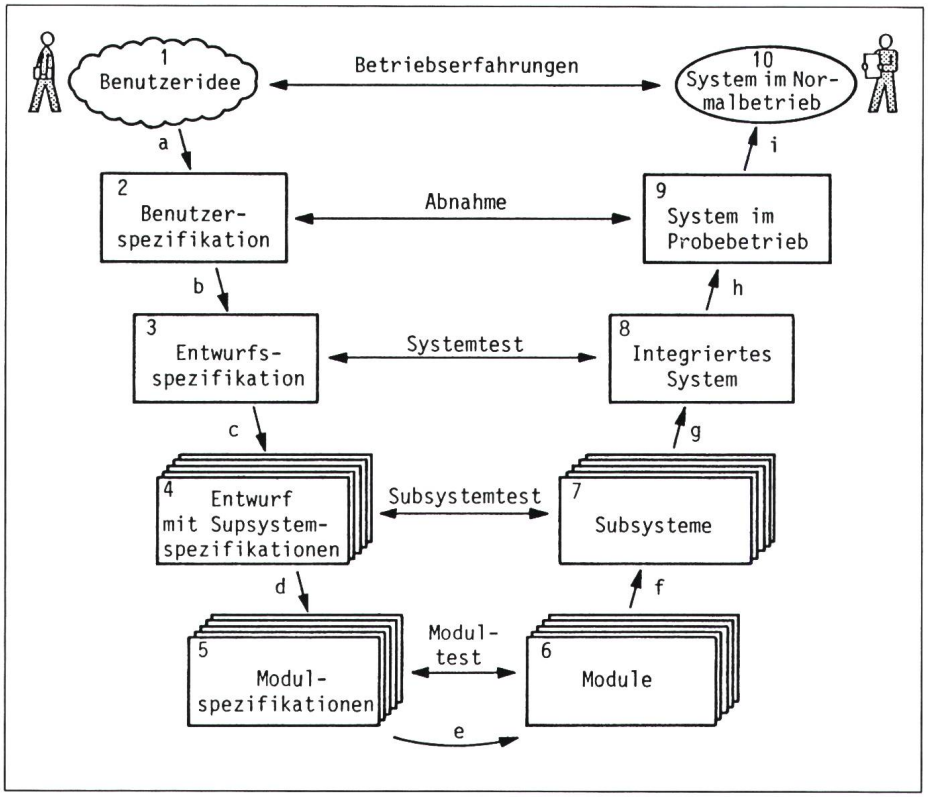
sind, noch dass alle Subsysteme und ihre Verbindungen ohne Fehler funktionieren. Eine wesentliche Anforderung an das gesuchte Modell muss deshalb sein Vermögen betreffen, asynchrone und zufällige Ereignisfolgen darstellen zu können. Dies bedeutet insbesondere, dass die an sich naheliegende Verwendung deterministischer endlicher Automaten als Modell verworfen werden muss.

Die beiden wohl wichtigsten Forderungen an irgendein Modellierungsverfahren (also nicht nur für die Beschreibung verteilter Systeme) sind nicht immer leicht unter einen Hut zu bringen. Verlangt werden nämlich die folgenden Eigenschaften [3]:

a. Eine möglichst hohe *Ausdruckskraft* (Modellierungskraft). Das Verfahren soll in verschiedener Hinsicht brauchbar sein. Gefordert werden insbesondere:
- Abdeckung möglichst vieler Klassen von Systemen, sei es konzeptueller Art (im Sinne von abstrakten, mathematischen Grundlagen), sei es aus der Sicht der Anwendung (z.B. Regelung chemischer Prozesse, Fabrikationsautomatisierung usw.),
- Abdeckung möglichst vieler Systemaspekte (z.B. Struktur, Verhalten, Leistung),
- Erweiterbarkeit (z.B. Verknüpfung mit anderen Verfahren).

b. Eine möglichst gute *Manipulierbarkeit*, d.h.
- Möglichkeit der Simulation durch den Computer,
- leichte Verständlichkeit für Menschen, zum Beispiel im Rahmen von Analysen, bei der Entwurfsarbeit und bei Transformationen,
- Ausgewogenheit zwischen Elementen, welche die Intuition gut ansprechen, und der strikten Befolgung mathematisch fundierter Zusammenhänge.

Auf jeden Fall muss heute gefordert werden, dass sich ein Modellierungsverfahren auf saubere mathematische Konzepte abstützt. Nur auf diese Weise wird es gelingen, die Modelle einer maschinellen Bearbeitung zugänglich zu machen, und nur so werden die Voraussetzungen für eine formale Verifikation erfüllt. Von besonderer Bedeutung sind dann sogenannte *bedeutungserhaltende Transformationen*, zum Beispiel die Abbildung eines Modelles auf eine neue Struktur unter Erhaltung der nach aussen gezeigten Eigenschaften.



Figur 2 Phasenweise Entwicklung grösserer Systeme

Zu den Begriffen und Definitionen

Im Laufe dieses Beitrages setzen wir uns mit einer Reihe von Begriffen auseinander, welche uns sehr wohl aus der Umgangssprache vertraut sind: Ereignis, Prozess, verteiltes System, Meldung, Zeit, Netz, ... Unwillkürlich wird der Leser deshalb auf vertraute Bilder aus seiner täglichen Umgebung zurückgreifen, wenn er unsere Darlegungen über Petri-Netze zu verstehen versucht. Dabei wird nicht zu vermeiden sein, dass scheinbar immer wieder Widersprüche auftauchen. Dies ist aber kein Grund zum Aufgeben! Die Widersprüche sind natürlich gerade darin zu suchen, dass mit den Petri-Netzen konsistente, auf mathematische Prinzipien abgestützte Konzepte und eine entsprechende Begriffswelt aufgebaut werden, wohingegen die Umgangssprache oft mehrere Interpretationen zulässt. Wo immer deshalb im vorliegenden Artikel Sätze auftreten wie zum Beispiel *eine Bedingung ist eine Vorbedingung eines Ereignisses, wenn...*, so müsste eigentlich die Einschränkung *definitionsgemäss* eingeschoben werden. Die Begriffswelt der Petri-Netze ordnet also gewohnten Bezeichnungen wie Ereignis eine ganz bestimmte, gegenüber der Umgangssprache eingeschränkte Bedeutung zu - hier zum Beispiel nimmt man an, dass ein Ereignis in infinitesimal kurzer Zeit vollzogen werde (eine sogenannte *unteilbare Aktion* mit verschwindender Dauer - engl. atomic action).

Eine weitere Quelle von Verständnisschwierigkeiten hat ihren Ursprung in der Thematik verteilter Systeme selbst. Unsere dem täglichen Leben und der Natur entnommenen Bilder «verteilter Systeme» gehen nämlich von der Vorstellung aus, dass die räumlich verteilten Ereignisse in einem solchen System in ein globales Zeitraster - eine globale Ordnung - eingebunden werden können (Ereignis *X* im Ort *A* erfolgte vor Ereignis *Y* im Ort *B*). Die heutigen Massenkommunikationsmittel und moderne Verfahren der Zeitbestimmung lassen uns eben glauben, dass eine solche Ordnung existiere. In Wirklichkeit handelt es sich aber nur um eine angenäherte Ordnung, und zwar darum angenähert, weil im allgemeinen die beobachteten Prozesse eine gegenüber der Geschwindigkeit der Informationsübertragung (der Lichtgeschwindigkeit) bedeutend kleinere Fortschrittsgeschwindigkeit besitzen. Diese Betrachtung muss dann versagen, wenn - wie dies natürlich bei Prozessen in elektronischen Systemen der Fall ist - der Prozessfortschritt mit den gleichen Einheiten gemessen werden muss wie die Geschwindigkeit der Informationsübermittlung. Tatsächlich haben wir nun ein *relativistisches System* vor uns, und eine globale Ordnung kann gar nicht definiert werden! Um so wichtiger ist es auch, Begriffe wie *Ereignis* und *Zeit* sauber zu definieren, wenn wir Trugschlüsse vermeiden wollen.

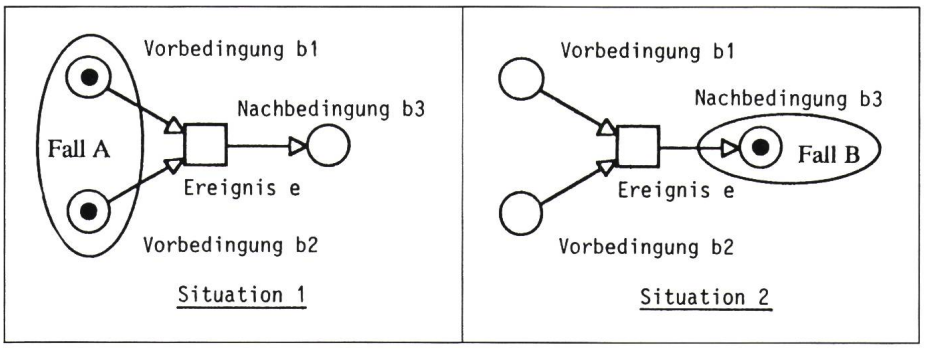
Tabelle I

Petri-Netze: Eine erste Übersicht

Ein Hauptproblem bei der Beschreibung räumlich verteilter Systeme liegt - wie bereits erwähnt - in der korrekten Beschreibung ihres Verhaltens bei parallelen bzw. gleichzeitigen Abläufen. Petri-Netze sind wegen ihrer Möglichkeit, Gleichzeitigkeit darzustellen, besonders geeignet, verteilte Systeme zu modellieren. Im Gegensatz zu vielen anderen Methoden werden in Petri-Netzen Daten- und Kontrollflüsse gemeinsam im gleichen Modell dargestellt. Zu ihren Vorteilen gehören vor allem die graphische Anschaulichkeit, die einfache Anwendbarkeit sowie ihr fundierter mathematischer Hintergrund. Besonders anschaulich präsentiert sich ein Petri-Netz durch die Tatsache, dass sowohl die *Struktur* wie auch das *dynamische Verhalten* eines Systems mit derselben graphischen Notation beschrieben werden. Zustände und Änderungen eines Systems sind jederzeit aus dem Modell ersichtlich.

Der mathematische Hintergrund der Petri-Netze ist eine inzwischen etablierte Systemtheorie, die *Netztheorie* [4]. *C.A. Petri* formulierte bereits im Jahre 1962 in seiner Dissertation zum Thema *Kommunikation mit Automaten* die Grundlagen einer Theorie über Kommunikation zwischen asynchronen Komponenten von Computersystemen [5] und legte damit den Grundstein zur Netztheorie. In der Zwischenzeit wurden unzählige weitere Netzmodelle definiert, die jeweils für spezielle Problemkreise und Abstraktionsbedürfnisse zweckmässig sind. Sie hängen alle untereinander durch gemeinsame Interpretationsmuster zusammen und können auch als Methodik zur Darstellung von Systemen mit verschiedenen Freiheitsgraden verstanden werden.

Als einfachstes Netzmodell gilt das *Bedingungs-Ereignis-Netz*. Es ist äusserst einfach zu handhaben, seine Modellierungskraft jedoch ist kaum ausreichend, um grössere Systeme zu beschreiben. Erweiterungen des Modells führen zu den *Stellen-Transitions-* und schliesslich zu den *Colored Petri-Nets* und den *Prädikat-Transitions-Netzen*. Sobald grössere Systeme mit Netzen beschrieben werden sollen, wird das Modell schnell unübersichtlich. Es muss daher eine zusätzliche Strukturierungsmöglichkeit eingeführt werden. *Hierarchische Netze* unterstützen bekannte Methoden wie z.B. stufenweises Verfeinern (Stepwise Refinement oder Top-Down-Design), aber auch den umgekehrten Vorgang im Sinne einer Vergröberung bzw. Abstraktion (Bottom-up-Design). Damit kann ein System schrittweise von einer informellen Beschreibung der Struktur zur präzisen formalen Spezifikation seines dynamischen Verhaltens übergeführt werden.



Figur 3 Grundlegende Begriffe

Bei vielen Anwendungen sollen Aussagen über das zeitliche Verhalten der Systeme gemacht werden. Ursprünglich erlaubten Petri-Netze keine Modellierung der Zeit. Diese Lücke ist aber inzwischen durch die Erweiterung zu *zeitbehafteten Netzen* (Timed Petri Nets) geschlossen worden.

Netzmodelle

Als typische Vertreter unzähliger Netzmodelle werden hier nur die wichtigsten und bekanntesten kurz vorgestellt.

1. Einfache Petri-Netze: B/E-Netze

Bedingungen und *Ereignisse* sind die primitiven Elemente der einfachsten Klasse von Petri-Netzen, der Bedingungs-Ereignis-Netze oder B/E-Netze (engl. C/E für Condition/Event). Bedingungen (S-Elemente[1]) werden dargestellt durch Kreise ○ und Ereignisse (T-Elemente[1]) durch Rechtecke □. *Pfeile* (gerichtete Kanten) verbinden Bedingungen mit Ereignissen (○→□) bzw. Ereignisse mit Bedingungen (□→○). *Marken* (Tokens) in gewissen Bedingungen ⊙ ergeben die *Anfangsmarkierung* (Anfangsfall) eines Netzes, d.h. die Bedingungen, die zu Beginn (Initialisierungszustand) der Ausführung erfüllt sind (Fig. 3). Eine Bedingung ist eine *Vorbedingung* eines Ereignisses, falls ein Pfeil von der Bedingung zum Ereignis existiert (b○→□e). Ebenso ist eine Bedingung eine *Nachbedingung* eines Ereignisses, falls ein Pfeil vom Ereignis zur Bedingung existiert (e□→○b).

Eine Bedingung in einem Netz ist in jeder Situation entweder erfüllt oder nicht erfüllt, und jede erfüllte Bedingung ist mit einer Marke gekennzeichnet. Die Summe aller in einem Netz erfüllten Bedingungen ergeben in einer bestimmten Situation einen *Fall*. Wenn alle Vorbedingungen eines Ereignisses in einem B/E-Netz erfüllt und alle seine Nachbedingungen nicht erfüllt sind, *kann* dieses Ereignis eintreten. Solche Ereignisse heissen *aktiviert*. Tritt ein aktiviertes Ereignis ein, so werden seine Vorbedingungen *unerfüllt* und seine Nachbedingungen *erfüllt*.

Die Figur 4 zeigt anhand eines anschaulichen Beispiels eine Situation je vor und nach dem Eintritt eines Ereignisses. Die Situation 1 zeigt den Fall vor dem Eintritt des Ereignisses mit den beiden erfüllten Vorbedingungen *Schraube vorhanden* und *Mutter vorhanden;* die Marken Mutter und Schraube liegen auf den Bedingungen. Die Nachbedingung *Mutter auf Schraube* ist nicht erfüllt. Da beide Vorbedingungen erfüllt sind und die Nachbedingung nicht erfüllt ist, kann das Ereignis *Mutter und Schraube zusammensetzen* eintreten. Situation 2 zeigt den Fall nach dem Ereignis. Nur noch die Nachbedingung Mutter auf Schraube ist erfüllt, die Vorbedingungen sind nicht mehr erfüllt.

Wenn zwei aktivierte Ereignisse mindestens eine gemeinsame Vorbedingung oder Nachbedingung besitzen, können sie miteinander in *Kon-*

[1] Durch diese Bezeichnungen wird von Anfang an die Beziehung zu den Begriffen, *Stellen* und *Transitionen hergestellt.*

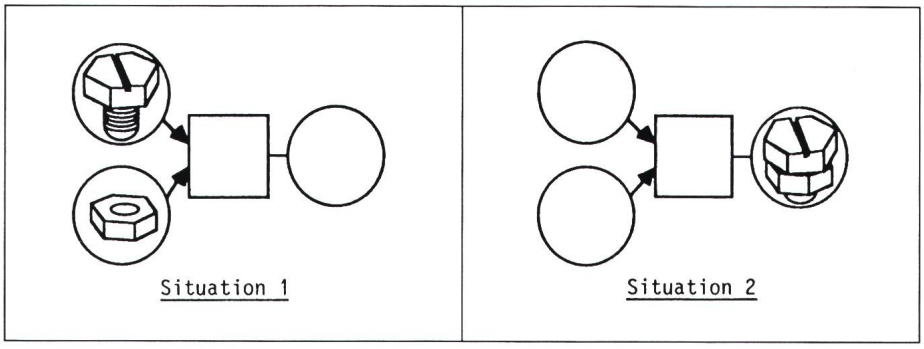
flikt stehen, sie konkurrieren miteinander. Tritt nun eines der Ereignisse ein, so ist das andere nicht mehr aktiviert. Das tatsächliche Eintreten konkurrierender Ereignisse ist zufällig und das Verhalten des Netzes ist somit nicht mehr deterministisch. Die Figur 5 zeigt drei typische Konfliktsituationen.

Wie ein weiteres Beispiel in Figur 6a illustriert, kann es vorkommen, dass zwar die Vorbedingungen a eines Ereignisses T2 erfüllt sind, aber auch mindestens eine seiner Nachbedingungen g. In dieser Situation kann das Ereignis T2 nicht eintreten; es entsteht eine sog. *Kontaktsituation*. *Kontaktfrei* sind Netze nur, wenn nie ein Kontakt entstehen kann. Um dies zu erreichen, darf das Eintreten eines Ereignisses nur von seinen Vorbedingungen, nicht aber von einer seiner Nachbedingungen abhängen. Wird ein nicht-kontaktfreies B/E-Netz durch *Komplemente* erweitert, so wird es kontaktfrei. Das Netz in Figur 6b ist die kontaktfreie Erweiterung von Figur 6a. Die zusätzliche Vorbedingung h ist stets komplementär zur Nachbedingung g und sorgt dafür, dass die Vorbedingungen des Ereignisses T2 nur dann erfüllt sind, wenn seine Nachbedingung nicht erfüllt ist.

Eingangs haben wir einen Prozess als eine Folge von Aktionen definiert. Tatsächlich können wir nun in Figur 6 drei sich immer wiederholende (also zyklische) Aktionsfolgen identifizieren, nämlich einen Produzentenprozess P und zwei Konsumentenprozesse K1 und K2. Das Fortschreiten dieser Prozesse ist im wesentlichen durch die folgenden Ereignisketten definiert:

P: $T2 \rightarrow T1 \rightarrow T2 \rightarrow T1 \rightarrow ...$
K1: $T5 \rightarrow T6 \rightarrow T5 \rightarrow T6 \rightarrow ...$
K2: $T4 \rightarrow T3 \rightarrow T4 \rightarrow T3 \rightarrow ...$

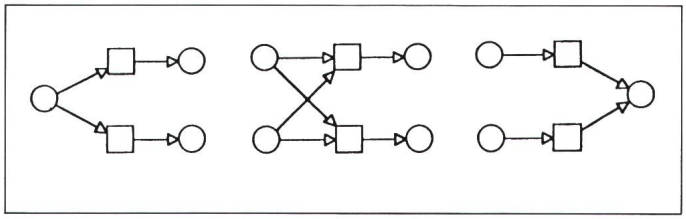
Die drei Prozesse können aber je für sich nicht frei ablaufen, da ihre Kopplung über die Bedingungen g (Figur 6a) bzw. g und h (Figur 6b) dazu führt, dass P nur voranschreiten (produzieren) kann, wenn auch K1 und/oder K2 voranschreiten (konsumieren). Man beachte, dass die Systembeschreibung von Figur 6 an sich keine Aussagen über das Verhältnis der Fortschrittsgeschwindigkeiten von K1 und K2 erlaubt, da eines der Ereignisse T5 und T3 beim Erfüllen der Bedingung g zufällig eintritt. Anderseits hat die Prozesskopplung zur Folge, dass im Mittel T2 genau gleich häufig eintritt wie T5 und T3 zusammen.



Figur 4 Anschauliches Beispiel eines einfachen Netzes

Vgl. Figur 3. Erst wenn die Verbindungen (Schraube vorhanden, Mutter vorhanden) erfüllt sind, und die Nachbedingung unerfüllt, d.h. der Platz des Mutter-Schrauben-Produkts leer ist, kann das Ereignis (Verbinden von Schrauben und Mutter) eintreten.

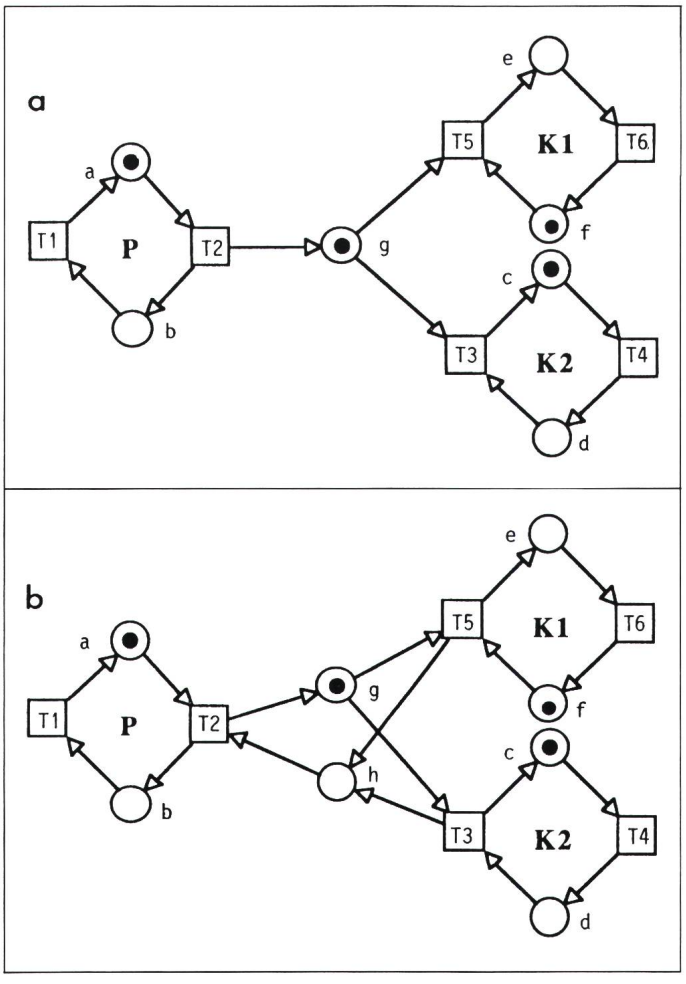
Figur 5
Typische Konfliktsituationen



Figur 6
Modell mit einem Produzenten und zwei Konsumenten

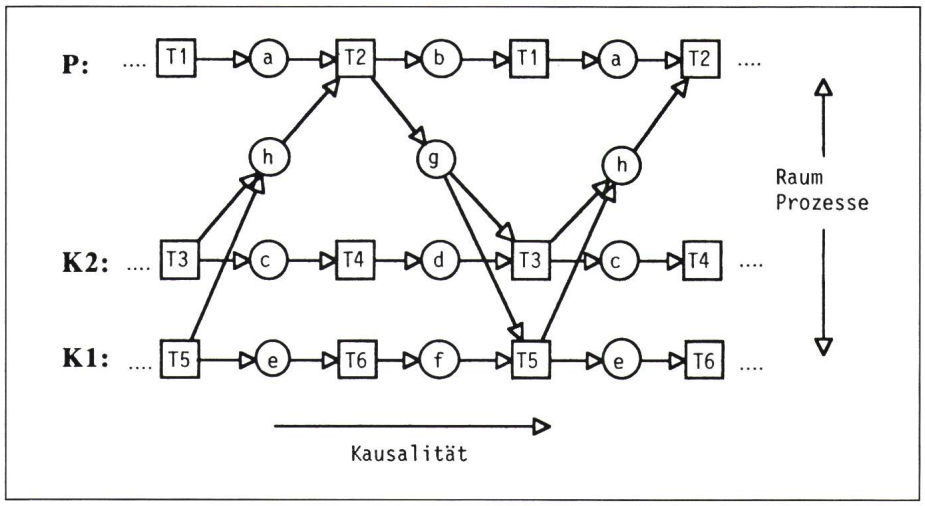
a B/E-Netz mit Kontaktstelle g

b Kontaktfreie Erweiterung durch Komplementierung von g durch h. T_2 kann erst eintreten, wenn die Bedingung h erfüllt, d.h. wenn g unerfüllt ist.

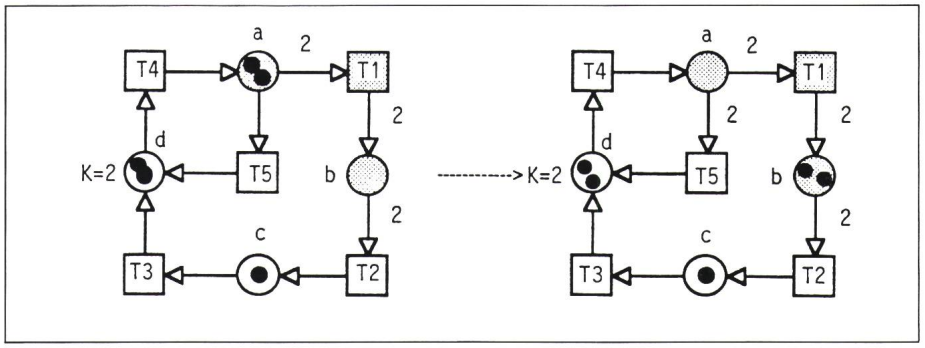


Die Netztheorie erlaubt nun, eine formalere Definition des Begriffes Prozess vorzunehmen, welche für kontaktfreie Netze gelten soll. Die Figur 7 zeigt, wie ein Prozessgraph zum Netz von Figur 6b konstruiert werden kann. Prozesse in einem B/E-Netz falten das Netz auf und zeigen Schritt für Schritt den Fortgang der Ereignisse auf. Da in einem verteilten System a priori *keine* totale Ordnung der Ereignisse (Transitionen) existiert, definiert ein Prozess nur partielle Reihenfolgen von Ereignissen. Wo Reihenfolgen oder Ordnungen in einem System erwünscht sind, müssen sie erzwungen werden; andernfalls soll nicht mit ihnen gerechnet werden, da ja der Vorteil von Netzen gerade das unabhängige (parallele, nebenläufige, konkurrente) Eintreten von Ereignissen oder Transitionen ist.

Ein Prozess wird so dargestellt, dass jedem Ereignis ein Rechteck und jeder erfüllten Bedingung ein Kreis zugeordnet wird. Pfeile beschreiben dabei den kausalen Zusammenhang zwischen den erfüllten Bedingungen und den eintretenden Ereignissen. Die *Modellierungskraft* einfacher Petrinetze (B/E-Netze) reicht jedoch nicht aus, um komplexere Systeme einigermassen übersichtlich zu beschreiben.



Figur 7 Prozessgraph
zu Figur 6b



Figur 8 Stellen-Transitions-Netz

Stellen-Transitions-Netze

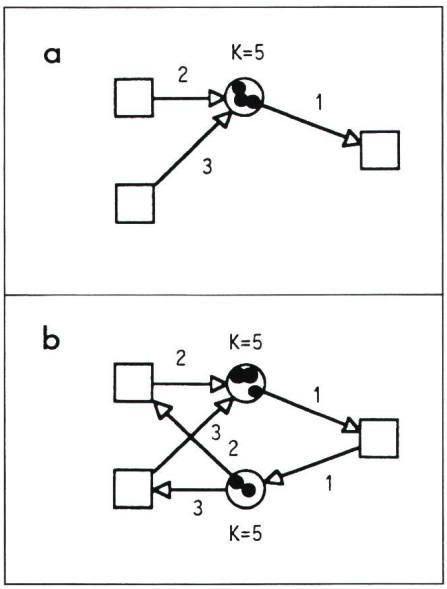
Stellen-Transitions-Netze (S/T-Netze, engl. P/T für Place/Transition) [6] unterscheiden sich von B/E-Netzen, indem S-Elemente (Stellen) mehr als eine Marke tragen und T-Elemente (Transitionen) entsprechend der Gewichte an den gerichteten Kanten an einer Stelle mehrere Marken hinzufügen oder wegnehmen können. Bedingungen werden zu *Stellen*, Ereignisse zu *Transitionen*. Stellen können mehrere Marken enthalten, eine Bedingung kann somit mehrfach erfüllt sein. Durch die gewichteten Kanten können in einer Transition ebenso mehrere erfüllte Bedingungen konsumiert, bzw. produziert werden.

Eine Stelle kann zusätzlich eine Kapazität ungleich eins erhalten (notiert als K = ...). Die Kapazität definiert eine maximale Anzahl von Marken, die auf einer Stelle liegen dürfen; das Feuern einer Transition darf somit gar nicht erfolgen, wenn diese Kapazität überschritten würde. Kapazitäten entsprechen den Nachbedingungen in B/E-Netzen. B/E-Netze können also durch S/T-Netze mit Kapazitäten und Gewichten gleich eins dargestellt werden. Die Anfangsmarkierung legt die Anfangszahl der Marken jeder Stelle fest (sie muss stets kleiner oder gleich der Kapazität der betreffenden Stelle sein). Aktiviert werden Transitionen nur, wenn bei keiner Stelle im Vorbereich die Kantengewichte die Anzahl vorhandener Marken übersteigt und nach dem Ereignis die Kapazitäten der Stellen im Nachbereich nicht überschritten werden.

Eine Transition kann *schalten* (feuern), wenn sie aktiviert ist. Schaltet eine Transition, so werden die Marken auf den Eingangsstellen um die Gewichte der Pfeile vermindert und die Marken der Ausgangsstellen um die Gewichte der Pfeile erhöht (Fig. 8). Wie bei B/E-Netzen kann auch bei S/T-Netzen durch geeignete Massnahmen (Komplementierung) dafür gesorgt werden, dass jedes Netz kontaktfrei wird (Fig. 9). Mit unendlichen Kapazitäten können S/T-Netze im Gegensatz zu B/E-Netzen einen unendlichen Zustandsraum erhalten (Fig. 10). S/T-Netze sind geeignet, Ausschnitte der realen Welt insofern zu modellieren, als nur das Verhalten aller Objekte unabhängig von ihren Attributen interessiert. Nicht die Individualität, sondern lediglich die Anzahl der Marken sind deshalb in S/T-Netzen von Bedeutung. Diese Einschränkung kann bei der nachfolgend beschriebenen Netzklasse fallengelassen werden.

Prädikat-Transitions-Netze

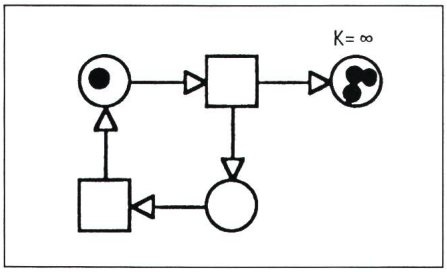
Der Vollständigkeit halber soll auch noch auf eine der modellkräftigsten Netzklassen, die Prädikat-Transitions-Netze (Pr/T-Netze) [7; 8] kurz eingegangen werden. Pr/T-Netze sind formale Objekte, die in mathematischer Art und Weise interpretiert und manipuliert werden können, vergleichbar mit logischen Formeln und algebraischen Ausdrücken. In einem Pr/T-Netz kann jede Marke ein individuelles Objekt sein, das mit einem oder mehreren Attributen charakterisiert wird. Die Anzahl Attribute einer Marke wird durch die Stelle, auf der sie



Figur 9 Komplementierung von S/T-Netzen

a S/T-Netz mit Kontaktstelle

b Kontaktfreie Erweiterung durch Komplementierung



Figur 10 Stelle mit unendlich grosser Kapazität führt zu einem unendlichen Zustand

liegt, bestimmt. Stellen werden zu Prädikaten, die, angewandt auf eine individuelle Marke, entweder wahr oder falsch sein können. Einfache Gewichte an den gerichteten Kanten (wie bei S/T-Netzen) werden durch eine formale Summe von Tupeln individueller Marken ersetzt. Nur Marken, deren Attribute in den Tupeln an den beschrifteten Kanten enthalten sind, können Transitionen aktivieren; die Kanten-Beschriftungen wirken also wie Filter. Sollen beliebige Marken (d.h. mit völlig frei wählbaren Attributen) oder wenigstens Marken mit teilweise freien Attributen zur Aktivierung zugelassen werden, so müssen freie Variablen in den Kantenbeschriftungen deklariert sein. Die Figur 11 zeigt den Übergang von einfachen B/E-Netzen zu einem Pr/T-Netz.

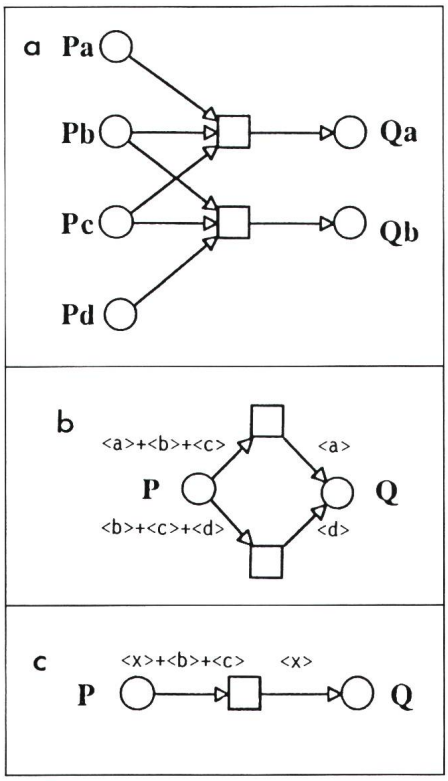
Transitionen werden durch die Individualisierung der Marken, sobald sie aktiviert werden, zu Instanzen von Transitionen. Alle Variablen der Transitionen werden dabei durch individuelle Symbole oder Marken, die mit den Beschriftungen an den Kanten vereinbar sind, ersetzt. Treten Variablen in einer Transition mehrfach auf, so werden sie stets durch dasselbe Symbol ersetzt. Zur Aktivierung einer Transition können Transitionen selber weitere Transitionsbedingungen enthalten. Erst wenn auch diese Bedingungen erfüllt sind, kann die Transition feuern (Fig. 12).

Hierarchische Erweiterung

Eine zusätzliche Strukturierung kann in einem Netz durch die Einführung von Hierarchien verwirklicht werden. Systeme können somit schrittweise *verfeinert* bzw. *vergröbert* werden. Unter Verfeinerung versteht man das Ersetzen von Netzknoten durch ein detaillierteres Unter-Netz. Vergröberung beschreibt den umgekehrten Vorgang. Dabei könnten grundsätzlich sowohl S- wie auch T-Elemente verfeinert bzw. vergröbert werden. Wie im Beitrag [9] in diesem Heft gezeigt wird, ist eine Verfeinerung der Transitionen vorzuziehen, entspricht dies doch einer besseren zeitlichen Auflösung eines auf der höheren Ebene noch als unteilbar (atomisch) betrachteten Vorganges.

Wie die Figur 13 zeigt, werden die oberen Netzebenen in einer hierarchischen Beschreibung oft als *Kanal-Instanzen-Netze* bezeichnet, wobei Kanäle den S-Elementen und Instanzen den T-Elementen entsprechen. Die Beschriftungen von Kanälen und Instanzen werden im allgemeinen noch als informell betrachtet und erst in der Verfeinerung formalisiert. Kanäle stellen passive Systemkomponenten dar, die Informationen speichern können; Instanzen repräsentieren aktive Systemkomponenten, die Objekte oder Marken verändern können. Pfeile oder gerichtete Kanten zeigen die logischen Zusammenhänge der Komponenten.

Hierarchische Erweiterungen sind *keine* Erweiterung des Netz-Modells selber, sondern blosse methodische Hilfen beim Entwurf von Systemen. Bei der Verfeinerung bzw. Vergröberung sind jedoch gewisse Regeln zur *Konsistenzerhaltung* des Modells zu beachten. Das Netz sollte vor der Operation kontaktfrei sein. Gerichtete Kanten können zusätzlich nur innerhalb der aktuellen Hierarchie zwischen S- und T-Elementen eingefügt werden. Kanten, die von einem Unter-Netz weg- oder zu ihm hinführen,

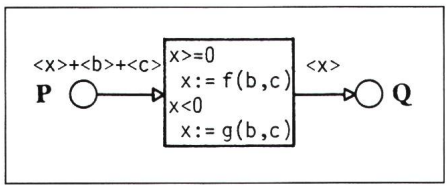


Figur 11 Übergang von einem B/E-Netz zu einem Pr/T-Netz

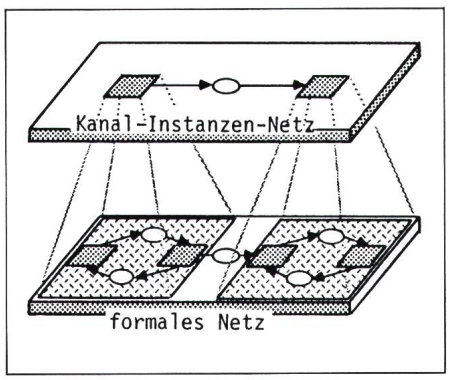
a Einfaches B/E-Netz

b S/T-Netz mit Summen von Attributen an den Konnektoren

c Pr/T-Netz mit freier Variable an Konnektoren



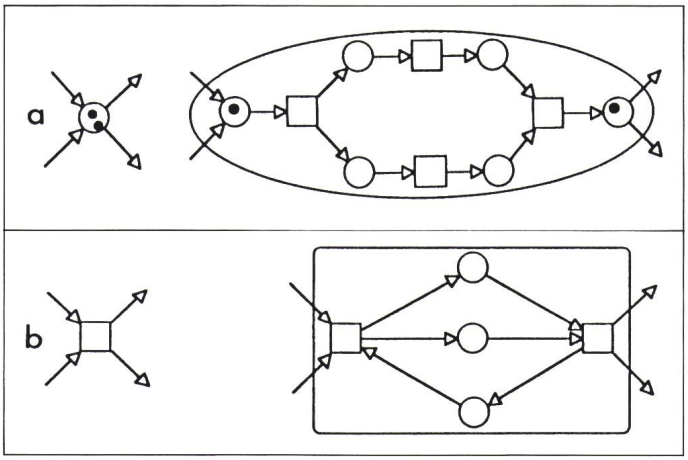
Figur 12 Transition mit zusätzlichen Bedingungen



Figur 13 Verfeinerung von Stellen und Transitionen

Obere Ebenen: Höhere Abstraktionsstufen

müssen bereits vor der Verfeinerung vorhanden sein und die vorgegebenen Richtungen müssen berücksichtigt werden (Fig. 14). Weiter muss beachtet werden, dass die Verfeinerung *markentreu* ist. Eine Verfeinerung soll gleich viele Marken abgeben wie sie konsumiert.



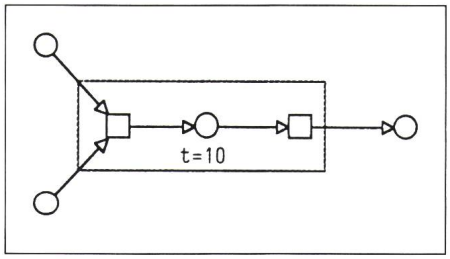
Figur 14
Markentreue Verfeinerung
a Verfeinerung einer Stelle
b Verfeinerung einer Transition

Zeitbehaftete Petri-Netze

Da Petri-Netze keiner zeitlichen, sondern einer *kausalen Ordnung* gehorchen, können sie weder den Zeitpunkt eines Ereignisses noch eine zeitliche Abhängigkeit zwischen Ereignissen bestimmen. In einem verteilten System gibt es eben *keine globale Zeit!* Aus diesem Grund ist es vom theoretischen Standpunkt aus auch nicht sinnvoll, eine solche einzuführen. In Petri-Netzen bedeutet der Begriff Gleichzeitigkeit von Ereignissen eigentlich Unabhängigkeit von Ereignissen: Zwei Ereignisse erfolgen gleichzeitig, bzw. zwei Prozesse laufen parallel ab, wenn keines der beiden Ereignisse nur als Folge des andern auftreten kann.

In einem verteilten System ist die Zeit immer an einen bestimmten Ort gebunden. Es gibt deshalb keine globale, jedoch beliebig viele lokale Zeiten: Da die Kommunikation zwischen den lokalen Teilsystemen immer eine endliche Zeit erfordert, kann auch keine Koordination (Synchronisation) dieser Teilsysteme zu einer wirklich globalen Ordnung führen.

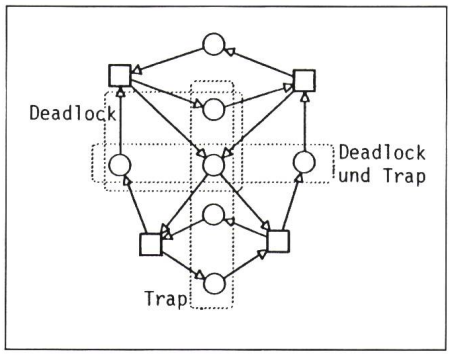
Zur Untersuchung der Leistungsfähigkeit oder zur Simulation eines Systems muss jedoch oft die Dauer einer Aktion oder eines Ereignisses spezifiziert werden können. Atomare Ereignisse oder Transitionen sind per Definitionem beliebig schnell und benötigen somit keine Zeit. Hingegen kann ein zusammengesetztes Ereignis oder eine Transition eine bestimmte Zeitdauer benötigen. Um diese Eigenschaft in einem Modell darzustellen, müssen die Marken eine bestimmte Zeitdauer ($t>0$) auf einer Stelle verharren, bevor sie von einer Transition konsumiert werden können (Fig. 15). Die Zeitintervalle können dazu in den Stellen oder den Transitionen festgelegt werden, verharren müssen die Marken aber jeweils auf den Stellen. Würden Transitionen Marken konsumieren und erst nach einer gewissen Zeitdauer wieder Marken an die Stellen in ihrem Nachbereich weitergeben – was auf den ersten Blick naheliegend wäre –, so würden die atomaren Eigenschaften einer Transition verletzt und Netz-Invarianten wären zu gewissen Zeiten nicht mehr erfüllt.



Figur 15 Zeitbehaftetes Netz

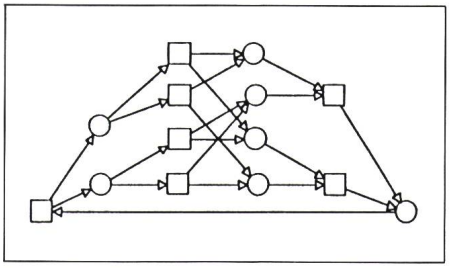
Analyse von Netzen

Bei der Analyse von Petri-Netzen sollen spezielle Eigenschaften wie z.B. *Lebendigkeit* (Liveness) und *Fairness* untersucht werden. Wir beschränken uns bei dieser Einführung auf Methoden, die für die Analyse von S/T-Netzen relevant sind. Um festzustellen, ob Teile eines S/T-Netzes lebendig sind, kann eine *Erreichbarkeitsanalyse* durchgeführt werden. Ausgehend von einem Anfangszustand wird festgestellt, ob gewisse Zustände in einem Netz erreichbar sind. Bei Lebendigkeitsuntersuchungen sind Netzteile interessant, die nie markiert werden oder die nie alle Marken verlieren. Eine Stellenmenge S heisst *Deadlock*, falls sie – einmal ohne Marken – niemals mehr markiert werden kann (Fig. 16). Deadlocks sind besonders kritische Systemteile, weil – wenn einmal ohne Marken – Transitionen in ihrem Nachbereich nicht mehr aktivierbar sind. Eine Stellenmenge S heisst *Trap*, wenn sie – einmal markiert – niemals mehr alle Marken verlieren kann. Ein Trap tritt dann auf, wenn jede Transition, die Marken aus S entnimmt, auch mindestens wieder eine in S ablegt [4].



Figur 16 Deadlocks und Traps in einem Netz

Mit der Einführung von *Free-Choice-Netzen* [10] erhält man eine Netzklasse, die sich wegen Strukturbeschränkungen einfacher untersuchen lässt. Es kann gezeigt werden, dass ein Free-Choice-Netz genau dann lebendig ist, wenn jeder seiner Deadlocks einen markierten Trap enthält. Free-Choice-Netze sind Netze, in denen Transitionen einer vorwärts verzweigten Stelle nicht rückwärts verzweigt sein dürfen. In solchen Netzen wird zwischen den Transitionen eines Konfliktes frei und unabhängig von anderen Stellen ausgewählt (Fig. 17).



Figur 17 Free-Choice-Netz (Beispiel)

Werden in einem System Zustände selten erreicht oder werden gewisse Transitionen anderen gegenüber unbeabsichtigt bevorzugt, ist das Modell nicht fair. Fairness kann entweder mit

einer Erreichbarkeitsanalyse oder durch statistische Auswertung von Simulationsergebnissen untersucht werden. Leider kann die Erreichbarkeitsanalyse bereits bei mittelgrossen Netzen nicht mehr erfolgreich angewandt werden. Bei B/E-Netzen ist eine derartige Untersuchung grundsätzlich noch sinnvoll; der Zustandsraum wird zwar schnell sehr gross, bleibt jedoch endlich. Bei höheren Netzen (S/T- und PrT-Netze) kann der Zustandsraum unendlich werden.

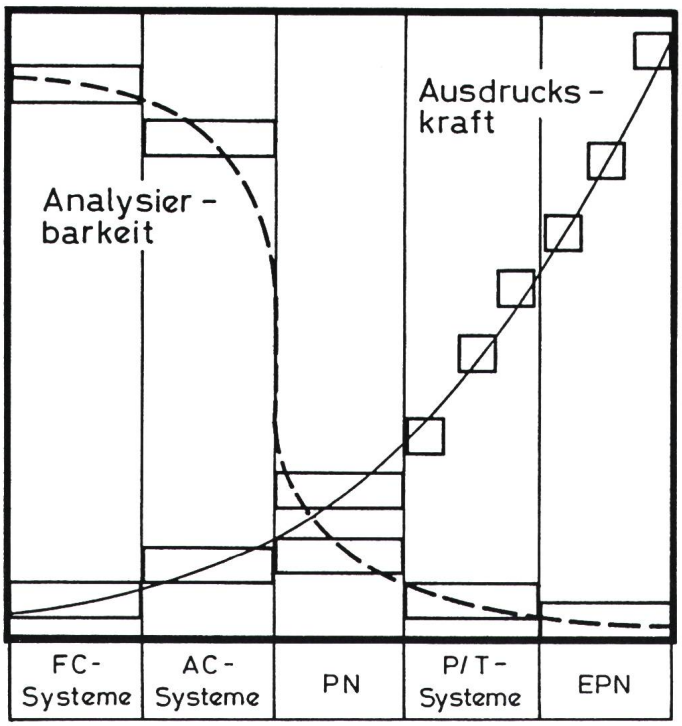
Eine weitere Methode zur Analyse von Netzen sind *Invarianten*. Stellenmengen von Netzen, deren Gesamtzahl an Marken unverändert bleibt, wenn Transitionen schalten, heissen *S-Invarianten*. Sie unterstützen die Analyse auf Lebendigkeit sowie weitere Systemeigenschaften. Neben S-Invarianten können auch *T-Invarianten* formuliert werden. Sie geben an, wie oft welche Transition schalten muss, um die Anfangsmarkierung des Netzes zu reproduzieren [4].

Eine vollständige Analyse von Netzen kann leider nur bei einfachen (B/E-Netze) oder gar eingeschränkten Netzen (Free-Choice-Netzen) erfolgreich durchgeführt werden. Überschreitet ein Modell eine gewisse Grösse, so ist die vollständige Analyse von vornherein aussichtslos. Es gelten daher die in Figur 18 aufgezeigten qualitativen Zusammenhänge.

Figur 18
Zusammenhang zwischen Ausdruckskraft und Analysierbarkeit für verschiedene Netzklassen

Qualitative Beurteilung mit willkürlichem Massstab

FC Free Choice
AC Asymmetric Choice
PN Petri-Netze (im engeren Sinne)
P/T Stellen/Transitionen (Place/Transition)
EPN Erweiterte Petri-Netze



Schlussbemerkungen

Aus Platzgründen musste sich dieser Beitrag darauf beschränken, einen ersten Eindruck von Petri-Netzen zu vermitteln. Der Leser wird sich aber erst von der Nützlichkeit eines Modells überzeugen lassen, wenn er anhand von Anwendungsbeispielen erkennt, dass die vorher aufgestellten Anforderungen - vor allem bezüglich Modellierungskraft und Manipulierbarkeit - wirklich erfüllt werden. Darüber hinaus wird er heute verlangen, dass das Erstellen eines Netzmodelles und dessen Weiterbearbeitung rechnergestützt erfolgen können. Beiden Anliegen wird im Beitrag [9] in dieser Zeitschrift entsprochen. Darüber hinaus verweisen wir auf die in den letzten Jahren sprunghaft angewachsene Literatur, so vor allem auf die Berichte über regelmässige internationale Tagungen [11], ein spezialisiertes Mitteilungsblatt [12] sowie auf die Bücher [13; 14] und [15].

Nicht eingehen konnten wir in diesem Beitrag auf sozusagen konkurrierende Verfahren aus dem Bereiche der sogenannten Prozessalgebren wie CCS [16] und CSP [17] sowie die damit stark verwandte, von der ISO entwickelte Spezifikationsmethode Lotos [18]. Auch für einen Vergleich mit dem in der Nachrichtentechnik verbreiteten SDL-Verfahren [19], welches teilweise auf einem Modell mit erweiterten endlichen Automaten beruht, mussten wir aus Platzgründen verzichten.

Literatur

[1] *J. Gutknecht:* Modulare Programmierung mit Modula-2. Bull. SEV/VSE 78(1987)1, S. 8...14.

[2] *B.W. Boehm:* A spiral model of software development and enhancement. IEEE Computer 21(1988)5, p. 61...72.

[3] *R.T. Boute:* Elements for the formal description of systems. In: Embedded systems. New approaches to their formal description and design. An advanced course, Zurich, March 1986. Edited by: A. Kündig, R.E. Bührer and J. Dähler. Lecture notes in computer science 284. Berlin a.o., Springer-Verlag, 1987; p. 63...89.

[4] *H.J. Genrich, K. Lautenbach* and *P.S. Thiagarajan:* Elements of general net theory. In: Net theory and applications. Proceedings of the advanced course on general net theory of processes and systems. Hamburg, October 8...19, 1979. Edited by W. Brauer Lecture notes in computer science 84. Berlin a.o., Springer-Verlag, 1980; p. 21...163.

[5] *C.A. Petri:* Kommunikation mit Automaten. Dissertation der Technischen Hochschule Darmstadt, 1962.

[6] *W. Reisig:* Place/transition systems. In: Petri nets: Central models and their properties. Advances in Petri nets 1986. Part I: Proceedings of an advanced course. Bad Honnef, September 1986. Edited by W. Brauer, W. Reisig and G. Rozenberg. Lecture notes in computer science 254. Berlin a.o., Springer-Verlag, 1987; p. 117...141.

[7] *H.J. Genrich* and *K. Lautenbach:* System modelling with high-level Petri nets. Theoretical Computer Science 13(1981)1, p. 109...136.

[8] *H.J. Genrich:* Predicate/transition nets. In: Petri nets: Central models and their properties. Advances in Petri nets 1986. Part. I. Proceedings of an advanced course. Bad Honnef, September 1986. Edited by W. Brauer, W. Reisig and G. Rozenberg. Lecture notes in computer science 254. Berlin a.o., Springer-Verlag, 1987; p. 207...247.

[9] *J. Dähler:* Petri-Netze. Teil 2: Ein interaktives Simulations- und Programmgenerierungswerkzeug für höhere Petri-Netze. Bull. SEV/VSE 79(1988)17, S. 1035...1040.

[10] *M. Hack:* Analysis of production schemata by Petri nets. Technical report No. 94. Project MAC. Cambridge/Mass. Massachusetts Institute of Technology, 1972.

[11] European workshop on applications and theory of Petri nets. Venice 22...24 June 1988. (Tagungsberichte auch früherer Konferenzen, werden im Springer-Verlag publiziert).

[12] Petri net Newsletter. Herausgegeben von der Gesellschaft für Informatik, Special Interest Group on Petri Nets and related System Models. (Anschrift: Postfach 1669, D-5300 Bonn 1).

[13] *W. Reisig:* Systementwurf mit Netzen. Berlin u.a., Springer-Verlag, 1985.

[14] *J.L. Peterson:* Petri net theory and the modelling of systems. Englewood Cliffs, Prentice-Hall, 1981.

[15] *W. Reisig:* Petri Netze. Berlin u.a., Springer-Verlag, 1982.

[16] *R. Milner:* A calculus of communicating systems. Lecture notes in computer science 92. Berlin/Heidelberg/New York, Springer-Verlag, 1980.

[17] *C.A.R. Hoare:* Communicating sequential processes. Englewood Cliffs, Prentice-Hall, 1985.

[18]18 Lotos - a formal description technique based on the temporal ordering of observational behaviour. ISO draft proposal 8807.

[19] Functional specification and description language (SDL). CCITT recommendations Z 100. CCITT red book. Volume VI, fascicule VI.11. Geneva, International Telecommunication Union, 1985.