Validierung eines Petrinetz-basierten Steuerungssystems*

Tobias Vesper Michael Weber

Humboldt-Universität zu Berlin
Institut für Informatik
Unter den Linden 6
10099 Berlin

*gefördert durch die Deutsche Forschungsgemeinschaft (DFG) im Rahmen der Forschergruppe „Petrinetz-Technologie“

Inhaltsverzeichnis

Einleitung

Seit nunmehr fast zwei Jahren befaßt sich die Gruppe TheoPAn der DFG-Forschergruppe PETRINETZ-TECHNOLOGIE mit der Validierung eines industriellen Systems, das mit dem Einsatz von Petrinetzen in den Jahren von 1991 bis 1995 an der Universität der Bundeswehr in München mit industrieller Unterstützung entwickelt wurde.

Das Ziel unserer Validierung war eine Schnittstelle zwischen der praktischen Entwicklung des Systems einerseits und der Entwicklung theoretischer Resultate andererseits. Konkreter wollten wir Ansatzpunkte für neue, direkt anwendbare theoretische Resultate herauskristallisieren, sowie andererseits untersuchen, wo bekannte theoretische Resultate eingesetzt wurden beziehungsweise werden können. Unsere Vorgehensweise bestand hierbei darin, zuerst Probleme zu erkennen, mit dem Anwender zu diskutieren und die Probleme und deren Lösungen später verallgemeinert zu formulieren.

Dieser Bericht bündelt unsere Erfahrungen. Er enthält keine theoretischen Resultate. Er dokumentiert auch kein Softwaresystem. Wir versuchen hier Ansatzpunkte für theoretische und Theorie-Praxis übergreifende Forschungsthemen aufzuzeigen, die sich direkt aus einer industriellen Fallstudie ergeben. Wir zeigen noch keine vollständigen Lösungen, sondern wir beschreiben die Probleme abstrakt und zeigen ansatzweise Lösungsideen.

Aufgrund der Komplexität und der vielen beim Systementwurf relevanten Themen gliedern wir diesen Bericht in verschiedene Sichtweisen, die im DFG-Antrag der Forschergruppe [WER95] formuliert wurden. Wir untersuchen in Abschnitt 1 das *Vorgehensmodell*, nach dem die Fallstudie entwickelt wurde. In Abschnitt 2 beschreiben wir typische *Kontrollstrukturen*, die so verallgemeinerbar sind, daß sie auch in anderen Projekten eine Rolle spielen können. In Abschnitt 3 beschreiben wir die *Strukturierung* der Fallstudie aus pragmatischen und Petrinetz-technischen Gesichtspunkten. In den Abschnitten 4 bis 7 beschreiben wir weitere Sichtweisen: *Datenstrukturen*, *Zeitaspekte*, *Simulation*, *Analyse* sowie *Verifikation*. Einen wichtigen Überblick über die derzeit (bzw. zur Zeit der Entwicklung der Fallstudie) mögliche Unterstützung durch Werkzeuge wird, bezogen auf die Fallstudie, in Abschnitt 8 gegeben.

Die Fallstudie CASSY

Das Cockpitassistenzsystem (CASSY) wurde an der Universität der Bundeswehr unter der Leitung von Prof. Dr. R. Onken und Dr. W. Ruckdeschel mit industrieller Unterstützung entwickelt. Ausgangspunkt seiner Entwicklung war die Beobachtung, daß 75% der Flugzeugunfälle durch Überforderung oder Fehler der Besatzung verursacht werden [RO94].

Bisherige Lösungen dieses Problems bestanden in der Automatisierung von Piloten-

tätigkeiten. Eine Folge dessen war eine Zunahme der Anzahl und der Komplexität von Überwachungseinrichtungen für automatische Prozesse. Dadurch wurde es der Besatzung erschwert, Wesentliches und Unwesentliches voneinander zu trennen.

> „Der große Automatisierungsaufwand im Flugzeugcockpit führte ... letztlich zu keiner Erhöhung der Flugsicherheit." [Ruc96]

Mit der Automatisierung veränderte sich also die *Art* der auftretenden Fehler. Waren es bis dahin Fehler des fertigkeitsbasierten Verhaltens (also Bedienfehler), liegt die Ursache der neuartigen Fehler in der Anwendung des Wissens [RO95].

Daher bietet es sich an, eine Flugzeugbesatzung bei ihrer Arbeit zu *unterstützen*, indem sie auf Fehler hingewiesen wird. Eine Möglichkeit, Fehler einer Flugzeugbesatzung durch ein automatisches System zu erkennen, ist der Vergleich des Verhaltens der Flugzeugbesatzung mit einem erwarteten Verhalten. Solch ein erwartetes Verhalten kann ein regelbasiertes System liefern, da im Flugwesen aufgrund sensibler Sicherheitsanforderungen bereits formulierte und validierte Regeln existieren. Regeln finden sich in Flughandbüchern, Flugverkehrsregeln und in der Ausbildung von Piloten. Es lag also nahe, das Cockpitassistenzsystem CASSY mit einem regelbasierten *Referenzpiloten* zu versehen, der seine Vorschläge mit dem aktuellen Verhalten der Besatzung vergleicht.

Der Referenzpilot wurde ausschließlich mit Petrinetzen modelliert. Diese Wahl basierte auf einem umfassenden Vergleich bestehender Beschreibungsmittel [Ruc96]. Wir heben hier insbesondere die folgenden Gründe für die Verwendung von Petrinetzen in CASSY hervor (vgl. [Web97]):

- Anschaulichkeit des Beschreibungsmittels durch Visualisierbarkeit des Modells und der Simulation,
- wirklichkeitsgetreue Modellierbarkeit von Nebenläufigkeit,
- einfache, problembezogene hierarchische Konstruktionsprinzipien,
- leichte Implementierbarkeit aufgrund der einfachen Semantik.

Bevor wir die Sichtweisen auf die Fallstudie im einzelnen betrachten, möchten wir die zwei wesentlichen Intentionen dieses Berichts, nämlich den Transfer praktischer Resultate in die Theorie und umgekehrt zu unterstützen, anhand von zwei Analogien verdeutlichen:

Archimedes beobachtete, daß schwere Güter auf Rollen transportiert wurden. Er untersuchte die theoretischen Zusammenhänge und erfand den Flaschenzug. Mit diesem konnte ein ganzes Schiff von Hand bewegt werden, was bis dahin als undenkbar galt.

Die zweite Analogie ist hoch aktuell: Im Jahre 1997 führte die NASA eine sehr erfolgreiche Expedition zur Erforschung des Mars durch. Die Software für den „Pathfinder" (ein satellitengesteuertes Roboterfahrzeug) war fehlerhaft. Dies führte zu mehreren Total-Resets des Systems, was jeweils mit einem massiven Verlust von Meßdaten einherging. Glücklicherweise konnte der Fehler später behoben werden. Das Problem, auf dem diese Systemabstürze beruhten, war in der Theorie schon seit langem (1980) bekannt und

gelöst. Es wurde jedoch noch nicht in der Praxis umgesetzt.[1]

In diesem Sinne möchten wir zwar nicht das Rad neu erfinden; aber einen Beitrag dazu liefern, Petrinetz-Theorie und -Praxis über eine Petrinetz-Technik zu verbinden.

Danksagungen

Die Autoren möchten an dieser Stelle vielen Unterstützern danken. An erster Stelle danken wir dem Entwickler der Fallstudie, Dr. Wilhelm Ruckdeschel, für seine hervorragende und sorgfältige Arbeit, die uns das Verständnis der Fallstudie ungemein erleichterte. Weiterhin danken wir ihm dafür, daß er uns in seiner Freizeit beraten hat. Dr. Jörg Desel vermittelte den Kontakt zu Prof. Onken und Dr. Ruckdeschel. Dr. Rolf Walter prophezeite uns zu recht, daß wir komplexe Teilmodelle viel besser verstehen würden, wenn wir sie als einfache high-level Netze modellieren würden. Dr. Ekkart Kindler stand uns jederzeit unterstützend, organisierend und beratend zur Verfügung. Prof. Dr. Wolfgang Reisig diskutierte mit uns die Einbettung des ECHO-Algorithmus in das Gesamtmodell (vergl. Abschnitt 2.2). Dr. Karsten Schmidt konstatierte, daß disjunkte S-Invarianten beim Aufräumen helfen (Abschnitt 2.1). Ihnen allen möchten wir an dieser Stelle danken.

Berlin, März 1998 Die Autoren

[1] Eine ausführliche Beschreibung des Problems, des theoretischen Hintergrundes und der Lösung kann man in den Bulletins der comp.risks-Newsgroup, vol. 19, iss. 49 bis iss. 54, 1997/1998 nachlesen.

Die Fallstudie CASSY

1 Vorgehensmodell

Im folgenden Abschnitt beschreiben wir das Vorgehensmodell, nach dem die Fallstudie entwickelt wurde. Es wurde bei der Entwicklung der Fallstudie kein vorhandenes Vorgehensmodell eingesetzt; das hier beschriebene ergab sich aus den protokollierten Gesprächen mit dem Entwickler der Fallstudie [Mar97c].

Das Vorgehensmodell ist graphisch in Abbildung 1 dargestellt. Die Projektentwicklung läßt sich grob in drei Phasen einteilen. Wir werden diese Phasen zuerst kurz umreißen und in den nachfolgenden Abschnitten detailiert beschreiben.

In der ersten Phase (*Akquise*) werden ausgehend von einer groben funktionalen Anforderung seitens des Auftragsgebers Wissensquellen gesammelt, die bei der Modellierung und Validierung helfen können. Danach werden auf der Basis reichhaltiger informeller Beschreibungen ein formales Beschreibungsmittel (hier Stellen/Transitionsnetze) festgelegt und erste Modelle entworfen.

In der zweiten Phase (*simulierte Einbettung*) werden die einzelnen Teilmodelle am Rechner simuliert. Nachdem das Modell weitgehend fehlerfrei simuliert werden konnte, wurde es in eine simulierte Umgebung (hier ein Flugsimulator) eingebettet und dort vom Anwender validiert. Zum Abschluß dieser Phase wird anhand eines nun festgelegten funktionalen Modells ein umfangreiches Redesign der Modellstruktur vorgenommen.

In der dritten Phase (*Einbettung in die Systemumgebung*) wird das Modell an die reale Umgebung angepaßt: Die Kopplungen mit der realen Umgebung werden in das Modell integriert, nach weiteren Simulationen und Analysen wird das Modell in der realen Umgebung (hier im Fluzeug) getestet und von Anwendern abschließend validiert.

1.1 Akquise

In der Fallstudie wurde die erste Phase im Vorgehensmodell durch drei Aufgaben bestimmt: der Wissensakquisition, der Aufgabenanalyse, die ein Beschreibungsmittel festlegt, sowie der Auswahl von geeigneten Werkzeugen zur Unterstützung der Modellierung und Simulation.

Da das Pilotenmodell als eine Referenz während eines Fluges dienen soll, kommt der Wissensakquisition eine besondere Bedeutung bei. Sie wurde unter Ausnutzung verschiedener heterogener Quellen durchgeführt: des eigenen „Wissens“ (also der Erfahrung) des Entwicklers als Flugpilot, des Wissens, das in Flughandbüchern verschiedener Fluggesellschaften beschrieben ist, sowie des Wissens, welches sich aus Interviews mit Berufspiloten ergab. Insbesondere letzteres Wissen war sehr wichtig, da die Piloten in der Praxis die

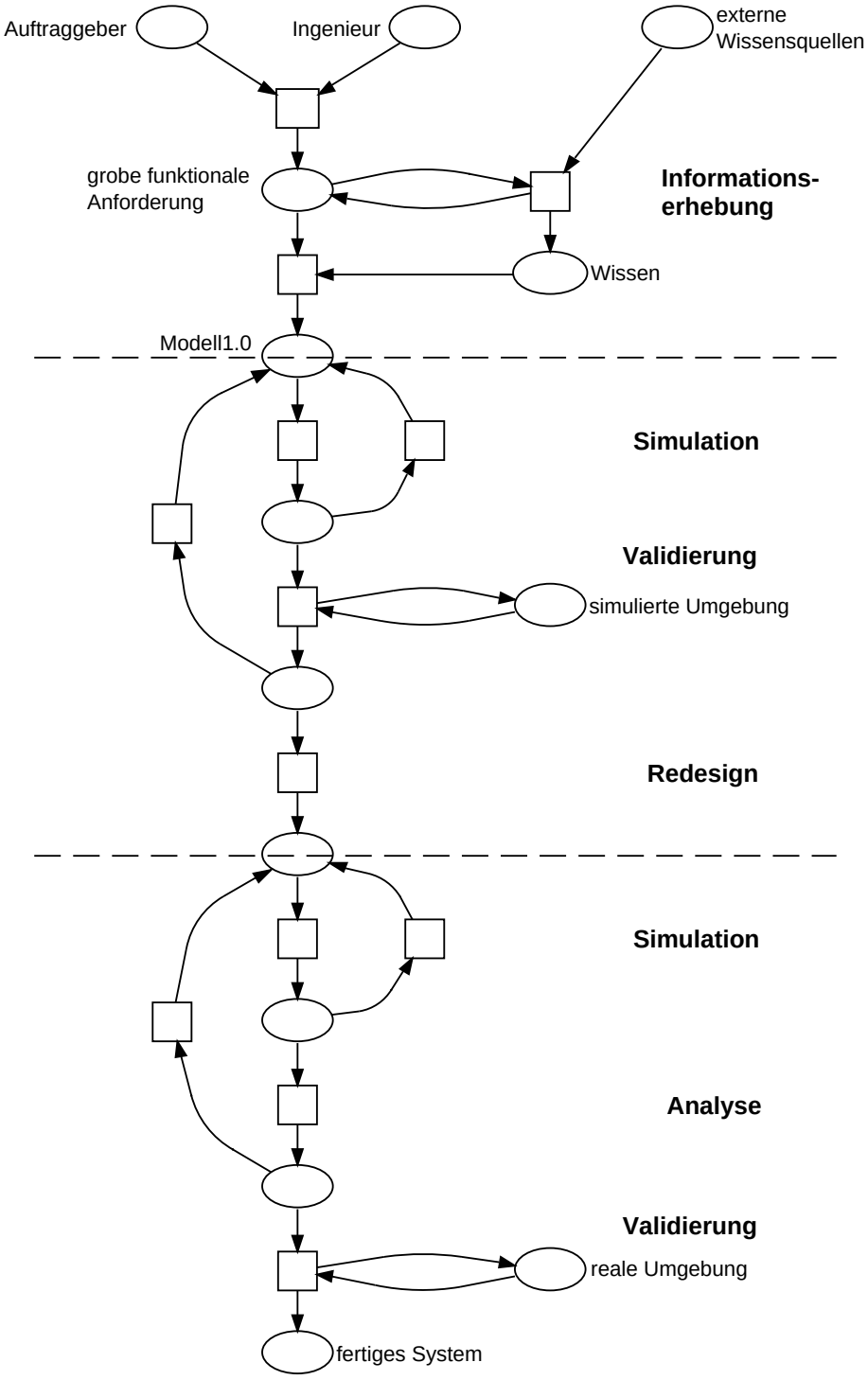

Abbildung 1: Das Vorgehensmodell der Fallstudie

Vorschriften der Flughandbücher aufgrund ihrer Erfahrungen abweichend auslegen. Das gesammelte Wissen liefert die Vorgaben für das Verhalten des Referenzpiloten.

Das vorhandene Wissen (also das Verhalten des Referenzpiloten) mußte in das Modell integriert werden. Die Auswahl des Beschreibungsmittels basierte hierbei auf den folgenden Beobachtungen ([Ruc96]):

- Pilotverhalten ist *stark nebenläufig*. Beispielsweise laufen alle Handlungen zur Situationserkennung unabhängig von der Flugrichtungsbestimmung oder der Bedienung der Landeklappen.
- Das in der Wissensakquisition gesammelte Wissen läßt sich durch *Regeln* beschreiben. Eine Regel hat hierbei die Form: „Wenn ..., dann ...". Ein Beispiel ist: „Wenn die Geschwindigkeit x unterschritten wird, dann verändere den Landeklappenwinkel auf y Grad."
- Der gesamte Zustandsraum kann als *diskret* betrachtet werden. So ist beispielsweise nie eine aktuelle Geschwindigkeit eines Flugzeuges von Interesse, sondern lediglich eine Über- bzw. Unterschreitung eines bestimmten Grenzwertes.
- Pilotenverhalten kann in verschiedenen Abstraktionsstufen betrachtet werden. Das Modell sollte ebenfalls verschiedene Abstraktionsmöglichkeiten bieten, um so überschaubar dargestellt werden zu können.

Aufgrund einer umfassenden Analyse vorhandener Beschreibungsmittel ([Ruc96]) fiel die Wahl auf Petrinetze:

- Ein großer Teil des Wissens kann mit Bedingungs/Ereignissystemen modelliert werden.
- Bei mehrfachen Ressourcen bietet sich der Einsatz von Stellen/Transitionssystemen an.
- Die Formalisierung mit Bedingungs/Ereignisnetzen führt in einigen Fällen zu sich wiederholenden Netzstrukturen. Hier sind high-level Netzklassen sinnvoll.

Wir illustrieren im folgenden an einem Beispiel, wie regelbasiertes Pilotenverhalten modelliert und in ein operationelles Modell eingebunden wurde. Die folgende Regel stellt einen Teil des gesammelten Wissens dar:

> „Wenn die Geschwindigkeit beim Anflug den Referenzwert x unterschreitet, dann soll der Landeklappenwinkel von $0°$ auf $5°$ verändert werden."

Der konkrete Wert von x ist hierbei vom Flugzeugtyp abhängig, in dem CASSY eingesetzt werden soll. Das Petrinetzmodell für diese Regel ist in Abbildung 2 dargestellt. Die zusätzlichen Pfeile an der Transition Switch sollen symbolisieren, daß das Schalten der Transition an eine externe Schaltnebenbedingung gekoppelt ist.

Folgende Aspekte des regelbasierten Verhaltens sind durch dieses Petrinetz beschrieben:

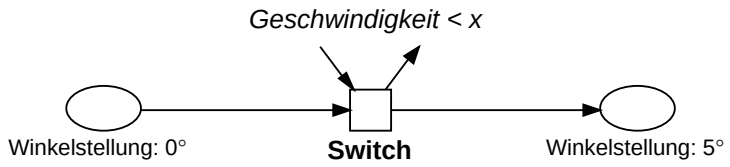

Abbildung 2: Regelbasiertes Verhalten als Petrinetz

- Eine Stelle des Petrinetzes entspricht einer möglichen Flugsituation.
- Eine Transition entspricht einem möglichen Zustandsübergang (der wiederum eine Entsprechung als Regel in der Wissensbasis hat).
- Eine Markierung entspricht einer aktuellen Flugsituation.

Die Integration solcher Regelnetze in ein operationelles Petrinetzodell eröffnet vielfältige Simulations- und Analysemöglichkeiten. Beispielsweise kann die Konsistenz der Wissensbasis überprüft werden, indem untersucht wird, ob ein bestimmter erwünschter Zustand (z.B. die Landeklappen sind ausgefahren) überhaupt erreichbar ist. Weiterhin können auch Abhängigkeiten (bzw. Unabhängigkeiten) zwischen einzelnen Regeln in einem (bzw. verschiedenen) Teilmodellen dargestellt werden.

Bei der dritten Aufgabe, der Auswahl geeigneter Werkzeuge, spielte die grobe funktionale Anforderung eine entscheidende Rolle. Aufgrund von Sicherheitsvorkehrungen war es erforderlich, alle Werkzeuge zur Modellierung und Simulation selbst zu entwickeln. Dies beschränkte die Benutzung von high-level Netzen, da dies im Rahmen des geforderten Zeitplans nicht zu realisieren war. Diese Aufgabe bildete den Hauptarbeitsaufwand: Währernd zur Erstellung der Version 1.0 des Modells ein Mannmonat notwendig war, benötigte die Erstellung des Parsers für die Netzdaten sowie des Simulators 1 Mannjahr.

1.2 Simulierte Einbettung

In der zweiten Phase, der simulierten Einbettung, stand die Simulation des Verhaltens des Referenzpiloten im Flugsimulator im Vordergrund. Wir beschreiben in diesem Abschnitt die Schwerpunkte aus Sicht des Vorgehensmodells, d.h. welche wesentlichen Entscheidungen getroffen wurden und welche Arbeitsschritte für die einzelnen Ergebnisse durchgeführt wurden.

Um die Bewertungen der Berufspiloten möglichst frühzeitig in den Entwicklungsprozeß mit einbeziehen zu können, bestand als eine wichtige Aufgabe der schnelle Einsatz der Modelle in einem Flugsimulator im Sinne des Rapid Prototyping. Die ersten Modelle wurden simuliert und mit dem Petrinetz-Analysewerkzeug INA analysiert, das unter der Leitung von Prof. Starke an der Humboldt-Universität zu Berlin entwickelt wurde. In mehreren Iterationsstufen wurden die Modelle zusammen mit Berufspiloten an einem Flugsimulator validiert. Hierbei stellte sich beispielsweise heraus, daß es sinnvoll ist, erweiterte Toleranzschranken direkt in das Petrinetz-Modell zu integrieren. (Allow/Force-Konzept, vergl. [Ruc96]).

In dieser Phase wurde weiterhin die Modulzerteilung festgelegt. Ausgangspunkt waren hierfür zwei Richtlinien. Einerseits sollte jedes Teilmodell auf einer Seite darstellbar sein. Andererseits sollten bezüglich der Koppelung bzw. Kohäsion die einzelnen Teilmodelle in einem vernünftigen Verhältnis zueinander stehen. Die aus diesen Anforderungen gewählte Strukturierungstechnik ist im Abschnitt 3 im Detail beschrieben. Weiterhin wurden die Teilmodelle in vier *funktionale Netztypen, [RO95, Mar97a]* unterschieden. Diese Unterscheidung von Teilmodellen hinsichtlich ihres Zwecks hilft bei der Festlegung einer geeigneten Modularisierung.

Nachdem die Technik der Kopplung der einzelnen Teilmodelle festgelegt wurde, konnten auch graphische Konventionen geschaffen werden, die die Anschaulichkeit der einzelnen Teilmodelle erhöhen. Diese sind in [Mar97c] ausführlich erläutert.

Zum Abschluß dieser Phase lag ein „festgeklopftes" funktionales Modell vor, d.h. alle wesentlichen funktionalen Bestandteile des Systems waren modelliert.

1.3 Einbettung in die Systemumgebung

In dieser Phase standen die Anbindung des Modells an die Systemumgebung (also CASSY selbst) sowie das Testen im Vordergrund.

Zuerst wurde die Kommunikation mit der Umgebung realisiert. An die Transitionen wurden externe Schaltnebenbedingungen geknüpft. Weiterhin konnten durch die Transitionen Aktionen ausgelöst werden, die als Signale der Umgebung mitgeteilt wurden. Diese Anbindungen an die Systemumgebung wurden in C programmiert.

In der Testphase wurde das Modell in mehreren Zyklen simuliert, insbesondere in Bezug auf das Zusammenspiel mit der Umgebung. Danach wurden einzelne Teilmodelle mit dem Werkzeug PEP analysiert. Soweit der damalige Stand der Technik es erlaubte, wurden auch verbundene Teilmodelle analysiert (vergleiche hierzu Abschnitt 7). Hierbei zeigte sich, daß noch jedes 10. Teilmodell Fehler in sich barg, die während der Simulation nicht erkannt wurden.

Nach Abschluß der Analyse wurde das Modell bei realen Flugversuchen getestet und von Piloten abschließend validiert.

2 Kontrollstrukturen

Die vollständige Analyse der Fallstudie ist aufgrund der Komplexität des Gesamtmodells unmöglich. Daher ist ein kompositionaler Ansatz notwendig, um aus der Analyse der Teilmodelle auf ein Verhalten des Gesamtnetzes schließen zu können. Wir untersuchen in diesem Abschnitt, inwiefern typische, in vielen Teilmodellen auftretende Netzstrukturen (*Kontrollstrukturen*) einen kompositionalen Ansatz für eine Analyse des Gesamtmodells ermöglichen.

Unseren Ansatzpunkt bilden die zwei wesentlichen Ursachen der Komplexität des Gesamtmodells: der Umfang der zu modellierenden Daten und die aufwendige Konstruktion einer Rücksetz- bzw. Aufräumkomponente.

2.1 Aufräumkomponenten

Die meisten Netze der Fallstudie müssen eine Möglichkeit besitzen, von jedem Zustand wieder in ihren Anfangszustand zurückgesetzt zu werden. Dies bedeutet, daß eine Möglichkeit geschaffen werden muß, alle Marken des Netzes einzusammeln, d. h. es muß möglich sein, das Netz *aufzuräumen*. Diese Möglichkeit wird durch die Modellierung erweiterter Netzstrukturen (*Aufräumkomponenten*) erreicht. Bei allen Netzen der Fallstudie ist offensichtlich, daß die Aufräumkomponente nach einem bestimmten Prinzip modelliert wurde. Dieses Prinzip werden wir im folgenden an einem Beispiel erläutern.

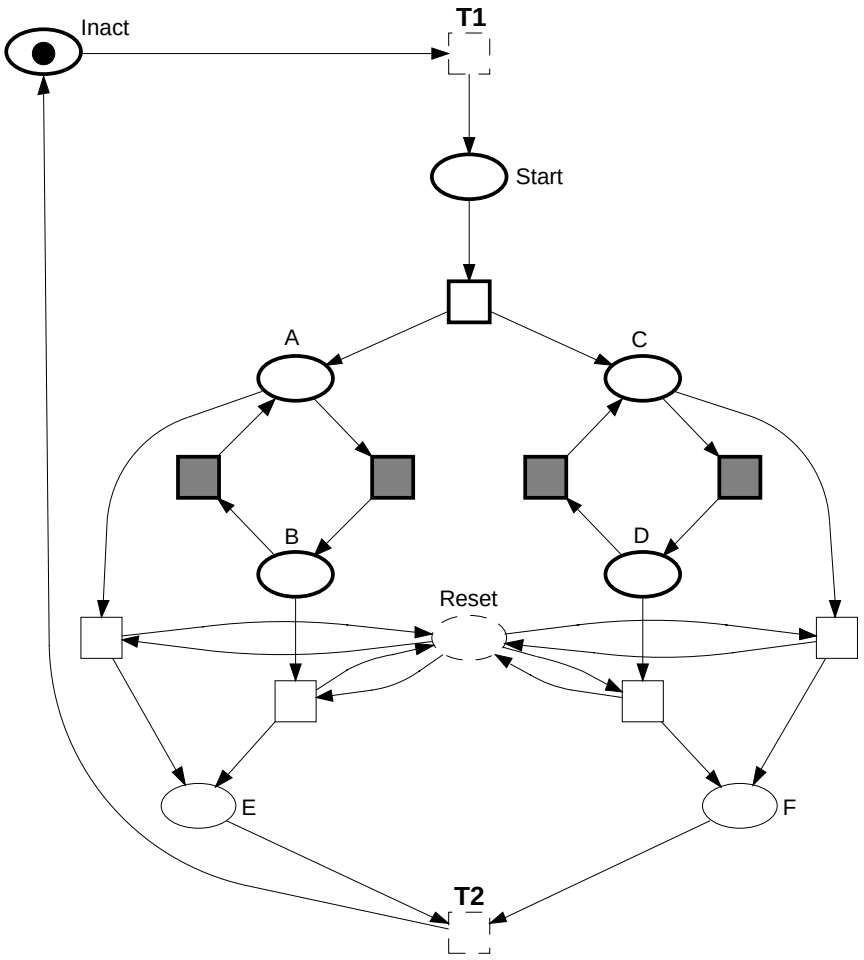

Abbildung 3: Ein Beipielmodell

Das in Abbildung 3 dargestellte Netz zeigt ein typisches Teilnetz der Fallstudie. Wir erläutern kurz die hier gewählte Darstellungsweise. Gestrichelte Stellen und Transitionen beschreiben eine Fusion mit Stellen beziehungsweise Transitionen eines anderen Teilnetzes. Betrachten wir vorerst nur die fett umrandeten Netzelemente. Sie beschreiben die eigentliche Funktionalität des Netzes: Nach dem Start werden zwei unabhängige Zyklen

gestartet. Eine graue Schattierung einer Transition bedeutet, daß an diese Transition eine externe Schaltnebenbedingung gebunden ist. Dadurch kann gegebenenfalls eine Transition nicht schalten, obwohl sie im dargestellten Modellausschnitt aktiviert ist. Weiterhin wird angenommen, daß sich in den Zyklen die Schaltnebenbedingungen wechselseitig ausschließen, so daß zwischen dem Schalten von zwei grauen Transitionen Zeit vergeht. Bei allen anderen aktivierten Transitionen ist durch den zugrundeliegenden Petrinetz-Interpreter (vgl. Abschnitt 8) gesichert, daß sie sofort schalten. Insgesamt ist in den Modellen wichtig, daß keine Zyklen von sofort schaltbaren Transitionen möglich sind.

Die Anforderungen an die Aufräumkomponente können nun wie folgt formuliert werden: Das dick umrandete Netz soll derart erweitert werden, daß aus jedem möglichen Zustand des Ursprungsnetzes bei Markierung der Stelle Reset durch eine Folge von sofort schaltenden Transitionen ein Zustand erreicht wird, in dem eine Marke auf der Stelle Inact liegt und alle anderen Stellen (außer Reset) unmarkiert sind. In der Abbildung ist eine Lösung für dieses konkrete Netz angegeben (alle Stellen und Transitionen, die mit einfacher Strichstärke dargestellt sind).

Aus theoretischer Sicht ist das oben geschilderte Problem der Konstruktion einer Aufräumkomponente für 1-sichere S/T-Systeme kein Problem: Zu jeder Stelle wird eine Komplement-Stelle konstruiert. Das neue Netz ermöglicht dann das Einsammeln aller Marken, da mithilfe der Komplement-Stellen für jede Stelle des Originalnetzes entschieden werden kann, ob sie markiert ist oder nicht. Diese theoretische Lösung hat aus praktischer Sicht mehrere Nachteile. Der wichtigste ist ihre Komplexität: Die Netzgröße wird mehr als verdoppelt. Dies schränkt die Darstellbarkeit, die Verständlichkeit und die Analysierbarkeit ein. Zudem ist durch das gesamte Netz nicht mehr notwendig 1-sicher. Das ist ein Nachteil, weil dies eine sehr erwünschte Analyseeigenschaft ist (vergl. Abschnitt 7).

In der Fallstudie wurde die Aufräumkomponente für jedes Netz individuell und adhoc eingeführt. Dadurch war dieser Teil der Modellierung fehleranfällig. Wir schlagen im folgenden eine Richtlinie vor, die in vielen Fällen zu den in der Fallstudie gewählten Aufräumkomponenten führt:[2]

1. Wir können jene Teilmenge der Stellen auszeichnen, die markiert sein können, ohne daß Transitionen in ihrem Nachbereich sofort schalten. Dies sind genau die Stellen im Vorbereich einer Transition mit einer externen Schaltnebenbedingung. In unserem Beispiel sind das die Stellen A, B, C und D. Insbesondere haben alle betrachteten Netze der Fallstudie eine solche Struktur, daß nach endlich vielen Schaltvorgängen nur noch auf solchen Stellen Marken liegen. Daher genügt es also, alle Marken von solchen Stellen abzuräumen.

2. Weiterhin sind insbesondere S-Invarianten interessant, die eine Teilmenge der unter 1. ermittelten Stellenmenge enthalten. In unserem Beispiel gilt die S-Invariante $\mathsf{Inact} + \mathsf{Start} + \mathsf{A} + \mathsf{B} = 1$. Daher sind Markierungen unerreichbar, die zugleich die Stellen A und B markieren. Nun können wir auf der Stelle E alle Marken eines Teils des Originalnetzes sammeln. Für die Stellen C und D funktioniert dieses Verfahren analog. Falls durch diese S-Invarianten alle relevanten Stellen abgedeckt sind, müs-

[2] Wir skizzieren hier nur die Idee informell, ihre Formalisierung übersteigt den Rahmen dieses Berichts.

sen wir nur noch eine Transition (hier T2) modellieren, die von allen Sammelstellen entsprechend viele Marken abzieht und eine Marke auf Inact legt.

3. Falls alle unter 2. betrachteten S-Invarianten bezüglich der relevanten Stellen nur Einträge mit dem Wert 1 haben, führt dieses Verfahren sogar zu einem 1-sicheren und lebendigen Netz.

Die derart systematisch konstruierten Aufräumkomponenten bilden eine typische Kontrollstruktur der Fallstudie.

2.2 Rücksetzkomponenten

Eine Hauptanforderung an die Fallstudie war nicht nur die Möglichkeit, Netze aufräumen zu können, sondern dies sollte auch noch entsprechend der Strukturierung geschehen (vgl. Abschnitt 3). Übergeordnete Netze entscheiden über die Auslösung eines Resets, teilen dies den untergeordneten Netzen mit und überwachen das Aufräumen. Um die Korrektheit solcher Konstruktionen überprüfen zu können, mußten sehr komplexe Teile des Modells zusammenhängend analysieret werden. Dies scheiterte jedoch häufig am Fassungsvermögen der Werkzeuge.

Eine Möglichkeit, die Verifikation hierbei zu unterstützen, bieten Beweise von Petrinetz-Modellen verteilter Algorithmen [Wal95, Rei98, WWV+97]. Viele dieser Algorithmen werden in einer Spezialform in der Praxis angewendet, manchmal ohne explizites Wissen des Modellierers. In der Fallstudie spiegelt die Modellierung der Rücksetzkomponente einen Spezialfall des Echo-Algorithmus wider [KRVW97]. Hierbei kann das gesamte Netzmodell als ein Agentennetzwerk aufgefaßt werden, in dem ein Agent einem Teilnetz und eine Kante zwischen zwei Agenten einer Über- bzw. Unterordnung gemäß der gewählten Strukturierung der Fallstudie entspricht. Die wesentliche Idee des Echo-Algorithmus, nämlich das Weiterreichen einer Nachricht im Netzwerk und die Rückmeldung des Erhalts dieser Nachricht, findet sich in Form von Reset-Nachrichten auf den Reset-Stellen der Teilnetze wieder.

Bislang fehlen noch anwendbare Techniken, die kompositionale Beweise aufgrund solcher Beobachtungen erlauben. Die Einbettung des Echo-Algorithmus in die Fallstudie sowie mögliche Schlußfolgerungen für die Korrektheit werden derzeit noch untersucht.

3 Strukturierung

Ein Petrinetz-Modell, das eine Größe wie die vorliegende Fallstudie mit insgesamt 2 068 Stellen und 2 941 Transitionen hat, muß sinnvollerweise strukturiert werden. So wurde das Petrinetz-Modell der Fallstudie in 175 Netze aufgeteilt [Ruc96].

Schon bei der Aufgabenanalyse ergaben sich Anforderungen an die Dekomposition des Gesamtsystems in Subsysteme. Es sollte trotz Dekompostion eine weitgehende Analyse des Modells und vor allem seine Ausführbarkeit möglich sein.

Das stark nebenläufige Pilotenverhalten wurde primär nach Tätigkeitsbereichen, wie dem Ausfahren der Landeklappen, strukturiert. Die dadurch entstandenen Teilmodel-

le konnten weitgehend unabhängig voneinander entwickelt, getestet und gegebenenfalls ausgetauscht werden.

Eine andere Strukturierung, wie sie in Flughandbüchern und Schulungsmaterial verwendet wird, gliedert Pilotenverhalten sequentiell nach Situationen. Dies umfaßt zwei Komponenten – *Flugphasen* (z. B. Reiseflug, Anflug, Landung) und *Flugplanvorgaben* (z. B. „proceed to station X"). Die einzelnen Situationsausprägungen sowohl innerhalb der Flugphasen als auch innnerhalb der Flugplanvorgaben schließen sich gegenseitig aus.

3.1 Teilmodelle

Das Pilotenverhalten wurde, wie oben gezeigt, in Situationsklassifikation (Exklusivität von Flugphase bzw. Flugplanvorgabe) und Handlungsablauf (starke Nebenläufigkeit) unterteilt. Die Petrinetz-Modelle der Handlungsabläufe beziehen sich meist auf einen bestimmten Situationskontext (z. B. die *Handlung* Landeklappenbedienung in der *Flugphase* Anflug). Die Teilmodelle von Handlungsabläufen werden über *Kopplungsnetze* aktiviert und deaktiviert, die Situationsmerkmale zu Handlungsabläufen zuordnen. [Mar97b]

Außerdem werden *Steuernetze* eingesetzt, die alle anderen Teilmodelle steuern. Sie veranlassen bspw. das Rücksetzen von Teilmodellen bei Änderung des Flugplans.

Folgende funktionalen Modellierungstypen wurden eingestzt:

- Netze zur Situationsklassifikation
- Netze für Handlungsabläufe
- Kopplungsnetze
- Steuernetze

3.2 Kopplung der Teilmodelle

Eine wichtige Anforderung an die Modellierung war die Vermeidung von Markenfluß zwischen den Teilmodellen. Dies ist sowohl für die modulare Simulation als auch für die Analyse von Teilmodellen eine sinnvolle Forderung. In der Fallstudie wurde diese Forderung mit Stellenfusion umgesetzt; ein Teilnetz kann dabei auf die Stellen anderer Teilnetze nur lesend zugreifen. In den Petrinetzmodellen wurde der lesende Zugriff auf diese „virtuellen" Stellen mit Schlingen dargestellt.

Synchrone Kopplung wurde mit Transitionsfusion modelliert. Aus Gründen der Übersichtlichkeit der hierarchischen Einbettung von Unternetzen wird eine der beteiligten Transitionen als „Haupttransition" der Fusion. [Mar97b]

3.3 Kopplung mit der Umgebung

Ein wesentliches Merkmal der Fallstudie ist die Einbettung des Petrinetz-Modells Referenzpilot in eine reale Umgebung. Diese Umgebung liefert einerseits Informationen über ihren Zustand; andererseits soll der Referenzpilot das erwartete Pilotenverhalten deutlich werden lassen.

Informationen aus der Modellumgebung werden mit Hilfe von Schaltnebenbedingungen an Transitionen gekoppelt. Ihr Schalten hängt also von der nicht modellierten Systemumgebung ab (vgl. die externen Transitionen in [Rei95, WWV+97]). Ein Beispiel für solch eine Schaltnebenbedingung ist eine bestimmte Flughöhe.

Durch die Einbindung des Referenzpiloten in CASSY wird die Ausführung „realer" Aktionen in der Modellumgebung nötig. Vor allem wird dem Überwachungsmodul von CASSY das erwartete Pilotenverhalten durch das Schalten bestimmter Transtitionen mitgeteilt.

Sowohl Schaltnebenbedingungen als auch „reale" Aktionen in der Modellumgebung werden durch C-Funktionen eingebunden.

3.4 Dekomposition

Ein wesentliches Element der Strukturierung des Referenzpiloten war die Dekomposition des Systems. Aus den Anforderungen an die Modellierung, wie der Analyse von Teilmodellen und der hierarchischen Gliederung des Gesamtsystems, ergaben sich drei wesentliche Grundkonstruktionen der Dekomposition von Petrinetzen [Mar97b].

Dekomposition ohne Nebenläufigkeit

Das Konzept der Dekomposition ohne Nebenläufigkeit orientiert sich am klassischen Prozedurkonzept sequentieller Programme. Eine solche Prozedur erledigt eine klar definierte Teilaufgabe und kommuniziert mit dem aufrufenden Programm über Eingabe- und Ausgabeparameter.

Für den Referenzpiloten wurde dieses Konzept zur Verfeinerung von Zuständen in Netzen zur Situationsklassifikation benutzt. Diese Netze sind zueinander nicht nebenläufig. Das Netz, welches die zu verfeindernde Stelle enthält wird *Obernetz* genannt; und das Netz, das die Stelle verfeinert, heißt *Unternetz*. Jede Transition im Vor- und Nachbereich der Stelle wird mit je einer „Start-" bzw. „Endtransition" eines Unternetzes fusioniert. Das Unternetz stellt die aufzurufende Prozedur dar. Über die Transitionen als Ein- bzw. Ausgabeparameter kann die Prozedur entsprechend verschiedenes Verhalten zeigen. Das Obernetz „wartet" auf die Beendigung des Unternetzes.

Abbildung 4 zeigt ein solches Zusammenspiel von Ober- und Unternetz. Eine der fusionierten Transitionen aufrufen1, ..., aufrufenX startet das Unternetz. Durch Schalten einer der Transitionen beenden1, ..., beendenX teilt das Unternetz dem Obernetz ein bestimmtes Ergebnis mit. Das Unternetz geht dabei in den Zustand Inact über.

Bei der Modellierung der Unternetze muß darauf geachtet werden, daß keine andere Stelle als Inact markiert ist, wenn das Netz inaktiv ist.

Dekomposition mit Rücksetzkonstruktion

Im Abschnitt 3.1 haben wir vier verschiedene Modellierungstypen eingeführt. Netze der verschiedenen Typen können hierarchisch aufeinander aufbauen. Da Netze verschiedener

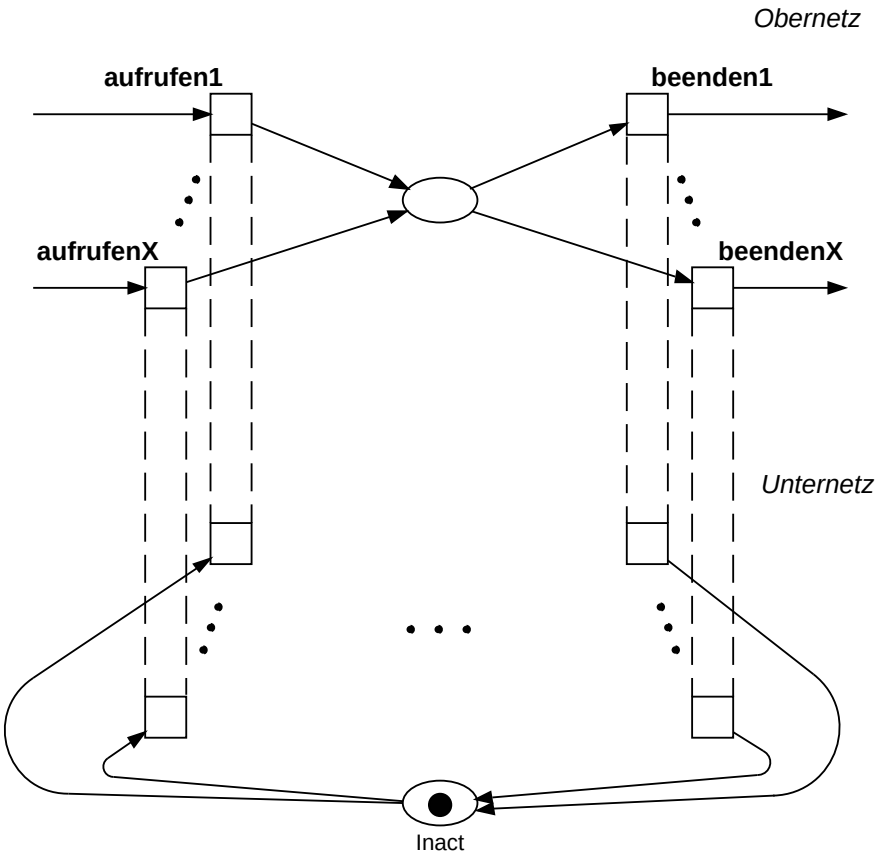

Abbildung 4: Dekomposition ohne Nebenläufigkeit

Typen jedoch zueinander nebenläufig sind, reicht das im vorangegangen Abschnitt vorgestellte Konzept der Dekomposition nicht aus. Das Obernetz muß Unternetze abbrechen können.

Ein Beispiel für die Notwendigkeit eines Abbruchkonzeptes ist eine Flugplanänderung, die den Piloten auffordert, den Landeanflug abzubrechen und den Reiseflug fortzusetzen. Die Landeklappen dürfen nicht weiter ausgefahren und müssen eingefahren werden.

Um einen solchen (nebenläufigen) Abbruch des Unternetzes zu erreichen, wird zusätzlich zu dem schon bekannten Dekompositionskonzept eine *Reset*-Stelle in das Obernetz eingefügt, die das Unternetz per Stellenfusion importiert. Entsprechend Abschnitt 3.2 kann im Unternetz nur lesend auf diese Stelle zugegriffen werden.

Das Unternetz schaltet bei einer Markierung der Reset-Stelle solche Transitionen, die erreichbare Zustände abfangen, und teilt dem Obernetz durch Schalten einer besonderen „Rückgabetransition“ den Erfolg des Rücksetzens mit.

Ein Beispiel eines Unternetzes mit Rücksetzkonstruktion war bereits in Abschnitt 2 zu sehen.

Unternetzbaustein

Auf der untersten Ebene der Modellhierarchie (in Netzen für Handlungsabläufe bzw. zur Situationsklassifikation) kommt es in unterschiedlichen Situationen häufig zu elementaren Verhaltensmustern. Beispielsweise ist die Änderung des Steuerkurses sowohl auf Anforderung der Luftsicherung als auch beim Fliegen einer Warteschleife nötig.

Durch den unterschiedlichen Kontext ist Nebenläufigkeit solcher Verhaltensmuster ausgeschlossen. Es lag daher nahe, aufrufbare Netzbausteine zu verwenden, die von mehreren Obernetzen aufgerufen werden können. Der Kontext schließt eine überlappende Aktivierung aus.

Im Gegensatz zu den oben dargestellten Dekompositionskonzepten verwendeten die Autoren des Referenzpiloten ein anderes Verfahren für den Aufruf von Netzbausteinen. Statt Transitionsfusion als Start eines Unternetzbausteines wird je eine Stelle aus den Obernetzen mit je einer Stelle im Unternetzbaustein fusioniert. Diese sogenannte signalgebenden Stellen aktivieren die eigentlichen Starttransitionen im Unternetzbaustein. Das Konzept der signalgebenden (im Unternetzbaustein nur lesbarer) Stelle wird außerdem verwendet, um dem Unternetz weitere Nachrichten (z. B. eine Resetaufforderung) aus dem Obernetz zukommen zu lassen. Der Unternetzbaustein wird ebenfalls mit einer signalgebenden Stelle beendet – diesmal als Export der Stelle in das Obernetz.

In Abbildung 5 ist ein solcher Unternetzbaustein zu sehen. Die signalgebenden Stellen Signal1 und Signal2 lösen jeweils mittelbar (über die Stelle Start) die Abarbeitung des Unternetzes aus. Schließlich teilt die signalgebende Stelle Ende dem entsprechenden Obernetz mit, daß das Unternetz seine Abarbeitung beendet hat.

Auch bei dieser Konstruktion muß bei der Entwicklung darauf geachtet werden, daß mit einer Markierung von Ende keine andere Stelle des Unternetzes markiert ist.

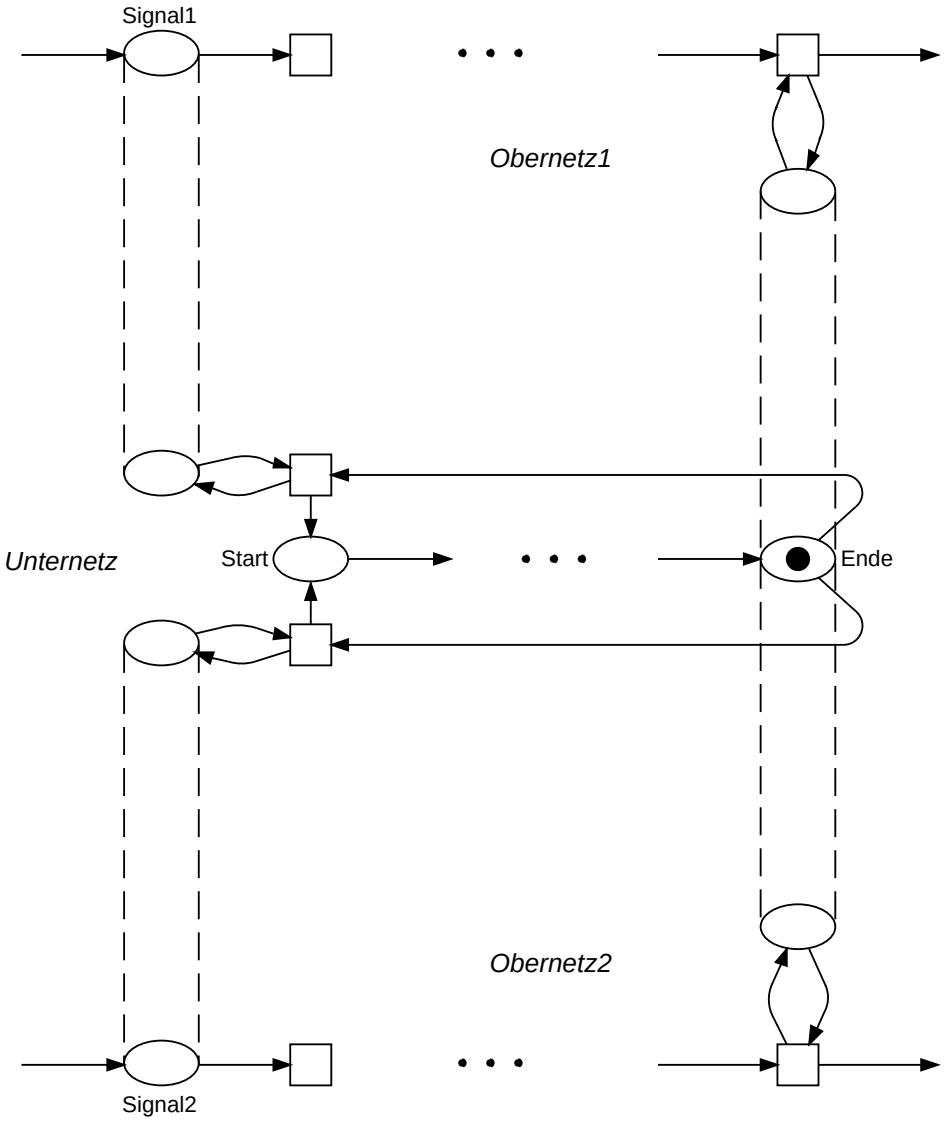

Abbildung 5: Unternetzbaustein

4 Datenstrukturen

In der Fallstudie werden Daten als Wahrheitswerte bzw. seltener als geringfügige Anzahl von Marken aufgefaßt. Da die Modellierungsaufgabe u. a. darin bestand, regelbasiertes Pilotenverhalten mit Hilfe von Petrinetzen auszudrücken, lag es nahe, Bedingungs-/Ereignis-Netze zu verwenden. Die Produktionsregeln, die der Aufgabenstellung zugrunde lagen, können leicht mit Petrinetzen reformuliert werden (s. a. 1.1) [Web97].

Selten war es notwendig, mehrere gleichartige Ressourcen wie Navigationsinstrumente zu modellieren. Zur Sicherung eines konsistenten Beschreibungsmittels wurden deshalb (überwiegend 1-sichere) Stellen-/Transitionen-Netze verwendet.

5 Zeitaspekte

Eine wichtige Anforderung an das Modell wurde durch das Projektumfeld (also CASSY) gestellt. Ohne Reimplementierung in einer konventionellen Programmiersprache sollte das Modell in Echtzeit ausführbar sein [Ruc96].

Der verwendete Interpreter schaltet das Petrinetz-Modell des Referenzpiloten schnell genug, um das Verhalten von Besatzung und Referenzpilot miteinander vergleichen zu können. In der verwendeten Netzklasse wird also vom zeitlosen Schalten der Transitionen ausgegangen.

6 Simulation

Die Simulation der Netze der Fallstudie war eine wichtige Teilaufgabe bei der Entwicklung der Modelle. Durch Simulation wichtiger Flugsituationen konnten frühzeitig die meisten Modellierungsfehler erkannt werden. Dazu wurden Flugszenarien entwickelt, an denen der Referenzpilot erprobt wurde.

Wesentliches Hilfsmittel bei der Simulation und vor allem beim Einsatz der Netze war ein echtzeitfähiger Interpreter in Verbindung mit einem Monitor (siehe Abschnitt 8).

7 Analyse und Verifikation

Der für die Flugsicherheit relevante Kontext des Referenzpiloten stellte hohe Anforderungen an die Korrektheit des Gesamtmodells. Auch mit umfangreichen Tests in Simulationsläufen kann jedoch nicht die Abwesenheit schwerwiegender Modellierungsfehler sicher festgestellt werden. Aus diesem Grunde wurden neben der Simulation auch Verfahren der Analyse und (in Ansätzen) der Verifikation verwendet.

Die größte Schwierigkeit bei der Analyse des Modells ergab sich aus seiner Größe. Insbesondere die (rechnergestützten) Analyseverfahren, die auf der Auswertung des Erreichbarkeitsgraphen beruhen, werden ab einer bestimmten Größe des zu analysierenden Netzes ineffektiv und nicht mehr verwendbar.

Aufgrund der Größe der Netzmodelle wurden zunächst die Teilnetze einzeln untersucht, später auch einige gekoppelte Netze. Folgende Anforderungen wurden dabei an die Eigenschaften der Teilnetze gestellt:

- stark zusammenhängend
- beschränkt (endliche Erreichbarkeitsmenge)
- reversibel (Erreichbarkeit eines Grundzustandes)
- (meistens) 1-sicher
- lebendig (alle Transitionen sind erneut aktivierbar)

Für die Entwickler überraschend erfüllten nach einer ersten Analyse des gößten Teils aller Netze 5% der analysierten Netze *nicht* alle geforderten Eigenschaften. Und dies obwohl bereits ein Jahr lang fehlerfreie Simulationsläufe des Modells durchgeführt worden waren.

Viele Modellierungsfehler entstanden erst bei der Kopplung von Teilmodellen. Deshalb sollte idealerweise im *bottom-up*-Verfahren zunächst die einzelnen Teilmodelle dann Gruppen von Teilmodellen und schließlich das Gesamtmodell analysiert werden. Die Größe des vorliegenden gesamten Modells und die Nebenläufigkeit seiner Teilsysteme machte es unmöglich das Gesamtmodell vollständig zu analysieren, da der Erreichbarkeitsgraph nicht mehr berechnet werden konnte. Die Teilsysteme, die in sich nur schwach nebenläufig sind, konnten jedoch auf die beschriebenen Eigenschaften hin analysiert werden. Auf diese Weise konnten für das Gesamtzystem wenigstens einige (durch Simulation nur schwer erkennbare) Modellierungsfehler ausgeschlossen werden. Die Korrektheit des Gesamtsystems konnte damit jedoch nicht nachgewiesen werden.

Der Erreichbarkeitsgraph der Modelle und die gesuchten Eigenschaften wurden mit dem Werkzeug INA [RS97] berechnet. Teilsysteme, deren Erreichbarkeitsgraph nicht mehr berechnet werden konnte, wurden mit einem Model-Checker [Esp93] aus dem Werkzeug PEP [GB96] überprüft. Dadurch wurde die Verklemmungsfreiheit des entsprechenden Teilsystems nachgewiesen.

Die Entwickler des Referenzpiloten bemängeln, daß „vielversprechende Ansätze“ wie temporale Logik für Petrinetz-Modelle [Rei89, Rei95, Rei98] noch nicht von leistungsfähigen Werkzeugen unterstützt werden. Die Bedeutung formaler Methoden der Verifikation für sicherheitsrelevante verteilte Systeme unterstreicht eindrücklich die Forderung nach Werkzeugunterstützung.

8 Werkzeugunterstützung

Die Netze des Modells *Referenzpilot* wurden mit dem graphischen Editor *Design/OA* Version 3.0 erstellt. Der Editor hat eine Schnittstelle zu einer ASCII-basierten Netzbeschreibungssprache. Für die Umformung dieser Beschreibungssprache in eine effiziente interne Repräsentation wurde ein Parser entwickelt.

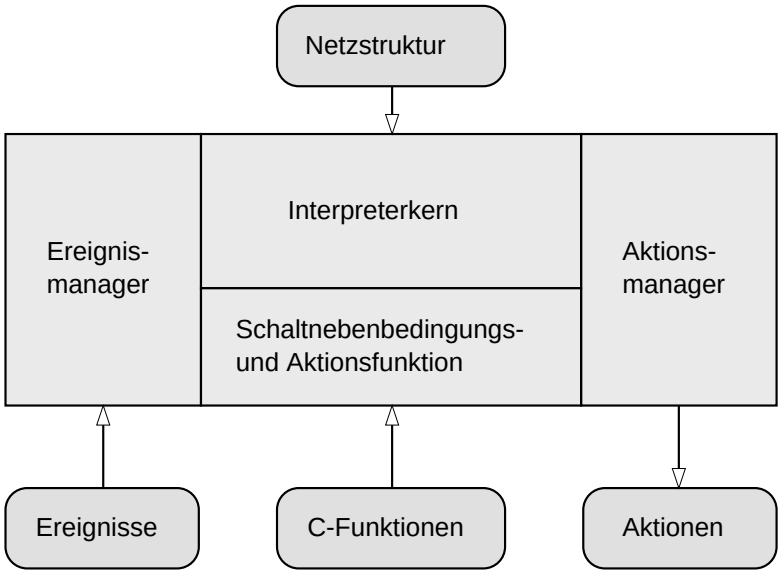

Abbildung 6: Struktur des Interpreters (nach [Ruc96])

Die interne Repräsentation der Netze wird schließlich von dem echtzeitfähigen Interpreter benutzt. Die Struktur des Interpreters ist in Abbildung 6 zu sehen. Der Interpreterkern sorgt für das Aktivieren, Deaktivieren und Schalten von Transitionen entsprechend der Schaltregel.

Das Modul *Ereignismanager* bindet externe Ereignisse an. Durch die Aktivierung von Transitionen des Netzes, die zum Schalten des Netzes außerdem noch externe Schaltnebenbedingungen benötigen, wird dem Ereignismanager durch den Interpreterkern mitgeteilt, welche Ereignisse erwartet werden. Der Ereignismanager teilt dem Interpreterkern dann das Eintreten von Ereignissen mit und löst damit mittelbar das Schalten aktivierter Transitionen aus.

Entsprechend löst das Modul *Aktionsmanager* externe Aktionen aus. Dafür wird es vom Interpreterkern aufgerufen. Der Aktionsmanager wird vor allem verwendet, um dem Überwachungsmodul von CASSY das erwartete Verhalten der Besatzung mitzuteilen.

Um die Aktionen des Interpreters darzustellen bzw. um externe Ereignisse der Petrinetz-Modelle zu simulieren, wurde ein Monitor entwickelt, der Abläufe (das Markenspiel) der Netze visualisiert.

Zur Analyse der Teilmodelle wurden die Werkzeuge INA [RS97] und PEP [GB96] sowie eine Eigenentwicklung verwendet. Für die Benutzung von INA und PEP wurden Konvertierungswerkzeuge verwendet.

Und schließlich wurden zwei Hilfswerkzeuge entwickelt. Ein Struktur-Monitor stellt die Netzhierarchie dar und verdeutlicht Transitions- und Stellenimporte bzw. -exporte. Ein Netz-Koppler löst die Fusionen zwischen mehreren Teilnetzen auf und erzeugt ein neues, alle fusionierten Teilnetze enthaltendes Netz.

Zusammenfassung

In diesem Bericht haben wir Erfahrungen zusammengefaßt, die wir im Rahmen der DFG-Forschergruppe PETRINETZ-TECHNOLOGIE bei der Validierung der Fallstudie CASSY gesammelt haben.

Unsere Erfahrungen lassen sich in drei Gruppen unterteilen:

1. theoretische Konzepte, die in der Praxis erfolgreich verwendet werden Hier möchten wir exemplarisch nennen, wie erfolgreich die Analyse auf Rücksetzbarkeit und Lebendigkeit sich auf das Fehlerverhalten ausgewirkt hat. 5% der Netze wiesen Fehler auf, die bei der Simulation noch nicht erkannt wurden!

2. praktische Lösungen, die zu neuen theoretischen Resultaten führen können Dies betrifft beispielsweise die Konstruktion der Rücksetz- und Aufräumkomponenten, von denen wir uns weiterführende theoretische Arbeiten versprechen.

3. Ideen für eine Neukonzeption der Fallstudie Bei der Validierung der Fallstudie entstanden natürlich auch viele Ideen dazu, welche Techniken man neu entwickeln beziehungsweise einsetzen könnte um die Erstellung ähnlicher Fallstudien zu vereinfachen. Eine dieser Ideen ist beispielsweise die Entwicklung von Techniken zur kompositionalen Verifikation. Die Erkenntnis, daß der ECHO-Algorithmus „irgendwie in das operationelle Modell eingebettet“ ist, ermöglicht diese Einbettung natürlich noch nicht. Hier sind weiterführende theoretische Arbeiten erforderlich. Weitere Reduzierung des Modellierungs- und Verifikationsaufwandes erhoffen wir uns vom Einsatz von high-level Netzen sowie von der Ausnutzung von Symmetrien bei der Erreichbarkeitsanalyse. Diese Techniken haben wir jedoch noch nicht abschließend anhand der Fallstudie validieren können.

Wir hoffen, daß sich diese Erfahrungen, gemeinsam mit den Erfahrungen, die sich im Rahmen der Forschergruppe aus anderen Fallstudien ergeben, in eine zu entwickelnde anwendungsorientierte Petrinetz-Technologie einbetten, mit der zukünftige Anwendungen effektiv unterstützt werden.

Literaturverzeichnis

[Esp93] Javier Esparza. *A Partial Order Approach to Model Checking.* Habilitation, Universität Hildesheim, 1993.

[GB96] Bernd Grahlmann und Eike Best. PEP — more than a Petri net tool. In T. Margaria und B. Steffen, Herausgeber, *Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, Band 1055 aus *LNCS*, Seiten 397–401. Springer-Verlag, Marz 1996.

[KRVW97] Ekkart Kindler, Wolfgang Reisig, Hagen Völzer und Rolf Walter. Petri net based verification of distributed algorithms: An example. *Formal Aspects of Computing*, 9:409–424, 1997.

[Mar97a] Axel Martens. Einführung in den Aufbau der Fallstudie Pilot. Internes Arbeitspapier, Humboldt-Universität zu Berlin, Institut für Informatik, 1997.

[Mar97b] Axel Martens. Strukturierungsaspekte in der Fallstudie Pilot. Internes Arbeitspapier, Humboldt-Universität zu Berlin, Institut für Informatik, 1997.

[Mar97c] Axel Martens. Vorgehensmodell in der Fallstudie Pilot. Internes Arbeitspapier, Humboldt-Universität zu Berlin, Institut für Informatik, 1997.

[Rei89] Wolfgang Reisig. Towards a temporal logic for causality and choice in distributed systems. In J.W. de Bakker, W.-P. de Roever und G. Rozenberg, Herausgeber, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Band 354 aus *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1989.

[Rei95] Wolfgang Reisig. Petri net models of distributed algorithms. In Jan van Leeuven, Herausgeber, *Computer Science Today. Recent Trends and Developments*, Band 1000 aus *Lecture Notes in Computer Science*, Seiten 441–454. Springer-Verlag, Berlin, 1995.

[Rei98] Wolfgang Reisig. *Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets.* Springer Verlag, 1998. In Vorbereitung.

[RO94] Wilhelm Ruckdeschel und R. Onken. Modelling of pilot behaviour using petri nets. In R. Valette, Herausgeber, *Application and Theory of Petri Nets 1994*, Band 815 aus *LNCS*, Seiten 436–453. Springer-Verlag, Juni 1994.

[RO95] Wilhelm Ruckdeschel und R. Onken. Analyse eines Petrinetz-basierten Pilotenmodells. In E. Schnieder, Herausgeber, *Entwurf komplexer Automatisierungssysteme*, Institut für Regelungs- und Automatisierungstechnik, Seiten 315–346. TU Brauschweig, Juni 1995.

[RS97] Stephan Roch und Peter Starke. *INA. Integrierter Netz Analysator Version 1.7. Handbuch*, April 1997. http://www.informatik.hu-berlin.de/lehrstuehle/automaten/ina/.

[Ruc96] Wilhelm Ruckdeschel. *Modellierung regelbasierten Pilotenverhaltens mit Petrinetzen.* Dissertation, Universität der Bundeswehr München, 1996.

[Wal95] Rolf Walter. *Petrinetzmodelle verteilter Algorithmen. Beweistechnik und Intuition*, Band 2 aus *Edition VERSAL*. Bertz Verlag, 1995. Dissertation, Humboldt-Universität zu Berlin.

[Web97] Michael Weber. CASSY: Einige Gründe zur Verwendung von Petrinetzen. In Hartmut Ehrig, Wolfgang Reisig und Herbert Weber, Herausgeber, *Move-On-Workshop der DFG-Forschergruppe Petrinetz-Technologie*, Nummer 97-21 in Forschungsberichte des Fachbereiches Informatik, Seiten 167–173, Technische Universität Berlin, 1997.

[WER95] Herbert Weber, Hartmut Ehrig und Wolfgang Reisig. Konzeption, theoretische Fundierung und Validierung einer anwendungsbezogenen Petrinetz-Technologie. Antrag an die DFG auf Förderung einer Forschergruppe, September 1995.

[WWV+97] Michael Weber, Rolf Walter, Hagen Völzer, Tobias Vesper, Wolfgang Reisig, Sibylle Peuker, Ekkart Kindler, Jörn Freiheit und Jörg Desel. DAWN. Petrinetzmodelle zur Verifikation Verteilter Algorithmen. Informatik-Bericht 88, Humboldt-Universität zu Berlin, 1997.