

Leistungsbewertung mit PEPSY-QNS und MOSES

Gunter Bolch, Matthias Roessler und Raimund Zimmer
Universität Erlangen-Nürnberg; IMMD IV; Martensstraße 1; D-91058 Erlangen
{bolch,roessler}@immd4.informatik.uni-erlangen.de

Hermann Jung
Siemens AG Erlangen
Abt.: ANL A433-SI
Werner-von-Siemens-Strasse 60
D-91052 Erlangen

Stefan Greiner
Duke University
Dept. of Electrical Engineering
Durham, NC 27708-0291, USA
Box 90291

Zusammenfassung: In diesem Beitrag werden mit PEPSY-QNS und MOSES zwei Werkzeuge zur Leistungsbewertung vorgestellt, die sich in ihrem Anwendungsbereich sehr gut ergänzen.

Dynamische Systeme, die sich durch Warteschlangennetze modellieren lassen wie z.B. Rechensysteme oder Fertigungssysteme, können mithilfe von PEPSY-QNS (Performance Evaluation and Prediction SYstem for Queueing NetworkS) analysiert werden. PEPSY-QNS stellt zur Eingabe des Warteschlangennetzes die graphische Oberfläche XPEPSY zur Verfügung und zur Berechnung der Leistungsgrößen eine große Zahl exakter und approximativer analytischer Verfahren und die Simulation.

Das Verhalten vieler Systeme läßt sich aber durch Warteschlangennetze nicht oder nur sehr ungenau modellieren. Z.B. wenn man bei Rechensystemen das Ausfallverhalten, Synchronisation oder Blockierungen berücksichtigen will. In solchen Fällen kann das Markov-Analyse-Tool MOSES (MOdelling Specification and Evaluation System) eingesetzt werden. Die Beschreibung des Systems erfolgt dabei in der neu entwickelten Systembeschreibungssprache MOSLANG. MOSES erzeugt aus dieser Spezifikation das zugehörige Markov'sche Gleichungssystem, das die Berechnung der Zustandswahrscheinlichkeiten und damit auch der gesuchten Leistungsgrößen ermöglicht.

Abstract: In this article the two tools PEPSY-QNS and MOSES, whose areas of application complement one another, are introduced.

Dynamic systems, which can be described by queueing networks (like computer or manufacturing systems) can be analyzed with PEPSY-QNS (Performance Evaluation and Prediction System for Queueing Networks). To describe the system in PEPSY-QNS, the graphical user interface XPEPSY is used. To compute the performance measures of the network, PEPSY-QNS provides many exact and approximate analytical methods as well as simulation.

In many cases modeling a system using queueing networks is not accurate enough to describe the system behaviour or it is even impossible to model the system (consider e.g. computer systems with server failure, synchronization or blocking). In this cases the Markov analysis tool MOSES (Modelling Specification and Evaluation System) is used. The system is described with the newly developed specification language MOSLANG. From this specification, MOSES generates the underlying system of equations, computes the steady state probabilities and all performance measures.

Schlüsselwörter: Systembeschreibungssprache, Systemspezifikation, Markov'sches Gleichungssystem, Lösungsmethoden, Zustandswahrscheinlichkeiten, Leistungsbewertung, Warteschlangen, Netzwerke

1. Einleitung

Leistungsbewertung ist bei heutigen Rechen-, Kommunikations- und Fertigungssystemen aufgrund der ständig steigenden Komplexität von besonderer Bedeutung und sollte Hand in Hand mit dem Entwurf des realen Systems gehen. Da Messungen während der Entwurfs- und Entwicklungsphase häufig nicht möglich sind müssen Modellbildungstechniken angewendet werden.

Warteschlangennetze sind sehr gut geeignet zur Modellierung und Leistungsbewertung, da sie sehr leicht verständlich sind und das System in einer kompakten Weise beschreiben. Für eine beschränkte Klasse von Warteschlangennetzen, den Produktformnetzen, existieren effiziente Analysealgorithmen (z.B. Mittelwertanalyse, SCAT- und Faltungsalgorithmus) um die Leistungsgrößen des Systems zu bestimmen. Viele Warteschlangennetze, sog. Nichtproduktformnetze, erfüllen jedoch die Voraussetzungen für Produktformnetze nicht. Sie können häufig mittels approximativer Algorithmen analysiert werden. Sind diese Algorithmen nicht genau genug oder auf das spezielle Problem nicht anwendbar - dies kann z.B. der Fall sein, wenn das Netz ein Blockiernetz ist oder wenn die Bedienzeiten nicht exponentiell verteilt sind - so verwendet man üblicherweise diskrete ereignisorien-

tierte Simulationen mit den bekannten Nachteilen (Rechenzeitbedarf, Unsicherheit bezüglich der Genauigkeit der Ergebnisse).

PEPSY-QNS (Performance Evaluation and Prediction **S**ystem for **Q**ueueing **N**etwork**S**) ist ein Leistungsbewertungstool, das auf der Basis solcher Warteschlangenmodelle arbeitet, wobei die Auswertung durch die obengenannten exakten oder approximativen Analysetechniken bzw. Simulation erfolgt.

Viele Systeme lassen sich nur durch Nichtproduktformnetze modellieren, für die keine analytischen Verfahren existieren, oder sind durch Warteschlangennetze überhaupt nicht ausreichend genau zu modellieren. Dies kann z.B. der Fall sein wenn man bei Rechensystemen die Synchronisation, das Ausfallverhalten oder Blockierungen berücksichtigen will. Dann kann man neben der Simulation, mit den oben erwähnten Nachteilen, die Markovanalyse anwenden. Zur Markovanalyse benötigt man die möglichen Zustände des Systems und die Übergangsraten zwischen diesen Zuständen und kann dann daraus mittels des Markov'schen Gleichungssystems die Zustandswahrscheinlichkeiten und die Leistungsgrößen berechnen.

Obwohl der Zustandsraum bereits für einfache Systeme sehr groß sein kann und exponentiell in der Systemkomplexität wächst, wird die Markovanalyse aufgrund steigender Rechnerleistungen und verbesserter mathematischer Verfahren immer wichtiger. Dies ermöglicht es, sowohl steady-state als auch transiente Leistungsgrößen in akzeptablen Rechenzeiten zu erhalten. Das Problem dieser Vorgehensweise ist jedoch, daß es nahezu unmöglich ist, das Markovmodell für komplexe Systeme von Hand zu konstruieren.

Eine Möglichkeit zur Lösung dieses Problems ist die Modellierung des Systems mittels stochastischer Petrinetze. Diese besitzen eine größere Mächtigkeit als Warteschlangennetze, da man damit beispielsweise Synchronisationsvorgänge oder das Ausfallverhalten beschreiben kann, was mittels Warteschlangennetzen nicht möglich ist.

In diesem Beitrag wird ein anderer Ansatz zur Spezifikation solcher Systeme gewählt. Mittels der neuen Beschreibungssprache **MOSLANG** werden der Zustandsraum und die Zustandsübergänge der zugrundeliegenden Markovkette beschrieben. Das Markovanalysetool **MOSES** (**MO**delling **S**pecification and **E**valuation **S**ystem) erzeugt daraus automatisch das zugehörige Markov'sche Gleichungssystem und löst dies.

2. Das Markov-Analysetool MOSES

Das Markov-Analysetool **MOSES** erhält als Eingabe die Systemspezifikation in der Systembeschreibungssprache **MOSLANG** und erzeugt das zugrundeliegende Markov'sche Gleichungssystem $pQ = 0$ automatisch.

Hierbei sind p der Vektor der Zustandswahrscheinlichkeiten und Q die Matrix der Übergangsraten zwischen den verschiedenen Zuständen der Markovkette. Q wird in parametrisierter Form erzeugt und das System kann daher mit verschiedenen Parametern analysiert werden, ohne das Markovsystem jedesmal neu erzeugen zu müssen. Die Lösung dieses Systems liefert die Zustandswahrscheinlichkeiten. Hierfür werden von **MOSES** fünf verschiedene Analyseverfahren angeboten. Aus den Zustandswahrscheinlichkeiten werden dann alle stationären Leistungsgrößen, die vorher ebenfalls mit **MOSLANG** spezifiziert wurden, automatisch berechnet.

Mittels der neuen Beschreibungssprache **MOSLANG** werden der Zustandsraum und die Zustandsübergänge der zugrundeliegenden Markovkette beschrieben. Der Kern von **MOSLANG** besteht hierbei aus Konstrukten, die die Spezifikation der möglichen Systemzustände sowie der erlaubten Zustandsübergänge ermöglichen. Diese Methode erlaubt eine sehr kompakte Systemspezifikation. Nach etwas Einarbeitung können auch umfangreiche Systeme in kurzer Zeit beschrieben werden.

Bei Verwendung von **MOSES** ist durch die verschiedenen Beschreibungskonstrukte eine sehr flexible Modellierung unterschiedlichster Systeme möglich (Warteschlangennetze, Petrinetze, Blockiernetze, Prioritätsnetze, fehlertolerante Systeme, Betriebssysteme, Fertigungssysteme, Kommunikationssysteme u.a.m.).

2.1. Die Modellbeschreibungssprache MOSLANG

Die Modellbeschreibungssprache **MOSLANG** besteht aus einer Reihe von Konstrukten, die in vier Abschnitte, dem *declaration part*, dem *vector description part*, dem *rules part* und dem *results part* aufgeteilt werden können. Die Semantik der vier Parts wird im Folgenden näher erläutert.

Declaration part: Dieser Abschnitt enthält sämtliche Deklarationen des zu spezifizierenden Systems. Dabei können symbolische Namen für jede Komponente des Zustandsvektors definiert werden. Konstanten werden wie in C definiert. Hier müssen auch sämtliche Leistungsgrößen und Parameter spezifiziert werden. Dabei werden Typ und Name der Leistungsgrößen angegeben.

Vector description part: In diesem Abschnitt werden die Dimension und der Wertebereich des Zustandsvektors festgelegt. Desweiteren werden in diesem Abschnitt die verbotenen Zustände spezifiziert.

Rules part: Im rules part werden die Zustandsübergänge durch eine Reihe von RULE Konstrukten spezifiziert. Die RULE Konstrukte bestehen aus einem condition- und einem action part. Sobald alle Bedingungen erfüllt sind, wird die spezifizierte Aktion ausgeführt. Der action part enthält auch die Transitionsrate und die Übergangswahrscheinlichkeit. Ein RULE Konstrukt spezifiziert mehrere Zustandsübergänge. Es ist daher nicht nötig, für jeden Zustandsübergang ein RULE Konstrukt zu spezifizieren. Die sog. immediate transitions werden unmittelbar, d.h. zeitlos, ausgeführt sobald die spezifizierten Bedingungen erfüllt sind. Damit ist z.B. die Modellierung von Verdrängung und Prioritäten sehr einfach. Immediate transitions werden durch die Transitionsrate -1.0 charakterisiert. Zeitlose Übergänge haben Priorität vor Übergängen mit positiver Übergangsrate. Durch die Verwendung von immediate transitions erreicht man eine Entkopplung der verschiedenen Knoten des Systems. Dies bewirkt eine ganz erhebliche Reduzierung des Aufwands für die Modellbeschreibung.

Results part: In diesem Abschnitt werden die zu berechnenden Leistungsgrößen spezifiziert. Zu diesem Zweck stehen im results part die Konstrukte RESULT, RESULT MEAN, RESULT DIST und FINAL zur Verfügung. Mittels des RESULT Konstrukts werden die grundlegenden Leistungsgrößen (diese können aus den Zustandswahrscheinlichkeiten berechnet werden) spezifiziert. Durch das RESULT MEAN- und das RESULT-DIST Konstrukt wird die mittlere Anzahl von Aufträgen bzw. die Verteilung der Anzahl von Aufträgen in einer Station festgelegt. Mittels des FINAL Konstrukts werden Leistungsgrößen spezifiziert, die Funktionen der einfachen Leistungsgrößen und der Systemparameter sind, z.B. die mittlere Warteschlangenlänge, die Auslastung und der Durchsatz.

2.2. Analyse eines fehlertoleranten Multiprozessor Systems mit MOSES

In diesem Kapitel soll die Benutzung von **MOSES** am Beispiel eines symmetrischen fehlertoleranten Multiprozessorsystems mit endlicher Pufferkapazität erläutert werden. Desweiteren wird die Semantik der einzelnen Konstrukte anhand der Spezifikation erklärt.

2.2.1. Systemannahmen

- Das System besitzt eine endliche Ankunfts warteschlange der Länge L .
- Die Ankunftsrate an der Warteschlange ist λ
- Die Bedienrate eines Prozessors ist μ
- Die Zeit zwischen den Serverausfällen ist negativ exponentiell verteilt mit der Rate $1/mtbf$.
- Die Zeit, um einen Server zu reparieren ist negativ exponentiell verteilt mit der Rate $1/mttr$.
- Wenn ein Server ausfällt und dieser gerade einen Job bedient, so wird der Job auf einem freien Server erneut gestartet. Ist kein Server frei, so wird gewartet bis der Server repariert ist oder bis ein Server frei wird.

2.2.2. Beschreibung der einzelnen MOSLANG Konstrukte

Define Block

Hier kann der Anwender symbolische Namen für jede Komponente des Zustandsraumes festlegen. Dabei sind alle gültigen C-Bezeichner als Identifier erlaubt. Eine Komponente des Zustandsraums hat hierbei grundsätzlich den Wertebereich Integer. Desweiteren werden im define Block Konstanten definiert (falls erforderlich).

VAR Block

Für jede zu deklarierende Variable muß ein VAR Konstrukt spezifiziert werden. Dabei muß der Anwender Name und Typ der Variablen angeben. Die spezifizierten Variablen stellen die Modellparameter des Systems dar und müssen vom Anwender zu Beginn der Systemanalyse interaktiv eingegeben werden.

OUTPUT Block

Für jede zu berechnende Leistungsgröße muß ein OUTPUT Konstrukt angegeben werden.

DIM Block

Hier wird die Dimension des Zustandsvektors festgelegt.

MIN/MAX Konstrukt

Mit diesen beiden Konstrukten werden die minimalen und maximalen Werte jeder Komponente des Zustandsraumes festgelegt.

START Konstrukt

Durch dieses Konstrukt wird der Startzustand für die Zustandsraumgenerierung festgelegt.

NOT Konstrukt

Mittels dieses Konstrukts werden die verbotenen Zustände der Markovkette (Zustände die in der Markovkette nicht auftreten dürfen) spezifiziert. Der Anwender kann so viele NOT Konstrukte wie nötig spezifizieren. Ein NOT Konstrukt besteht aus einem oder mehreren Prädikaten und das gesamte Prädikat ist erfüllt, wenn **alle** separaten Prädikate des NOT Konstrukts erfüllt sind.

RULE Konstrukt

In diesem Abschnitt werden die Regeln spezifiziert, die das Verhalten des Systems festlegen. Ein Rule Konstrukt besteht dabei aus einem condition part und einem action part. Der condition part besteht aus einem oder mehreren Prädikaten. Der action part beschreibt die Zustandsübergänge im System. Er besteht aus einer oder mehreren Aktionen, der Übergangsrate und einer Übergangswahrscheinlichkeit. Die Übergangswahrscheinlichkeit gibt an, mit welcher Wahrscheinlichkeit man in eine andere Station (z.B. eines Warteschlangennetzes) wechselt. Der action part wird ausgeführt, wenn alle spezifizierten Prädikate im condition part erfüllt sind. Durch die Verwendung von immediate transitions kann der Modellierungsaufwand beträchtlich reduziert werden.

RESULT Konstrukt

In diesem Teil der Spezifikation werden die Leistungsgrößen spezifiziert, die direkt aus den Zustandswahrscheinlichkeiten abgeleitet werden können.

RESULT-MEAN Konstrukt

Mit Hilfe dieses Konstrukts wird die mittlere Anzahl von Aufträgen in den Knoten des Systems berechnet.

FINAL Konstrukt

Die berechneten Leistungsgrößen, die auf dem Bildschirm oder im Results File erscheinen sollen, müssen hier spezifiziert werden.

2.2.3. Spezifikation des Systems

```
#define queue    VAL[0] /* number of jobs in the queue */
#define active  VAL[1] /* number of active jobs */
#define working VAL[2] /* number of non defect CPU's */
#define immediate = -1.0 /* indicates a immediate transition */

VAR int    L ; /* buffer capacity */
VAR double mue ; /* service rate of each CPU */
VAR double lambda ; /* arrival rate of jobs at the queue */
VAR double coverage ; /* coverage factor */
VAR double mtbf ; /* mean time between failure */
VAR double mttr ; /* mean time to repair */
```

```

OUTPUT double qquer ; /* average queue length */
OUTPUT double rho ; /* utilization of the system */
OUTPUT double throughput /* throughput of the system */
OUTPUT double p_working ; /* probability that the system works */

DIM 3;
MIN 0 0 0 ;
MAX L 3 3 ;

START 0 0 3 ;

NOT { queue + active > L + 3 } ;

/* Rule for the arrival of jobs at the system */
RULE { queue < L } - > queue += 1 : lambda : 1.0 ;

/* Rule for the failure of a component */
RULE {working > 0 } - > working -= 1 : 1/mtbf : 1.0 ;

/* Rule for repairing a failed component */
RULE {working < 3 } - > working += 1 : 1/mttr : coverage ;

/* At least one server is free to service a job */
RULE { queue > 0 }{ working - active > 0 } - > active += 1 queue -= 1 : immediate :
1.0 ;

/* Rules for finishing jobs */
RULE { active == 1 } { working >= 1 } -> active -= 1 : 1*mue : 1.0 ;
RULE { active == 2 } { working == 1 } -> active -= 1 : 1*mue : 1.0 ;
RULE { active == 2 } { working >= 2 } -> active -= 1 : 2*mue : 1.0 ;
RULE { active == 3 } { working == 1 } -> active -= 1 : 1*mue : 1.0 ;
RULE { active == 3 } { working == 2 } -> active -= 1 : 2*mue : 1.0 ;
RULE { active == 3 } { working >= 3 } -> active -= 1 : 3*mue : 1.0 ;

RESULT {queue+active > 0 } - > rho += PROB ;
RESULT {working > 0 } - > p_working += PROB ;
RESULT DIST queue ;
RESULT qquer = MEAN queue ;

FINAL throughput = 3*rho*mue*p_working ;

```

2.3. Leistungsgrößen

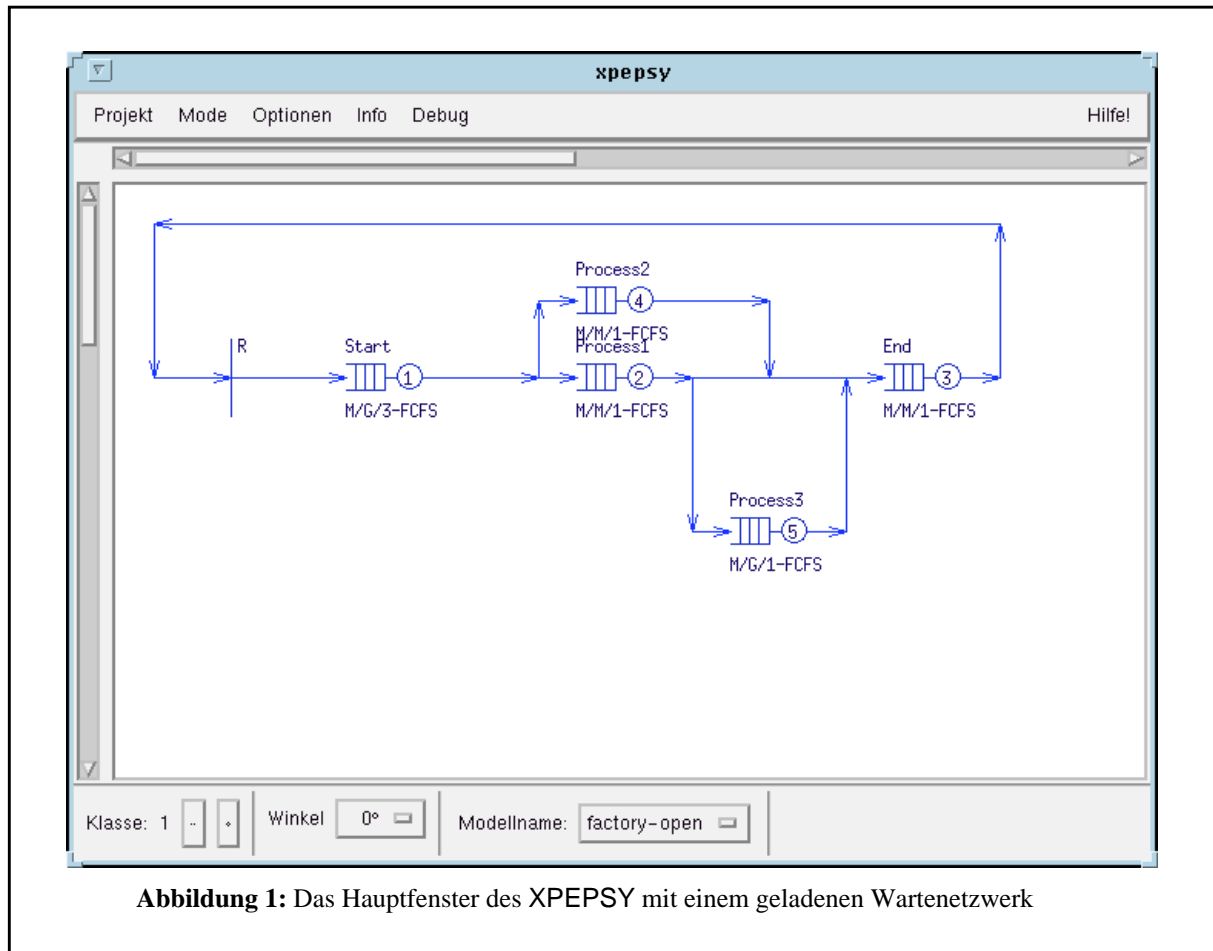
MOSES erzeugt aus der Spezifikation des Systems mittels **MOSLANG** automatisch die Generatormatrix Q [8]. Als nächstes werden die Zustandswahrscheinlichkeiten des Markovmodells durch Lösen des Gleichungssystems $pQ = 0$ berechnet [6] und daraus dann die mit **MOSLANG** spezifizierten Leistungsgrößen. Zur Lösung des Gleichungssystems stehen zur Zeit die Jakobi-Methode [10], das Gauss-Seidel-Verfahren [10], die LPU-Methode [4], die Crout-Methode [9] und die Grassmann-Methode [5] zur Verfügung.

3. PEPSY-QNS

Das Leistungsbewertungstool **PEPSY-QNS** kann immer dann eingesetzt werden, wenn sich das zu untersuchende System durch ein Warteschlangennetzwerk modellieren läßt. **PEPSY-QNS** gliedert sich in zwei Teile, das Basis-**PEPSY** und das graphische Front-End **XPEPSY**. Dadurch wird gewährleistet, daß die volle Funktionalität sowohl auf einfachen Terminalarbeitsplätzen als auch auf schnellen Graphikworkstations zur Verfügung steht, und ein Wechsel der Arbeitsumgebung jederzeit möglich ist.

3.1. Modellerstellung

Die Arbeit mit PEPSY-QNS beginnt mit dem Entwurf des Warteschlangenmodells. Es läßt sich mit XPEPSY einfach eingeben (Abbildung 1) und wird in der Modellbeschreibungdatei abgelegt [7].



Es lassen sich offene, geschlossene oder gemischte Netzwerke eingeben. Dabei ist eine Klasse mit einer Kette (chain) gleichzusetzen, d.h. es sind keine Wechsel der Aufträge einer Klasse in eine andere Auftragsklasse möglich. Dies stellt keine Einschränkung dar, da sich Netzwerke mit Klassenwechsel der Aufträge in Netze ohne umformen lassen. Die geschlossenen Klassen werden durch die Anzahl der im Netzwerk umlaufenden Aufträge beschrieben, die offenen Auftragsklassen durch die Spezifikation des Ankunftsprozesses.

Die Wartesysteme werden in PEPSY-QNS durch ihre Abarbeitungsstrategie, sowie die ersten zwei Momente des Bedienprozesses beschrieben. Zur Wahl stehen zum Einen die bekannten BCMP-Knotentypen (M/M/m-FCFS, M/G/1-Processor-Sharing, M/G/-Infinite-Server, M/G/1-LCFS) sowie FCFS-Wartesysteme mit beliebigen Bedienungszeitverteilungen (./G/m-FCFS). PEPSY-QNS stellt aber auch Knotentypen mit unterbrechenden oder nichtunterbrechenden Prioritätsstrategien sowie Mehrbedienersysteme mit inhomogenen Bedienprozessen bereit (M/M/m-FCFS-ASYM, M/G/m-FCFS-ASYM). Das Routing der Aufträge wird durch Angabe der Übergangswahrscheinlichkeiten spezifiziert.

3.2. Analysemethoden

In PEPSY-QNS gibt es eine Vielzahl von möglichen Analysemethoden zur Berechnung von Leistungsgrößen der Wartetzwerke. Im Allgemeinen haben die Verfahren gewisse Einschränkungen und Randbedingungen, die eine universelle Anwendbarkeit auf alle Modellbeschreibungen ausschließt. Es kann vom Anwender nicht erwartet werden, diese Restriktionen zu kennen und zu entscheiden, welche Methode zur Leistungsanalyse geeignet ist. Deshalb ist im PEPSY-QNS eine unterstützte Verfahrenauswahl vorgesehen, die dem Benutzer nur diejenigen

Verfahren nennt, die auf das zu untersuchende Modell anwendbar sind. Neben dieser einfachen Möglichkeitsscheidung sind die implementierten Verfahren auch noch grob in zwei Klassen unterschieden worden, so daß bei der einfachen Auswahl nur diejenigen genannt werden, die sich in irgendeiner Weise bewährt oder unentbehrlich gemacht haben.

Im Laufe der Jahre sind sehr viele bekannte Analyseverfahren und Neu- oder Weiterentwicklungen implementiert worden. Sie lassen sich in mehrere große Gruppen einteilen: [1]

- Methoden, die die Leistungsgrößen über Zustandswahrscheinlichkeiten berechnen (Markov-Analyse),
- Verfahren zur Berechnung der Leistungsgrößen über die Normalisierungskonstante (Faltung, RECAL - Recursion by Chain ALgorithm),
- die Mittelwertanalyse MVA und davon abgeleitete Verfahren (MVA, SCAT - Self Correcting Approximation Technique, Bard-Schweitzer-Algorithmus, LINEARIZER)
- Dekompositionsverfahren (Kühn, Chylla, Whitt, Pujolle, Gelenbe, Summationsmethoden, Marie, MEM - Maximum Entropie Methode),
- Produktformapproximationen (Diffusionsapproximation, EPM - Erweiterte Produktformmethode),
- Methoden zur Berechnung der oberen und unteren Grenzen der Leistungsgrößen (ABA - Asymtotic Bounds Analysis, BJB - Balanced Job Bounds analysis)
- Simulation.

Der Benutzer des Programmsystems läßt sich für seine Modellbeschreibung eine Liste der anwendbaren Methoden ausgeben. Diese Liste wird aufgrund der Eigenschaften des Modells, des Verfahrens und auch der der Implementation erstellt, so daß der Anwender keine detaillierten Kenntnisse über die Analysemethoden benötigt um Leistungsgrößen berechnen zu lassen. Er wird jedoch durch eine Datenbasis mit Beschreibungen der einzelnen Verfahren unterstützt. Sie enthält jeweils wichtige Literaturverweise, eine kurze Methodenbeschreibung, die Einschränkungen der Verfahrens bei der Anwendung sowie Hinweise auf Implementierungsdetails und verwandte Verfahren

3.3. Darstellung der Ergebnisse

Der Aufruf von PEPSY-QNS erfolgt beim Basis-PEPSY durch Angabe des Verfahrensnamens und der Modellbeschreibungdatei oder beim XPEPSY durch die Auswahl des Verfahrensnamens aus einer Liste.

Die Ergebnisse werden in einheitlicher Form präsentiert, so daß ein einfacher und direkter Vergleich der Ergebnisse unterschiedlicher Verfahren möglich ist. Alle Verfahren berechnen den Durchsatz, die Besuchshäufigkeit, die mittlere Bedienzeit, die Auslastung, die mittlere Verweilzeit, die mittlere Anzahl von Aufträgen, die mittlere Wartezeit und die mittlere Warteschlangenlänge zum Einen für jeden Knoten für jede Auftragsklasse getrennt zum Anderen knotenweise aufsummiert über alle Auftragsklassen.

Als Netzwerkgrößen werden für jede Klasse der Durchsatz, die mittlere Anzahl der Aufträge und die mittlere Durchlaufzeit berechnet. Neben diesen allgemein berechneten Werten erzeugen einige Verfahren besondere, zusätzliche Leistungsgrößen oder geben zusätzliche Informationen, wie z.B. die Anzahl der Iterationen eines Algorithmus oder die Genauigkeit der Simulation.

Parallel zur Entwicklung des XPEPSY ist auch ein graphisches Tool, XGAD, zur Aufbereitung der gewonnenen Leistungsgrößen entwickelt worden. Im einfachsten Fall werden damit die Ergebnisse einer Analyse durch Balkengraphiken visualisiert. Es lassen sich aber auch die berechneten Werte von bis zu acht unterschiedlichen Analysemethoden oder Modellen gleichzeitig darstellen und vergleichen. Eine wichtige Funktion ist die parameterabhängige Analyse und Darstellung. Dabei läßt sich das Verhalten des Modells bei wachsender Auftragszahl, Ankunftsrate oder Variation anderer Parameter untersuchen.

4. Zusammenfassung und Ausblick

MOSES ist ein sehr mächtiges Tool zur Spezifikation und Analyse von komplexen Systemen. Wenn der Anwender damit vertraut ist, kann er auch umfangreiche Systeme problemlos spezifizieren. Im Moment können mit **MOSES** nur steady state Analysen durchgeführt werden. Es ist aber geplant, das Tool auch auf transiente Analysen zu erweitern. In diesem Beitrag wird anhand eines einfachen fehlertoleranten Multiprozessorsystems der Um-

gang mit **MOSES** und insbesondere die Spezifikation eines Systems mit der Beschreibungssprache **MOSLANG** erläutert. Mit **MOSLANG** können die unterschiedlichsten Systeme spezifiziert werden wie z.B. Warteschlangennetze, Petrinetze, Präzedenzgraphen, Markov-Reward-Modelle, fehlertolerante Systeme, Produktionssysteme, Kommunikationssysteme oder Betriebssysteme. Z. Zt. wird an einer neuen Version von **MOSLANG** gearbeitet, die eine noch kompaktere und leichter lesbare Beschreibung erzeugt. Weiterhin ist geplant, neue Techniken zur Ermittlung der Generatormatrix und der Lösung des Gleichungssystems zu implementieren.

Die Leistungsanalyse einer großen Klassen von Wartenetzwerken ist mit **PEPSY-QNS** einfach durchzuführen. Dabei werden exakte oder approximative Berechnungsmethoden eingesetzt, so daß sich in kurzer Zeit eine große Zahl von Modellen oder Modellvarianten mit hinreichender Genauigkeit untersuchen lassen. Zusätzlich ist im Programmsystem eine Simulationskomponente enthalten, mit der sich approximative analytische Ergebnisse validieren oder verbessern lassen. Durch die Integration der auf **MOSES** basierenden Markov-Analyse für Warteschlangennetze wurde der Anwendungsbereich von **PEPSY-QNS** erheblich erweitert.

Weitere und aktuelle Informationen zu **PEPSY-QNS** und **MOSES** und deren Verfügbarkeit können beim ersten Autor angefordert werden. Detailliertere Angaben finden sich in [2] und [3].

5. Literatur

- [1] Bolch, G.: *Leistungsbewertung von Rechensystemen mittels analytischer Warteschlangenmodelle*, B. G. Teubner, Stuttgart, 1989.
- [2] Bolch, G., Kirschnick, M.: *PEPSY-QNS - Performance Evaluation and Prediction System for Queueing Networks*, Universität Erlangen-Nürnberg, Institut für Mathematische Maschinen und Datenverarbeitung IV, Tech. Report TR-I4-21-92; Oct. 1992
- [3] Bolch, G.; Greiner, S.: *The Markov Analyzer MOSES*, Universität Erlangen-Nürnberg, Institut für Mathematische Maschinen und Datenverarbeitung IV, Tech. Report TR-I4-10-94; June 1994
- [4] George, A.; NG., E.: *An Implementation of Gauss elimination with partial pivoting for sparse systems*, SIAM Journal of Scientific and Statistical Computing, 6, pp. 390-409, 1985.
- [5] Grassmann, W. K.: *Finding Transient Solutions in Markovian Event Systems Through Randomization*, Proceedings the First International Conference on the Numerical Solution of Markov Chains, Raleigh, 1990.
- [6] Hennecke, C.: *Parallele Implementierung der numerischen Analyse*, Diplomarbeit am IMMD IV, Erlangen, 1988.
- [7] Kirschnick, M.: *XPEPSY - Handbuch*, Universität Erlangen-Nürnberg, Institut für Mathematische Maschinen und Datenverarbeitung IV, Tech. Report TR-I4-93-18, Sept. 1993.
- [8] Munkert, F.: *Effiziente Erzeugung von Generatormatrizen für zeitkontinuierliche Markovketten*, Studienarbeit am IMMD IV, Erlangen, 1991.
- [9] Schmeisser, G.: *Praktische Mathematik*, de Gruyter, Berlin, New York, 1976.
- [10] Stewart, W. J.: *A Comparison of Numerical techniques in Markov Modelling*, Communications of the ACM, Vol. 21, pp. 144-152, No. 2 Feb. 1978.

Dr.-Ing. Gunter Bolch, Studium der Nachrichtentechnik in Karlsruhe und Berlin, wiss. Ass. am Institut für Regelungstechnik der Uni Karlsruhe, Akad. Rat und seit 1982 Akad. Direktor am Lehrstuhl für Betriebssysteme der Uni Erlangen, Arbeitsgebiete: Quantitative Analyse von Rechensystemen und Prozeßautomatisierung, mehr als 60 Veröffentlichungen darunter 3 Fachbücher, zahlreiche Industriekooperationen, mehrere Forschungsaufenthalte an ausländischen Universitäten u.a. in Brasilien, UdSSR und USA

Dipl.-Inf. Stefan Greiner, geb. 1966 in Kümmersbruck/Oberpfalz, Studium der Informatik an der Uni Erlangen mit Schwerpunkt Betriebssysteme und Nebenfach Mathematik, Diplomarbeit: Ein analytisches Modell für das Betriebssystem BS2000, seit 1994 Studien- und Forschungsaufenthalt an der Duke University, Durham NC, USA.

Dr.-Ing. Hermann Jung, geb. 1960 in Mitwitz/Oberfranken, Studium der Informatik an der Uni Erlangen mit Schwerpunkt Betriebssysteme, 1986 -91 wiss. Ass. am Lehrstuhl für Betriebssysteme der Uni Erlangen, Dissertation: Leistungsbewertung UNIX-basierter Multiprozessorbetriebssysteme, seit 1991 Siemens AG Erlangen Bereich Anlagentechnik

Dipl.-Inf. Matthias Roessler, geb. 1966 in Frankfurt am Main, Studium der Informatik mit Nebenfach Fertigungsautomatisierung an der Uni Erlangen, seit 1991 wiss. Ass. am Lehrstuhl für Betriebssysteme der Uni Erlangen, Arbeitsgebiete: Quantitative Analyse von Rechen- und Fertigungssystemen, Entwicklung des Tools PEPSY-QNS

Dipl.-Inf. Raimund Zimmer, geb. 1963 in Nabburg/Oberpfalz, Studium der Informatik an der Uni Erlangen mit Schwerpunkt Algorithmische Sprachen, Diplomarbeit: Implementierung einer Auswertemethode für Markovmodelle von Rechensystemen, seit 1994 Softwareentwickler bei der Siemens AG Erlangen Bereich Medizintechnik