— **Hands on Tools** —

Hands on HIT

Dokumentversion 2.2.00

H. Beilner J. Mäter (ed.) M. Sczittnick Ch. Wysocki

July 2004

Hands on HIT

Zusammenfassung:

Der vorliegende Text führt in die praktische Benutzung der Modellierungsumgebung HIT ein. Mit diesem Ziel wird eine einfache Leistungssbewertungstudie beschrieben und begleitet. Die Studie stützt sich im Problembereich auf den Entwurf eines hypothetischen Literatur-Recherche-Systems mit Datenhaltung auf CD-ROMs und untersucht dabei auftretende Fragen.

Im Modellierungsbereich wird der Benutzer zunächst anhand eines einfachen initialen Minimalmodells (durch leichte Veränderung getroffener Problemannahmen) auf die Notwendigkeit der Verfügbarkeit unterschiedlicher Modell-Analysetechniken gestoßen (einfache Wartesysteme, separable Wartenetze, numerische Markoff-Analyse, Simulation). Anschließend werden schrittweise, durch verfeinerte Annahmen im Problembereich motiviert, umfangreichere Modelle aufgebaut und bezüglich ihrer Leistungsfähigkeit analysiert.

Dabei erweist sich zum einen die ausgeprägte Strukturierung von HIT-Modellen als hilfreich (den Modell-Spezifikationsaufwand mindernd), zum anderen tritt erneut die Notwendigkeit des Einsatzes unterschiedlicher Analysetechniken zutage, die in HIT ohne wesentliche Neuspezifikation zugreifbar sind.

Die Problem- (und Modell-) Verfeinerung erreicht schließlich einen Punkt, der funktionale (einen potenziellen Deadlock betrefende) Probleme aufdeckt. Ein Querverweis auf das Nachbarpraktikum „Hands on QPN“ eröffnet die Möglichkeit, Fragen dieses Typs systematisch (und Tool-gestützt) zu untersuchen. Zwei alternative Lösungsmöglichkeiten werden zurückgeliefert, die im vorliegenden Praktikum bezüglich wechselseitiger, leistungsorientierter Vor- und Nachteile untersucht werden.

Kontaktadresse:
Universität Dortmund
Informatik IV
Dipl. Inform. J. Mäter
D-44221 Dortmund

Telefon: +49 231 755 2411
Telefax: +49 231 755 4730
E-Mail: Juergen.Maeter@udo.edu

Inhaltsverzeichnis

Kapitel 1

Einführende Bemerkungen

Wir richten uns mit diesem Dokument an zukünftige HIT/HITGRAPHIC Anwender, die anhand eines ausformulierten Modellierungsbeispiels die Handhabung des Tools kennenlernen wollen.

Das Dokument ist so konzipiert, daß parallel zum Lesen die erläuterten Sachverhalte sofort am Rechner nachzuvollziehen sind. Die Verfügbarkeit einer Workstation mit dem Tool HIT inklusive HITGRAPHIC ist dazu erforderlich.

Als Zeitrahmen für das Praktikum sind etwa vier Stunden angesetzt. Hinweise zum Überspringen bestimmter Abschnitte beziehen sich auf diesen Rahmen.

Es wird eine Anleitung zum praktischen Arbeiten gegeben, die mit einem einfachen Beispiel beginnt, das im folgenden ausgebaut wird. Detaillierte Hilfen zur Benutzung des Tools sind enthalten. Die prinzipielle Vorgehensweise ist bei allen Beispielen ähnlich:

- Spezifikation eines Modelltyps mit Last- und Maschinenbeschreibung
- Spezifikation der Auswertungsbeschreibung
- Spezifikation der Experimentbeschreibung
- Experimentlauf

Voraussetzungen für ein sinnvolles Verstehen dieses Dokuments sind Grundkenntnisse im Bereich Rechnerarchitektur, in höheren imperativen Programmiersprachen und in der Workstationhandhabung. Weiterhin wird Erfahrung in der Modellierung vorausgesetzt, ebenso wie ein grundsätzliches Wissen über Lösungsverfahren (d.h., Techniken der Modellanalyse) wie Simulation und analytische Techniken.

Neben der üblichen Untergliederung des Textes in Kapitel und Abschnitte ist die Anleitung weiter untergliedert durch verschiedenartig eingerückte Absätze und Überschriften. Dies soll hier kurz erläutert und demonstriert werden.

Absätze, die sich mit der Beschreibung des Modellierungsproblems befassen, sind nicht eingerückt.

Modellbildung

Absätze mit dieser Überschrift und dieser Einrückung befassen sich mit grundlegenden Aspekten der Modellbildung mit dem Tool.

Toolumsetzung

Die Hinweise zur Umsetzung des Problems in eine vom Tool akzeptierte Beschreibung erfolgt in dieser Einrückungsstufe und unter dieser Überschrift. Hier werden auch detaillierte Anweisungen und Hilfen zur Handhabung des Tools gegeben.

In der zweiten Einrückungsstufe werden Hinweise zur eingesetzten Methodik gegeben. Dabei werden die grundlegende Theorie oder Eigenschaften eingesetzter Algorithmen kurz angesprochen.

Kapitel 2

Literaturrecherche: HIT-Modell 1

Einige Einsatzmöglichkeiten des Tools HIT sollen an einem Fallbeispiel „Literaturrecherche“ erprobt werden. Unterschiedliche Modellannahmen erfordern vier verschiedene Szenarien/Varianten. Die Varianten werden mit den von HIT unterstützten Lösungsmethoden analysiert (analytisch-algebraisch, analytisch-numerisch und simulativ).

2.1 Erste Modellbeschreibung

Wir versetzen uns in die Situation eines Anbieters einer Dienstleistung Literaturrecherche. Nutzer dieser Dienstleistung stellen Anfragen; ein Rechensystem, das die Dienstleistung erbringt, liefert die entsprechenden Antworten zurück. Der Anbieter der Dienstleistung wünscht, das „passende“ System für die Literaturrecherche zu finden: Zum einen soll das System zur Zufriedenheit der Nutzer ausreichend schnell antworten, zum anderen sowohl hinreichend ausgelastet sein, als auch gewisse Reserven für eine zukünftige Steigerung der Nachfrage besitzen.

Zur Lösung dieses Leistungsbewertungsproblems sind weitere Informationen notwendig: Wie ist die Belastung des Systems, wieviele Benutzer stellen Anfragen, welche und wieviele Operationen beinhaltet eine Benutzersitzung? Zum anderen ist Wissen darüber nötig, wie das Rechensystem auf Anfragen reagiert, d.h. hier, wie schnell die Abarbeitung der Operationen erfolgen kann. Aus einer groben Systemanalyse ergeben sich folgende Basisdaten:

- Die Benutzungsrate liegt bei 0.5 Sitzungen pro Sekunde, d.h. etwa alle 2 Sekunden beginnt eine neue Sitzung.
- Eine Sitzung beinhaltet durchschnittlich 10 Anfrageoperationen.
- Das Rechensystem kann etwa 8 Operationen pro Sekunde erbringen.

Mit diesen groben, ersten Informationen ergibt sich die Möglichkeit, ein entsprechend grobes, erstes Modell zu erstellen.

Modellbildung

In der Modellwelt von HIT werden Systeme hierarchisch betrachtet. An jeder Ebene der Hierarchie werden Dienste angeboten, die von darüberliegenden Schichten genutzt werden können. Die Realisierung der angebotenen Dienste befindet sich in darunterliegenden Schichten.

Wir unterscheiden in jeder Schicht **Last** und **Maschine**. Eine Last, z.B. ein Prozeß, ein Job, ein Kunde, ein Dienst, belastet das System mit Anforderungen, die von angebotenen Diensten der Maschine zu erfüllen sind. Die Last wird in HI-SLANG (der Spezifikationssprache von HIT) durch **Services** (Lasttypen) beschrieben. Jedes erzeugte (d.h. zur Ausführung gebrachte) „Exemplar“ dieses Lasttyps verhält sich gemäß seines spezifischen (in HI-SLANG angegebenen) Verhaltens- oder Prozeßmusters.

Die Maschine, die aus wechselseitig unabhängigen **Komponenten** besteht, bietet Dienste zur Nutzung an. Jede dieser Komponenten hat ihr eigenes dynamisches Verhalten, das durch Kontrollprozeduren geregelt ist; so legen die Kontrollprozeduren z.B. Bedienstrategie, Bediengeschwindigkeit u.ä. fest.

Im ersten zu modellierenden HIT-Modell besteht die Systemlast (die Benutzeraktivität) aus „Sitzungen“, in denen Dienstleistungen vom Rechensystem angefordert werden. Jede Sitzung verläuft ähnlich: Es werden durchschnittlich ca. 10 Anfrageoperationen durchgeführt. Durch diese Information ist das Prozeßmuster vorgegeben.

Die Maschine besteht in diesem Fall aus einer einzelnen Komponente, dem Rechensystem, das in einer Sekunde 8 Operationen ausführen kann.

Toolumsetzung

Sie setzen nun die vorgegebenen Informationen in eine vom Tool HITGRAPHIC akzeptierte Beschreibung um. HITGRAPHIC ist die graphische Benutzungsoberfläche zu HIT, einschließlich der Datenhaltung.

Nachdem Sie eine eigene Datenbank erstellt und die Environmentvariable `DB_DIR` gesetzt haben (vgl. First Steps in [2]), können sie das Tool mit Ihrer Datenbank aufrufen:

```
hitgraphic <your_database>
```

Es erscheinen das Start- und das Environment-Fenster. HITGRAPHIC unterstützt verschiedene Environments (Arbeitsumgebungen). Erstellen Sie ein neues Environment durch Aktivieren des Buttons *new environment* mit der linken Maustaste. Eine Dialogbox fordert Sie auf, den Namen des Environments einzugeben. Mit einem Mausklick (linke Maustaste) auf *OK* verschwindet die Box, und in der Datenbank wird Ihr Arbeitsenvironment angelegt.

Modellbildung

Das Environment-Fenster verwaltet für Sie die Modellierungsobjekte und stellt Funktionen zur Manipulation des Environments und aller Modellierungsobjekte zur Verfügung. Sie können von diesem Fenster aus auch HIT starten, um Experimente entsprechend Ihrer Beschreibungen auszuführen.

Toolumsetzung

Das Tool interagiert mittels Menüs und Dialogboxen mit dem Benutzer. Die Menüs können Sie mit der rechten Maustaste aktivieren

- auf Titelbalken,
- auf Objekten, die durch ein Label oder graphisches Symbol repräsentiert werden und
- auf dem Fenster-Hintergrund.

Dialogboxen erscheinen z.B., wenn zusätzliche Informationen abgefragt werden müssen, oder wenn Sie ein Objekt löschen wollen. Diese Boxen müssen von Ihnen durch einen Mausklick mit der linken Maustaste auf *OK* bestätigt werden, bevor Sie weiterarbeiten können. Die linke Maustaste wird auch für andere Funktionen verwendet, z.B. zur Auswahl oder zur Positionsmarkierung bei der *move*-Operation.

Beginnen Sie nun mit der Spezifikation des ersten Modelltyps. Dazu klicken Sie mit der rechten Maustaste auf den Titelbalken mit dem Label MODEL TYPES. Wählen Sie aus dem Popup Menü den Eintrag *new model type*, und benennen Sie den Modelltyp ('**bib1**'). Bestätigen Sie den Namen mit einem Mausklick auf *OK* oder mit *Return*. Return bewirkt bei Dialogboxen die Ausführung des hervorgehobenen Buttons. Der Modelltyp wird in die Liste der Modelltypen eingefügt, und Sie können anschließend das Popup Menü auf dem Listenlabel 'bib1' aktivieren. Wählen Sie die *open*-Funktion. Es erscheint ein leeres Typ-Graphik-Fenster, in dem Sie Ihre Modellbeschreibung erzeugen können.

Beschäftigen Sie sich zunächst mit der Systemlast des ersten Modelltyps, also den Benutzeraktivitäten. Die Systemlast wird durch Services beschrieben. Selektieren Sie bitte aus dem Hintergrundmenü den Eintrag *new service*, und nennen Sie die Aktivität '**sitzung**'. Der neue Service wird als Parallelogramm über dem ACTIVITIES-Symbol eingezeichnet.

Eine Sitzung besteht durchschnittlich aus 10 Anfragen. Diese Subaktivität, die eine Anforderung an das System stellt, wird als *used service* bezeichnet. Aktivieren Sie das Popup Menü auf dem Service-Symbol, wählen Sie *new used service*, und geben Sie den Namen '**anfrage**' ein. Ein horizontaler Pfeil symbolisiert die Subaktivität. Die „Größe" der Anforderung wird mittels eines formalen Parameters spezifiziert. Daher selektieren Sie den Eintrag *formal parameters* in dem Popup Menü auf 'anfrage'. Der HITGRAPHIC Editor ermöglicht Ihnen nun die textuelle Eingabe dieses Parameters, den Sie bitte im oberen Subeditor (PARAMETERS) eingeben:

```
amount : REAL
```

Der untere Editor (RESULTS) wird für Ergebnis-Parameter verwendet, die in diesem Beispiel keine Verwendung finden.

Der HITGRAPHIC Editor ermöglicht textuelle Eingaben (HI-SLANG Code). Er ist mit einer *check*-Funktion ausgestattet, die eine lokale syntaktische und

semantische Überprüfung des HI-SLANG Codes vornimmt. Der Editorinhalt wird mit der *save*-Operation im Menü des Subeditor-Titels oder mit *save all* im Menü des Editor-Titels gesichert. Verlassen können Sie den Editor mit der *quit*-Operation. Die Operation *save all & quit* faßt diese beiden Aktionen zusammen.

Nachdem Sie den Editor für die formalen Parameter geschlossen haben, können Sie das Verhaltensmuster der Aktivität 'sitzung' eingeben. Dazu wählen Sie die *open*-Operation im Popup Menü des Service-Symbols. Es wird ein unterteiltes Editor-Fenster aufgeblendet. Da der Service weder formale Parameter noch Ergebnis-Parameter oder lokale Deklarationen besitzt, bleiben die entsprechenden Subeditoren (auf der linken Seite) frei. Das Verhaltensmuster wird im Subeditor BODY eingegeben:

```
anfrage (negexp (1/10));
```

Dynamisch bedeutet das, daß ein Aufruf des *used service* 'anfrage' die Ausführung des angebundenen *provided service* in einer Komponente der tieferen Schicht auslöst. Die Bearbeitung auf der oberen Schicht kann erst fortfahren (bzw. hier: ist erst abgeschlossen), wenn der Dienst auf der tieferen Schicht abgearbeitet ist (vgl. Aufruf in sequentiellen Programmiersprachen).

Da im Durchschnitt 10 Operationen benötigt werden, wählen Sie als aktuellen Parameter für die Subaktivität 'anfrage' die Exponentialverteilung mit der Rate 1/10.

> Über die Größe der Anforderung ist nur der Mittelwert bekannt. Für eine Auswertung ist aber die Vorgabe einer Verteilung notwendig. Dieser Freiheitsgrad ist hier durch die Wahl der Exponentialverteilung genutzt, so daß die analytischen Lösungsverfahren leicht zugänglich sind. Im folgenden werden wir wiederholt nach diesem Gesichtspunkt eine Verteilung auswählen. Es sollte aber bewußt sein, daß die Art der Verteilung durchaus Einfluß auf die Ergebnisse haben kann.
>
> Als Kritikpunkt ist zu bemerken, daß damit eine kontinuierliche Verteilung für eine ursprünglich diskrete Größe, die Zahl der Operationen, benutzt wird.

Toolumsetzung

Sichern und verlassen Sie den Editor wie oben beschrieben.

Die Beschreibung der Last ist nun abgeschlossen. Die explizite Lasterzeugung geschieht im Editor für den Modelltyp. Öffnen Sie den Editor (*open* auf dem ACTIVITIES-Symbol), und geben Sie im ACTIVITIES Editor das CREATE-Statement ein:

```
CREATE 1 PROCESS sitzung EVERY negexp (0.5);
```

Damit wird, wie in der Problemstellung beschrieben, mit variablen Zeitabständen (im Mittel) alle 2 Sekunden ein neuer Service 'sitzung' gestartet. Sichern und verlassen Sie den Editor.

Nun wenden Sie sich der Maschinenkomponente Ihres Modelltyps zu. Diese besteht aus einem Rechensystem, das 8 Operationen pro Sekunde bearbeiten kann. Das Hintergrundmenü stellt die Funktion *new component* zur Verfügung. Sie werden aufgefordert, einen Komponententyp zu selektieren. Dazu sehen Sie bitte im Environment-Fenster unter der Komponentenliste nach. Gewisse Standard-Komponententypen werden Ihnen von HITGRAPHIC angeboten. (Sie haben natürlich auch die Möglichkeit, selbst Komponententypen zu definieren, die Sie dann in hierarchischen Modellen verwenden können.) Selektieren Sie bitte den *server* mit der linken Maustaste, und bestätigen Sie die Auswahl. Geben Sie der Komponente den Namen '**rechensystem**'. Die Komponente wird in der Graphik unten als Rechteck eingezeichnet.

Der *server* bietet den Dienst 'request' als *provided service* an, der von den Subaktivitäten (den *used services* dieser Schicht) verwendet werden kann. Die Bindung von *used* und *provided* service wird mit der *set*-Operation am entsprechenden Kreuzungspunkt der sie repräsentierenden Linien realisiert. Zur Kennzeichnung erscheint ein schwarzer Kreis über dem Schnittpunkt.

Die Geschwindigkeit der Maschine (Sie erinnern sich, 8 Operationen pro Sekunde) wird mit Hilfe der *control procedures* festgelegt. Wählen Sie diesen Eintrag aus dem Popup Menü auf dem Komponenten-Symbol. Das Anlegen der Kontrollprozeduren erfordert eine *save*-Operation, deren Anforderung von Ihnen bestätigt werden muß. Auch einige weitere Funktionen erfordern ein vorheriges *save*, was jeweils durch eine Aufforderung zur Bestätigung erkennbar ist. Wir werden dies an den entsprechenden Stellen nicht mehr gesondert erwähnen.

Wie das Fenster für die Spezifikation der Kontrollprozeduren zeigt, besteht eine Komponente aus vier Bereichen:

- **Announce Queue** für Anmeldungen von Prozessen,
- **Entry Area** als Warteraum für Prozesse,
- **Service Area** zur Bedienung von Prozessen und
- **Exit Area** für fertig bearbeitete Prozesse.

Die Kontrollprozeduren beschreiben Bedingungen für und Art von Übergängen:

- **Accept** für die Annahme,
- **Schedule** zur Übernahme in die Bedienung bzw. zur Verdrängung aus der Bedienung,
- **Dispatch** zur Festlegung der Bediengeschwindigkeit und
- **Offer** für das Abgabeverhalten zur nächsten Komponente.

Das Rechensystem ermöglicht die (quasi-)parallele Abarbeitung der Operationen.

Toolumsetzung

Die Kontrollprozeduren sind mit Default-Werten belegt. Wir wollen für die Komponente 'rechensystem' eine gleichmäßige Aufteilung der Bedienkapazität auf alle parallel bearbeiteten Prozesse erreichen. Wählen Sie bitte die Dispatch-Prozedur *shared* aus dem Popup Menü auf dem Label *dispatch*.

> Für Kenner von Warteschlangensystemen: Diese Strategie ist dort als processor sharing bekannt.

Toolumsetzung

Sie müssen noch die Geschwindigkeit der Bedienung (8 Operationen pro Sekunde) als aktuellen Parameter für die Prozedur angeben. Dazu öffnen Sie den Parametereditor durch Ausführung der *open*-Operation auf dem Parameterfeld. Nach Ausführung der angeforderten *save*-Operation, wird der Editor aufgeblendet, und Sie können im unteren Subeditor den aktuellen Wert des Parameters *speed* verändern:

```
LET speed := 8.0
```

Der Editor wird gesichert und verlassen (*save all & quit*). Sichern und schließen Sie auch das Kontrollprozedur-Fenster mit dem Menü auf dem Titelbalken.

Damit ist die Spezifikation Ihres ersten Modelltyps abgeschlossen. Ihr Graphikfenster sollte aussehen wie in Abbildung 2.1 dargestellt. Sichern und schließen Sie nach einem Vergleich Ihr Graphikfenster.

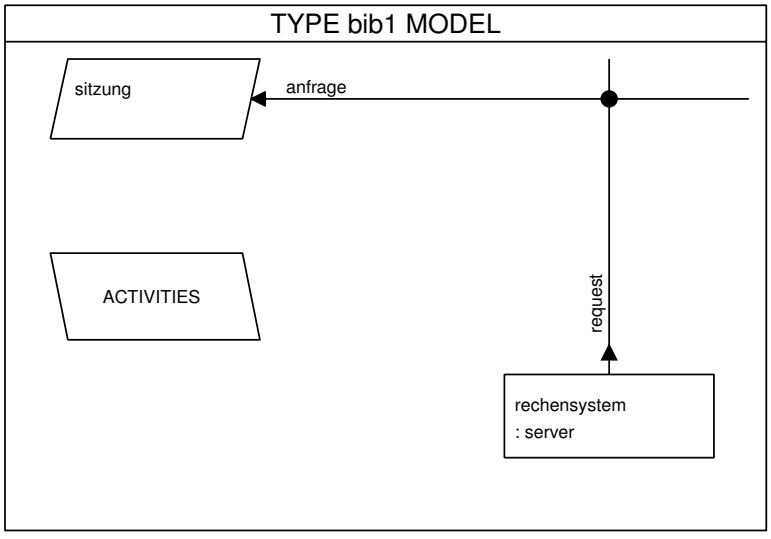

Abbildung 2.1: Typ des HIT-Modells 1

Vom Environment-Fenster aus können Sie sich mit der Funktion *show survey* eine Übersicht Ihres Modelltyps anzeigen lassen. Das Survey-Fenster unterscheidet die Typ- und die Objektstruktur und stellt die hierarchische Beziehung der verwendeten Typen bzw. der Objektinstanzen dar. Sie können

mit den Buttons unter dem Titelbalken zwischen den beiden Darstellungen wechseln (Mausklick mit der linken Taste). Auch das Survey-Fenster wird mit der *quit*-Funktion verlassen.

> Hier ist nun der Punkt, an dem die Parallele zu den Warteschlangensystemen fortgesetzt werden kann. Das spezifizierte Modell entspricht dem bekannten Wartesystem vom Typ M/M/1-processor sharing. Die Abstände der Ankünfte (im CREATE-Statement) sind unabhängig exponentiell verteilt mit Rate 0.5. Damit ergibt sich als Ankunftsprozeß ein Poissonstrom mit Ankunftsrate $\lambda = 0.5$. Die Bedienwünsche sind ebenfalls unabhängig exponentiell verteilt; die Bedienrate μ ergibt sich als Quotient von Bediengeschwindigkeit und mittlerer Bedienanforderung, also $\mu = 8.0/10.0$.

Modellbildung

Wir wollen das Modell bewerten, müssen also auch spezifizieren, welche Leistungsmaße an welcher Komponente ermittelt werden sollen. Diesen Zweck erfüllt die Auswertungsbeschreibung, die Sie nun erzeugen müssen.

Toolumsetzung

Eine Auswertungsbeschreibung ist immer einem speziellen Modelltyp zugeordnet, wobei zu einem Modelltyp auch mehrere Auswertungsbeschreibungen existieren können.

Führen Sie die Funktion *new evaluation* im Environment-Fenster auf der Überschrift EVALUATIONS aus. Nachdem Sie Ihren gerade erzeugten Modelltyp selektiert und einen Namen für die Auswertungsbeschreibung (‘**bib1_eva**’) eingegeben haben, können Sie diese Auswertungsbeschreibung, die in die Liste eingefügt wird, mit der *open*-Funktion öffnen. Die Graphik im Auswertungsfenster zeigt Ihnen die Objektstruktur des Modells.

Erzeugen Sie ein Auswertungsobjekt an der Komponente ‘rechensystem’, indem Sie im Popup Menü auf dem entsprechenden Komponentensymbol den Eintrag *new evaluation object* auswählen. Geben Sie dem Auswertungsobjekt den Namen ‘**rs_reaktion**’. Eine kleine Box wird an die Komponente angehängt, die Sie mit *open* öffnen können.

Nun wird das Auswertungsobjekt-Fenster angezeigt, in dem Sie Ihre Analysewünsche bzgl. dieses Auswertungsobjekts spezifizieren können.

Das Fenster ist in die folgenden Bereiche aufgeteilt:

- **Area:** Die Auswertung für ein Auswertungsobjekt kann sich auf die gesamte Komponente oder auf einen einzelnen, per Mausklick selektierbaren Bereich (Announce Queue, Entry Area, Service Area, Exit Area) beziehen. Initial wird die gesamte Komponente betrachtet.
- **Streams:** Diese Liste enthält die verfügbaren Leistungsmaße. Dazu gehören die Standardströme (Turnaroundtime, Population ...) und die vom Modellierer im Typ-Graphik-Fenster selbstdefinierten Ströme vom

Typ Event (Zeitreihe), State (Zustandstrajektorie) und Count (zeitbezogene Ereigniszählung).

- **Estimator:** Die verfügbaren Schätzer können vom Modellierer ausgewählt werden, z.B. Mittelwerte oder Standardabweichung und Konfidenzintervall (nur für den simulativen Löser).
- **Hierarchy:** In dieser Liste werden alle vom Modellierer in der Auswertungsbeschreibung erzeugten Lasthierarchien für die Komponente aufgeführt. Damit kann eine Analyse differenziert nach Verursachern ausgeführt werden, z.B. können Verweilzeiten für Lese- und Schreibzugriffe getrennt bewertet werden.
- **Start/Stop:** Für die Simulation können Start- und Stopkriterien angegeben werden, die die Messung beeinflussen. In Abschnitt 2.3 wird ein Beispiel für eine Anwendung gezeigt.

In dem Beispielszenario ist der Betreiber des Systems am Durchsatz, an der Antwortzeit und an der Auslastung interessiert. Deshalb selektieren Sie in dem Subfenster STREAMS die Ströme *THROUGHPUT*, *TURNAROUNDTIME* und *UTILIZATION* mit der linken Maustaste. Es erscheint ein kleiner Pfeil, der anzeigt, welche Ströme angefordert werden. Voreingestellt ist der Strom *POPULATION*.

Die übrigen Einstellungen übernehmen Sie bitte unverändert. Wir kommen in den weiteren Modellbeispielen noch auf einige zurück.

Sie können jetzt das Auswertungsobjekt-Fenster sichern und schließen, ebenso das Auswertungsfenster.

Modellbildung

Vor der Analyse wenden wir uns der Experimentbeschreibung zu. Sie enthält die Wahl des Lösungsverfahrens, etwa erforderliche Steuerungsparameter und die Belegung von Modellparametern. Auch die Beschreibung von Experimentserien zu einem oder zu verschiedenen Modellen ist möglich.

Toolumsetzung

Die Experimentbeschreibung können Sie ebenfalls mit *new experiment* im Environment-Fenster in der entsprechenden Spalte erzeugen. Nennen Sie das Experiment '**bib1_exp**', und öffnen Sie es. Im Experiment-Fenster werden die zu verwendende Methode und Auswertung beschrieben. Unser erstes grobes Modell kann analytisch gelöst werden. Wählen Sie den Löser *DOQ4* mit der linken Maustaste.

> DOQ4 ist ein Löser, der verschiedene Algorithmen enthält. Die Hauptdomäne ist die exakte Lösung von Warteschlangennetzen mit Produktformlösung. Eine etwas erweiterte Modellklasse ist ebenfalls noch lösbar, allerdings sind dann die Ergebnisse nicht mehr exakt, sondern liefern eine Näherung bzw. Abschätzung der exakten Lösung. Das vorliegende Modell fällt in die Klasse der exakt lösbaren Modelle.

Toolumsetzung

Im Unterfenster EVALUATIONS werden die zu verwendenden Auswertungen aufgelistet. Sie fügen 'bib1_eva' mit der Funktion *insert evaluation* im Popup Menü des Titelbalkens ein. Dazu müssen Sie wieder im Environment-Fenster die entsprechende Auswertungsbeschreibung durch einen Mausklick mit der linken Taste selektieren. Das EVALUATE-Statement wird automatisch (nach Bestätigung) in den Rumpf des Experiments (BODY) eingetragen.

Der Experimentbody enthält die Ausführungsbeschreibung und kann natürlich auch editiert werden. Er wird mit der *open*-Funktion des Titelmenüs aufgeblendet:

```
EVALUATE bib1_eva;
```

Sie könnten an dieser Stelle auch mehrere Auswertungen angeben oder bei parametrisierten Modellen Experimentserien spezifizieren. Das EVALUATE-Statement muß immer in einer Zeile stehen und mit einem Semikolon abgeschlossen werden.

Sichern und schließen Sie den Editor und anschließend das Experiment-Fenster.

Jetzt können Sie vom Environment mittels des Menüpunktes *transform & run* im Popup Menü für das Experiment einen Analyselauf starten. Die graphische Beschreibung wird in ein HI-SLANG Programm (HI-SLANG ist die Spezifikationssprache für HIT) überführt. Nach einer fehlerfreien Transformation wird das HIT RUN CONTROL Fenster aufgeblendet, das Ihnen im Popup Menü des Titels mit *run* die Möglichkeit bietet, HIT aufzurufen. Im Experiment Run Control Fenster können Sie die einzelnen Schritte von HIT verfolgen. Der HI-SLANG Code wird in Simula überführt, der Löser wird gebunden, und das Analyseprogramm wird ausgeführt.

Wenn Sie alles fehlerfrei eingegeben haben, können Sie sich nach dem HIT-Lauf die Ergebnisse, die automatisch in die Datenbank eingefügt werden, ansehen. Wählen Sie dazu die Funktion *show table* im Popup Menü vom Experiment 'bib1_exp'. HIT liefert in einer Tabelle die folgenden Werte:

	Mean Value
POPULATION	1.67
THROUGHPUT	0.50
TURNAROUNDTIME	3.33
UTILIZATION	0.63

Da wir bereits erkannt haben, daß man das Modell als M/M/1-System mit processor sharing sehen kann, können wir die von HIT ermittelten Ergebnisse mit den nach der Theorie zu erwartenden vergleichen. Die Auslastung ρ ergibt sich als Quotient von Ankunftsrate und Bedienrate, also $\rho = \frac{\lambda}{\mu} = \frac{0.5}{0.8} = 0.625$. Da die Auslastung kleiner 1 ist, kann sich Verkehrsflußgleichgewicht einstellen, sodaß der Durchsatz gleich der Ankunftsrate ist, also bei 0.5 Sitzungen pro

Sekunde liegt. Die mittlere Population liegt bei $\frac{\rho}{1-\rho} = 1.\overline{6}$. Schließlich kann man mit dem Satz von Little die mittlere Antwortzeit des Systems als Quotient von mittlerer Population und Durchsatz ermitteln; sie ergibt sich zu $\frac{1.\overline{6}}{0.5} = 3.\overline{3}$. Diese Ergebnisse sollten sie auch mit HIT erhalten haben.

Wenn Sie für die Modellierung und Analyse Ihres ersten HIT-Modells nur 40 Minuten gebraucht haben, können Sie in den Abschnitten 2.2 und 2.3 den numerischen und den simulativen Löser anhand leicht modifizierter Modelle ausprobieren.

Andernfalls sollten Sie diese Abschnitte überspringen, da Sie sonst vielleicht zu wenig Zeit für die aus Modellierungssicht interessanteren Kapitel 3 und 4 haben. Falls am Ende noch genügend Zeit zur Verfügung steht, können Sie problemlos wieder hier aufsetzen, da das Modell in der Datenbank bestehen bleibt.

2.2 Modell mit beschränktem Benutzerzugang

Im tatsächlichen Betrieb werden auch Hochlastphasen erwartet, die eine längere Zeitdauer anhalten. Wir nehmen einmal eine Verdoppelung der Last, d.h. eine Verdoppelung der Benutzungsrate an. Damit die Antwortzeiten und damit ggf. auch die Leitungskosten für den Nutzer nicht zu hoch werden, soll die maximale Zahl der parallel bearbeiteten Sitzungen auf 10 begrenzt werden. Überzählige Anforderungen werden abgewiesen. Dabei stellt sich die Frage, ob dann nicht auch im Normalbetrieb zu viele Benutzer abgewiesen werden und ob die Antwortzeiten in Hochlastphasen wirklich im akzeptablen Rahmen bleiben.

Modellbildung

Der Anbieter möchte sein System auf 10 gleichzeitige Nutzer beschränken und den Niedriglastfall mit dem Hochlastfall vergleichen. Sie müssen also die Erzeugung der Systemlast entsprechend beschränken und dann beide Lastfälle betrachten.

Toolumsetzung

Da die Modelländerungen nur minimal sind, erzeugen Sie eine Kopie des ersten Modelltyps mit der angebotenen *copy*-Funktion im Menü auf dem Listeneintrag 'bib1'. Nennen Sie den kopierten Modelltyp '**bib1_low**' (low steht für eine niedrige Last, low load).

Die Änderungen betreffen nur die Lasterzeugung, d.h. Sie müssen nur das CREATE-Statement in ACTIVITIES des Modelltyps (*open* auf dem ACTIVITIES-Symbol) ändern:

```
CREATE 1 PROCESS sitzung LIMIT 10 EVERY negexp (0.5);
```

Sichern und schließen Sie den Modelltyp, und erzeugen Sie die zugehörige Auswertungsbeschreibung ('**bib1_low_eva**'). Sie können dazu nicht die

Copy-Funktion verwenden, denn eine Auswertungsbeschreibung ist immer an einen speziellen Modelltyp gebunden.

Nachdem Sie das Auswertungsobjekt ('**rs_reaktion**') an der Komponente 'rechensystem' erzeugt haben, wählen Sie wie im ersten Beispiel die Ströme *THROUGHPUT*, *TURNAROUNDTIME* und *UTILIZATION*. Den Strom *POPULATION* können Sie mit einem Mausklick abschalten. Sichern und schließen Sie Auswertungsobjekt- und Auswertungsfenster.

Erzeugen Sie nun die Experimentbeschreibung ('**bib1_low_exp**').

> Mit der Beschränkung der Prozesse bei der Erzeugung verläßt das Modell die Klasse der von DOQ4 lösbaren Modelle. Allerdings brauchen wir die analytischen Verfahren noch nicht zu verlassen, denn der numerische Löser kann dieses Modell analysieren. Dazu betrachtet er die dem Modell unterliegende zeitkontinuierliche Markovkette. Zustandsraum und Generatormatrix werden aufgebaut, zur Ermittlung der stationären Zustandswahrscheinlichkeiten wird ein lineares Gleichungssystem gelöst.

Toolumsetzung

Aufgrund der Beschränkung bei den Ankünften können Sie den numerischen Löser *MARKOV* benutzen. Die Werte ACCURACY und CPUTIME können Sie unverändert übernehmen.

Fügen Sie die neue Auswertungsbeschreibung mit Hilfe des Menüs im Titelbalken ein, und füllen Sie (manuell oder automatisch) den BODY für das Experiment mit:

```
EVALUATE bib1_low_eva;
```

Sichern und schließen Sie den Editor und das Experiment-Fenster, und starten Sie die Transformation und HIT vom Environment aus (*transform & run* und *run*).

> Für Kenner von numerischen Lösungsverfahren auf Basis von Markovketten sind am Ende des Listings (Menüpunkt *show* und Unterpunkt *protocol*) einige interessante Informationen zur Generatormatrix und zum Ablauf der Lösung angegeben.

Toolumsetzung

Nun müssen Sie noch den Hochlastfall betrachten. Führen Sie dazu die folgenden Schritte aus:

- Kopieren Sie den Modelltyp 'bib1_low' nach '**bib1_high**'.
- Ändern Sie im CREATE-Statement die Rate von 0.5 auf 1.0.
- Erzeugen Sie die Auswertungsbeschreibung '**bib1_high_eva**' mit Auswertungsobjekt und Stromauswahl.

- Erzeugen Sie die Experimentbeschreibung '**bib1_high_exp**'. Dazu können Sie auch die *copy*-Funktion verwenden, anschließend im Experiment-Fenster die alte Evaluation löschen (*remove* auf dem Listeneintrag im Subfenster), die neue einfügen und den Experimentbody aktualisieren. Denken Sie daran, daß das EVALUATE-Statement nur angehängt wird, d.h. Sie müssen das ursprüngliche Statement manuell löschen, wenn Sie kopiert haben.

Nach einem *transform & run* können Sie den geforderten Vergleich der Werte durchführen. Die Resultate für die zwei Lastfälle sind in der folgenden Tabelle aufgeführt.

	Niedriglast	Hochlast
THROUGHPUT	0.50	0.78
TURNAROUNDTIME	3.22	9.00
UTILIZATION	0.62	0.98

Die Verlustrate ist nicht direkt in der Ergebnistabelle aufgeführt, läßt sich aber leicht aus der Differenz von Angebot (0.5 bzw. 1.0 Sitzungen pro Sekunde) und dem tatsächlich erzielten Durchsatz ermitteln.

Die Einschätzung der Ergebnisse ist nun subjektiv vom Dienstanbieter zu leisten. Er muß beurteilen, ob zu viele Anfragen abgewiesen werden und ob die Antwortzeiten akzeptabel sind.

2.3 Modell mit modifizierter Lastcharakteristik

Zu einigen Aspekten der bisherigen Modellannahmen seien nun genauere Informationen verfügbar geworden bzw. liegen zusätzliche Erfahrungswerte vor. Folgende Beobachtungen wurden gemacht: Die Folge der Benutzerankünfte weist Unregelmäßigkeiten auf, die sich in einem Variationskoeffizienten der Zwischenankunftszeit von 1.5 ausdrücken. Auch ist es sicher realistischer, für die Zahl der Anfrageoperationen je Sitzung eine diskrete Verteilung zu fordern. Wir nehmen eine Gleichverteilung über den ganzen Zahlen von 1 bis 19 an. Diese Version soll nur unter Hochlast betrachtet werden.

Toolumsetzung

Auch für diese Modellvariante können wir von einem schon bestehenden Modelltyp ausgehen. Kopieren Sie den Modelltyp 'bib1_high' nach '**bib1_verteilung**' (das Suffix steht für die unterschiedlichen Verteilungen).

Die Ankünfte haben sich im Vergleich zum vorigen Modell geändert. Die geforderte Ankunftsverteilung kann mit Hilfe der Cox-Verteilung modelliert werden. Dazu müssen Sie in ACTIVITIES des Modelltyps im CREATE-Statement eine Änderung vornehmen:

```
CREATE 1 PROCESS sitzung LIMIT 10 EVERY cox(1.0,1.5);
```

Cox-verteilte Zeiten ergeben sich als durch Wahrscheinlichkeiten gesteuerte Abfolge von Phasen, deren Dauern negativ-exponentiell verteilt sind. Die hier benutzte kurze Notation mit Mittelwert und Variationskoeffizient als Parameter legt die Cox-Verteilung noch nicht eindeutig fest. Das Tool wählt eine Verteilung aus, die mit möglichst wenig Phasen auskommt und den angegebenen Parametern entspricht.

Toolumsetzung

Die zweite Änderung betrifft die Verteilung für die Zahl der Anfrageoperationen, die Sie bitte im Body des Dienstes 'sitzung' eingeben:

```
anfrage (randint (1,19));
```

An diesem Punkt kann man sich bereits Gedanken um das einzusetzende Lösungsverfahren machen. Die benutzten Verteilungen machen den Einsatz eines analytischen Lösungsverfahrens unmöglich. In der Terminologie der Wartesysteme liegt ein GI/G/1/n-System mit processor sharing vor, dessen theoretische Behandlung komplex ist.

Als Ausweg bietet sich das Lösungsverfahren an, das nahezu keine prinzipiellen Beschränkungen aufweist, die Simulation.

Toolumsetzung

Beschäftigen wir uns mit der zum Modelltyp gehörenden Auswertungsbeschreibung, die Sie wieder neu erzeugen müssen ('**bib1_vert_eva**'). Für das Auswertungsobjekt '**rs_reaktion**' wählen Sie wie bisher die Ströme *THROUGHPUT*, *TURNAROUNDTIME* und *UTILIZATION*.

Da das Modell aufgrund der geänderten Verteilungen mit dem simulativen Löser analysiert werden soll, sollten Sie als Schätzer jetzt auch ein Konfidenzintervall anfordern. Selektieren Sie daher im Subfenster ESTIMATOR den Schätzer *CONFIDENCE LEVEL*. Damit erhalten Sie zusätzlich zum Mittelwertschätzer auch die Standardabweichung und ein Konfidenzintervall für den Mittelwert mit der voreingestellten Vertrauenswahrscheinlichkeit von 95%.

Die Bedeutung von Konfidenzintervallen für die Interpretation der Ergebnisse ergibt sich aus der Natur der Simulation. Eine Simulation kann, je nach Wahl und Initialisierung eines Zufallszahlengenerators, nur einen oder mehrere konkrete Abläufe des Modellverhaltens nachspielen, während als Ergebnis Aussagen über alle möglichen Abläufe gefragt sind. Statistische Verfahren helfen, diese Lücke auszufüllen. Sie basieren entweder auf mehreren Simulatorläufen (Ensemble-Analyse) oder, wie auch bei HIT, auf einem einzigen Simulatorlauf (Zeitreihenanalyse).

Ein Konfidenzintervall rechtfertigt die Aussage, daß der tatsächliche Mittelwert mit vorgegebener Vertrauenswahrscheinlichkeit in diesem Intervall liegt. Hier ist wiederum etwas Vorsicht geboten, denn auch die Grenzen des Konfidenzintervalls sind Schätzungen.

Zur Konfidenzintervallschätzung wird in HIT ein autoregressives Modell benutzt, dessen Auswertung nur eine gewisse Stationarität voraussetzt. Es handelt sich dabei um ein Online-Update-Verfahren, bei dem keine großen Mengen von Daten gespeichert werden müssen.

Modellbildung

Für die Simulation ist eine Stopbedingung unerläßlich. In HITGRAPHIC haben Sie die Möglichkeit, zusätzlich zur CPU- und Modellzeit (im Experiment-Fenster), die sozusagen als Notstop dienen, auch die Breite des Konfidenzintervalls als Abbruchkriterium zu verwenden.

Toolumsetzung

Die Breite des Konfidenzintervalls wird als simulative Stopbedingung (*simulation STOP*) angegeben. Führen Sie dazu auf dem Symbol ∞ im unteren Bereich des Auswertungsobjekt-Fensters die Funktion *simulation STOP* aus. Das Symbol wird zu einem Stop-Symbol, und es erscheint ein Bereich zur Eingabe der Bedingung. Schalten Sie die Bedingung um auf WIDTH, indem Sie *width* aus dem Popup Menü des Bereichs auswählen. Die Simulation soll erst dann enden, wenn die Konfidenzintervalle für alle angewählten Ströme eine maximale Breite von $\pm 5\%$ um den Mittelwert haben. Ändern Sie die Breite durch Editieren des Textfeldes, und geben Sie als aktuellen Wert 5 ein.

Sichern und schließen Sie das Auswertungsobjekt-Fenster, und schließen Sie das Auswertungsfenster.

Die Experimentbeschreibung '**bib1_vert_exp**' können Sie entweder kopieren oder neu erzeugen. Nehmen Sie dann die folgenden Änderungen vor:

- Als Löser verwenden Sie die Methode *SIMULATIVE*. Der Parameter CPUTIME wird auf 120 gesetzt, denn Sie sind sicher ungeduldig und wollen nicht viel länger als 2 Minuten warten. Für MODELTIME ist 28800 ein sinnvoller Wert, die Betrachtung des Modells über 8 Stunden (Modellzeit) sollte ausreichen.

  Die Simulation stoppt, wenn die Konfidenzintervalle den geforderten Breiten genügen (simulative Stopbedingung im Auswertungsobjekt-Fenster) oder die CPU-Zeit oder die Modellzeit abgelaufen sind.
- Fügen Sie die Auswertungsbeschreibung 'bib1_vert_eva' ein.
- Spezifizieren Sie den Experimentbody:

```
EVALUATE bib1_vert_eva;
```

Nun können Sie einen weiteren HIT-Lauf aus dem Environment starten und sich die Resultate der Simulation anzeigen lassen.

	Mean Value	Konf. Int. 95%
THROUGHPUT	0.75	± 2.50%
TURNAROUNDTIME	8.68	± 4.80%
UTILIZATION	0.93	± 1.86%

Diese Werte ergeben sich unter Nutzung des voreingestellten Initialwerts (seed) des in HIT benutzten Zufallzahlengenerators. Bei anderen Initialwerten ergeben sich verständlicherweise andere Ergebnisse.

Wenn Sie einen Blick auf die Ausgabetabelle werfen, finden Sie auf der ersten Seite die erreichte Modellzeit und die benötigte CPU-Zeit. Diese ist deutlich höher als bei den analytischen Lösungsverfahren. Damit bestätigt sich selbst in diesem kleinen Beispiel die Erfahrung, daß Simulation recht rechenintensiv ist.

Dies ist hier auch bedingt durch die Art des Modells. Als Folge der recht hohen Auslastung ergeben sich für die Antwortzeiten höhere Autokorrelationen. Diese führen letztendlich zu größeren Konfidenzintervallen. Da aber die Konfidenzintervallbreite als durchaus sinnvolles Abbruchkriterium eingesetzt wurde, muß die Vergrößerung der Konfidenzintervalle durch eine größere Stichprobe kompensiert werden. Dies bedeutet praktisch eine längere Laufzeit der Simulation.

Kapitel 3

Literaturrecherche: HIT-Modell 2

Mit dem groben Modell ließen sich erste interessante Ergebnisse erzielen. Die einfache Darstellung des Rechensystems läßt aber vermuten, daß gewisse Effekte, die sich aus den internen Abläufen im Rechensystem ergeben, vernachlässigt worden sind. Deshalb soll im folgenden das Rechensystem genauer betrachtet, d.h. verfeinert werden.

3.1 Verfeinerung des Rechensystems

Wir gehen von einer festen Zahl von 10 Terminals bzw. Verbindungen aus, die ständig besetzt sind. Zwischen den Anfragen macht sich jeder Benutzer Notizen, was im Mittel 20 Sekunden in Anspruch nimmt. Die Verfeinerung des Rechensystems soll so erfolgen, daß die als Hauptkomponenten identifizierten Teile (die CPU sowie ein sekundäres Speichersystem, auf dem große Datenmengen zugreifbar sind) aufgenommen werden. Eine Anfrage erfordert, daß im Durchschnitt 20 Blöcke von den Datenträgern zu lesen und im Mittel etwa 60000 Instruktionen auf der CPU durchzuführen sind. Das Lesen eines Blocks erfordert etwa 100 ms, die durchschnittliche Instruktion kann in etwa einer Mikrosekunde durchgeführt werden.

Modellbildung

Das zweite zu modellierende System besteht aus drei Schichten. Die oberste Schicht bildet das Bibliotheksystem 'bib2', die mittlere Schicht beinhaltet den Benutzer und das Rechensystem, die unterste Schicht ist die Verfeinerung des Rechensystems, bestehend aus CPU und Speichersystem.

HITGRAPHIC unterstützt unter anderem die bottom-up und auch die top-down Methode bei der Modellbildung. Wir wollen hier in top-down Richtung vorgehen.

Toolumsetzung

Betrachten wir zuerst die oberste Schicht, den Modelltyp '**bib2**', den Sie in der Modelliste erzeugen müssen. Öffnen Sie das Typ-Graphik-Fenster, um den Modelltyp zu spezifizieren.

Die Last des Modelltyps wird wie vorher durch die Aktivität '**sitzung**' beschrieben, die sich nun aus den Subaktivitäten '**anfrage**' und '**notieren**'

zusammensetzt. Erzeugen Sie den *service* mit den *used service*s. Der *used service* 'notieren' benötigt wieder einen formalen Parameter (amount : REAL). Der *used service* 'anfrage' bleibt hingegen parameterlos, da dieser Dienst an einen angebotenen Dienst gebunden wird, der erst in einer tiefer liegenden Schicht durch das Rechensystem realisiert wird.

Die Problembeschreibung geht davon aus, daß die Terminals ständig besetzt sind. Sie müssen demnach diese Annahme auch in der Prozeßbeschreibung im Body des Dienstes 'sitzung' berücksichtigen und die Aktivitäten in eine Endlosschleife einbetten:

```
LOOP
  anfrage;
  notieren (negexp (1/20));
END LOOP;
```

Die Prozeßerzeugung in ACTIVITIES des Modelltyps lautet gemäß Problemstellung:

```
CREATE 10 PROCESS sitzung;
```

> Wir erzeugen hier also eine feste Anzahl von Prozessen, die permanent laufen und sich alle nach demselben Muster verhalten. Dies bedeutet natürlich nicht, daß alle immer dasselbe machen. Dies wird schon durch die Benutzung von Zufallsvariablen klar, deren Realisierungen unterschiedlich ausfallen werden.
>
> Auch läßt sich eine Parallele zu einer anderen Modellwelt entwickeln, diesmal zu den Warteschlangennetzen. Dort spricht man von geschlossenen Systemen, weil dort wie hier zu den permanenten Prozessen keine weiteren „von außen" hinzutreten und kein Prozeß terminiert und damit „das System verläßt". Das endlos laufende Muster wird bei den Warteschlangennetzen geschlossene Kette genannt.

Toolumsetzung

Wenden Sie sich nun der Maschinenspezifikation zu. Der Benutzer selbst bildet eine der beiden Komponenten, die Sie als '**benutzer**' vom Typ *server* modellieren können. Da durch das Notieren ein reiner Zeitverbrauch modelliert werden soll, können Sie die Voreinstellung der Komponente übernehmen (d.h. es ist keine Spezifikation der Kontrollprozeduren notwendig, der Default beschreibt eine reine zeitliche Verzögerung).

> In Warteschlangennetzen wird dafür eine Station vom Typ Infinite Server eingesetzt.

Toolumsetzung

Die zweite Komponente ist das Rechensystem. Hier verwenden Sie einen selbstdefinierten Komponententyp, der erst noch erzeugt und spezifiziert werden muß. Wählen Sie im Environment aus dem Menü auf dem Titel der *self*

defined COMPONENT TYPES die Funktion *new component type*. Geben Sie dem Komponententyp den Namen '**retrieval2**'. Sie können nun im Typ-Graphik-Fenster des Modelltyps die zweite Komponente mit dem Namen '**rechensystem**' anlegen. Sie bietet noch keinen Dienst an, denn Sie müssen den Komponententyp noch spezifizieren.

Öffnen Sie Ihren selbstdefinierten Komponententyp vom Environment aus. Die Last in dieser Schicht besteht aus der Aktivität '**antwort**', die an die höherliegenden Schichten zur Nutzung angeboten werden soll. Erzeugen Sie den Service, und führen Sie die Funktion *provide* im Popup Menü auf dem Service aus. Als Zeichen dafür, daß dieser Dienst angeboten wird, d.h. daß er von höher liegenden Schichten genutzt werden kann, erscheint in der Graphik ein kleiner, nach links weisender Pfeil. Ein Service in einem Komponententyp ist entweder *provided* oder *internal*. Modelltypen kennen nur interne Dienste.

Die Subaktivitäten, also die *used services*, sind '**lesen**' und '**suchen**', jeweils versehen mit einem formalen Parameter (amount : REAL). Aus der Problemstellung können Sie direkt das Verhalten (also den Body für den Dienst 'antwort', der eine sequentielle Abfolge von 'lesen' und 'suchen' auszudrücken hat) herauslesen:

```
lesen (negexp (1/20));
suchen (negexp (1/60000));
```

> Auch hier benutzen wir für die Bedienanforderung, die letztendlich von einem *server* abgearbeitet wird, die für eine analytische Lösung günstige Annahme der Exponentialverteilung.

Toolumsetzung

Bei jeder Nutzung des Dienstes 'antwort' (initiiert durch eine darüberliegende Schicht) wird diese Anweisungsfolge ausgeführt.

Der Last wird auch hier eine Maschine, bestehend aus den unabhängigen Komponenten '**cpu**' und '**speichersystem**', gegenübergestellt, die die notwendigen Dienste bereitstellt. Erzeugen Sie diese Komponenten vom Typ *server*. Das dynamische Verhalten wird durch Kontrollprozeduren beschrieben.

Für die CPU, die mit der PS (processor sharing) Disziplin bedient, wählen Sie als Dispatch-Prozedur *shared*. Laut Problemstellung wird für das Ausführen einer Instruktion eine Mikrosekunde benötigt. Das ergibt eine Geschwindigkeit von 1 000 000 Instruktionen pro Sekunde. Geben Sie diesen Wert als aktuellen Parameter für die Geschwindigkeit ein:

```
LET speed := 1E6
```

Das Speichersystem arbeitet gemäß der Scheduling-Strategie FCFS (first come first served) und erfordert 100 ms für das Lesen eines Blocks. Das ergibt eine Geschwindigkeit von 10 Blöcken pro Sekunde. Sie wählen also als Schedule-Prozedur *fcfs* und als Dispatch-Prozedur *equal* mit dem aktuellen Parameter:

```
LET speed := 10
```

Für einen *server* mit Strategie FCFS ist es absolut notwendig, daß die Bedienanforderungen an diesen *server* alle negativ-exponentiell mit der gleichen Rate verteilt sind, damit das Modell in der Klasse der von DOQ4 exakt lösbaren Produktformmodelle bleibt. DOQ4 kann auch Modelle lösen, die diese Restriktion nicht in so drastischer Form erfüllen müssen und zudem Prioritätsstrategien enthalten dürfen. Die Lösung ist dann allerdings approximativ.

Toolumsetzung

Vergessen Sie nicht, die Bindungen zu setzen. Der *used service* 'lesen' wird an den *provided service* 'request' der Komponente 'speichersystem' gebunden, der *used service* 'suchen' an den *provided service* 'request' der Komponente 'cpu' (führen Sie die *set*-Operation an den entsprechenden Schnittpunkten aus). Überprüfen Sie zur Sicherheit Ihren Komponententyp mit der Abbildung 3.1.

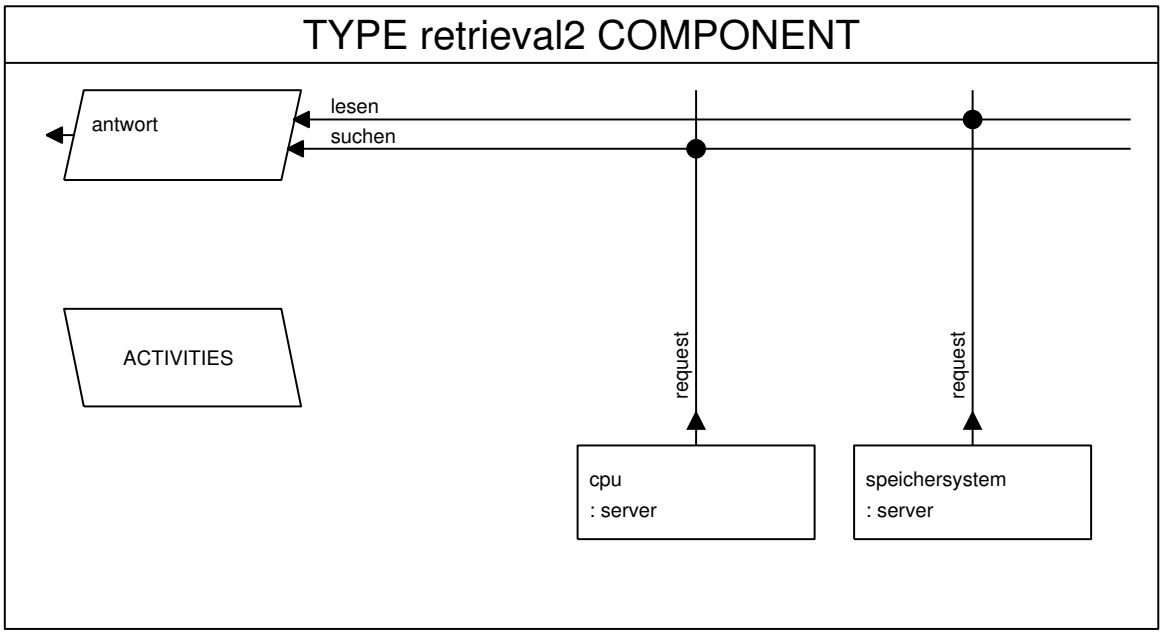

Abbildung 3.1: Komponententyp 'retrieval2' für HIT-Modell 2

Nachdem Sie den Komponententyp gesichert und verlassen haben, kehren Sie wieder zum Modelltyp 'bib2' zurück. Führen Sie hier ebenfalls eine *save*-Operation aus, um den Fensterinhalt auf den neuesten Stand zu bringen. Beim Neuzeichnen des Fensterinhaltes wird der aktuelle Datenbestand verwendet, d.h. der von Ihnen selbstdefinierte Komponententyp wird nun mit dem angebotenen Dienst 'antwort' gezeichnet. Sie können nun auch in diesem Graphikfenster die Bindung der Dienste vornehmen: 'anfrage' und 'antwort', 'notieren' und 'request'. Vergleichen Sie wieder Ihren Modelltyp mit der Abbildung 3.2.

Nach Sichern und Schließen sehen Sie sich vielleicht einmal das Übersichtsfenster Ihres Modelltyps an (*show survey* im Environment auf dem Modelltyp).

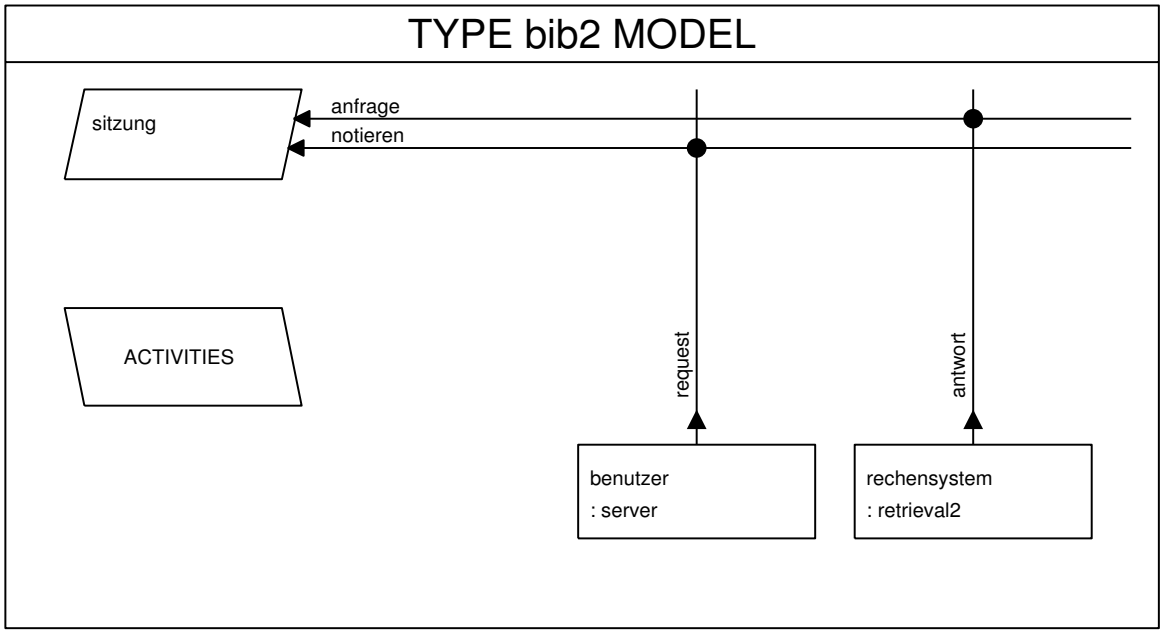

Abbildung 3.2: Typ des HIT-Modells 2

Sie sehen hier sehr gut den hierarchischen Aufbau anhand der Objektstruktur.

Wenden Sie sich dann der Auswertungsbeschreibung ('**bib2_eva**') zu, die Sie in der Auswertungsliste im Environment erzeugen müssen. Legen Sie für die Komponente Rechensystem das Auswertungsobjekt '**rs_reaktion**' an und für die Komponente Speichersystem das Objekt '**ss_reaktion**'.

Sie sollten für 'rs_reaktion' die Ströme *THROUGHPUT* und *TURNAROUNDTIME* anfordern, für 'ss_reaktion' *TURNAROUNDTIME* und *UTILIZATION*.

Für die zugehörige Experimentbeschreibung '**bib2_exp**' wählen Sie den analytischen Löser *DOQ4*, fügen die Auswertungsbeschreibung in die Liste ein und editieren den Body für das Experiment (`EVALUATE bib2_eva;`).

> DOQ4 benutzt für dieses Modell einen Algorithmus, wie er auch zur Lösung von Warteschlangennetzen mit Produktform eingesetzt wird. Damit zeigt sich das Phänomen, daß unterschiedliche Modellbeschreibungsparadigmen durchaus zu derselben Lösungsproblematik führen können. Man kann zu jedem „Produktformmodell“ von HIT ein entsprechendes Warteschlangennetz mit denselben Ergebnissen konstruieren. Dies mag auch für Kenner von Warteschlangennetzen nicht so offensichtlich sein, deshalb ein kleiner Tip: *server* werden natürlich zu Stationen, als Klasse wird jede Aufrufmöglichkeit, d.h. jeder Aufrufstack vom Erzeugungspunkt der Prozesse bis zum request eines *server* gesehen.
>
> Aber urteilen Sie selbst: Ist nicht die Modellbeschreibung in HIT dem Problem angemessener und somit der Weg über ein Warteschlangennetz „indirekter“?

Toolumsetzung

Starten Sie mit *transform & run* die Transformation und HIT. Nach einem hoffentlich erfolgreichen Analyselauf können Sie die Ergebnisse mit denen der Tabelle vergleichen.

Auswertungsobjekt	rs_reaktion	ss_reaktion
THROUGHPUT	0.39	–
TURNAROUNDTIME	5.51	5.45
UTILIZATION	–	0.78

3.2 Das verfeinerte System unter Hochlast

Von besonderem Interesse kann es für den Dienstanbieter sein, herauszufinden, wo ein leistungsmäßiger Engpaß im System entstehen kann. Experimentell läßt sich dies bestimmen, indem man die Last erheblich steigert und z.B. von 1000 angeschlossenen Terminals ausgeht.

Toolumsetzung

Für den Hochlastfall können wir von einer Kopie des Modelltyps 'bib2' ausgehen. Führen Sie bitte folgende Schritte aus:

- Kopieren Sie den Modelltyp 'bib2' nach '**bib2_high**'.
- Ändern Sie im CREATE-Statement die Anzahl der zu erzeugenden Prozesse von 10 auf 1000.
- Erzeugen Sie die Auswertungsbeschreibung '**bib2_high_eva**' mit den Auswertungsobjekten '**rs_reaktion**' und '**ss_reaktion**' und entsprechender Stromauswahl.
- Erzeugen Sie die Experimentbeschreibung '**bib2_high_exp**'. Dazu können Sie wie auch im ersten Beispiel die *copy*-Funktion verwenden und anschließend im Experiment-Fenster die alte Auswertung löschen, die neue einfügen und den Experimentbody aktualisieren.
- Wählen Sie als Lösungsmethode den Löser *LIN2*.

  Der Löser LIN2 ist bis auf kleine Ausnahmen für alle (separablen) Produktformmodelle einsetzbar. Er liefert approximative Ergebnisse, wobei die Approximationsgüte mit der Zahl der permanent laufenden Prozesse steigt. Dadurch ist er für das vorliegende Modell gut geeignet. Zudem ergibt sich eine Laufzeitreduzierung gegenüber dem exakten Löser DOQ4, insbesondere wenn mehrere nach verschiedenen Mustern ablaufende Prozesse (mittels mehrerer CREATE-Statements) erzeugt werden.

Toolumsetzung

Starten Sie HIT mit *transform & run*, und sehen Sie sich die Resultate an.

Auswertungsobjekt	rs_reaktion	ss_reaktion
THROUGHPUT	0.50	–
TURNAROUNDTIME	1980.00	1979.94
UTILIZATION	–	1.00

In den Ergebnissen werden Sie feststellen, daß die Auslastung von 'speichersystem' auf praktisch 1.0 angestiegen ist und der Hauptanteil der Prozesse an dieser Komponente verweilt (dort wartet). Sie ist also der Flaschenhals in dem Modell.

> Wenn Sie genügend Zeit haben, dann machen Sie die Probe aufs Exempel für das Lösungsverfahren LIN2. Kopieren Sie das Experiment und lösen Sie noch einmal mit DOQ4. Ein Ergebnisvergleich sollte allerhöchstens geringe Abweichungen ergeben, ein signifikanter Laufzeitgewinn sollte sich allerdings ebenfalls nicht einstellen, da das Modell nur eine Art von Prozessen kennt und daher das Potential zur Effizienzsteigerung (durch den approximativen Löser LIN2) gering ist.

Kapitel 4

Literaturrecherche: HIT-Modell 3

Die bisherigen Ergebnisse haben gezeigt, daß der Hauptengpaß des Systems beim sekundären Speichersystem liegt. Da die Gesamtleistungscharakteristik des Modells deshalb recht sensitiv gegenüber der Darstellung des sekundären Systems ist, scheint hier eine weitere Verfeinerung sinnvoll. Dabei setzen wir beim ersten Beispiel des vorigen Kapitels an.

4.1 Verfeinerung des sekundären Speichersystems

Aus Kompaktheits- und Verfügbarkeitsgründen benutzt das Literaturrecherchesystem CD-ROMs. Dafür stehen 3 Laufwerke mit Blocklesezeiten von 50 ms sowie ein automatisiertes Ladesystem, das die Verbindung zwischen dem CD-ROM-Archiv und den Laufwerken herstellt, zur Verfügung. Das Entfernen und Laden einer CD-ROM kann das System in etwa einer Sekunde durchführen, allerdings kann jeweils nur ein Laufwerk zu jeder Zeit bestückt werden. Das Laden einer CD-ROM ist nur in 10% aller Fälle nötig, in den anderen Fällen befindet sie sich bereits in einem der freien Laufwerke. Ein Laufwerk muß nun, bevor es benutzt werden kann, reserviert werden. Bei den Anfragen hat sich eine bestimmte Art von Ablauf als typisch herausgestellt.

Primär wird eine CD-ROM benötigt, die die wesentlichen Informationen zum Interessengebiet des Benutzers enthält. Die durchgeführte Operation erfordert das Lesen von ca. 20 Blöcken und die Durchführung von ca. 60000 Instruktionen. In etwa 50% der Anfragen wird dabei auf eine zweite CD-ROM verwiesen, was wiederum das Lesen von ca. 20 Blöcken und etwa 60000 Instruktionen erfordert. In etwa 50% aller Fälle ist nach einer Operation die Anfrage beendet, anderenfalls wiederholt sich das Muster mit einer neuen Operation auf der primären CD-ROM des Interessengebiets.

Modellbildung

Für die Modellierung dieser Verfeinerung gehen Sie in bottom-up Reihenfolge vor und erzeugen einen Komponententyp, der die verfeinerte Modellierung des Speichersystems enthält. Dieser Komponententyp kann dann in der darüberliegenden Schicht (im Rechensystem) eingesetzt werden.

Toolumsetzung

Erzeugen Sie im Environment den neuen Komponententyp '**cd_rom3**', der die Verfeinerung des Speichersystems modellieren wird, und öffnen Sie das Graphikfenster.

Wenden Sie sich zuerst der Maschinenkomponente des neuen Typs zu. Die Maschine besteht aus einem Ladegerät und drei Laufwerken. Das Ladegerät '**lader**' wird als *server* mit der Scheduling-Prozedur *random* modelliert.

Für die Laufwerke ('**laufwerke**') verwenden Sie ebenfalls einen *server* mit der Scheduling-Prozedur *random*, die Dispatch-Prozedur ist *equal* mit einem aktuellen Wert für den Parameter *speed* von 20 (die Blocklesezeit ist mit 50 ms angegeben; das entspricht einer Geschwindigkeit von 20 Blöcken pro Sekunde).

Um das Belegen und Freigeben von Laufwerken zu verwalten, steht mit dem Komponententyp *counter* eine Komponente zur Verfügung, die das Resourcenhandling unterstützt.

> Komponenten vom Typ *counter* beinhalten einen Vektor von ganzzahligen Zustandsvariablen. Über den angebotenen Dienst 'change' kann ein Änderungsvektor für die Zustandsvariablen angegeben werden. Die Änderung kann nur durchgeführt werden, wenn der Zustandsvektor nach Addition des Änderungsvektors noch in den erlaubten Grenzen bleibt, die als Aktualparameter der Komponente anzugeben sind. Anderenfalls muß der Prozess warten, bis die Änderung möglich wird.
>
> Komponenten von Typ *counter* erlauben dadurch die Synchronisation von Prozessen. Sie sind damit Semaphoren oder Tokenpools ähnlich, für die in HIT ebenfalls vordefinierte Komponententypen existieren. Die Wahl von *counter* erlaubt hier allerdings noch den Einsatz des analytisch-numerischen Verfahrens bei Durchführung der Analyse. Die Möglichkeit, Synchronisationsaspekte einzubringen und trotzdem eine analytische Lösung zu erhalten, ist eine besondere Stärke der analytisch-numerischen Verfahren.

Toolumsetzung

Nennen Sie die Komponente vom Typ *counter* '**lw_reservierung**'. Der Counter besitzt drei formale Parameter vom Typ ARRAY OF INTEGER, die die minimale, die maximale und die initiale Belegung festlegen und die Sie mit aktuellen Werten belegen müssen. Da Sie nur die Belegung einer Resource betrachten müssen, reicht ein degeneriertes (einelementiges) Array aus. Wählen Sie aus dem Popup Menü auf der Komponente den Eintrag *actual parameters*, und geben Sie im unteren Subeditor die aktuellen Werte ein:

```
[0], [3], [3]
```

Der Counter bietet den Dienst *change* mit den formalen Parametern

```
amount : ARRAY OF INTEGER;
prio   : INTEGER DEFAULT 32767
```

an. Sie können sich diese Parameter mit der Funktion *show formal parameters* anzeigen lassen (Menü auf dem angebotenen Dienst *change*).

> Wie der Parameter prio, den Sie in diesem Beispiel nicht benötigen, bereits andeutet, sind auch Prioritätsregelungen bei der Auswahl der Prozesse möglich, die eine Änderung durchführen können. Der voreingestellte Wert 32767 steht für die „niedrigste“ Priorität.

Toolumsetzung

Betrachten Sie nun die Last der neuen CD-Komponente. Sie geht aus der Problemstellung hervor, wenn Sie sich bewußt machen, welche Dienste diese Komponente dem Rechensystem anbieten muß. Sie muß die Zugriffe auf die drei Laufwerke verwalten (‘**belegen_laden**’ und ‘**freigeben**’) und einen Zugriff auf eine eingelegte CD bieten (‘**lesen**’). Erzeugen Sie diese drei Dienste mittels der *new service*-Funktion.

Der Dienst ‘belegen_laden’ belegt ein Laufwerk und lädt in 10% der Fälle eine CD. Sie benötigen also die *used service*s ‘**allocate**’ und ‘**laden**’. Der genutzte Dienst ‘allocate’ wird an die Komponente ‘lw_reservierung’ gebunden. Seine formalen Parameter müssen zu den Parametern des angebotenen Dienstes ‘change’ kompatibel sein:

```
amount : ARRAY OF INTEGER;
prio   : INTEGER DEFAULT 32767
```

Sie können sich das Eingeben der Parameter ersparen, indem Sie statt der Funktion *set* die Funktion *set with transfer* für das Setzen der Bindung verwenden. Die formalen Parameter werden dann automatisch (natürlich nach einer Bestätigung) übertragen.

Der genutzte Dienst ‘laden’ wird an den Lader gebunden und hat daher ebenso einen formalen Parameter:

```
amount : REAL
```

Das Verhalten des Dienstes ‘belegen_laden’ wird im entsprechenden Body eingegeben (*open* auf dem Menü des Service-Symbols):

```
allocate ([-1]);
IF draw (0.1)
THEN
  laden (negexp (1/1));
END IF;
```

Der Service ‘lesen’ nutzt den Dienst ‘**lesen**’ mit dem formalen Parameter:

```
amount : REAL
```

Der *used service* wird an den angebotenen Dienst der Komponente 'laufwerke' gebunden. Im Mittel werden 20 Blöcke gelesen, das Prozeßverhalten sieht damit folgendermaßen aus:

```
lesen (negexp (1/20));
```

Der Service 'freigeben' nutzt den Dienst '**release**', der die gleichen formalen Parameter wie 'allocate' hat und ebenfalls an die Komponente 'lw_reservierung' gebunden wird:

```
amount : ARRAY OF INTEGER;
prio   : INTEGER DEFAULT 32767
```

Für das Prozeßverhalten gilt:

```
release ([1]);
```

Damit die Dienste des Komponententyps 'cd_rom3' auch von übergeordneten Schichten genutzt werden können, führen Sie bitte die Funktion *provide* für jeden der drei Dienste aus. Vergleichen Sie zur Sicherheit Ihren Komponententyp mit der Abbildung 4.1.

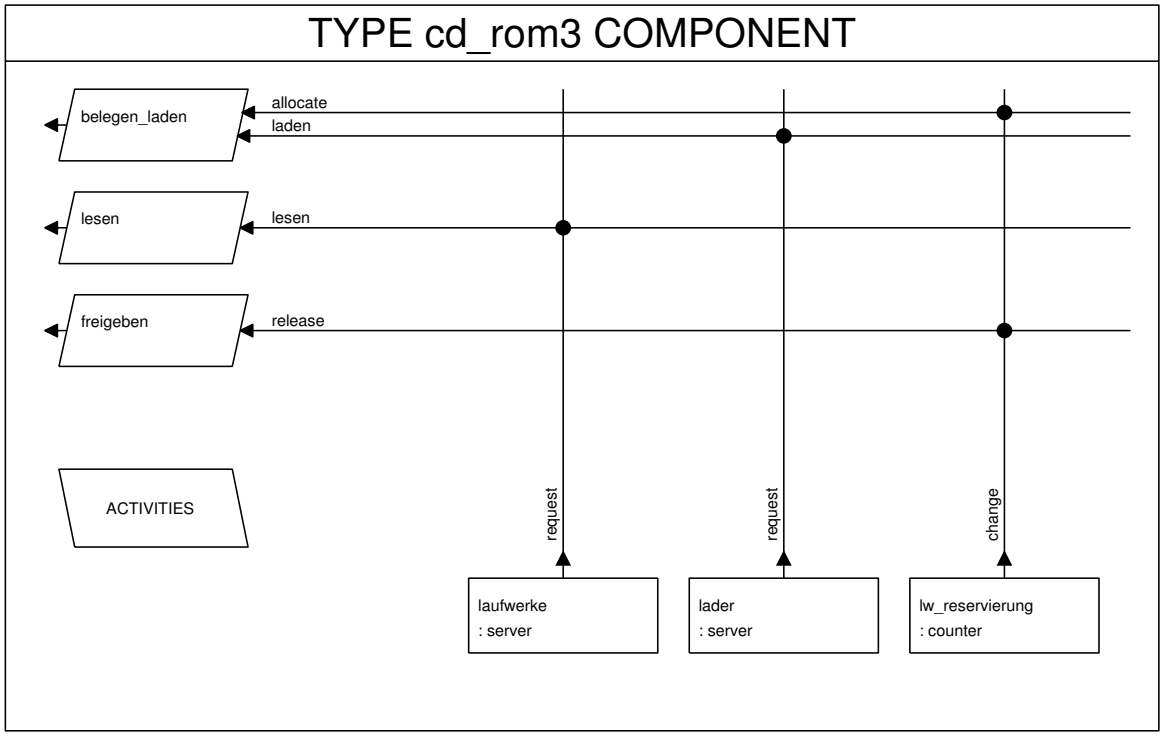

Abbildung 4.1: Komponententyp 'cd_rom3' für HIT-Modell 3

Mit dem Sichern und Schließen des Graphikfensters haben Sie einen neuen Komponententyp erzeugt, dessen angebotene Dienste nun in höheren Schichten genutzt werden können.

Modellbildung

Wir wollen den soeben erzeugten Komponententyp in unserem Rechensystem nutzen. Das Rechensystem war vom Typ 'retrieval2'. Sie können nun von diesem Typ ausgehend einen neuen Typ spezifizieren.

Toolumsetzung

Kopieren Sie den Komponententyp im Environment nach '**retrieval3**', und nehmen Sie die notwendigen Modifikationen im Graphikfenster vor:

- Löschen Sie die Komponente 'speichersystem' vom Type *server* mit der *delete*-Funktion im Menü auf dem Komponentensymbol.
- Erzeugen Sie eine neue Komponente '**cd_system**' vom Typ 'cd_rom3'. Die Komponente sollte, wenn Sie bis hierher der Vorlage korrekt gefolgt sind, die drei Dienste 'freigeben', 'lesen' und 'belegen_laden' anbieten.
- Der Dienst 'antwort' benötigt zwei weitere *used services*, '**belegen**' und '**freigeben**'.
- Löschen Sie die formalen Parameter für den schon bestehenden *used service* 'lesen'.
- Binden Sie die *used services* des Dienstes 'antwort' an die *provided services* der Komponente 'cd_system', den Namen entsprechend.
- Editieren Sie den Body des Dienstes 'antwort' gemäß den Anforderungen in der Problemstellung:

```
belegen;
LOOP
  lesen;
  suchen (negexp (1/60000));
  BRANCH
    PROB 0.5 : belegen;
               lesen;
               suchen (negexp (1/60000));
               freigeben;
  END BRANCH;
END LOOP UNTIL draw (0.5);
freigeben;
```

Nachdem Sie den Komponententyp modifiziert haben, können Sie Ihre Graphik mit Abbildung 4.2 vergleichen. Sichern und schließen Sie anschließend das Graphikfenster.

Sie können nun den neuen Modelltyp erzeugen. Als Vorlage verwenden Sie den Modelltyp 'bib2'. Erzeugen Sie eine Kopie '**bib3**' im Environment, und öffnen Sie den neuen Modelltyp. Die Komponente 'rechensystem' soll vom neu erzeugten Typ 'retrieval3' sein. Sie müssen also die alte Komponente im Graphikfenster löschen und eine neue Komponente anlegen. Vergessen Sie nicht, die Bindung wieder herzustellen.

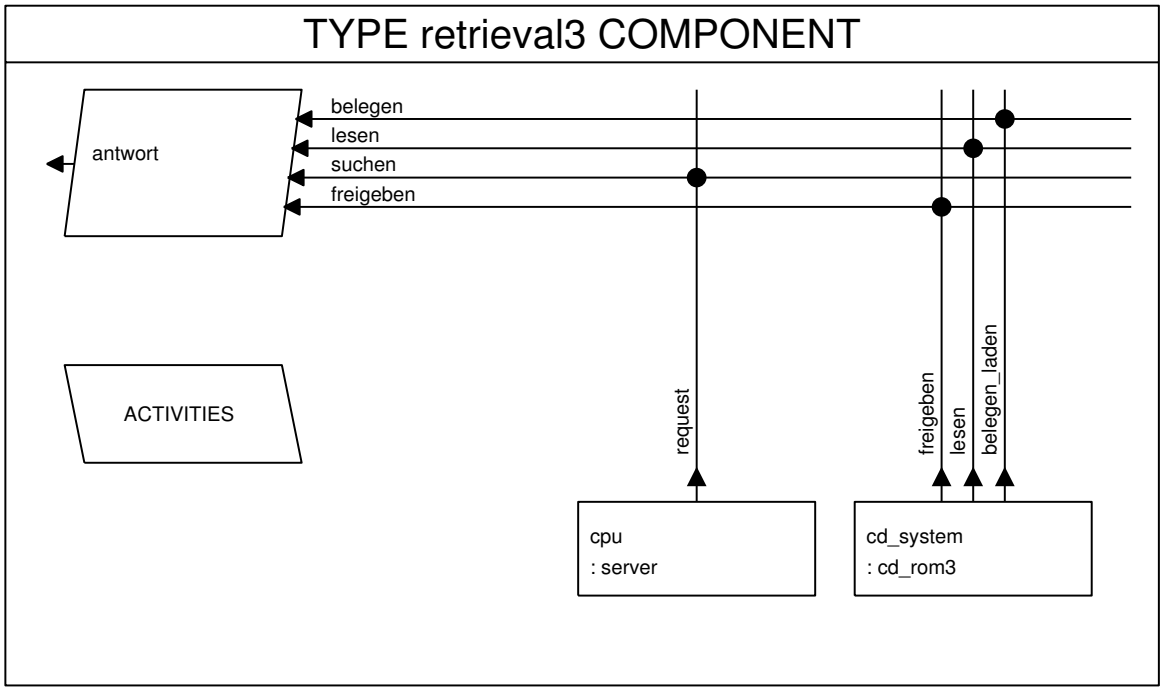

Abbildung 4.2: Komponententyp 'retrieval3' für HIT-Modell 3

Wie bereits erwähnt, zielen wir auf den Einsatz des analytisch-numerischen Verfahrens. Neben der Stärke des Einbringens von Synchronisationsaspekten hat das Verfahren die Schwäche, zur Zustandsraumexplosion zu neigen. Diese führt dann zu unerfüllbaren Speicheranforderungen oder unakzeptabel hohen Laufzeiten. Um nicht sofort in diese „Falle" zu laufen, sollten Sie versuchen, das Modell zunächst „klein" zu halten. Eine Möglichkeit besteht darin, mit kleinen Prozeßzahlen zu beginnen und diese langsam zu steigern. Ob die letztendlich angestrebte Modellgröße noch praktisch analysierbar ist, wird sich dabei herausstellen.

Selbst wenn das Modell letztendlich (praktisch) nicht analytisch-numerisch behandelbar ist, kann man wegen der vom Lösungsverfahren unabhängigen Modellbeschreibung von HIT leicht auf ein anderes Verfahren wechseln, hier z.B. auf die Simulation.

Toolumsetzung

Wir wollen uns an dieses recht detaillierte Modell bei der späteren Analyse vorsichtig herantasten und die Anzahl der Benutzer nur Schritt für Schritt steigern. Daher verwenden Sie bitte einen formalen Modellparameter. Führen Sie die Funktion *formal parameters* im Menü des Titelbalkens des Graphikfensters aus, und geben Sie einen formalen Parameter an:

```
anzahl_benutzer : INTEGER
```

Wenn Sie die ACTIVITIES des Modelltyps öffnen (*open* auf dem ACTIVITIES-Symbol), wird dieser Parameter im linken oberen Subeditor angezeigt. Sie verwenden ihn im CREATE-Statement:

```
CREATE anzahl_benutzer PROCESS sitzung;
```

Später können Sie dann im Experiment mit dem aktuellen Parameter die Anzahl der parallelen Sitzungen steuern.

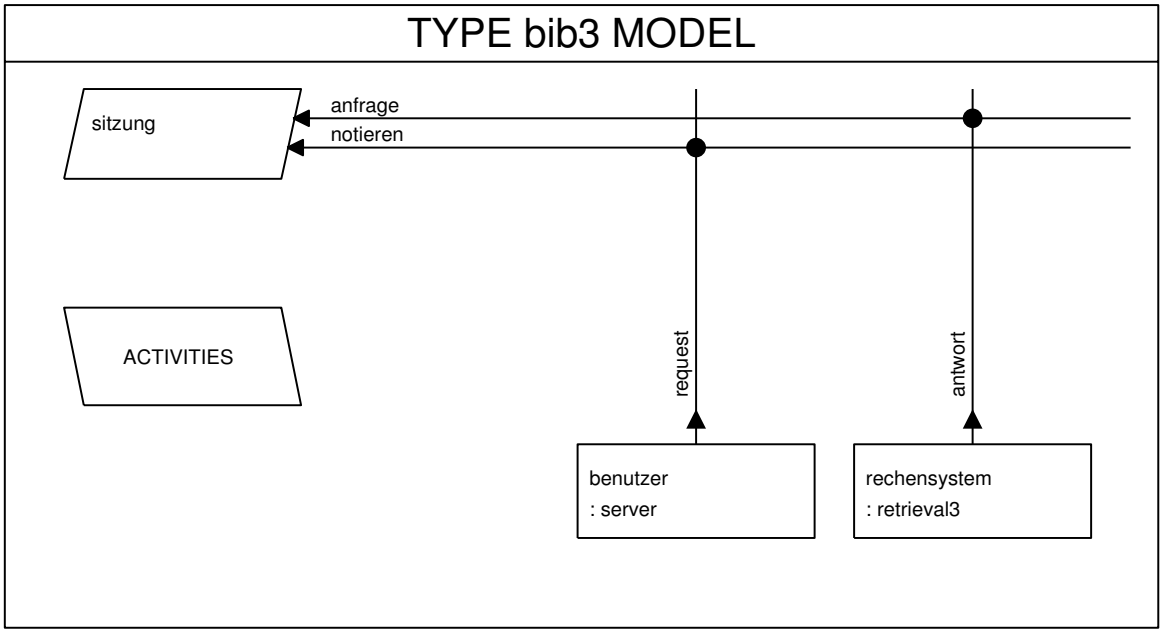

Abbildung 4.3: Typ des HIT-Modells 3

Weitere Änderungen sind nicht notwendig. Vergleichen Sie zum Abschluß Ihren Modelltyp mit Abbildung 4.3. Verlassen Sie das Fenster ordnungsgemäß, und werfen Sie auch einen Blick auf das Survey-Fenster Ihres Modelltyps (vgl. Abbildung 4.4).

Wir wenden uns nun der Auswertungsbeschreibung zu. Erzeugen und öffnen Sie '**bib3_eva**'. Der Betreiber des Bibliotheksystem ist nach wie vor an der Antwortzeit auf eine Anfrage interessiert. Erzeugen Sie also ein Auswertungsobjekt '**rs_reaktion**' mit der Funktion *new evaluation object* im Menü auf der Komponente 'rechensystem'. Selektieren Sie im Auswertungsobjekt-Fenster im Unterfenster STREAM den Strom *TURNAROUNDTIME*.

Sichern und schließen Sie alle die Auswertung betreffenden Fenster, und erzeugen Sie eine Experimentbeschreibung.

Da wir Schritt für Schritt vorgehen wollen, nennen Sie das erste Experiment '**bib3_exp1**'. Wählen Sie als Methode den numerischen Löser *MARKOV*, und fügen Sie die Auswertungsbeschreibung ein. Den Experimentbody können Sie wieder automatisch füllen lassen. Da das Modell einen formalen Parameter besitzt, müssen Sie nun den aktuellen Wert eingeben. Dazu führen Sie die Funktion *actual parameters* im Menü auf der automatisch eingefügten Auswertung aus. Setzen Sie den aktuellen Wert im unteren Subeditor auf '1' (bitte nur 1 eingeben).

Schließen Sie das Fenster ordnungsgemäß, und starten Sie die Analyse mit *transform & run* vom Environment aus.

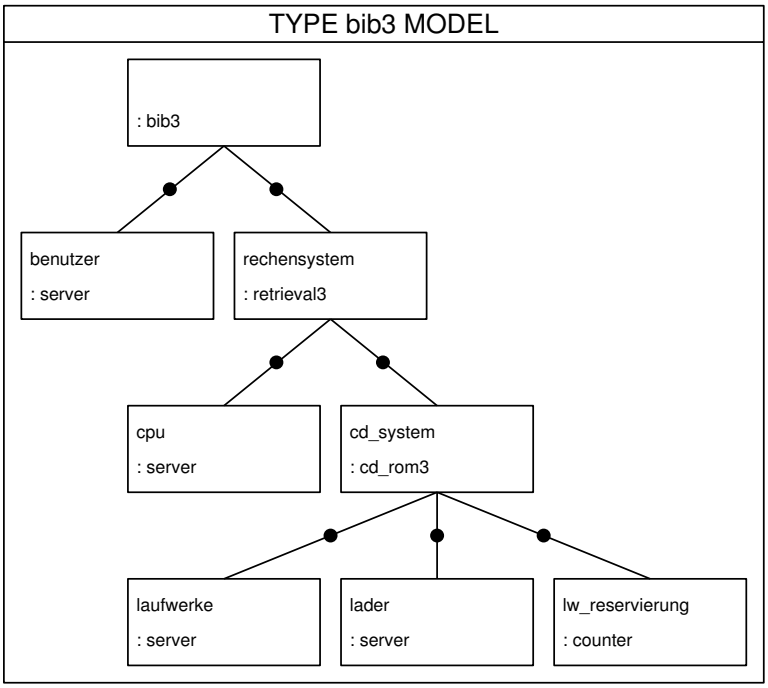

Abbildung 4.4: Objektstruktur für HIT-Modell 3

Nachdem der HIT-Lauf hoffentlich fehlerfrei beendet ist, können Sie eine weitere Experimentbeschreibung erzeugen. Kopieren Sie im Environment 'bib3_exp1' nach '**bib3_exp2**', und setzen Sie den aktuellen Wert für den formalen Modell-Parameter von '1' auf '2'.

Stellen Sie eine weitere Kopie her ('**bib3_exp3**'), und erhöhen Sie den aktuellen Wert auf '3'.

Nachdem Sie diese drei Experimentläufe durchgeführt haben, können Sie die Antwortzeiten vergleichen.

anzahl_benutzer	1	2	3
TURNAROUNDTIME	3.38	3.79	–

Beim dritten Lauf hat das HIT-System einen Deadlock aufgedeckt, es ist keine Ergebnisdatei verfügbar.

> Das Auftreten eines Deadlocks bedeutet, daß im Ablauf des Modells ein Zustand eintreten kann, in dem eine gewisse Menge von Prozessen keine Vorschritte mehr machen kann. Das analytisch-numerische Verfahren ist in der Lage, solche Situationen zu erkennen, falls alle Prozesse betroffen sind, und liefert eine entsprechende Fehlermeldung.
>
> Demgegenüber wäre es bei einer Simulation wegen der stochastischen Natur des Modells nicht sicher, daß man in die Deadlocksituation gerät. Diese würde dann, wenn alle Prozesse betroffen sind, natürlich gemeldet. Schwieriger verhielte es sich bei partiellen Deadlocks, in die nur ein Teil der Prozesse verwickelt ist.

Wir sind hier auf ein Problem gestoßen, das eigentlich nicht mehr in den Bereich der quantitativen Leistungsbewertung fällt. Das modellierte System funktioniert nicht korrekt, es erfüllt wesentliche qualitative Eigenschaften nicht. Deshalb muß es zunächst mit geeigneten Methodiken und Techniken für qualitative Probleme untersucht werden. Damit böte sich hier der Übergang zum Nachbar-Praktikum „Hands on QPN“ [1] an. Dort wird genau das hier auftretende qualitative Problem mit dem Tool QPN untersucht, dessen Modellwelt auf einer Kombination von Petrinetzen und Warteschlangennetzen basiert.

Vielleicht möchten Sie aber das Arbeiten mit HIT fortsetzen. Deshalb seien hier zwei Lösungsansätze für das Problem genannt. Zum einen können bei jeder Anfrage gleich beide Laufwerke belegt werden, zum anderen könnte, falls die Anfrage ein zweites Laufwerk benötigt, das erste erst wieder freigegeben werden, bevor dann beide benötigten Laufwerke gleichzeitig belegt werden.

Damit stellt sich sofort wieder ein Leistungsbewertungsproblem: Welches ist die leistungsfähigere Alternative? Bei der ersten Lösung wird man eine reduzierte Auslastung der Laufwerke erwarten, da sie auch belegt werden müssen, ohne tatsächlich genutzt zu werden. Bei der zweiten kann das Freigeben des ersten Laufwerks bewirken, daß nach der Wiederanforderung u.U. die CD-ROM physikalisch wieder eingeladen werden muß, was eine zeitaufwendige Operation ist und eine Verdoppelung von Arbeit bedeutet. Eine Modellierung mit HIT kann beantworten, welche der zwei vorgestellten Alternativen schwerere Leistungseinbußen zur Folge hat.

4.2 Erste Alternative des Speichersystems

Um sicherzustellen, daß ein Prozeß gegebenenfalls auch das zweite Laufwerk belegen kann, werden immer zwei Laufwerke belegt.

Toolumsetzung

Erzeugen Sie eine Alternative zum bestehenden Speichersystem ‘cd_rom3’ durch Kopieren im Environment, und nennen Sie den neuen Komponententyp ‘**cd_rom3_a**’. Wir wollen den zusätzlichen Dienst ‘doppel_belegen’ anbieten und die Aktionen ‘belegen’ und ‘laden’ getrennt betrachten, denn ‘laden’ soll nur bei Bedarf ausgeführt werden.

Führen Sie die folgenden Aktionen im Graphikfenster aus:

- Kopieren Sie zuerst den Dienst ‘belegen_laden’ nach ‘**doppel_belegen**’ und nach ‘**belegen**’.
- Benennen Sie den Dienst ‘belegen_laden’ in ‘**laden**’ um.
- Löschen Sie den *used service* ‘laden’ der Dienste ‘doppel_belegen’ und ‘belegen’ und den *used service* ‘allocate’ des Dienstes ‘laden’.

- Modifizieren Sie die Bodies von 'doppel_belegen'

```
allocate ([-2]);
```

- von 'belegen'

```
allocate ([-1]);
```

- und von 'laden'

```
IF draw (0.1)
THEN
  laden (negexp (1/1));
END IF;
```

- Verbinden Sie die zwei *used services* 'allocate' mit dem von der Komponente 'lw_reservierung' angebotenen Dienst.

Auf der Maschinenseite verändern Sie bitte die Kontrollprozeduren an den Komponenten 'laufwerke' und 'lw_reservierung'. Die Scheduling-Prozedur von 'laufwerke' wird auf *type default* zurückgesetzt, und 'lw_reservierung' arbeitet in dieser Version mit der Scheduling-Prozedur *cprio*.

> Der Wechsel auf die Scheduling-Prozedur *cprio* ist hier nötig, da jetzt der Zustandsvektor der Komponente vom Typ *counter* um betragsmäßig unterschiedliche Werte geändert werden soll. Dies ist beim numerischen Verfahren nur mit *cprio* durchführbar.

Toolumsetzung

Die neue Variante des Speichersystems (vgl. Abbildung 4.5) können Sie nun in die Rechensystem-Komponente einbringen. Kopieren Sie im Environment 'retrieval3' nach '**retrieval3_a1**', und vertauschen Sie den Komponententyp der Komponente 'cd_system' durch Löschen und erneutes Anlegen der Komponente (nun jedoch vom Typ 'cd_rom3_a').

Da das Belegen und Laden von CDs separat gehandhabt wird, benötigen Sie noch einen weiteren *used service* am Dienst 'antwort'. Erzeugen Sie den *used service* '**laden**'. Benennen Sie bitte den *used service* 'belegen' in '**doppel_belegen**' um (das dient nur zum besseren Verständnis).

Führen Sie die Bindungen durch. Der vom 'cd_system' angebotene Dienst 'belegen' bleibt ungebunden. Wir wollen ihn in einer späteren Version noch für eine Einzelbelegung verwenden.

Der letzte Schritt für den Komponententyp 'retrieval3_a1' betrifft das Prozeßmuster des Dienstes 'antwort'. Öffnen Sie den Editor, und ändern Sie den Inhalt des Bodies gemäß der Problemstellung:

```
doppel_belegen;
laden;
LOOP
```

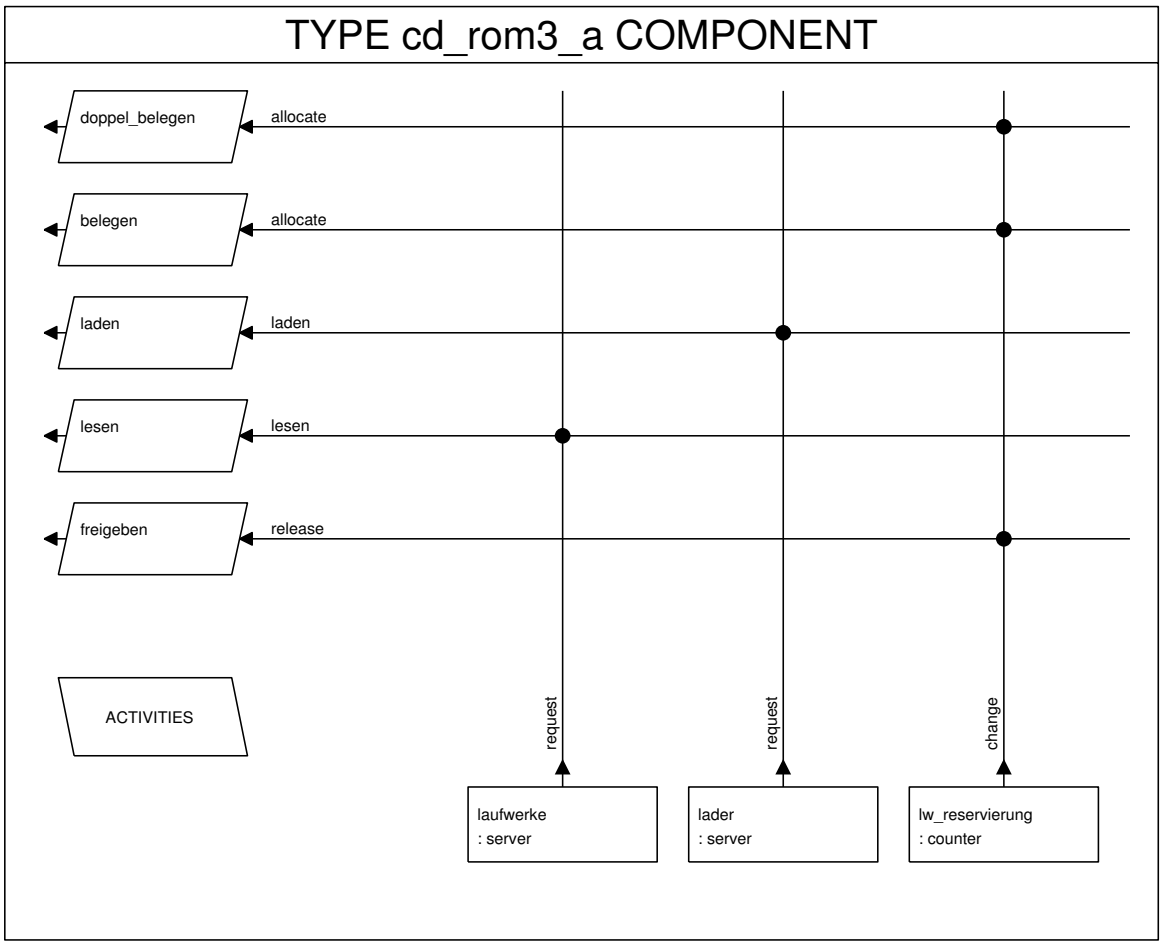

Abbildung 4.5: Variante zum Komponententyp für das CD-ROM-System

```
  lesen;
  suchen (negexp (1/60000));
  BRANCH
    PROB 0.5 : laden;
               lesen;
               suchen (negexp (1/60000));
  END BRANCH;
END LOOP UNTIL draw (0.5);
freigeben;
freigeben;
```

Wenden wir uns nun der höchsten Modellschicht zu, dem Modelltyp '**bib3_a1**', den Sie wieder mit Hilfe der *copy*-Funktion im Environment als Kopie von 'bib3' erzeugen. Tauschen Sie den Typ der Komponente 'rechensystem' mit dem von Ihnen soeben erzeugten Typ 'retrieval3_a1' aus, und stellen Sie die Bindung zwischen 'anfrage' und 'antwort' wieder her.

Erzeugen Sie die Auswertungsbeschreibung '**bib3_a1_eva**' mit dem Auswertungsobjekt '**rs_reaktion**' an der Komponente 'rechensystem', und fordern Sie die *TURNAROUNDTIME* an.

Legen Sie eine neue Experimentbeschreibung '**bib3_a1_exp**' an, wählen Sie die Lösungsmethode *MARKOV*, und fügen Sie die Auswertungsbeschreibung in die Auswertungsliste ein.

Wir wollen den formalen Modellparameter nutzen und eine Experimentserie starten, um die Ergebnisse für die Parameterbelegungen in einem HIT-Lauf zu erhalten. Wir wollen die Antwortzeiten für 1, 2, 3 und 10 parallele Benutzer. Dazu müssen Sie den aktuellen Modellparameter entsprechend variieren. Führen Sie die Funktion *actual parameters* auf der eingefügten Auswertung aus, und geben Sie im unteren Subeditor (ACTUAL VALUES) den Parameternamen 'anz' ein.

Öffnen Sie den Experimentbody, deklarieren Sie unter LOCALS diesen Parameter mit:

```
VARIABLE
  anz : INTEGER;
```

und geben Sie im BODY die Experimentspezifikation für die Experimentserie ein, wobei Sie nun den Aktualparameter nutzen:

```
FOR anz := 1, 2, 3, 10
LOOP
  EVALUATE bib3_a1_eva;
END LOOP;
```

Damit werden vier Analyseläufe mit den jeweils aktuellen Parametern gestartet. Bevor Sie im HIT RUN CONTROL Fenster HIT mit der *run*-Funktion starten, setzen Sie den Speicher für den Experimentlauf von 4096 kB auf 8192 kB hoch. Dadurch reduziert sich der Speicherverwaltungsaufwand bei der Analyse.

Sie können nach dem HIT-Lauf die Ergebnisse mit denen der ersten Version 'bib3' vergleichen.

anzahl_benutzer	1	2	3	10
TURNAROUNDTIME	3.38	3.83	4.40	15.05

4.3 Zweite Alternative des Speichersystems

Wenn die Zeit es noch erlaubt und Sie Spaß an der Modellierung mit HIT/HITGRAPHIC gefunden haben, können Sie auch die zweite vorgeschlagene Alternative des Speichersystems modellieren und analysieren.

Bei dieser Variante gibt der Prozeß das erste Laufwerk frei, falls er ein zweites benötigt, und beantragt dann beide Laufwerke gleichzeitig. Erst wenn zwei Laufwerke verfügbar sind, kann der Benutzer seine Sitzung fortsetzen.

Toolumsetzung

Änderungen für die zweite Alternative betreffen nur das Rechensystem, d.h. den Komponententyp 'retrieval3_a1' und natürlich durch dessen Verwendung auch den Modelltyp. Kopieren Sie 'retrieval3_a1' nach '**retrieval3_a2**' und 'bib3_a1' nach '**bib3_a2**'.

Im Komponententyp 'retrieval3_a2' müssen Sie der Problemstellung entsprechend das Prozeßmuster des Dienstes 'antwort' ändern:

```
belegen;
laden;
LOOP
  lesen;
  suchen (negexp (1/60000));
  BRANCH
    PROB 0.5 : freigeben;
               doppel_belegen;
               laden;
               laden;
               lesen;
               suchen (negexp (1/60000));
               freigeben;
  END BRANCH;
END LOOP UNTIL draw (0.5);
freigeben;
```

Sie benötigen nun auch wieder den vom 'cd_system' angebotenen Dienst 'belegen', für den Sie auf der Seite der Lastbeschreibung den *used service* '**belegen**' erzeugen müssen. Das funktioniert am einfachsten mit der *copy*-Funktion auf dem schon existierenden *used service* 'doppel_belegen'. Nachdem Sie den neuen genutzten Dienst an die Maschine gebunden haben, können Sie das Graphikfenster schließen.

In dem Modelltyp 'bib3_a2' ersetzten Sie bitte die Komponente 'rechensystem' vom Typ 'retrieval3_a1' durch eine Komponente des Typs 'retrieval3_a2'. Vergessen Sie nicht die Bindung.

Nach dem üblichen *save* und *quit* können Sie die Auswertungsbeschreibung '**bib3_a2_eva**' erzeugen. Um die Analysewerte vergleichen zu können, verfahren Sie wie in der vorherigen Modellalternative, d.h. Anlegen des Auswertungsobjekts '**rs_reaktion**' an der Komponente 'rechensystem' mit Auswertungsstrom *TURNAROUNDTIME*.

Die Experimentbeschreibung '**bib3_a2_exp**' können Sie von der schon vorhandenen Beschreibung 'bib3_a1_exp' kopieren. Modifizieren Sie die Auswertungsliste, den aktuellen Parameter eingeschlossen, und ebenso den Experimentbody.

Nach einer Analyse (wieder mit heraufgesetztem Speicher für den Experimentlauf) erhalten Sie die folgenden Ergebnisse:

anzahl_benutzer	1	2	3	10
TURNAROUNDTIME	3.48	3.55	3.64	5.53

4.4 Ergebnisauswertung

Abschließend können Sie noch mal die ermittelten Antwortzeiten der drei Modellvarianten vergleichen.

anzahl_benutzer	1	2	3	10
bib3	3.38	3.79	–	–
bib3_a1	3.38	3.83	4.40	15.05
bib3_a2	3.48	3.55	3.64	5.53

Da die Antwortzeiten für mehrere Benutzer in der zweiten Alternative wesentlich günstiger ausfallen, sollte diese den Vorrang haben.

Kapitel 5

Abschließende Bemerkungen

Wenn Sie das Dokument bis zu diesem Punkt durchgearbeitet haben, sollte Ihnen der Umgang mit HIT und HITGRAPHIC sowohl von der reinen Handhabung als auch von der Methodik, wie Modellierungsprobleme mit Hilfe des Tools angegangen werden können, vertrauter geworden sein.

Natürlich sind hier nicht alle Möglichkeiten des Tools demonstriert.

Bei der Modellbeschreibung sind gemeinsam genutzte (shared) Komponenten nicht erwähnt, die Bestandteil der Maschinen verschiedener Komponenten sind. Auch die Möglichkeit, vom Modellierer selbstdefinierte Ströme zu erzeugen, sich also eigene Leistungsmaße zu spezifizieren, ist nicht vorgestellt worden.

In der Auswertungsbeschreibung ist es möglich, Maße unterschieden nach sogenannten Verursacherhierarchien zu ermitteln.

Im Analysebereich kann zu einer Komponente mittels einer Voranalyse eine im leistungsmäßigen Verhalten ähnliche Ersatzkomponente (Aggregat) bestimmt werden, die an Stelle der ursprünglichen Komponente eingesetzt werden kann. Damit unterstützt HIT auch eine heterogene Modellierung unter Einsatz verschiedener Lösungsverfahren.

Wenn Sie daran interessiert sind, so sollten Sie sich mit dem Tutorial im „HITGRAPHIC User's Guide“ [2] beschäftigen. Dort wird, neben einer Einführung in HITGRAPHIC, auch auf die Aspekte gemeinsam genutzte Komponenten, Verursacherhierarchien und Aggregierung besonderer Wert gelegt.

Eine weitere Einführung, allerdings auf HI-SLANG-Sprachebene, bietet „HIT and HI-SLANG – An Introduction“ [3]. Dort finden Sie auch nähere Erläuterungen zu den HI-SLANG-Statements, den vordefinierten Komponententypen und der Bedeutung von Kontrollprozeduren. HI-SLANG ist eine HLL (high level language) und bietet neben den üblichen Kontrollstrukturen, wie z.B. IF-, CASE- oder LOOP-Statements spezielle für die Modellierung notwendige Konstrukte. Beispiele sind das CONCURRENT-Statement oder die verschiedenen Prozeduren zur Zufallszahlengenerierung (negexp, cox, normal, randint, ...).

Als Nachschlagewerke stehen Ihnen das „HI-SLANG Reference Manual“ [4] und der Referenzteil im „HITGRAPHIC User's Guide“ [2] zur Verfügung.

Die Nutzung von HIT ist hier anhand eines Problems aus dem Bereich der Rechensysteme demonstriert. HIT ist aber nicht auf diesen Bereich beschränkt; es wird auch zur Modellierung in Bereichen wie Kommunikationssysteme, Verkehrssysteme, Materialfluß- und Logistik-Systeme und anderen eingesetzt.

Bei Interesse ...

Für Universitäten ist der Einsatz von HIT für Zwecke der Forschung und Lehre kostenfrei (bis zu drei HIT-Lizenzen). Wenn Sie Interesse an einer HIT-Version haben oder weitere Literatur oder Dokumentation wünschen, wenden Sie sich bitte zur Vereinbarung der Modalitäten an:

Jürgen Mäter
Universität Dortmund
Informatik IV
D-44221 Dortmund

Telefon: +49 231 755 2411
E-Mail: Juergen.Maeter@udo.edu

Literaturverzeichnis

[1] H. Beilner, P. Buchholz, P. Kemper. *Hands on QPN.* Hands on Tools, Interne Berichte, Universität Dortmund, Informatik IV, 1993

[2] M. Sczittnick (ed.). *HITGRAPHIC User's Guide.* Version 2.1.00, Universität Dortmund, Informatik IV, 1993

[3] N. Weißenberg (ed.). *HIT and HI-SLANG: An Introduction.* Version 1.1.00, Universität Dortmund, Informatik IV, 1992

[4] M. Büttner (ed.). *HI-SLANG Reference Manual.* Version 3.4.00, Universität Dortmund, Informatik IV, 1994