UNIVERSITÄT DORTMUND

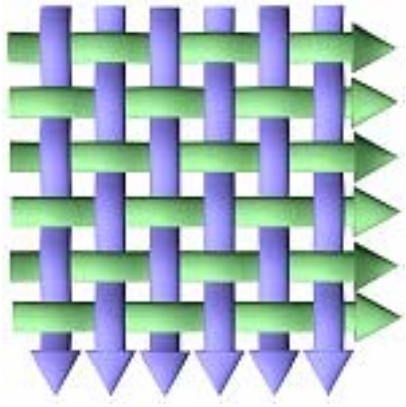
Technical Report 06003

ISSN 1612-1376

Hands On ProC/B-Tools –

Eine beispielorientierte Einführung in die Anwendung der ProC/B-Tools

Mirko Eickhoff, Michael Hierweck, Mathias Schwenke



Sonderforschungsbereich 559
Modellierung großer Netze in der Logistik

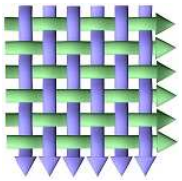
Universität Dortmund
44221 Dortmund

SFB 559 — Teilprojekt M1
LS Informatik IV
Universität Dortmund
27. September 2006
Version 1.1

Sonderforschungsbereich 559

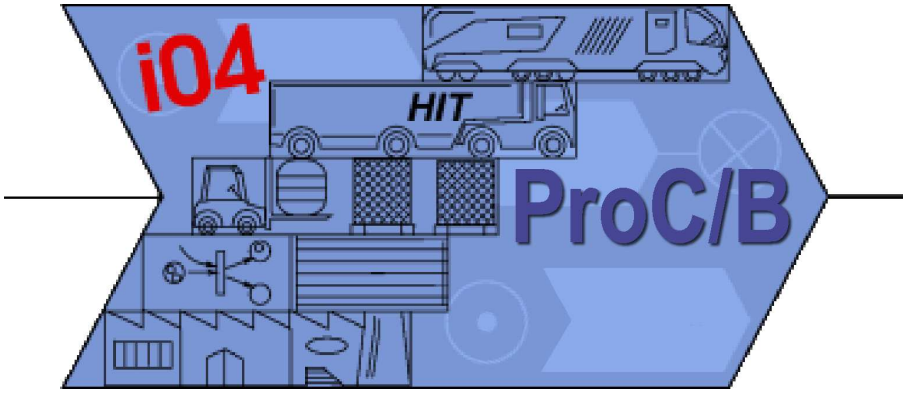
Modellierung großer
Netze in der Logistik

Technical Report 06003



Hands On ProC/B-Tools

Eine beispielorientierte Einführung
in die Anwendung der ProC/B-Tools



SFB 559 — Teilprojekt M1

Mirko Eickhoff
Michael Hierweck
Mathias Schwenke

Vorwort

Dieses Dokument soll Ihnen helfen, einen Einstieg in die Verwendung der Tools zur Modellierung und Analyse logistischer Systeme zu finden. Es bietet einen Überblick über alle Tools, und zwar aus der Sicht eines Anwenders, der erste Schritte mit diesen unternimmt, während er versucht, ein Modell inkrementell umzusetzen und zu analysieren. Wir werden dabei natürlich nicht alle Funktionen jedes Tools verwenden, aber nach dem Durcharbeiten dieses Dokuments sollte es Ihnen leicht fallen, alles Weitere durch Ausprobieren oder Studium der entsprechenden Programm-Dokumentationen herauszufinden.

Vorausgesetzt wird allerdings ein grundlegendes Verständnis des **ProC/B-Paradigmas** [5] sowie Erfahrung im Einsatz von Software im Allgemeinen. Sollten Sie noch keinen Überblick über die Ideen und Konzepte sowie die Bedeutung der einzelnen Modellelemente besitzen, so ist es empfehlenswert, sich diesen anzueignen, bevor Sie das Studium dieses Dokuments fortsetzen. Es wird nicht erwartet, dass Sie jedes Element inklusive der Attribute und deren Bedeutung exakt kennen.

Im Folgenden wird nun ein Schnellimbiss mit einigen Sitzplätzen, Küche und Personal modelliert, da wir annehmen, dass wohl jeder eine intuitive Vorstellung der Abläufe einer solchen Einrichtung besitzt. Weiterhin lässt sich unser Modell - wie wir später noch sehen werden - leicht inkrementell aufbauen, weiter präzisieren und erweitern.

Wir hoffen, dass dieses Dokument nicht nur trockene Theorie darstellt, sondern auch zum selbständigen Experimentieren mit den Tools einlädt. Ändern Sie einfach eine Einstellung im Modell und schauen Sie, welche Auswirkungen dies bei den Simulationsergebnissen mit sich bringt! Erweitern Sie das Modell um neue Speisen, fügen Sie Küchengeräte hinzu, und beobachten Sie die Auswirkungen auf den Personalbedarf! Lassen Sie dabei die Verweildauer der Kunden nicht außer Acht! Sie werden schnell erkennen, dass oft minimal erscheinende Änderungen großen Einfluss haben werden.

Zuletzt bleibt, Ihnen viel Freunde beim Lesen und Experimentieren sowie viel Erfolg bei der späteren Anwendung zu wünschen.

Dortmund, im Herbst 2006

Inhaltsverzeichnis

Kapitel 1

Modell eines Schnellimbisses

Angenommen, Sie erhalten den Auftrag, einen Schnellimbiss zu modellieren, um diesen anschließend im Sinne der **Leistungsbewertung** zu analysieren. Dies bedeutet, dass Sie versuchen sollen, einen konkreten Schnellimbiss mit den abstrakteren Elementen des **ProC/B-Paradigmas** so darzustellen, dass die für die Analyse relevanten Abläufe und Zusammenhänge erhalten bleiben. Bei der Analyse wird es Ihnen dann ermöglicht, durch **Simulation** verschiedene Informationen zu erhalten, beispielsweise die Verweildauer der Kunden im Schnellimbiss. Es besteht dann die Möglichkeit, durch Veränderungen am Modell die Auswirkungen auf die Messwerte zu untersuchen.

Bei dem zu modellierenden Schnellimbiss handelt es sich um einen kleinen Betrieb, der über Nacht schließt und mit wenig Personal, welches keiner speziellen Aufgabenteilung unterliegt, seine Kunden bedient. Diese haben die Möglichkeit, die Speisen mitzunehmen bzw. vor Ort zu verzehren. Existiert dieser Schnellimbiss bereits, so ist es sinnvoll, statistische Erhebungen über relevante Daten wie Kundenzahl, Bedienzeiten etc. durchzuführen, um der Realität möglichst nahe zu kommen und die Werte im Modell entsprechend einzustellen.

Bei der Modellierung gehen wir nun top-down vor. Dies bedeutet, mit einem (komplexen) Gesamtsystem zu beginnen und dieses schrittweise in kleinere einfachere Systeme zu zerlegen, welche dann zusammen das Gesamtsystem darstellen. Auf einer höhereren Abstraktionsebene muss man sich dann nicht mit den Details der kleineren Systeme beschäftigen. ProC/B-Modelle sind hierarchisch gegliedert und unterstützen somit diesen Ansatz.

Es ergibt Sinn, zwar einerseits das Modell weitgehend flexibel zu halten, sich andererseits aber nicht zu sehr auf Details zu konzentrieren. Grundsätzlich sollte das Ziel der Analyse im Auge behalten werden; in unserem Fall interessiert es den Auftraggeber insbesondere, ob Kunden zügig bedient werden und ob das Personal sinnvoll ausgelastet ist. Dennoch sollen später vielleicht auch weitere Auswertungen vorgenommen werden.

1.1 Die Umgebung

Stellen wir uns nun einen Schnellimbiss als Blackbox vor. Was sieht der Beobachter von außen? In erster Linie wird er feststellen, dass Personen den Schnellimbiss betreten und wieder verlassen. Beobachtet er dies genauer, so wird er feststellen, dass manche länger im Schnellimbiss verweilen, andere aber nach recht kurzer Zeit mit Tüten beladen wieder herauskommen. Es gibt also zwei verschiedene Typen von Kunden. Wir nennen die zuerst genannten *Sit-In-Gäste* und letztere *Take-Away-Gäste*. Beide Gast-Typen betreten und verlassen den Schnellimbiss. Also soll unser Schnellimbiss die Operation *besuchen* durch bestimmte Typen von Gästen unterstützen.

1.2 Prozesse

Ein Gast wird gemäß dem **ProC/B-Modellformalismus** als **Prozess** dargestellt, welcher gestartet wird und den Imbiss durchläuft, bis er anschließend terminiert. Während er den Imbiss durchläuft, greift er auf Ressourcen zurück und nutzt damit den Imbiss bzw. dessen Komponenten. Durchlauf und Nutzung orientieren sich dabei an der Handlungsabfolge, die ein Gast in einem Schnellimbiss üblicherweise vornimmt, beispielsweise: Betreten, Bestellen, Verzehren, Bezahlen, Verlassen. Dabei werden auch neue Prozesse initiiert, beispielsweise Kochen oder Kassieren, welche nicht durch den Gast vorgenommen werden, der Gast war jedoch Auslöser.

1.3 Ressourcen

Wie bereits angedeutet, benötigen die Prozesse entsprechende Ressourcen. Diese werden nun vorgestellt.

1.3.1 Der Schnellimbiss

Öffnen wir dazu die Blackbox und werfen wir einen Blick ins Innere. Wir stellen fest, dass in unserem Schnellimbiss zunächst einmal folgende für die logistischen Ressourcen relevante Objekte auftreten: Eine Theke, an der bedient wird, einige Tische mit Sitzplätzen für den Verzehr vor Ort, Personal und, nicht zu vergessen, die Küche. All diese Objekte fassen wir zunächst wieder als Blackbox auf.

1.3.2 Das Personal

Das Personal zeichnet sich zunächst einmal dadurch aus, dass es aus einer festen Anzahl von Personen besteht. Diese können nun zu bestimmten Tätigkeiten herangezogen, also angefordert werden. Wir gehen davon aus, dass unser Personal universell ausgebildet ist und jede Aufgabe übernehmen kann.

1.3.3 Die Sitzplätze

Ähnlich dem Personal steht eine feste Anzahl von Sitzplätzen zur Verfügung. Für unsere Zwecke soll es genügen, mitzuzählen, wie viele Plätze belegt sind, und im Falle der Vollauslastung den Gast warten zu lassen.

1.3.4 Die Küche

In der Küche wird gekocht. Wir beschränken uns zu diesem Zeitpunkt darauf, dass die unterstützte Operation *Kochen* Zeit in Anspruch nimmt.

1.3.5 Die Theke

An der Theke werden die entscheidenden Dienstleistungen erbracht. Zunächst werden Bestellungen aufgenommen, anschließend die zubereiteten Speisen ausgegeben und je nach Typ des Gastes sofort oder nach dem Verzehr kassiert.

1.3.6 Die Kasse

Zum Kassieren benötigt man eine Kasse. Die Kasse sollte deshalb modelliert werden, da es nicht möglich erscheint, dass mehrere Mitglieder des Personals diese gleichzeitig verwenden. Wir werden später erkennen, dass dieser simpel erscheinende Umstand Folgen haben wird. Auch bei der Kasse handelt es sich um eine Ressource, die belegt und freigegeben werden kann.

Kapitel 2

Auswahl geeigneter Modellelemente

Nachdem wir uns überlegt haben, wie die verschiedenen Komponenten unseres zunächst stark vereinfachten Schnellimbisses zusammenspielen und welche relevanten Eigenschaften in das Modell einfließen sollen, stehen wir nun vor der Aufgabe, geeignete Komponenten aus dem ProC/B-Modellformalismus zu wählen.

Zunächst einmal bietet es sich an, all das, was wir zunächst als Blackbox betrachtet haben, auch als solche umzusetzen. Das Modellelement **Funktionseinheit** eignet sich hierfür. Man kann es mit einer kleinen Programmbibliothek vergleichen, welche Funktionen zur Verfügung stellt. Den Funktionen können Parameter übergeben werden, und sie können Rückgabewerte liefern. Die Funktionen heißen hier **Dienste**, welche beim Aufruf gestartet und im Innern der **Funktionseinheit** durch **Prozessketten** dargestellt werden. Sobald diese terminieren, ist der Aufruf beendet. Wie in der Programmierung kann man **Funktionseinheiten** intern ohne Auswirkungen auf die Nutzung der **Dienste** innerhalb des **Modells** verändern, sofern man nur die Schnittstelle unverändert lässt. Wir modellieren die Schnittstelle, indem wir die zur Verfügung stehenden **Prozesse** benennen und deren **Parameter** angeben.

ProC/B-Modelle sind hierarchisch aufgebaut. Durch die **Funktionseinheiten** sind die Schnittstellen zwischen den Hierarchieebenen definiert. **Funktionseinheiten** bieten die Möglichkeit, ihre bereitgestellten **Dienste** auf der jeweils höheren Hierarchieebene zu nutzen.

Externe Funktionseinheiten können **Dienste**, die durch andere **Funktionseinheiten** bereitgestellt werden, sogar in entgegengesetzter Richtung importieren, d.h. Dienste, die auf der jeweils höheren Hierarchieebene bereit stehen, können innerhalb der **Externen Funktionseinheit** genutzt werden.

2.1 Die Umgebung

Die Umgebung stellt natürlich die oberste Ebene unserer Hierarchie dar. Wir verwenden zwei **Quellen**, welche die beiden Typen der Gäste mit bestimmter Häufigkeit auf die Reise schicken. Jeder Gast stellt somit einen **Prozess** dar, welcher den *Schnellimbiss* besucht und anschließend in einer **Senke** verschwindet.

2.2 Der Schnellimbiss

Der *Schnellimbiss* selbst wird durch eine **Funktionseinheit** repräsentiert. Innerhalb dieser wird der Besuch desselben modelliert. Ferner geben wir ihm eine selbständige **Prozesskette** mit der Bezeichnung *Aufraeumen*. Dieser regelmäßig startende **Prozess** soll Kräfte des Personals binden, da sich das Personal bekanntlich nicht ausschließlich den Gästen widmen kann, sondern noch Arbeiten, die unabhängig vom Erscheinen eines Gastes anfallen, erledigen muss. Diese Tätigkeiten benötigen zunächst einmal nur Zeit.

2.3 Das Personal

Bei der Wahl eines geeigneten Elements für das Personal entscheiden wir uns für einen **Server**. Ein **Server** stellt eine Ressource zur Verfügung, welche sich über den Aufruf *request* für einen Zeitraum anfordern lässt. Es sind mehrere Anforderungen zeitgleich möglich. Allerdings ist die Anzahl der zeitgleich möglichen Anforderungen beschränkt. Die Beschränkung ist einstellbar und entspricht in diesem Fall der Anzahl der zeitgleich eingesetzten Mitarbeiter des Schnellimbisses.

2.4 Die Sitzplätze

Bei den Sitzplätzen wählen wir das **Storage**. Dieses stellt einen mehrdimensionalen diskreten Raum dar. Im Fall der Sitzplätze benötigen wir nur eine Dimension. Es lassen sich die minimale und maximale Belegung vorgeben. Die minimale Belegung der Sitzplätze ist natürlich 0, die maximale Belegung ist die Anzahl der real verfügbaren Sitzplätze. Denkbar wäre es, eine weitere Dimension hinzuzufügen und damit zwischen Sitz- und Stehplätzen zu unterscheiden. Um das Modell nicht unnötig zu komplizieren, verzichten wir darauf.

2.5 Die Küche

Für die Küche bietet sich wieder eine **Funktionseinheit** an. Die Vorgänge in der Küche sind aufwendig, und es müssen auch viele Ressourcen verwaltet werden. Die Küche stellt außerdem eine eigenständige logische Einheit dar. Man

könnte sich sogar später einmal dazu entschließen, nicht selbst zu kochen, sondern ein Fremdunternehmen kochen zu lassen, die Küche auszulagern oder gar die Küche umzugestalten, ohne den Rest des Modells zu ändern. All dies wird von Funktionseinheiten leicht unterstützt. Zunächst einmal lassen wir die Küche sehr abstrakt und nehmen an, dass das Kochen lediglich Zeit erfordert und Personal des Schnellimbisses in Anspruch nimmt. Wir haben das Personal jedoch dem Schnellimbiss zugeordnet. Eine **Externe Funktionseinheit** ermöglicht es uns jedoch, auf diese Ressource über die aufrufende Hierarchieebene zuzugreifen.

2.6 Die Theke

Die Theke behandeln wir als zentralen Dienstleistungspunkt, der verschiedene Dienste anbietet und sich dabei, wie die Küche, auf die Ressource Personal abstützt. Allerdings sind die Abläufe nicht sonderlich komplex, weshalb wir sie schon in der ersten Fassung unseres Modells modellieren werden.

2.7 Die Kasse

Die Kasse ist - ähnlich den Sitzplätzen - eine weitere Ressource, welche geteilt werden muss. Wir verwenden einen **Counter**, der im Sinne eines Semaphors dafür sorgt, dass nicht zwei Mitglieder des Personals gleichzeitig buchen.

Kapitel 3

Modellierung mit dem ProC/B-Editor

Wir belassen es zunächst einmal bei diesem einfachen Modell und versuchen, es mit Hilfe der ProC/B-Elemente umzusetzen. Oftmals werden jetzt konzeptionelle Mängel deutlich, darum ergibt es Sinn, dies zu testen und auch probeweise Analysen durchzuführen.

3.1 Das ProC/B-GUI

Starten wir also das ProC/B-GUI, welches das zentrale Menü für alle weiteren Funktionen umfasst. Nach dem Start erscheint das in Abbildung 3.1 gezeigte Hauptmenü. Über die Schaltflächen lassen sich die verschiedenen Unterprogramme aufrufen.

3.2 Arbeiten mit dem ProC/B-Editor

Wenden wir uns zunächst einmal dem ProC/B-Editor zu. Für Details hinsichtlich der Verwendung des ProC/B-Editors verweisen wir auf das Handbuch [2], welches jeden Menüeintrag, jede Schaltfläche und Tastenkombination erläutert.

Bevor wir starten, müssen wir uns auf eine Zeiteinheit einigen, d.h. wir müssen die Beziehung zwischen der Modellzeit und der Echtzeit festlegen. Da alle Zeitangaben im Modell als Vielfache der Modellzeit-Einheit angegeben werden und in unserer Modellwelt viele nur kurze Zeit dauernde Aktivitäten vorliegen, bietet sich folgende Beziehung an: Eine Modellzeit-Einheit entspricht einer Sekunde der Realität.

3.2.1 Das Hauptfenster mit Hierachiebaum

Der ProC/B-Editor startet mit einem neuen leeren Projekt. Abbildung 3.2 zeigt das Hauptfenster mit dem zunächst noch leeren Hierachiebaum. Über den

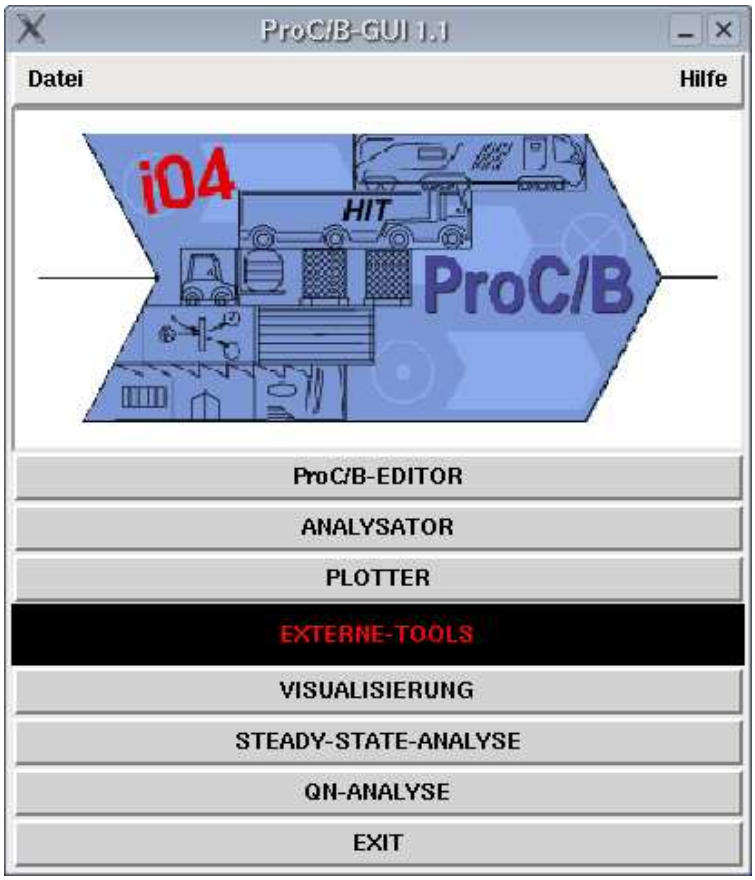


Abbildung 3.1: Hauptmenü des ProC/B-GUIs nach dem Start

Menüpunkt „Modell->Neu“ erhält man ein neues Arbeitsfenster zum Modellieren.

3.2.2 Verwendung des Arbeitsfensters

Nachdem sich das Arbeitsfenster geöffnet hat, erscheint eine Fläche, welche in mehrere Bereiche eingeteilt ist. Beim Modellieren verwenden wir den unteren Bereich für Funktionseinheiten, Server, Counter und Storages. Auf dem mittlereren Bereich können wir die restlichen Elemente platzieren. Dazu wählen wir ein Element aus der Buttonleiste unter dem Menü aus und klicken auf seine Zielposition.

Attribute dieser Elemente werden über Masken editiert, man erreicht diese Attributmasken, indem man mit der mittleren Maustaste das Element anklickt und aus dem Kontextmenü „Attribute“ auswählt. Viele Attributwerte sind voreingestellt.

Hat man die Werte vom Standardwert auf einen anderen geändert, so werden die Attribute in der Nähe der Elemente textuell angezeigt.

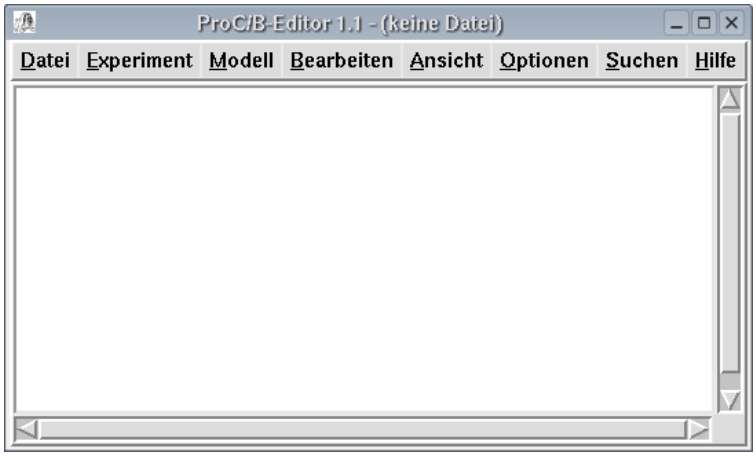


Abbildung 3.2: Hauptfenster des ProC/B-Editors nach dem Start mit leerem Hierarchiebaum

3.2.3 Die oberste Ebene: Der Imbiss

Die oberste Ebene enthält als wesentliche Elemente **Quellen**, **ProzessIDs** und **Senken**. Hier werden die **Prozesse** gestartet, welche ihren Weg durch unser Modell finden werden.

Über den Menüeintrag „Funktionseinheit->Bezeichner“ geben wir dem Modell die Bezeichnung *Imbiss*, legen mit dem Element **Quellen** die Quellen an, stellen in deren Attributmasken die Anzahl auf *randint(0,3)*[1] für die *Take-Away-Gäste* bzw. auf *randint*$(0, 2)$ für die *Sit-In-Gäste* ein sowie die Zeitangaben auf *negexp*$(1/120)$[2] bzw. *negexp*$(1/360)$.

Mit dem Element **ProzessID** legen wir die Schnittstellen der Prozessketten fest, in unserem Beispiel vergeben wir in der Attributmaske lediglich die Bezeichner *TakeAwayGast* und *SitInGast*. Beiden Prozessketten fehlen jetzt noch die **Prozesskettenelemente**, die den angebotenen Dienst der noch anzulegenden **Funktionseinheit** *Schnellimbiss* aufrufen werden. Den Elementen geben wir die Bezeichner *Schnellimbiss_besuchen*. Anschließend verbinden wir mit der rechten Maustaste die Prozesskettenbestandteile zu den beiden Prozessketten.

Beide **Prozessketten** sollen den Dienst *besuchen* der **Funktionseinheit** *Schnellimbiss* aufrufen, wobei ein **Parameter** übergeben wird. Dieser dient zur Unterscheidung der Gästetypen auf der unteren Ebene. Erstmal legen wir also eine **Funktionseinheit** an und geben ihr über die Attributmaske den Bezeichner *Schnellimbiss* sowie über den Eintrag „Port virtuelle Quelle anlegen“ einen Dienst. Im nächsten Abschnitt wird dann das Innenleben des Schnellimbisses beschrieben.

[1] Als Ergebnis wird mit gleicher Wahrscheinlichkeit die Anzahl 0, 1, 2 oder 3 gewählt, es werden hier also 0 bis 3 Prozesse gestartet.

[2] Im Mittel werden alle 120 Zeiteinheiten Prozesse der entsprechenden Anzahl gestartet, hier liegen also im Mittel 120 Sekunden zwischen 0 bis 3 neu gestarteten Prozessen der *Take-Away-Gäste*. Entsprechendes gilt für die *Sit-In-Gäste*.

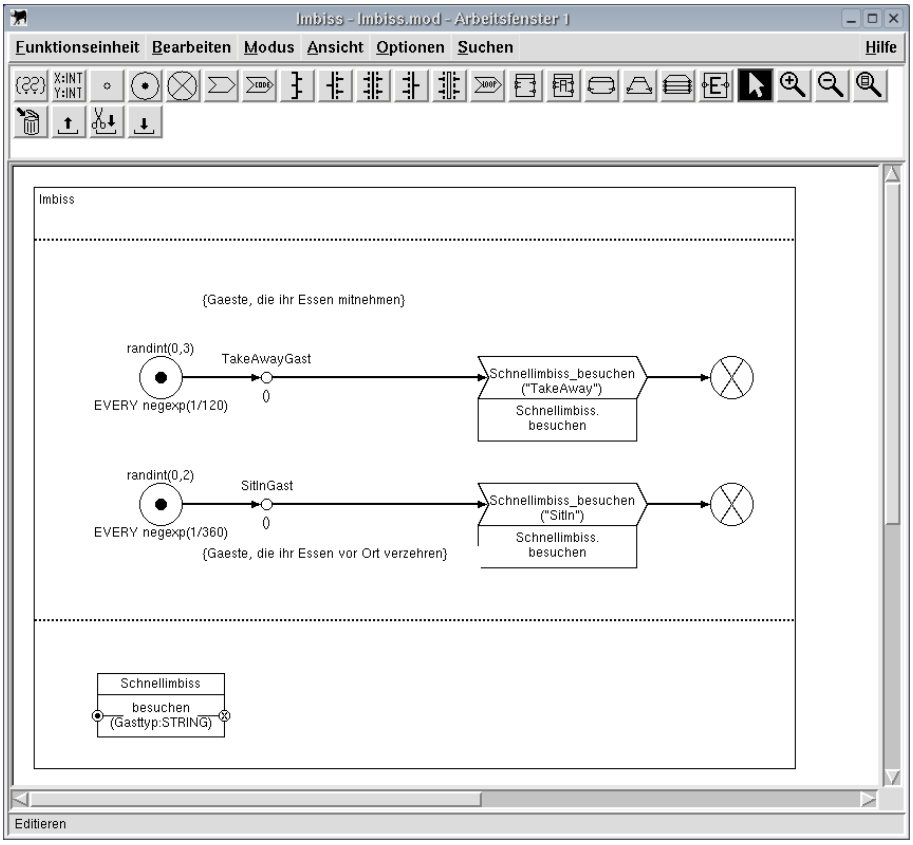


Abbildung 3.3: Editorfenster mit dem Modell der Außenansicht des Schnellimbisses

Die Anbindung der Prozesskettenelemente an die Funktionseinheit erfolgt über die rechte Maustaste; die Eingabe der dazugehörigen Parameter *TakeAway* bzw. *SitIn* in den Attributmasken der Elemente kann allerdings erst erfolgen, wenn zumindest die Schnittstelle der Funktionseinheit *Schnellimbiss* im nächsten Abschnitt angelegt wurde. Wir öffnen die Funktionseinheit zum Beispiel über den entsprechenden Eintrag deren Kontextmenü.

Die Abbildung 3.3 zeigt das Modell der Außenansicht des Schnellimbisses, wie im Entwurf beschrieben.

Die Kommentare sollen dem Betrachter helfen, das Modell schneller erfassen zu können. Sie lassen sich mit dem entsprechenden Werkzeug und dem Einfügen des Kommentars in die Kommentarliste der Attributmaske anlegen. In unserem Fall erläutern sie die Variablenbelegung.

Um die Übersichtlichkeit zu erhöhen, haben wir die beiden alternativ parallel ablaufenden Teilprozessketten auch parallel dargestellt, so dass sofort auffällt, dass hier ähnliches Verhalten modelliert wird.

3.2.4 Die Innenansicht: Der Schnellimbiss

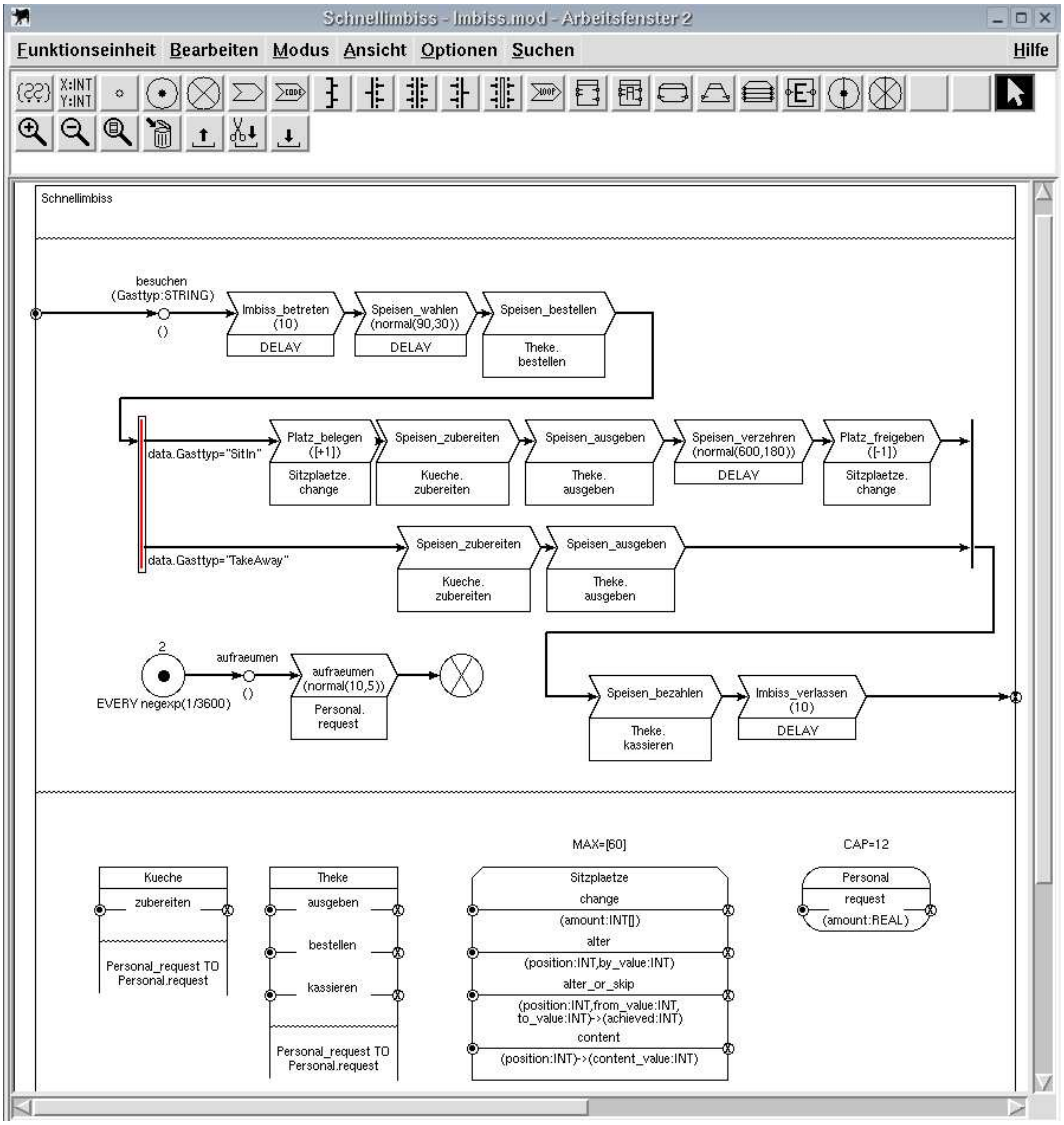


Abbildung 3.4: Editorfenster mit dem Modell der Innenansicht des Schnellimbisses

Die Abbildung 3.4 zeigt den erwartungsgemäß aufwändigeren Schnellimbiss.

Zunächst einmal fällt die **Prozesskette** *besuchen* auf. Deutlich zu erkennen ist die fast lineare Abfolge der Prozesskettenelemente, deren Reihenfolge sich am Ablauf eines Schnellimbissbesuchs orientiert. Ein Besucher des Schnellimbisses führt nacheinander bestimmte Handlungen durch. Diese werden durch **Prozesskettenelemente** dargestellt. Allerdings tritt eine Fallunterscheidung auf, welche durch den **Oder-Konnektor** modelliert wird. Diese alternativen Abläufe sind optisch deutlich erkennbar. Es fällt durch die Anordnung der **Prozesskettenelemente** innerhalb dieser Fallunterscheidung auch auf, dass die

Aktionen auf dem unteren Weg eine echte Teilmenge derer auf dem oberen Weg darstellen.

Die Schnittstelle der Prozesskette *besuchen* wurde erzeugt, indem in der Attributmaske der **ProzessID** der Eingabeparameter *Gasttyp* vom Typ String angelegt wurde. Nun lassen sich auch in den beiden Prozesskettenelementen *Schnellimbiss_besuchen* aus dem vorhergehenden Abschnitt die Werte *TakeAway* und *SitIn* angeben, die beim Aufruf dieses Dienstes als Parameter *Gasttyp* übergeben werden.

Die in der **ProzessID** angegebenen Parameter lassen sich als lokale Variablen in dieser Funktionseinheit benutzen, dazu muss vor den Parameternamen *data.* stehen. In unserem Modell benutzen wir diesen Parameter, indem wir als Werte der beiden ausgehenden Verbindungen des **Oder-Konnektors** *data.Gasttyp="SitIn"* bzw. *data.Gasttyp="TakeAway"* eintragen. Den Oder-Konnektor stellen wir auf den Typ *boolesch*.

Küche und Theke wurden als **Funktionseinheiten** modelliert, Personal und Sitzplätze als **Server** bzw. **Storage**. Das **Storage**, welches die Sitzplätze repräsentiert, wird zweimal aufgerufen. Sobald ein Gast Platz nimmt, erhöhen wir die Anzahl der belegten Sitzplätze, nach dem Verzehr geben wir sie wieder frei. Wir zählen hiermit also die belegten Sitzplätze. Alternativ hätte man auch die noch freien Sitzplätze zählen können. Dann sollte das **Storage** allerdings auch entsprechend vorbelegt werden.

Wie bereits in Abschnitt 2.4 erwähnt, verwalten **Storages** mehrdimensionale diskrete Räume, repräsentiert durch Vektoren, welche durch eckige Klammern gekennzeichnet werden. In unserem Fall enthält dieser Vektor nur eine Komponente für die Anzahl der belegten Sitze. In der Attributmaske legen wir den Initialwert [0], den Minimalwert [0] sowie das Maximum [60] fest. Beim Aufruf des **Storage** sollte man beachten, dass ebenfalls ein Vektor in eckigen Klammern übergeben wird.

Auch die interne **Prozesskette** *aufraeumen* ist bereits modelliert worden. Sie steht nicht mit dem aufrufenden **Prozess** in Verbindung, teilt sich mit diesem aber die Ressource *Personal*.

Bei den **Funktionseinheiten** in der Abbildung 3.4 erkennt man deutlich, dass der **Dienst** *Personal.request* jeweils importiert wird. Darauf wird der Abschnitt 3.2.6 genauer eingehen.

3.2.5 Zunächst vereinfacht: Die Küche

Der einzige **Dienst** der Küche *zubereiten* nimmt für einen dynamischen Zeitraum Personal in Anspruch, welches für diesen Zeitraum nicht mehr für weitere Anforderungen verfügbar ist. Wie sich das Anfordern des Personals, welches zunächst nur in der übergeordneten Funktionseinheit definiert ist, importieren lässt, beschreibt der folgende Abschnitt.

Zugegebenermaßen ist die Küche recht abstrakt modelliert. Arbeitet man mit mehreren Modellierern im Team, so wäre dies eine geeignete Schnittstelle zum

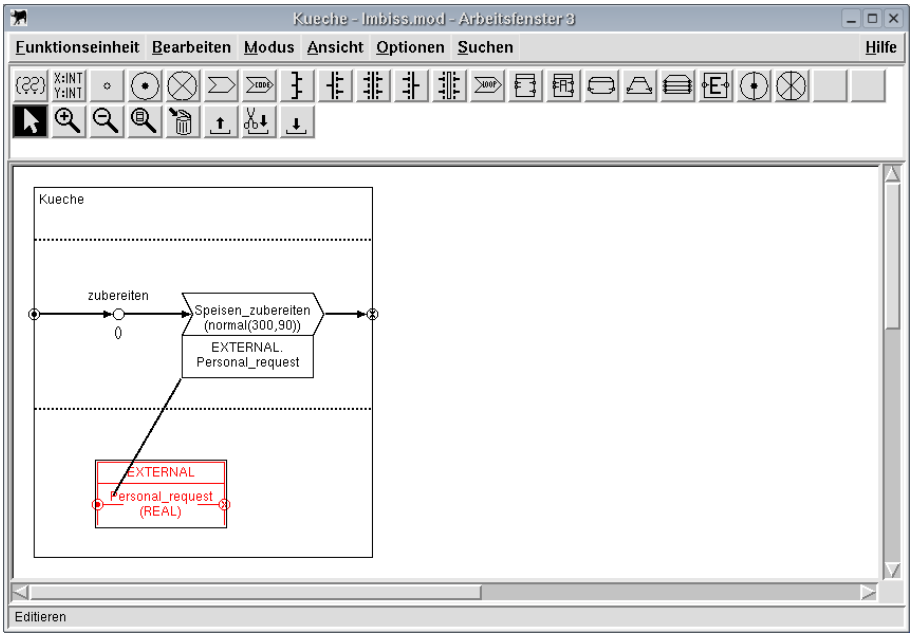


Abbildung 3.5: Editorfenster mit dem Modell der Küche

Modellierer, welcher sich mit der Umsetzung der Küche befasst, da diese ein eigenständiges System darstellt.

3.2.6 Gemeinsam verwendet: Das Personal

Die Küche verwendet die Ressource Personal, welche jedoch dem Schnellimbiss zugeordnet wurde. Dies ist auch von der Idee her korrekt, da das Personal auch für die übrigen Aktivitäten verfügbar sein muss. Verfügt die Küche über exklusives Personal, z.B. dedizierte Köche, so würde man dieses der Küche zuordnen. Wir müssen uns das Personal aber teilen.

Dazu verwenden wir eine **Externe Funktionseinheit**: Über die Attributmaske der **Funktionseinheit** *Küche* werden die importierten **Dienste** editiert. Wir benutzen die Einstellungen unter „Prozesszuordnung“, dort vergeben wir einen Namen für jeden zu importierenden Dienst, unter dem der Dienst in der externen Funktionseinheit ansprechbar sein soll. In unserem Fall ist dies *Personal_request*. Man bindet diesen an einen Dienst (bei uns *request*) einer Funktionseineit, eines Servers, eines Storages etc. (bei uns *Personal*) an, indem man festlegt, welcher Dienst welcher Funktionseinheit, welchen Servers oder welches Storages etc. importiert werden soll. Dieser Dienst kann über ein Menü ausgewählt werden. Dabei wird automatisch die virtuelle Parameterliste (unter „Parameter(virtuell)“) mit den Namen und Typen der passenden Parameter belegt. In unserem Fall wurde zum Prozess *Personal_request* der Parameter *amount* vom Typ *real* eingetragen.

Anschließend wird innerhalb der **Funktionseinheit** *Küche* im unteren Bereich

eine **Externe Funktionseinheit** automatisch eingeblendet. Sie bietet den importierten **Dienst** an. Der Zugriff auf diesen **Dienst** erfolgt dann genauso, als ob er von einer gewöhnlichen **Funktionseinheit** dieser Hierachieebene bereitgestellt würde. Die Abbildung 3.5 zeigt das Editorfenster mit dem vollständigen Modell der Küche.

3.2.7 Vollständig modelliert: Die Theke

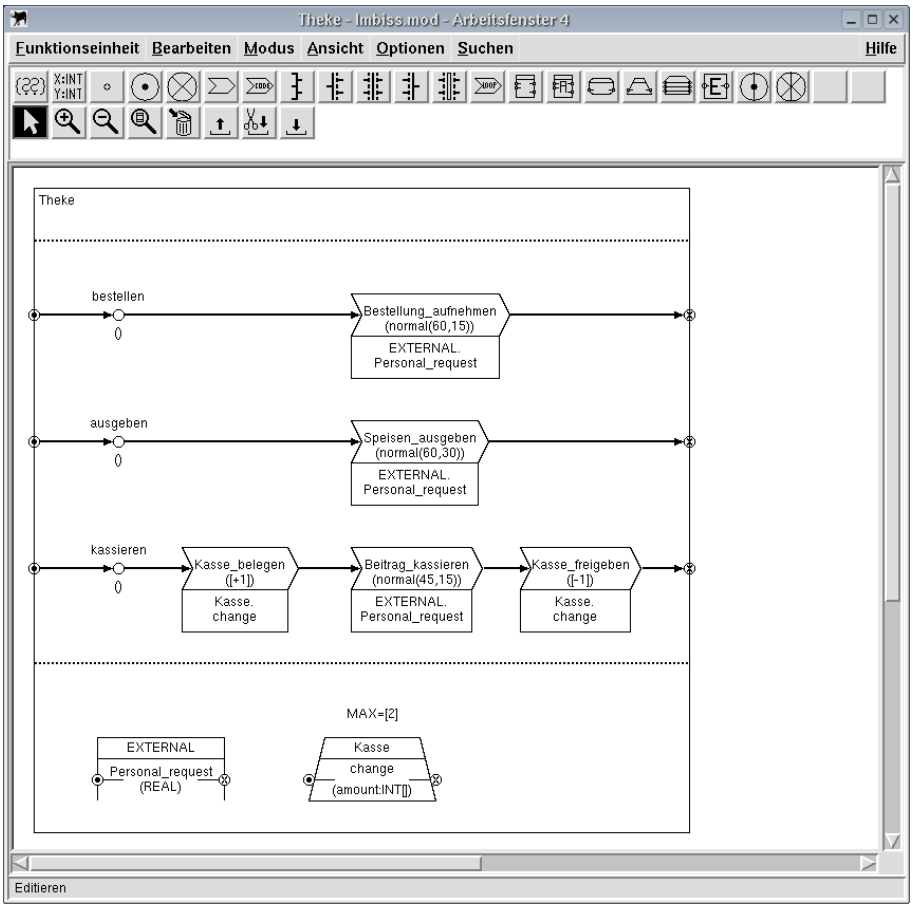


Abbildung 3.6: Editorfenster mit dem Modell der Theke

Das Aufnehmen der Bestellungen sowie die Ausgabe der Speisen benötigen offenbar Zeit und für die Dauer dieses Zeitraumes Personal. Gleiches gilt auch für den **Dienst** *kassieren*. Allerdings kann nur dann kassiert werden, wenn auch eine Kasse zur Verfügung steht. Wir verwenden einen **Counter** als Semaphor, allerdings werden sogar zwei gleichzeitige Verwendungen zugelassen; es existieren also zwei unabhängige Kassen.

Personal wird aber erst dann angefordert, wenn auch eine Kasse zur Verfügung steht. Solange dies nicht der Fall ist, kann das Personal andere Tätigkeiten ausüben. Die Abbildung 3.6 zeigt das Editorfenster mit dem Modell der Theke.

3.2.8 Zusammenfassung

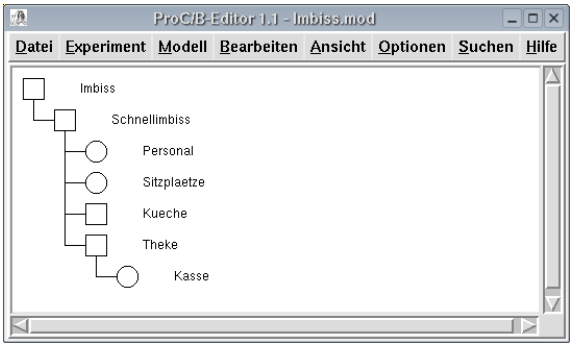


Abbildung 3.7: Hauptfenster des ProC/B-Editors mit Anzeige des Hierarchiebaums

Wir haben eine erste Version des Modells unseres Schnellimbisses modelliert. Dabei kommen verschiedene Modellelemente zum Einsatz, welche über Attribute für unsere Zwecke angepasst wurden.

Das hier vorgestellte Modell ist nur eine Möglichkeit unter vielen, einen Schnellimbiss zu modellieren. Vielleicht entwerfen Sie selbständig ein anderes Modell. In der Praxis wird man vielleicht mehrere Versuche starten, bis man zu einem Modell gefunden hat, welches den Anforderung entspricht.

Wir schließen nun die Editierfenster des ProC/B-Editors und speichern das Modell durch Auswählen des Menüpunktes „Speichern“ aus dem „Datei“-Menü unter dem Namen „Imbiss.mod“.

Die Abbildung 3.7 zeigt, wie im Hauptfenster des ProC/B-Editors nun der Hierachiebaum dargestellt wird.

Bevor wir unser Modell nun analysieren können, müssen wir zunächst noch definieren, was wir auswerten möchten.

Kapitel 4

Experimentdefinition mit dem ProC/B-Editor

Nachdem wir nun eine erste Version des Modells konstruiert haben, möchten wir eine Analyse durchführen, um mögliche Fehler zu finden und einen Überblick über die Leistungsfähigkeit unseres Modells zu gewinnen.

4.1 Start der Experimentdefinition

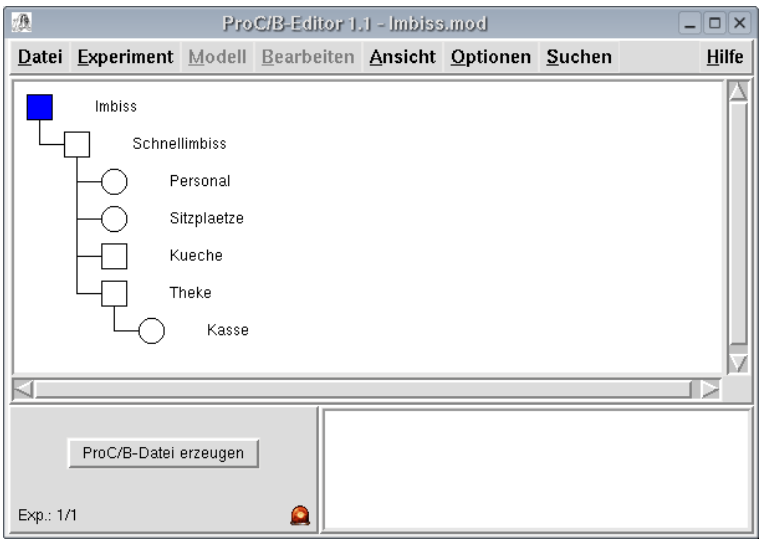


Abbildung 4.1: Experimentfenster mit Hierachiebaum

Zum Starten der Experimentdefinition wählen wir den obersten Knoten der zu analysierenden Hierarchie im Hierarchiebaum, der im Hauptfenster des ProC/B-Editors zu sehen ist, aus. Im lokalen Menü des obersten Knotens starten wir das Experiment durch Auswahl der Menüpunkte „Experimentserie anlegen ->

ProC/B-Analyse“. Abbildung 4.1 zeigt das Experimentfenster nach dem Start der Definition des Experiments. Dabei wird der oberste Knoten der zu analysierenden Hierarchie blau dargestellt.

4.2 Definition eines einfachen Messobjekts

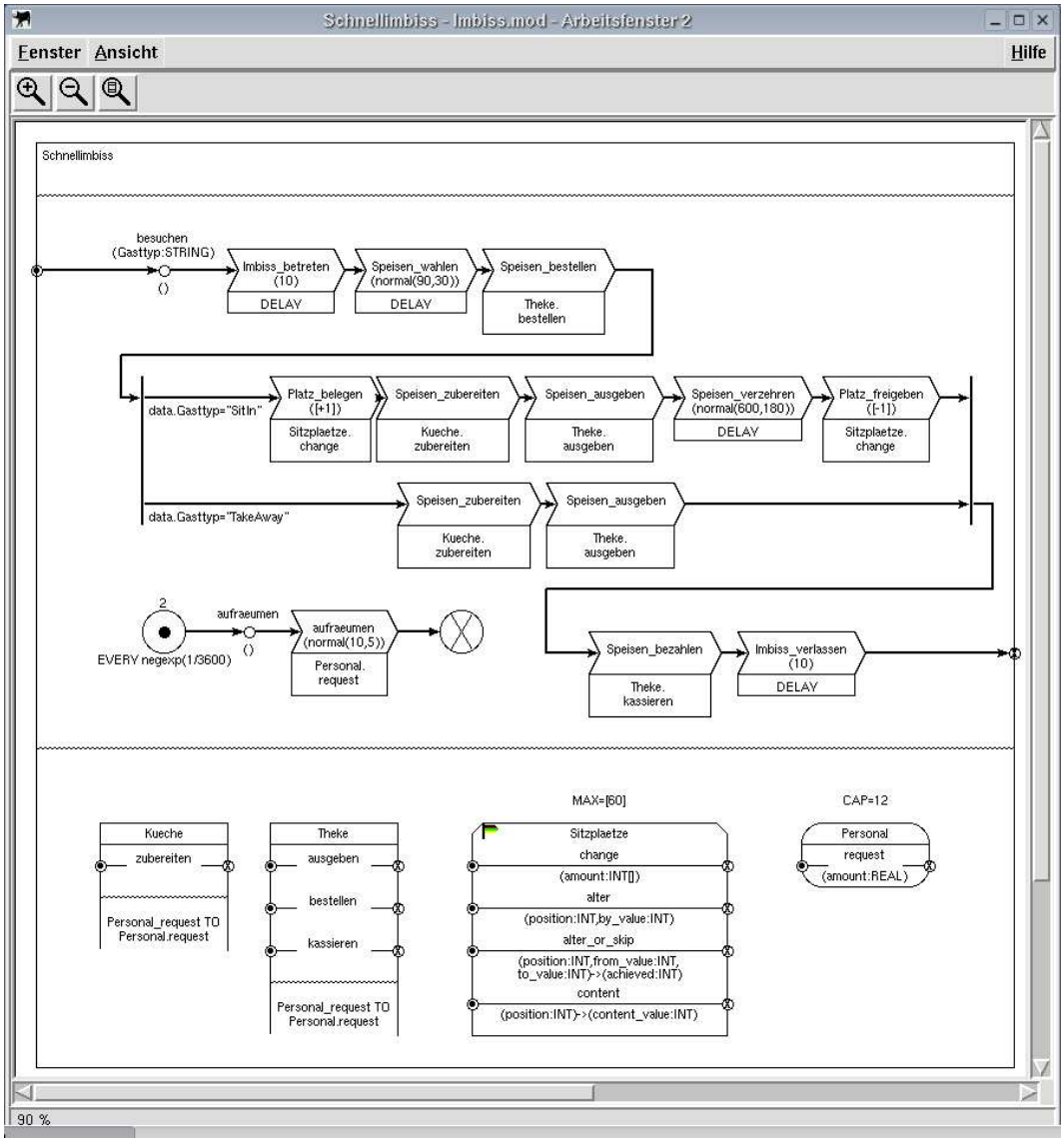


Abbildung 4.2: Experimentfenster mit der Innenansicht des Schnellimbisses

Zunächst einmal messen wir die Auslastung unserer Sitzplätze. Dazu öffnen wir das Fenster, welches die Innenansicht des Schnellimbisses zeigt. Wir öffnen mit der Maus am **Storage** *Sitzplaetze* ein lokales Menü. Hier wählen wir die Menüpunkte „Neuer Messpunkt“ - „STATE“ - „ALL“. Das bedeutet, dass wir

die Auslastung unabhängig vom Verursacher bestimmen wollen. Abbildung 4.2 zeigt das geöffnete Fenster. Das Fähnchen am **Storage** zeigt nach dem Abschluss der Definition, dass sich dort wenigstens ein Messobjekt befindet.

4.3 Definition eines Messobjekts mit Verursacherpfad

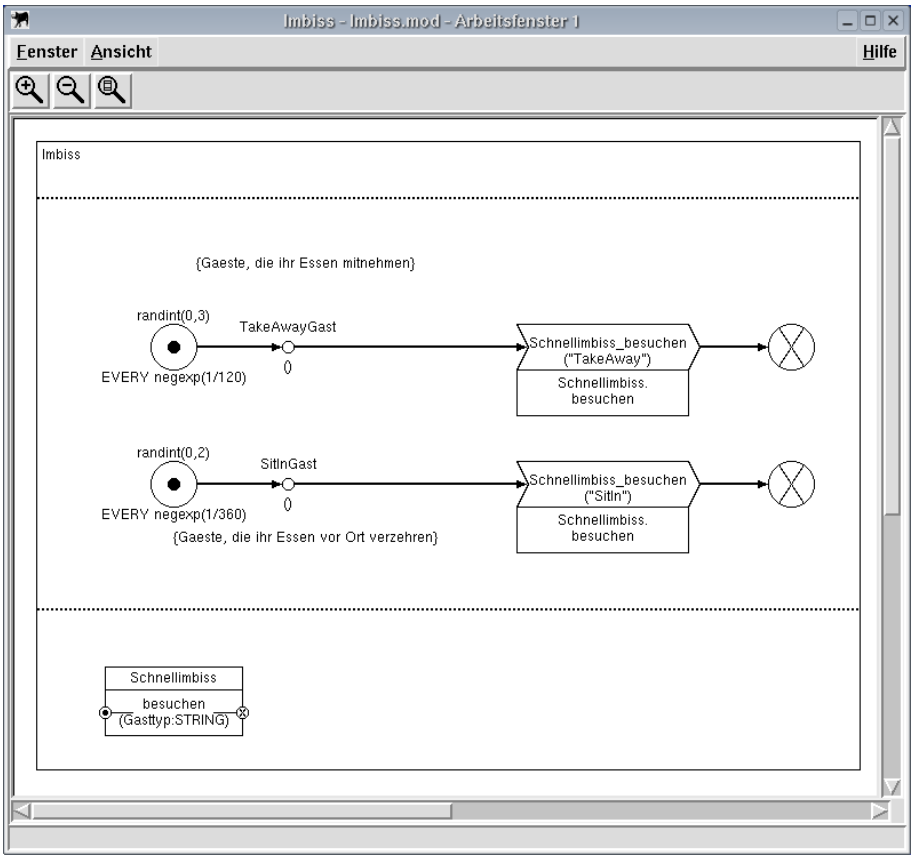


Abbildung 4.3: Experimentfenster mit der Außenansicht des Schnellimbisses

An der **Funktionseinheit** *Schnellimbiss* selbst möchten wir die Verweilzeiten messen. Wir definieren nun drei Messpunkte an der **Funktionseinheit**, die uns Aussagen über die Verweildauer allgemein sowie die Verweildauer aufgeschlüsselt nach den Gasttypen liefern.

Dazu öffnen wir das Fenster, welches die Außenansicht des Schnellimbisses wie in Abbildung 4.3 zeigt. Wir definieren den Messpunkt für die allgemeine Verweildauer analog zum Messpunkt am **Storage**, wählen diesmal allerdings „TURNAROUNDTIME“. Die Definition der Messpunkte für die aufgeschlüsselten Verweilzeiten ist ein wenig komplizierter:

Zunächst erzeugen wir über das lokale Menü an der **Funktionseinheit** einen

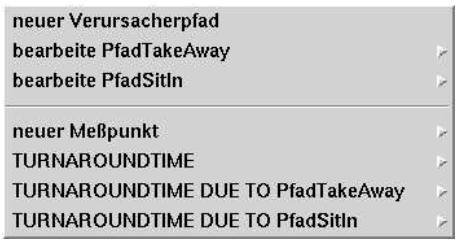


Abbildung 4.4: Lokales Menü der Funktioneinheit Schnellimbiss mit Anzeige der Messobjekte

neuen Verursacherpfad. Diesen nennen wir *PfadTakeAway*. Wir müssen nun nacheinander die Elemente angeben, welche den Verursacherpfad kennzeichnen. Dieses sind bei uns das **Prozesskettenelement** *Schnellimbiss_besuchen (TakeAway)* und die **Funktionseinheit** selbst. Wir wählen daher in dieser Reihenfolge aus den lokalen Menüs dieser beiden Elemente die Einträge „ergänzen mit...“. Nachdem wir die **Funktionseinheit** hinzugefügt haben, bekommen wir die Meldung, dass der Verursacherpfad vollständig und eingefügt worden ist. Anschließend können wir im lokalen Menü nicht nur allgemein (ALL), sondern auch speziell (PfadTakeAway) messen.

Bei der Definition des Messpunkts für den Gasttyp Sit-In verfahren wir analog mit dem **Prozesskettenelement** *Schnellimbiss.besuchen (SitIn)* und der **Funktionseinheit** selbst.

Abbildung 4.4 zeigt das lokale Menü der **Funktionseinheit** *Schnellimbiss* nach dem Abschluss der Experimentdefinitionen. Alle Messpunkte werden zusammen mit den Verursacherpfaden angezeigt.

4.4 Speichern der Experimentdefinition

Bevor wir uns nun der Simulation zuwenden können, müssen wir das Experiment speichern. Dies ermöglicht es uns auch, das Experiment später wieder aufzurufen, zu kontrollieren oder abzuändern. Vielleicht editieren wir auch nur einige Parameter und schauen die Auswirkungen an.

Zum Speichern wählen wir den Eintrag „Experiment speichern“ aus dem Menü „Experiment“. Vom Programm wird ein Name vorgeschlagen. Es ist sinnvoll, den vorgeschlagenen Namen zu benutzen, wir nennen unser Experiment „Imbiss.E1.exp“.

Kapitel 5

Analyse und Ergebnisdarstellung

Nachdem wir uns bisher mit Vorüberlegungen und Modellierung beschäftigt haben, wenden wir uns nun der Analyse und der Ergebnisdarstellung zu: Unsere bisherigen Bemühungen werden in gewisser Weise belohnt.

Wie die Modellierung werden auch Analyse und Ergebnisdarstellung durch die ProC/B-Tools unterstützt.

Die Analyse erfolgt in Form einer Simulation durch einen Löser.

5.1 ProC/B-Analysator

Aus dem Hauptmenü des ProC/B-GUIs starten wir zunächst den ProC/B-Analysator. Abbildung 5.1 zeigt das Hauptfenster des ProC/B-Analysators unmittelbar nach dem Start.

In der Eingabemaske „Datei“ wird das zu analysierende Experiment festgelegt. Wir wählen unsere zuvor gespeicherte Experimentdatei „Imbiss.E1.exp“ aus.

5.1.1 Einstellungen

Im oberen Bereich des Fensters werden zahlreiche Einstellungsmöglichkeiten angeboten, welche Einfluss auf den Verlauf und die Ergebnisse der Analyse haben. Die Analyse ist deterministisch. Modell- und Experimentbeschreibung sowie die folgenden Parameter legen das Ergebnis eindeutig fest. Abbildung 5.2 zeigt das Fenster des ProC/B-Analysators nach dem Eintragen der Einstellungen.

Modellzeit

Die Modellzeit gibt an, über wieviele Modellzeiteinheiten die Analyse durchgeführt werden soll. Wir hatten uns dazu entschieden, eine Modellzeiteinheit einer

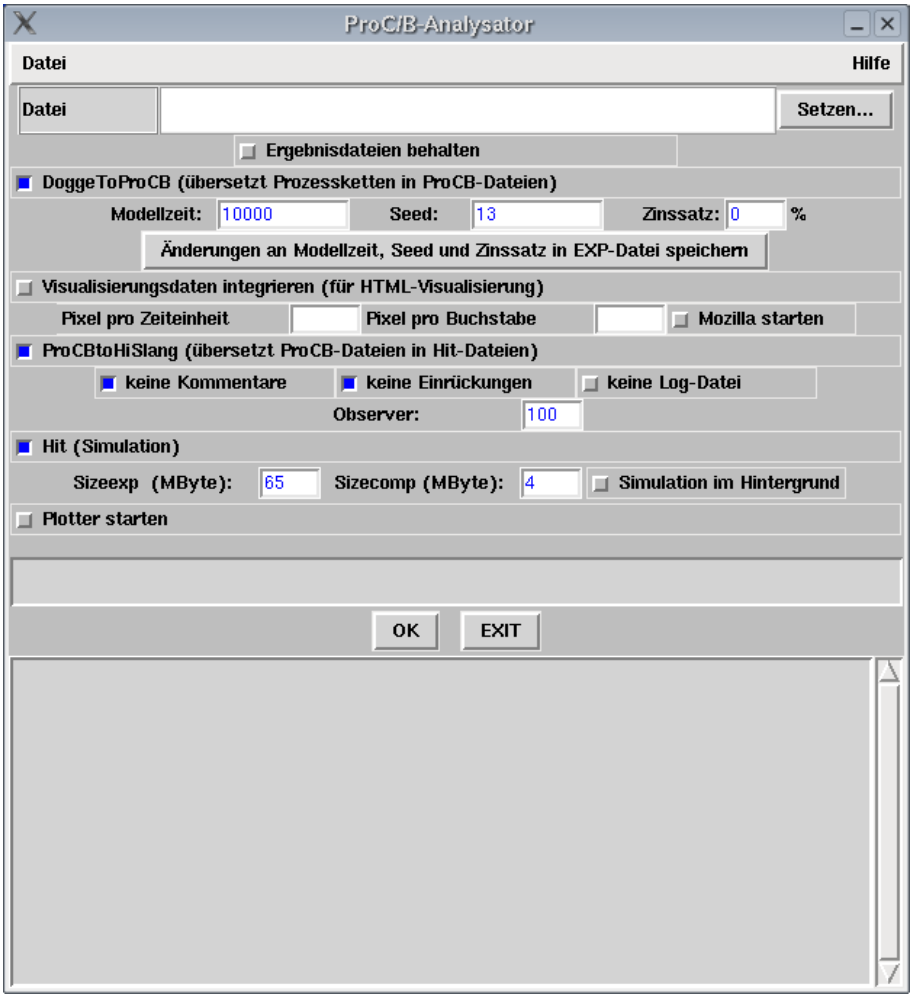


Abbildung 5.1: ProC/B-Analysator nach dem Start

Sekunde entsprechen zu lassen. Da wir nicht wissen, ob und wann die stationäre Phase erreicht wird, wählen wir 20 Stunden, was 72000 Modellzeiteinheiten entspricht; da unser Schnellimbiss auch nicht länger als 20 Stunden geöffnet hat, stellt dies auch keine Einschränkung dar.

Seed

Dies ist der Startwert für die Zufallszahlengenerierung des Analysetools. Wir belassen es beim Standardwert. Sinnvollerweise werden hier positive ungerade Zahlen verwendet, um den Zyklus der Zahlenfolge des Zufallszahlengenerators zu maximieren.

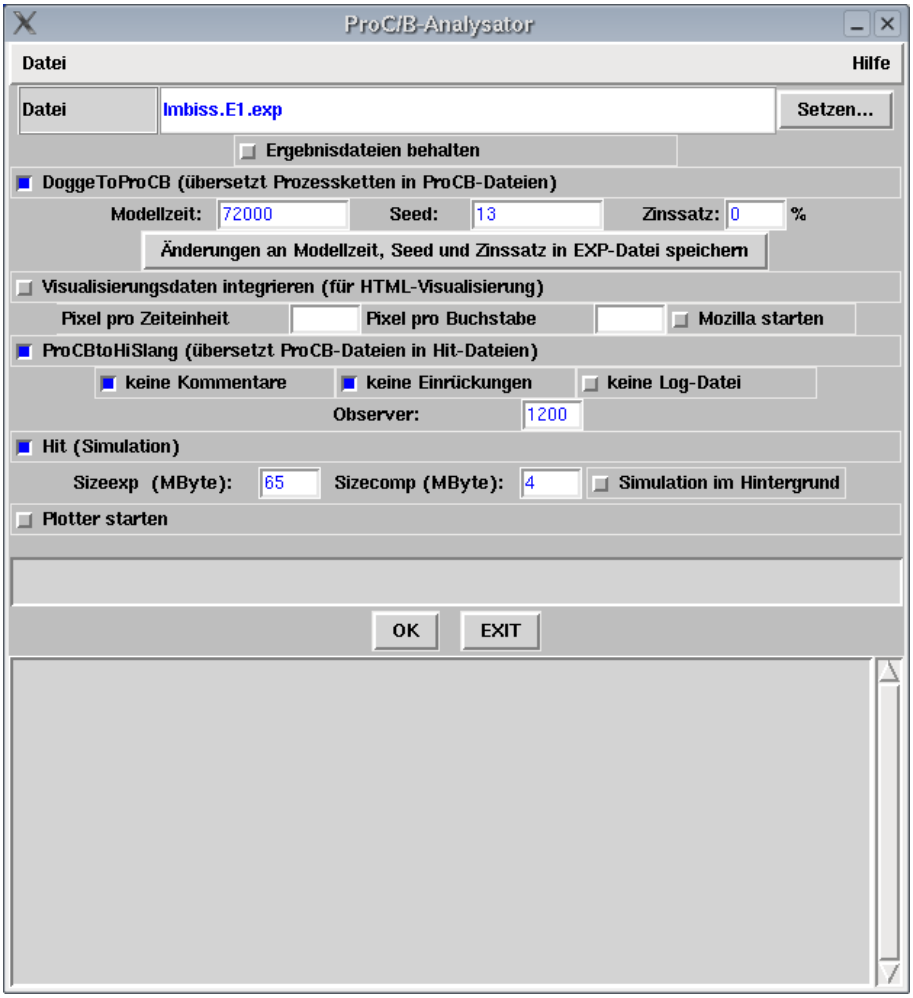


Abbildung 5.2: ProC/B-Analysator mit Einstellungen für den Start der Simulation

Zinssatz

Der Zinssatz wird für die Kostenrechnung benötigt. Für die aktuelle Analyse ist er nicht relevant.

Visualisierungsdaten integrieren

Diese Option ist auszuwählen, wenn eine die Analyseergebnisse als HTML-Datei ausgegeben werden sollen oder eine Analyse unter dem Gesichtspunkt der Kostenrechnung vorgenommen werden soll. Das folgende Kapitel 6 befasst sich mit diesem Thema.

keine Kommentare, keine Einrückungen und keine Log-Datei

Die Optionen „keine Kommentare“ und „keine Einrückungen“ sollten in der Regel ausgewählt sein. Für Debugging-Zwecke kann es unter Umständen sinnvoll sein, sie, sowie die Option „keine Log-Datei“, zu deaktivieren.

Observer

Die Ergebnisse werden in Form von Kurven in Diagrammen (Modellzeit -> Wert) dargestellt. Die Anzahl der Observerzeitpunkte gibt an, zu wievielen Modellzeitpunkten Messwerte ermittelt werden sollen. Je mehr Observerzeitpunkte angegeben werden, desto länger dauert die Analyse. Wir wählen 1200 Observerzeitpunkte. Dies bedeutet, dass alle 60 Modellzeiteinheiten, also jede volle Minute, Messwerte ermittelt werden.

Sizecomp

Hier wird festgelegt, wieviel Arbeitsspeicher dem Hit-Compiler zur Verfügung steht. Grundsätzlich gilt: Wenig Speicher verlangsamt die Analyse drastisch. Durch die Angabe von zuviel Arbeitsspeicher kann man allerdings den Rechner für andere Prozesse blockieren. Wir belassen es bei der Voreinstellung.

Sizeexp

Diese Einstellung gibt an, wieviel Arbeitsspeicher der Simulation zur Verfügung stehen soll. Die sinnvolle Wahl hängt von vielen Faktoren, auch von der Modellgröße ab. Als Richtwert gilt, wenigstens 4 MB und nicht mehr als die Hälfte des verfügbaren Realspeichers anzugeben. 65 MB haben sich bei unserem Modell als ausreichend erwiesen.

5.1.2 Simulation

Nachdem wir die Einstellungen vorgenommen haben, können wir die eigentliche Analyse, also die Simulation, starten. Wir wählen die Datei „Imbiss.E1.exp“ aus.

Während der Analyse werden Ausgabefenster geöffnet, die über den Fortschritt der Analyse informieren. Die Ausgabe entspricht der in Abbildung 5.3 gezeigten. Es werden die bisher abgearbeitete Modellzeit und verbrauchte CPU-Zeit angezeigt. Häufiges Auftreten der Garbage-Collection deutet daraufhin, dass der Speicher recht knapp bemessen ist, so dass man bei einer Wiederholung der Analyse des Modells doch mehr Speicher bewilligen sollte, sofern dies möglich ist.

Nach Abschluss der Simulation zeigt das Fenster des ProC/B-Analysators im unteren Bereich die Zusammenfassung der Ausgaben des Lösers, ähnlich der Ausgabe, welche in Abbildung 5.4 zu sehen ist.

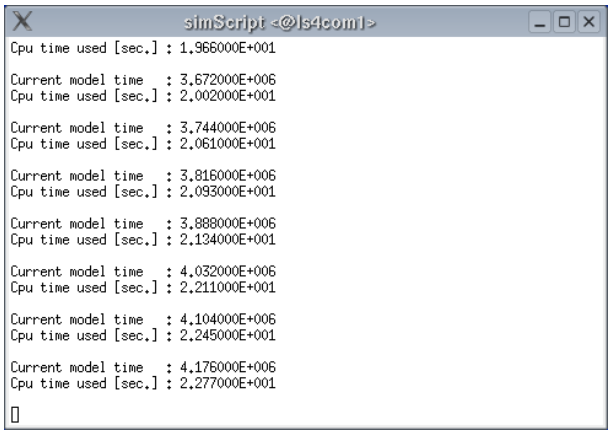


Abbildung 5.3: Ausgabe während der Simulation

5.2 ProC/B-Plotter

Nachdem wir die Simulation durchgeführt haben, können wir nun die Ergebnisse betrachten und auswerten. Dabei werden wir durch den ProC/B-Plotter unterstützt, welcher sich, wie die anderen Tools, aus dem Hauptmenü des ProC/B-GUIs starten lässt. Außerdem lässt sich der Plotter auch aus dem Analysator aufrufen. Abbildung 5.5 zeigt das Hauptfenster des ProC/B-Plotters.

Wir haben bei der Experimentdefinition zwei Auswertungen vorgesehen: die Verweildauer der Kunden innerhalb des Schnellimbisses und die Auslastung des Sitzplätze. Da es sich um Messergebnisse handelt, die nicht unmittelbar in Zusammenhang stehen, werden wir sie getrennt auswerten und darstellen.

Zunächst öffnen wir jedoch die Datei, welche unsere Ergebnisse enthält. Es handelt sich in Anlehnung an den Namen der Experimentbeschreibungsdatei um die Datei „t.Imbiss.E1.dum“.

Die Darstellung der Ergebnisse erfolgt in Form von Diagrammen. Dabei sind für die Kurven verschiedene Typen auswählbar, die aus den Ergebnissen berechnet werden. Als Graphtyp lassen sich Mittelwert (MeanT) und Mittelwert über dem Intervall zwischen zwei Observerzeitpunkten (MeanDeltaT) auswählen, als charakteristische Werte stehen zum Beispiel der Mittelwert (mean), positive und negative Konfidenz (confplus und confminus) sowie die Standardabweichung (standarddeviation) zur Auswahl.

5.2.1 Auswertung: Verweildauer im Schnellimbiss

Wenden wir uns zunächst der Verweildauer der Kunden im Schnellimbiss zu. Wir hatten drei Messungen in diesem Bereich definiert. Einerseits wollten wir die Verweildauer aller Gäste bestimmen, andererseits sollte dies nach GastTyp aufgeschlüsselt werden. Folglich werden wir drei Messkurven erhalten.

Abbildung 5.4: ProC/B-Analysator mit der Zusammenfassung der Ausgaben des Lösers

Kurvenauswahl

Wir wählen die Messkurven, wie in Abbildung 5.6 dargestellt wird, aus der Liste der Messkurven aus. Dies geschieht per Mausklick auf die Pfeile neben den Zeilen, welche die Kurven bezeichnen, im linken Bereich des Fensters. Im rechten Bereich ist es per Mausklick möglich, den Typ der Kurven und die Anzeigefarbe auszuwählen. Außerdem lassen sich die Kurven beschriften sowie Titel für das Diagramm sowie die beiden Achsen vergeben.

Diagramm

Nach dem Start des Plottvorgangs, wird das in Abbildung 5.7 dargestellte Diagramm angezeigt. Es zeigt die ausgewählten Messkurven und die Legende. Es

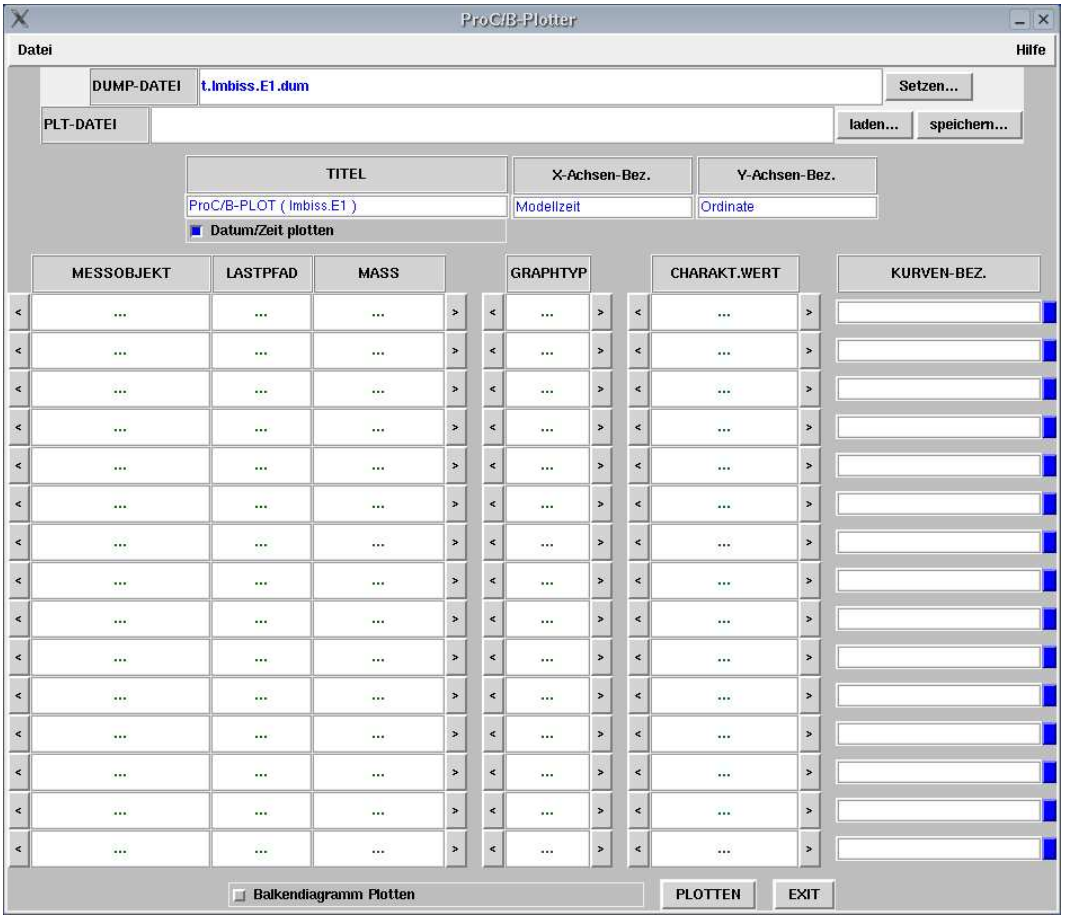


Abbildung 5.5: Proc/B-Plotter nach dem Start

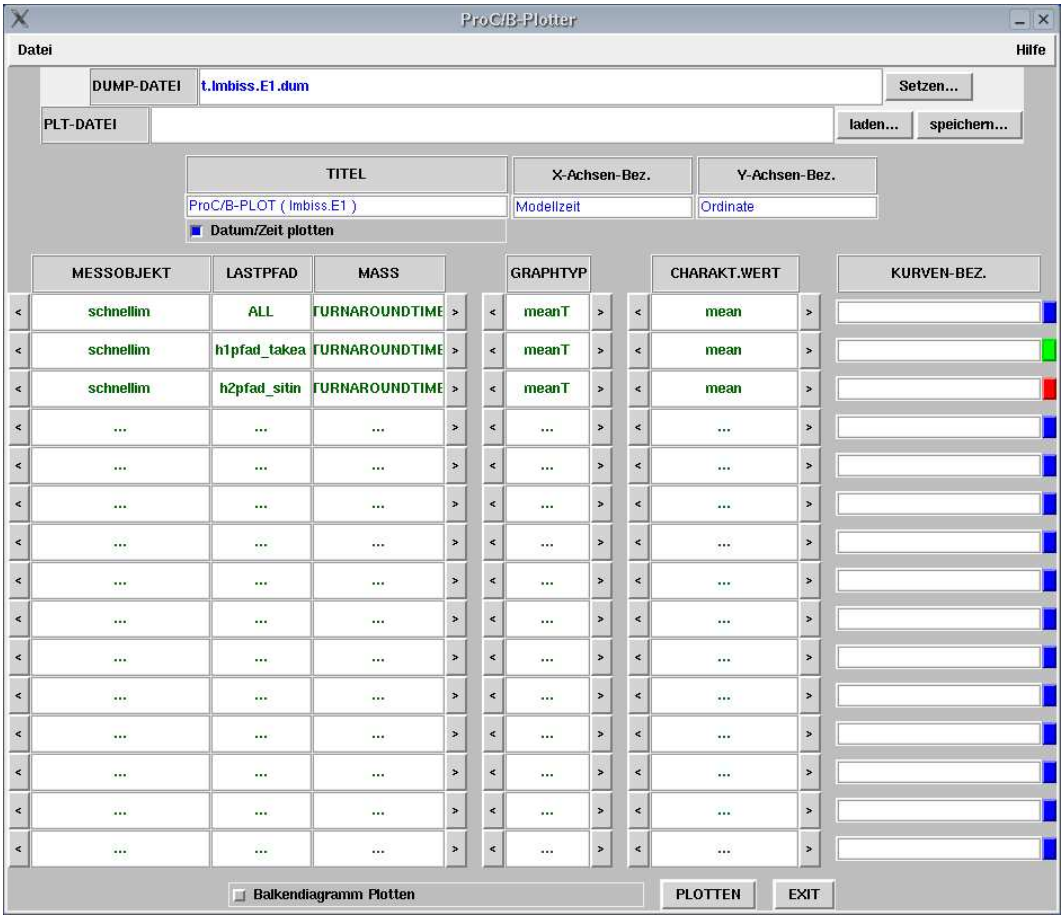


Abbildung 5.6: Auswahl der Kurven zur Auswertung der Verweildauer im Schnellimbiss

besteht, die Möglichkeit das Diagramm als Postscriptdatei zu exportieren und weiterzuverwenden.

Interpretationsversuch

Die Interpretation der Ergebnisse kann natürlich nicht durch die ProC/B-Tools erfolgen, sie bleibt dem Benutzer überlassen.

Zunächst einmal erkennen wir, dass die Kurven einen recht ähnlichen Verlauf haben. Sie steigen zunächst steil an und erreichen bald einen vermutlich stationären Bereich, d.h. sie bleiben dann annähernd auf einem Niveau. Diesen Bereich, ab dem sich die Entwicklung der Mittelwerte stabilisiert hat, untersuchen wir nun.

Die drei Kurven können wir nun als parallele Gerade sehen. Dies bedeutet zunächst, dass die Verweildauer nicht weiter steigt. Im Sinne unseres Schnellimbisses heißt das, dass die Aufenthaltsdauer vermutlich beschränkt ist und sich

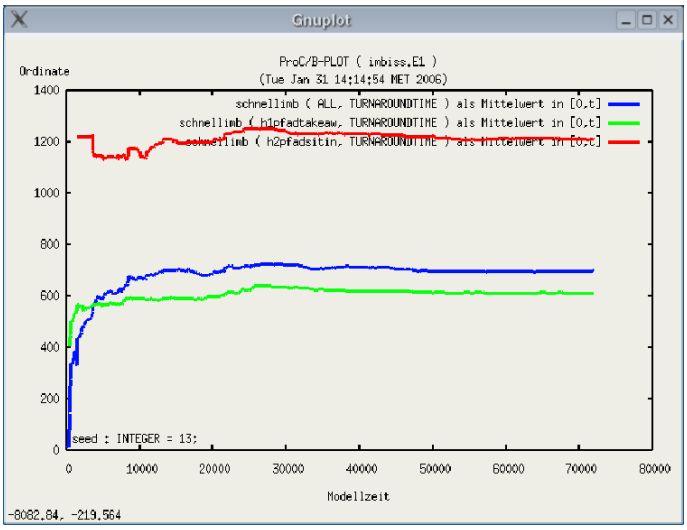


Abbildung 5.7: Diagramm mit Kurven zur Auswertung der Verweildauer im Schnellimbiss

- anschaulich betrachtet - keine immer länger werdende Schlange wartender Kunden bildet.

Aus der Legende bzw. der Farbcodierung geht hervor, dass sich die TakeAway-Gäste im Mittel etwa 600 Zeiteinheiten, also zehn Minuten lang, im Schnellimbiss aufhalten. Die SitInGäste benötigen etwa doppelt so lange.

Die mittlere Verweildauer aller Gäste liegt nur minimal oberhalb der Verweildauer der TakeAwayGäste. Dies liegt daran, dass das der Schnellimbiss viel häufiger von TakeAwayGästen besucht wird, so dass die Verweildauer der SitInGäste bei der Mittelwertbildung einen geringeren Einfluss hat.

5.2.2 Auswertung: Auslastung der Sitzplätze

Nach der Verweildauer werten wir nun die Auslastung der Sitzplätze aus.

Kurvenauswahl

Wir möchten uns nicht nur die Mittelwerte anschauen, sondern wählen, wie in Abbildung 5.8 gezeigt wird, auch die Kurventypen Standardabweichung sowie positive und negative Konfidenz aus.

Diagramm

Das Diagramm 5.9 zeigt die Kurvenschar an. Positive und negative Konfidenz werden erst nach dem ersten Observerzeitpunkt dargestellt.

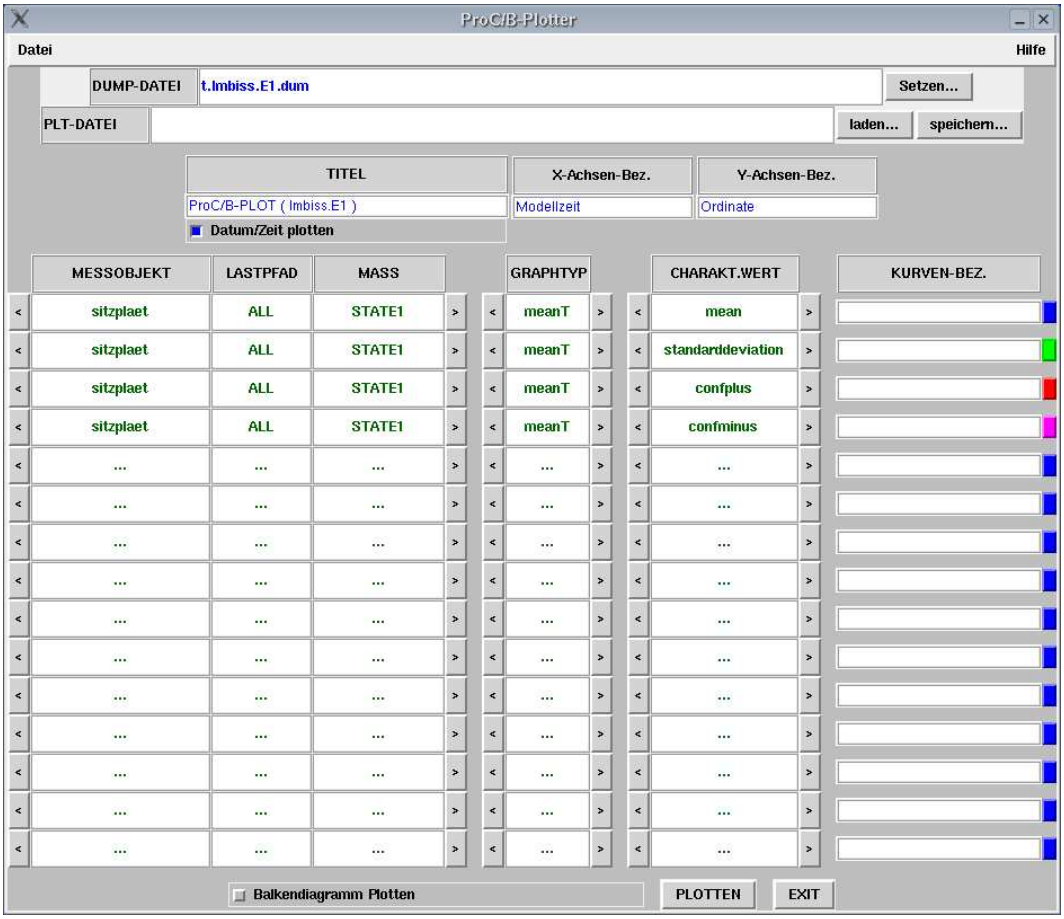


Abbildung 5.8: Auswahl der Kurven zur Auswertung der Auslastung der Sitzplätze

Interpretationsversuch

Auch diese Ergebnisse lassen sich interpretieren.

Versuchen wir erneut die stationäre Phase zu identifizieren. Der Mittelwert scheint sich bei 2,5 einzupendeln. Positive und negative Konfidenz bilden einen Schlauch um den Mittelwert herum. Die Standardabweichung liegt offenbar bei 1,5. Die stationäre Phase wurde jedoch noch nicht erreicht, möchte man genauere Ergebnisse erhalten, so müsste man die Simulation mit längerer Simulationsdauer wiederholen.

Dennoch kann man versuchen Antworten auf Fragen nach der erforderlichen Sitzplatzkapazität zu finden. Der Mittelwert schwankte um 2.5 - benötigt der Schnellimbiss also nur 2,5 Sitzplätze?

Theoretisch ja. Da es sich aber um den Mittelwert der Auslastung während der bisherigen Experimentdauer handelt und die tatsächliche Auslastung schwankt, die Standardabweichung liegt ja bei 1,5, sorgte eine Beschränkung auf zwei

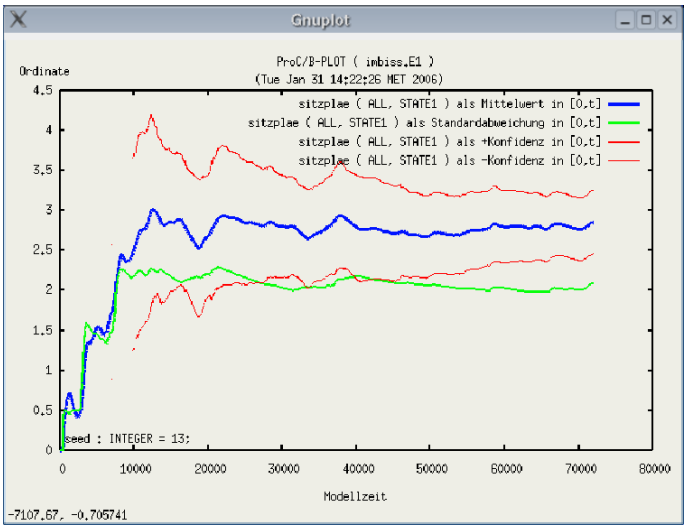


Abbildung 5.9: Diagramm mit Kurven zur Auswertung der Auslastung der Sitzplätze

Sitzplätze dafür, dass das Modell nicht mehr stationär ist, da die Schlange wartender Kunden immer länger würde. Auch drei Sitzplätze werden vermutlich zu langen Wartezeiten führen.

5.3 Zusammenfassung

Wir haben eine Simulation unseres Modells und dabei verschiedene Auswertungen durchgeführt.

Insgesamt sehen wir, dass sich die Ergebnisse mit Hilfe des Modells und seiner Parameter erklären lassen. Ferner decken sich die Ergebnisse offenbar mit den Erfahrungen, die wir beim Besuch eines Schnellimbisses gewonnen haben. Dies spricht dafür, dass das Modell eine gute Annäherung der Realität darstellt.

Nun können wir das Modell verfeinern oder abändern und die Auswirkungen beobachten. Interessant wäre beispielsweise, mit wie vielen oder besser wie wenig Sitzplätzen man noch vertretbare Verweilzeiten erhält. Auch der Personalbestand lässt sich vermutlich reduzieren. Was geschieht, wenn man eine Kasse schließt?

Wir werden diese Untersuchungen nicht vorführen, sondern überlassen sie Ihnen. Sie können sich dabei einfach an unserer Vorgehensweise orientieren und dabei ggf. auf die Online-Hilfen der Tools zurückgreifen.

Kapitel 6

Kostenrechnung

Bisher haben wir uns mit der Modellierung eines Schnellimbisses und dessen Auswertung unter Gesichtspunkten der Leistungsbewertung beschäftigt. Es ist jedoch auch möglich, Auswertungen hinsichtlich der Ökonomie durchzuführen. Wir werden im Folgenden das Modell so anpassen, dass diese Auswertungen möglich sind, eine Analyse durchführen und die Ergebnisse auswerten.

6.1 Einführung

Die Leistungsbewertung ermöglicht es, ein Modell hinsichtlich Auslastung, Durchsatz, Verweilzeiten und ähnlicher Aspekte auszuwerten. Ergebnisse lauten in unserem Fall beispielsweise, es sind ausreichend viele Sitzplätze vorhanden, oder das Schließen einer Kasse verursacht eine immer länger werdende Schlange wartender Kunden.

Interessant sind aber auch Antworten auf Fragen nach den Kosten, die beispielsweise durch das Öffnen einer zweiten Kasse entstehen oder durch die Bereitstellung weiterer Sitzplätze. Geschäftsleute stellen sich natürlich auch die Frage, ob es evtl. sinnvoll wäre, den Schnellimbiss zu schließen oder gar nicht erst zu eröffnen, weil der zu erwartende Gewinn unterhalb des Gewinns liegt, den man z.B. durch eine andere Kapitalanlage, z.B. Zinsen, erwirtschaften könnte.

Die Kostenrechnung soll dabei helfen, Antworten auf diese Fragen zu finden.

6.1.1 Unterstützung

Die Kostenrechnung ist ähnlich der Leistungsbewertung in die ProC/B-Tools integriert. Modellierung und Analyse werden auf die gleiche Weise durchgeführt, für die Ergebnisdarstellung existiert ein weiteres Tool. Leistungsbewertung und Kostenrechnung schließen sich nicht gegenseitig aus, sie können mit demselben Modell sogar zeitgleich durchgeführt werden. Die durch Simulation ermittelten Kosten können nach dem Verursacher aufgeschlüsselt ausgewertet werden.

6.1.2 Beschränkungen

Die Kostenrechnung unterstützt derzeit keine Externen Funktionseinheiten. Dies bedeutet, dass wir unser Modell des Schnellimbisses anpassen müssen, so dass es keine Externen Funktionseinheiten benötigt. In Zukunft soll diese Einschränkung entfallen.

6.2 ProC/B-Editor

Im ProC/B-Editor besteht die Möglichkeit, in den Attributmasken der Funktionseinheiten festzulegen, welche Fixkosten, welche Kosten pro Aufruf eines Dienstes bzw. Zeiteinheit, die der Aufruf des Dienstes in Anspruch nimmt, etc. anfallen. Dazu werden die entsprechenden Größen in den Attributmasken auf der Karteikarte „Kosten“ eingetragen.

6.2.1 Modell

Zunächst müssen wir einen Weg finden, der es uns ermöglicht, auf die Externen Funktionseinheiten zu verzichten. Wir haben die Ressource *Personal* gemeinsam genutzt. Eine mögliche Lösung besteht darin, diese Ressource aufzuspalten und den Funktionseinheiten, welche Personal benötigen, jeweils eigenständiges Personal zuzuordnen. Wir werden diese Möglichkeit nun umsetzen.

Die gemeinsam verwendete Ressource *Personal* befindet sich innerhalb der Funktionseinheit *Schnellimbiss*. Wir öffnen diese im ProC/B-Editor.

In der Attributmaske des Servers *Personal* setzen wir den Wert Kapazität auf 1. Damit steht der Prozesskette *Aufraeumen* nur noch ein Mitarbeiter zur Verfügung. An den Funktioneinseinheiten *Kueche* und *Theke* lösen wir nun die Verbindung zum Server *Personal*, indem wir in der Attributmaske die Prozesszuordnung und den jeweiligen virtuellen Parameter löschen.

Die Abbildung 6.1 zeigt die Innenansicht des Schnellimbisses nach der Änderung.

Nun müssen wir aber den Funktionseinheiten *Kueche* und *Theke* noch eigenes Personal zuordnen. Hierzu werden wir in beiden Funktionseinheiten lokal Server anlegen, welche das nun nicht mehr gemeinsam verwendete Personal repräsentieren.

Die Abbildungen 6.2 und 6.3 zeigen die Funktionseinheiten *Kueche* bzw. *Theke* nach den Änderungen.

Damit haben wir nun die notwendigen Vorbereitungen getroffen und das Modell so angepasst, dass es sich zum Einsatz mit der Kostenrechungsfunktion eignet.

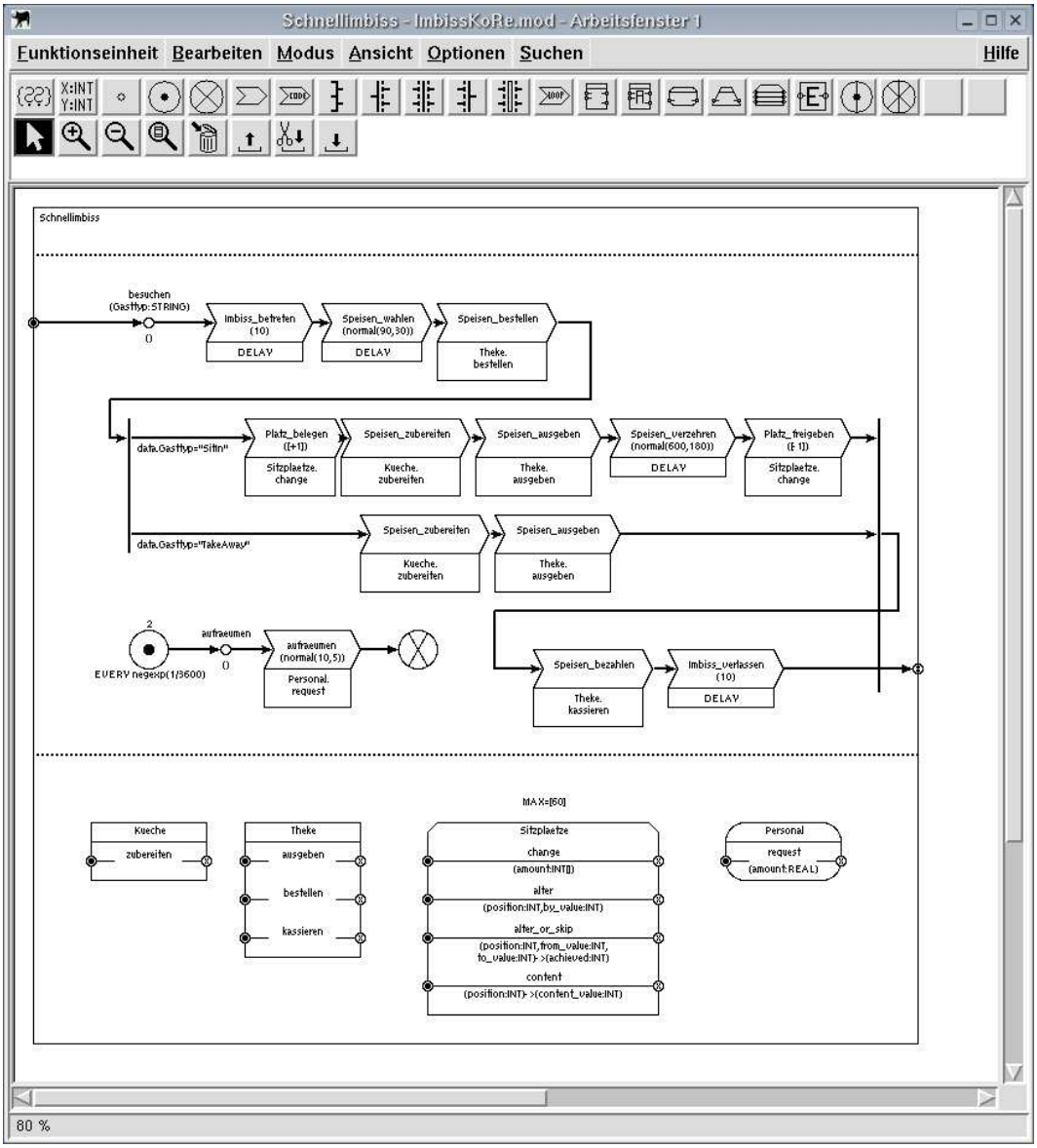


Abbildung 6.1: Innenansicht des Schnellimbisses nach der Änderung

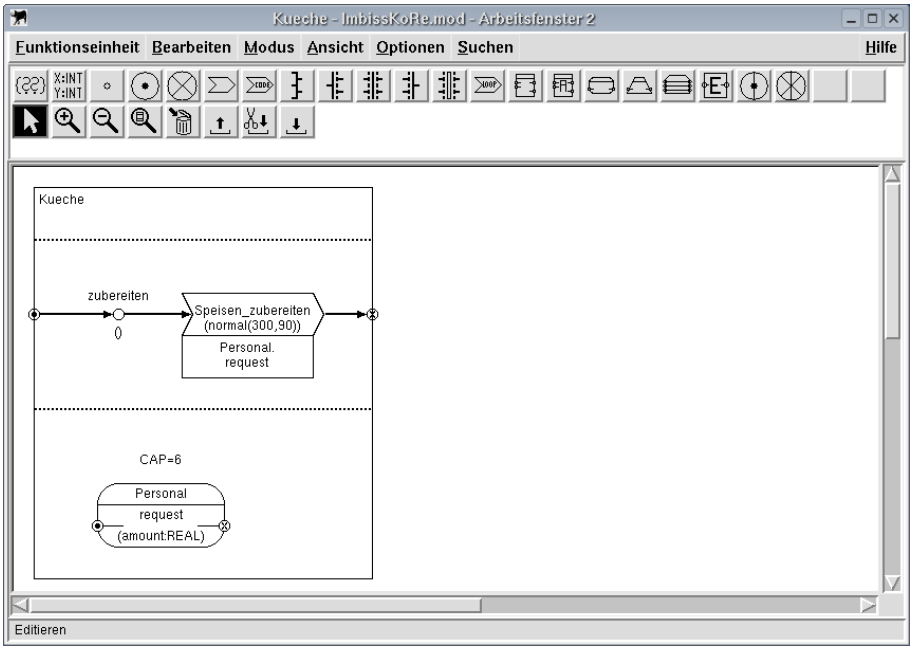


Abbildung 6.2: Ansicht der Küche nach der Änderung

6.2.2 Attribute

Für die Kostenrechnung müssen einige Attribute an den Modellelementen festgelegt werden. Die für die Kostenrechung relevanten Elemente, insbesondere die Funktionseinheiten, besitzen in der Attributmaske die Karteikarte „Kosten“. Dort lassen sich die Attribute, welche für die Kostenrechnung benötigt werden, einstellen.

Wir analysieren nun die Kosten, welche im Zusammenhang mit den Sitzplätzen entstehen. Wir wissen, dass die Sitzplätze dadurch Kosten verursachen, dass sie existieren, bei der Anschaffung oder durch die Miete des Raumes, die sie in Anspruch nehmen. Abhängig von der tatsächlichen Nutzung der Sitzplätze fallen weitere Kosten an, z.B. Verschleiß oder Reinigung. Gäste, die den Schnellimbiss besuchen, setzen einen bestimmten Betrag um, welcher bei Gästen, die ihre Speisen vor Ort verzehren, erfahrungsgemäß höher liegt. Ferner besteht die Möglichkeit, einen Zinssatz festzulegen, so dass man feststellen kann, ob die Investition überhaupt sinnvoll war.

Wir legen folgende Werte fest: Fixkosten in Höhe von 10000 und Betriebskosten (State) in Höhe von [3] bei den Sitzplätzen – die eckigen Klammer sind erforderlich, da es sich im einen Vektor handelt – sowie einen Umsatz von 5 bei den SitIn-Gästen und 3 bei den TakeAway-Gaesten.

Wir speichern das modifizierte Modell unter dem Namen „ImbissKoRe.mod“.

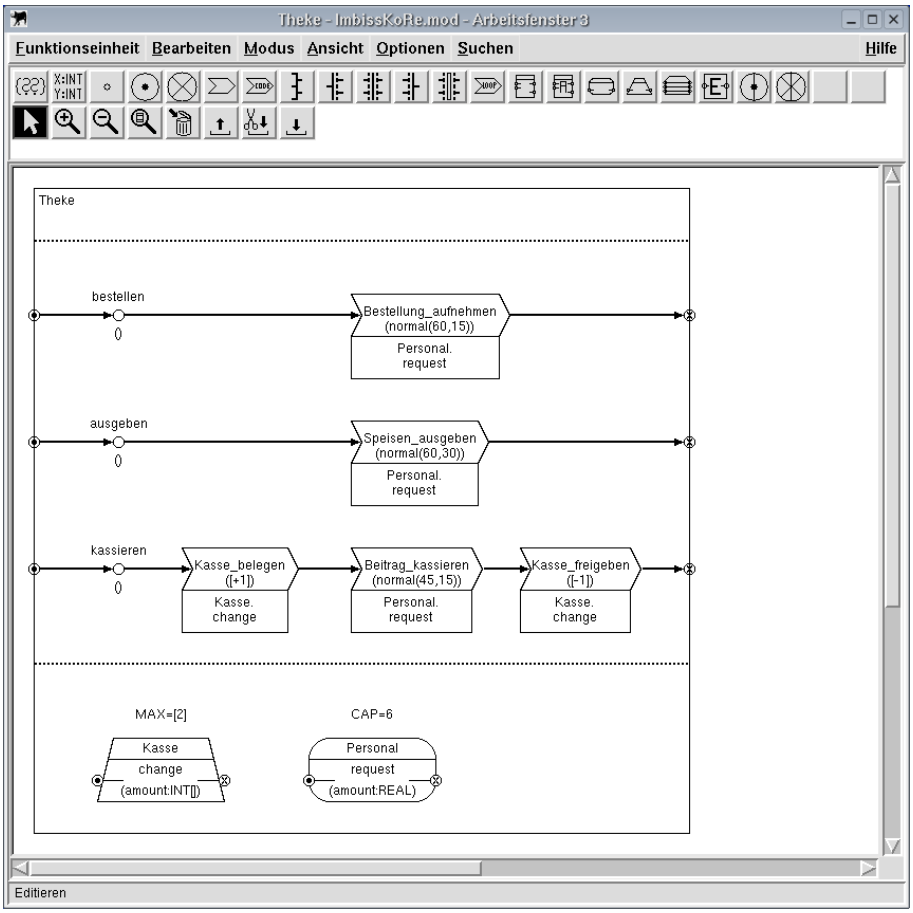


Abbildung 6.3: Ansicht der Theke nach der Änderung

6.3 Experimentdefinition

Wie bei der Leistungsbewertung definieren wir ein Experiment. In diesem müssen aber keine Messobjekte definiert werden, wenn man ausschließlich die Kosten analysieren möchte, da das Kostenrechnungsmodul alle notwendigen Messobjekte automatisch anlegt, d.h. weitere Schritte zur Experimentdefinition sind nicht erforderlich. Grundsätzlich ist es aber möglich parallel zur Kostenrechung auch eine Leistungsbewertung durchzuführen.

Das Experiment speichern wir unter dem Namen „ImbissKoRe.E1.exp“.

6.4 ProC/B-Analysator

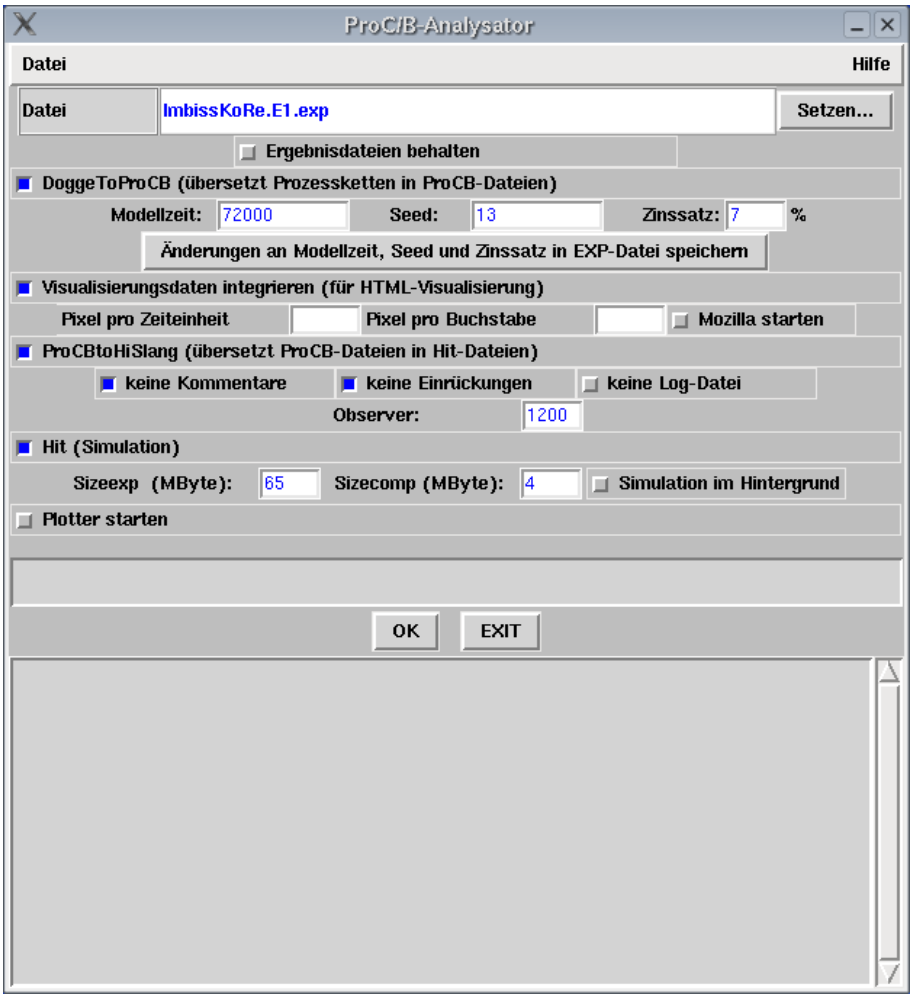


Abbildung 6.4: ProC/B-Analysator mit den Optionen der Kostenrechnung

Nach dem Start des ProC/B-Analysators, der auch in diesem Fall die Simulation starten wird, tragen wir die gleichen Parameter ein, wie beim vorhergegehen-

den Experiment. Allerdings heißt die zu analysierende Experimentdatei nun „ImbissKoRe.E1.exp“. Wir schalten nun die Option „Visualisierung“ ein.

Die Abbildung 6.4 zeigt den ProC/B-Analysator mit den Optionen der Kostenrechnung.

6.4.1 Auswahl der Messobjekte

Nach dem Start der Simulation werden zahlreiche Pfade zur Analyse vorgeschlagen. Es ist sinnvoll, alle auszuwählen und alle Fragen mit „ ja“ zu beantworten.

6.4.2 Simulation

Der Ablauf der Simulation entspricht dem aus Kapitel 5 bekannten Ablauf. Nachdem diese beendet wurde, wird wieder eine Zusammenfassung der Ergebnisse angezeigt.

6.5 Ergebnisdarstellung

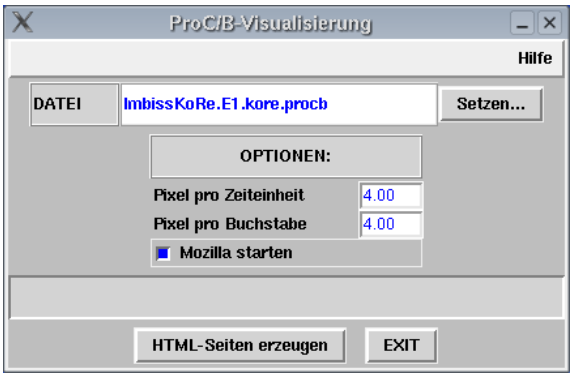


Abbildung 6.5: Einstellungen der Kostenvisualisierung

Da wir nun eine Auswertung in Form der Visualisierung der Ergebnisse vornehmen möchten, starten wir die Kostenvisualisierung aus dem Hauptmenü des ProC/B-GUI über den Menüpunkt „Visualisierung“.

Zunächst erscheint das in Abbildung 6.5 gezeigte Fenster, in welchem die zur Durchführung der Auswertung benötigten Parameter eingegeben werden können. Notwendig ist es, die entsprechende Datei auszuwählen. Ferner besteht die Möglichkeit, den Maßstab der Grafiken anzupassen.

Anschließend werden aus den Simulationsergebnissen HTML-Dateien erzeugt, welche über einen Webbrowser angezeigt und ausgedruckt werden können.

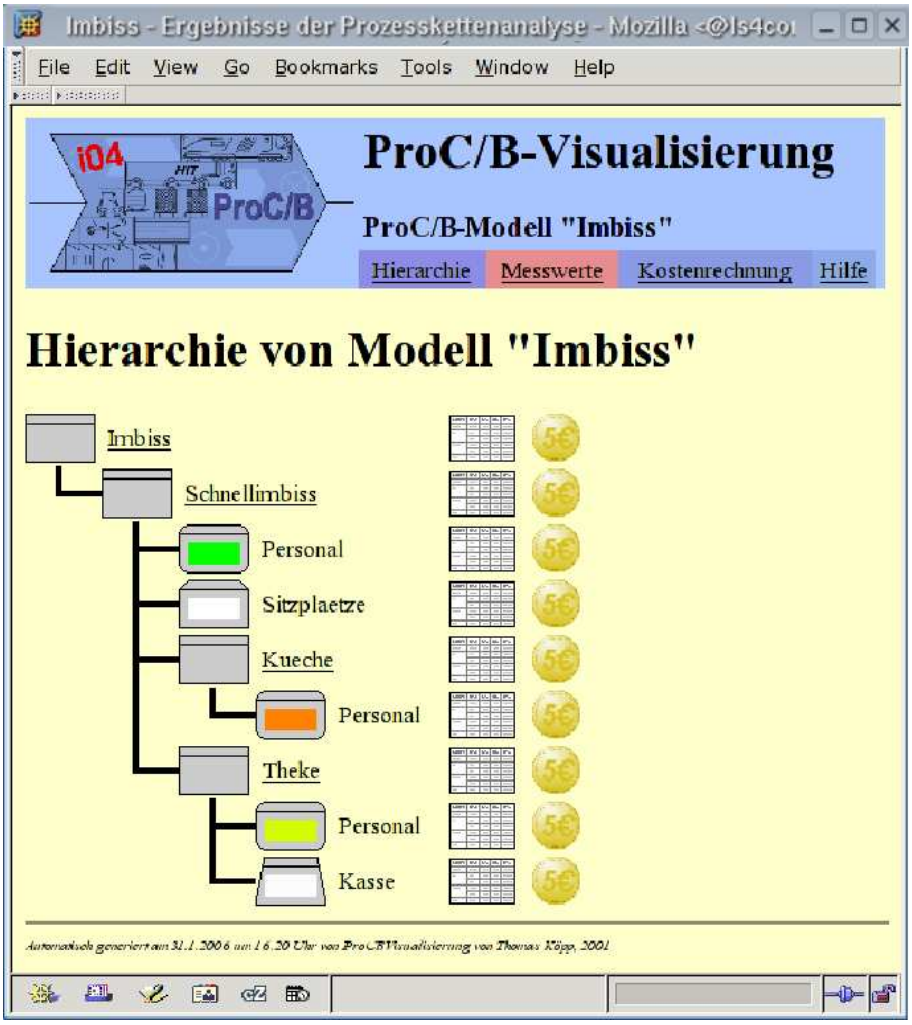


Abbildung 6.6: Webbrowser mit Überblick über die Hierarchie

6.5.1 Darstellung im Webbrowser

Der gestartete Webbrowser zeigt zunächst die in Abbildung 6.6 dargestellte Webseite mit dem Hierachiebaum des Modells an.

Im oberen Bereich der Seite werden vier Menüpunkte angeboten. Der Menüpunkt „Hierarchie“ ermöglicht die Rückkehr zur Startseite, der Menüpunkt „Messwerte“ bietet Zugang zur Visualisierung der Ergebnisse der Leistungsbewertung. Die eigentliche Kostenvisualisierung wird über den Menüpunkt „Kostenrechnung“ gestartet. Hinter dem Menüpunkt „Hilfe“verbirgt sich die Online-Hilfe der Kostenvisualisierung, in der die Handhandung des Tools, aber auch Bedeutung der Symbole und Abkürzungen erklärt werden.

Viele der Beschriftungen oder Grafiken sind mit Hyperlinks hinterlegt, so dass man auf weitere Seiten gelangen kann. Beispielsweise ist der Text „Schnellimbiss“ mit der Anzeige der Ergebnisse der Leistungsbewertung jener Funktionseinheit verknüpft.

6.5.2 Auswertung der Ergebnisse

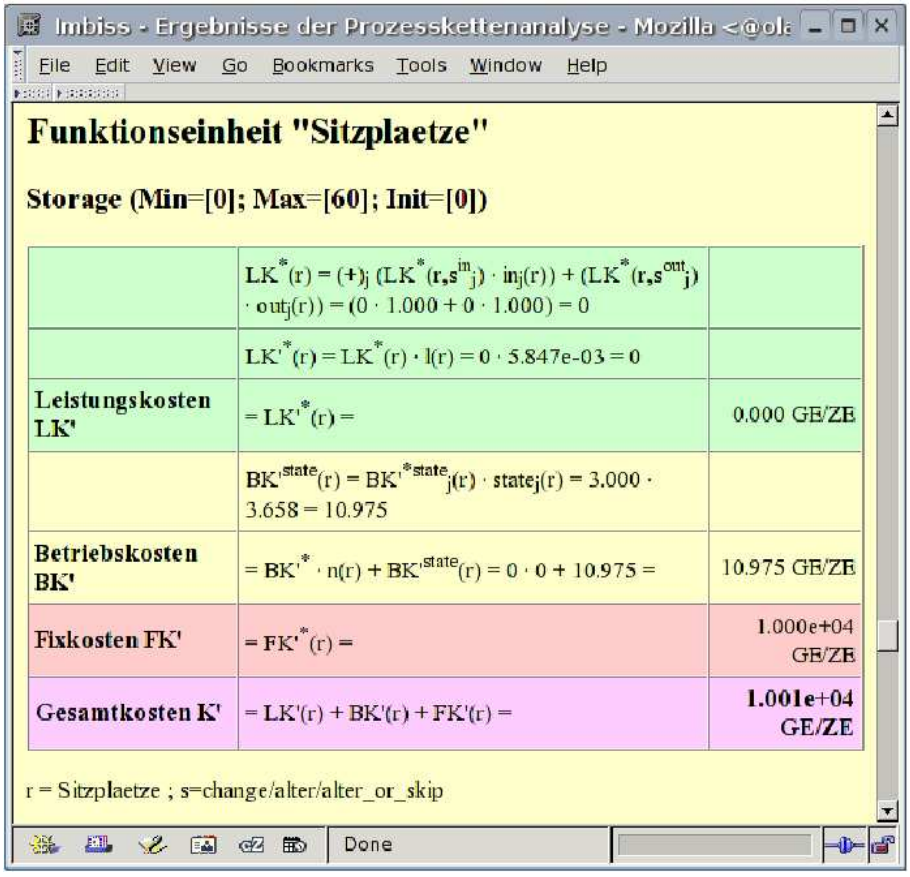


Abbildung 6.7: Webbrowser mit Kostenauswertung der Sitzplätze

Durch Auswahl des Menüpunkts „Kostenrechnung“ gelangen wir auf die Webseite „Prozesskostenrechnung für Imbiss“, an deren unterem Ende sich der in Abbildung 6.7 gezeigte Abschnitt mit Ergebnissen der Funktionseinheit *Sitzplaetze* befindet.

Die Ergebnisse werden in einer Tabelle dargestellt, in welcher auch die Formeln zur Berechnung angezeigt werden. Die Teilergebnisse sowie das Gesamtergebnis ergeben sich folgendermaßen:

Die Leistungskosten betragen 0 Geldeinheiten pro Zeiteinheit, da wir im Modell keine Leistungskosten angegeben hatten. Die Betriebskosten ergeben sich als Produkt der Kosten der Belegung der Sitzplätze von 3, die wir im Modell angegeben hatten, multipliziert mit der Belegung der Sitzplätze von 3,658, welche sich aus der Simulation ergibt. Somit betragen die Betriebskosten 10,975 Geldeinheiten pro Zeiteinheit. Die Fixkosten hatten wir im Modell mit 10.000 Geldeinheiten pro Zeiteinheit vorgegeben.

Die Höhe der Gesamtkosten ergibt sich aus der Summe der Leistungskosten, Betriebskosten und Fixkosten in Höhe von 10.010,975 Geldeinheiten pro Zeiteinheit an, welche in der Exponentialdarstellung als 1.001e+04 angezeigt wird.

Die Formelzeichen werden in der Online-Hilfe erklärt. Weitergehende Informationen findet man in [7].

Literaturverzeichnis

[1] H. Beilner:
Leistungsbewertung von Rechen- und Kommunikationssystemen,
Vorlesungsskript, Universität Dortmund, 1999.

[2] M. Hierweck, J. Finzel, A. van Almsick, J. Kriege, M. Schwenke:
ProC/B-Editor - Handbuch, SFB-Bericht 06002, Universität Dortmund, 2006, ISSN 1612-1376
lokales File auf dem Server des Lehrstuhls für praktische Informatik:
/app/unido-i04/SFB559_M1/Dokumentation/
Toolbenutzung/Handbuch.pdf
oder extern als Handbuch.pdf-File im Installationsverzeichnis des ProC/B-Editors

[3] H. Beilner:
HI-SLANG reference manual,
Universität Dortmund, 1999.

[4] A. M. Law, W. D. Kelton:
Simulation Modelling and Analysis,
McGraw-Hill Higher Education, 2000

[5] H. Beilner, F. Bause, H. Tatlitürk, A. van Almsick, M. Völker:
Zum B-Modellformalismus - Version B1, SFB-Bericht 99002, Universität Dortmund, 1999.
lokales File auf dem Server des Lehrstuhls für praktische Informatik:
/app/unido-i04/SFB559_M1/B1HIT/old/ERGEBNIS_DOCS/
DOKUMENTE/B1-Bericht.pdf

[6] M. Eickhoff:
Statistische Auswertung und Erkennung der stationären Phase in der Simulation zustands-diskreter Systeme,
Diplomarbeit, Universität Dortmund, 2002.

[7] Th. Köpp:
Visualization of Performance and Cost Measures for Logistic Systems Derived from Process Chain Paradigms,
Diplomarbeit, Universität Dortmund, 2001.

Abbildungsverzeichnis

Sonderforschungsbereich 559

Bisher erschienene Technical Reports

03034 Frank Laakmann, Iwo Riha, Niklas Stracke, Stefan Weidt: Workbench-gestützte Konstruktion von Beschaffungsketten

03035 Iwo Riha, Stefan Weidt: Entwicklung einer Bewertungssystematik für Beschaffungsketten

04001 André Alberti, Bernd Hellingrath, Stefan Weidt, Markus Witthaut: Ergebnisse und Schlussforderungen der Simulationsexperimente im Szenario Automobilindustrie

04002 Kay Hömberg, Dirk Jodin, Maren Leppin: Methoden der Informations- und Datenerhebung

04003 Carsten Tepper: Prozessablauf-Visualisierung von ProC/B-Modellen

05001 Jochen Bernhard, Miroslav Dragan, Sigrid Wenzel: Evaluation und Erweiterung der Kriterien zur Klassifizierung von Visualisierungsverfahren für GNL

05002 Bernd Hellingrath, Sana Mehicic-Eberhardt, Markus Witthaut: Entwicklung eines Anaylserahmens für die Untersuchung organisatorischer Aspekte in der Supply Chain

05003 Dennis Müller, Mathias Stöber, Axel Thümmler: Einsatz der Response Surface Methode zur Optimierung komplexer Simulationsmodelle

05004 Dirk Jodin, Andreas Mayer: Automatisierte Methoden und Systeme der Datenerhebung

05005 Thomas Fender, Anne Krampe, Sonja Kuhnt: Kriterien für die Kategorisierung statistischer Methoden im Rahmen eines Methodennutzungsmodells zur Informationsgewinnung in GNL

05006 Kay Hömberg, Dirk Jodin, Maik Langenbach, Christian Kellner: Konzept einer logistischen Informationsbedarfsanalyse mit Hilfe von Basisprozessen und standardisierten Logistikdaten

05007 Hans-Werner Graf: Festlegung der Abfahrts- und Ankunftszeiten (Fahrplangestaltung)

06001 Iwo Riha: Grundlagen des Cost-Benefit-Sharing

06002 Jens Finzel, Michael Hierweck, Andreas van Almsick, Jan Sören Kriege, Mathias Schwenke: ProC/B-Editor – Handbuch

06003 Mirko Eickhoff, Michael Hierweck, Mathias Schwenke: Hands On ProC/B-Tools – Eine beispielorientierte Einführung in die Anwendung der ProC/B-Tools

Alle Technical Reports können im Internet unter
http://www.sfb559.uni-dortmund.de/
abgerufen werden. Für eine Druckversion wenden Sie
sich bitte an die SFB-Geschäftsstelle
e-mail: andrea.grossecappenberg@iml.fraunhofer.de