Simulation von Web Services

H. Alvandi,
Swen Habenberger
Sommersemester 2004
TeleSeminar
Uni Mannheim/Karlsruhe

Gliederung

1) Motivation
2) Dienstgüte / Quality of Service (QoS)
3) Beispielsimulator für DAML-S: Karmasim
   - Situation Calculus Language
   - Petrinetze
   - Karmasimbeispiel
4) Beispielsimulator für WSFL: JSIM
   - Service Composition and Execution Tool (SCET)
   - JSIM im Detail
5) Ausblick

Motivation für Simulationen

- (Web-) Serviceprovider wollen den Kunden einen möglichst hohen Standard an Qualität und Zuverlässigkeit liefern.
- Simulationen am Computer können teure Tests ersparen, sofern solche Tests überhaupt möglich sind.
- Die Webservice-Komposition ist ein neuer Forschungs-bereich, der Web Service-Technologie und Prozess-Komposition kombiniert.

Dienstgüte / Quality of Service

- Verfügbarkeit (Availability)
- Erreichbarkeit (Accessibility)
- Integrität (Integrity)
- Performance
- Zuverlässigkeit (Reliability)
- Regelwerk (Regulatory)
- Sicherheit (Security)

Verfügbarkeit / Availability

Repräsentiert die Wahrscheinlichkeit, dass ein Webservice verfügbar oder für die unmittelbare Verwendung bereit ist.

Wichtigstes Messkriterium ist

Time-to-repair (TTR)

=> Web Service Management

=> Web Service Clustering

Erreichbarkeit / Accessibility

Erreichbarkeit bedeutet:

Anfragen werden bearbeitet.

Gemessen werden die erfolgreichen Zugriffe auf den Webservice.

=> Load Balancing

=> Service pooling

Integrität / Integrity

- Stellt die Korrektheit der Ergebnisse und Datenintegrität sicher.

- Bei Abbruch einer Transaktion, müssen getätigte Veränderungen rückgängig gemacht werden können.

Performance

- Kenngrößen:
  - Durchsatz (Throughput)
  - Latenz (Latency)

- Ziel: höherer Durchsatz und niedrigere Latenz

Zuverlässigkeit / Reliabilty

- Fehler pro Monat oder Jahr
- Anzahl der zugesicherten und in der richtigen Reihenfolge zugestellten Nachrichten

Regelwerk / Regulatory

- strikte Einhaltung der Standards (wie SOAP, UDDI, WSDL)
- Berücksichtigung der Gesetze, Servicelevelvereinbarungen
- Angabe der verwendeten Version (z.B. SOAP Version 1.2)

Sicherheit / Security

- Verschlüsselte Nachrichten
- Vertrauenswürdigkeit / Authentifizierung
- Zugangskontrolle
- Verschiedene Ebenen des Zugangs zum Webservice
- XML Encryption oder Key Management Specification
- P3P (Platform for Privacy Preferences)

QoS für Web-Service

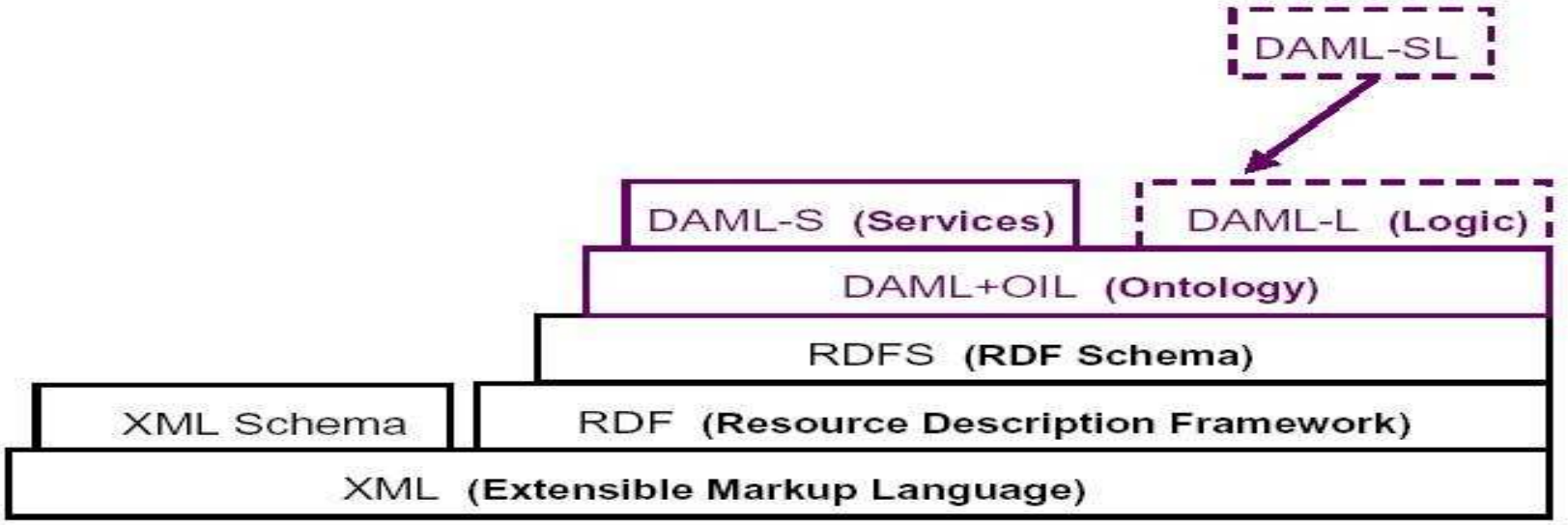
- Accessibility
- Availability
- Interoperability
- Security

Simulationsprogramme

- HaarnessIT for .Net
- Optimyz WebServiceTester 3.0 (BPEL4WS)
- SOAPscope
- SOAPtest
- Testmaker
- Sitescope
- PushToTest
- Karmasim
- Jsim
- OPnet
- ...

DAML-S

- DARPA Agent Markup Language (DAML)
- DAML-Services (DAML-S)
- 11/2003 Umbennung in OWL-S
- Aktuell 282-Ontologien vorhanden

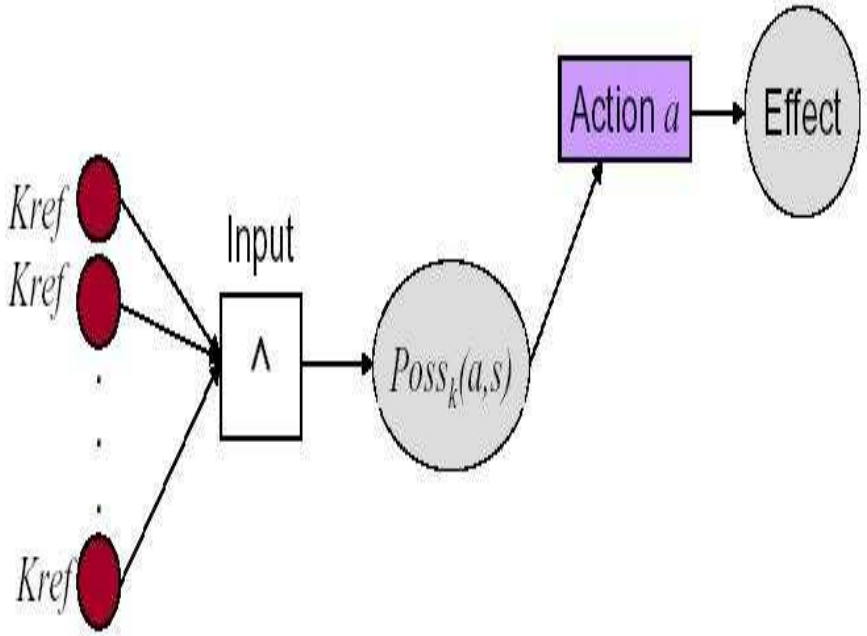


Situation Calculus Language

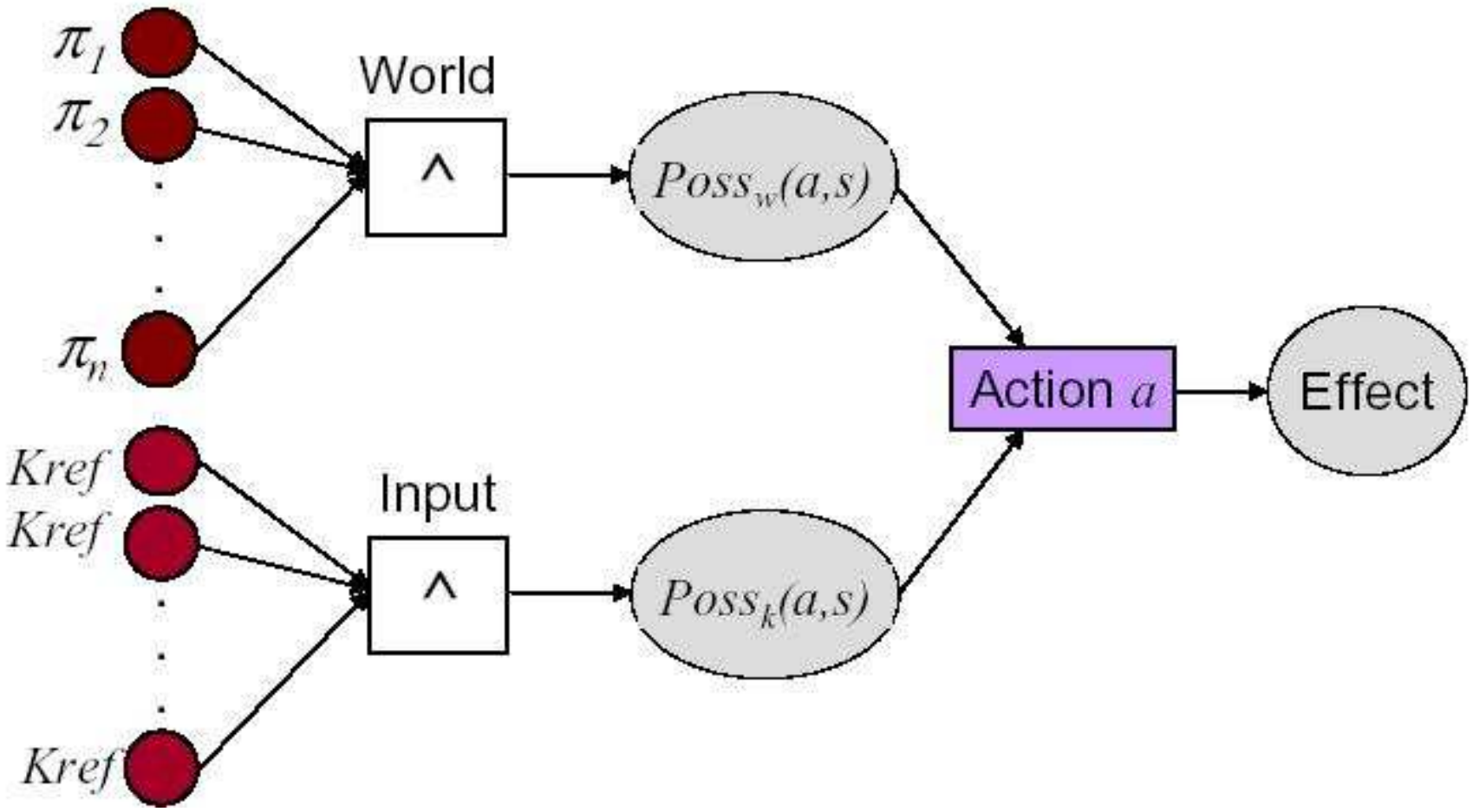
- Alle Veränderungen sind das Resultat einer Aktion, ausgelöst vom Agenten.
- Simulationen sind Sequenzen von Aktionen, beginnend von einem Ausgangsort.

- Preconditions und Input
- Conditional Effects und Output

Petri-Netze

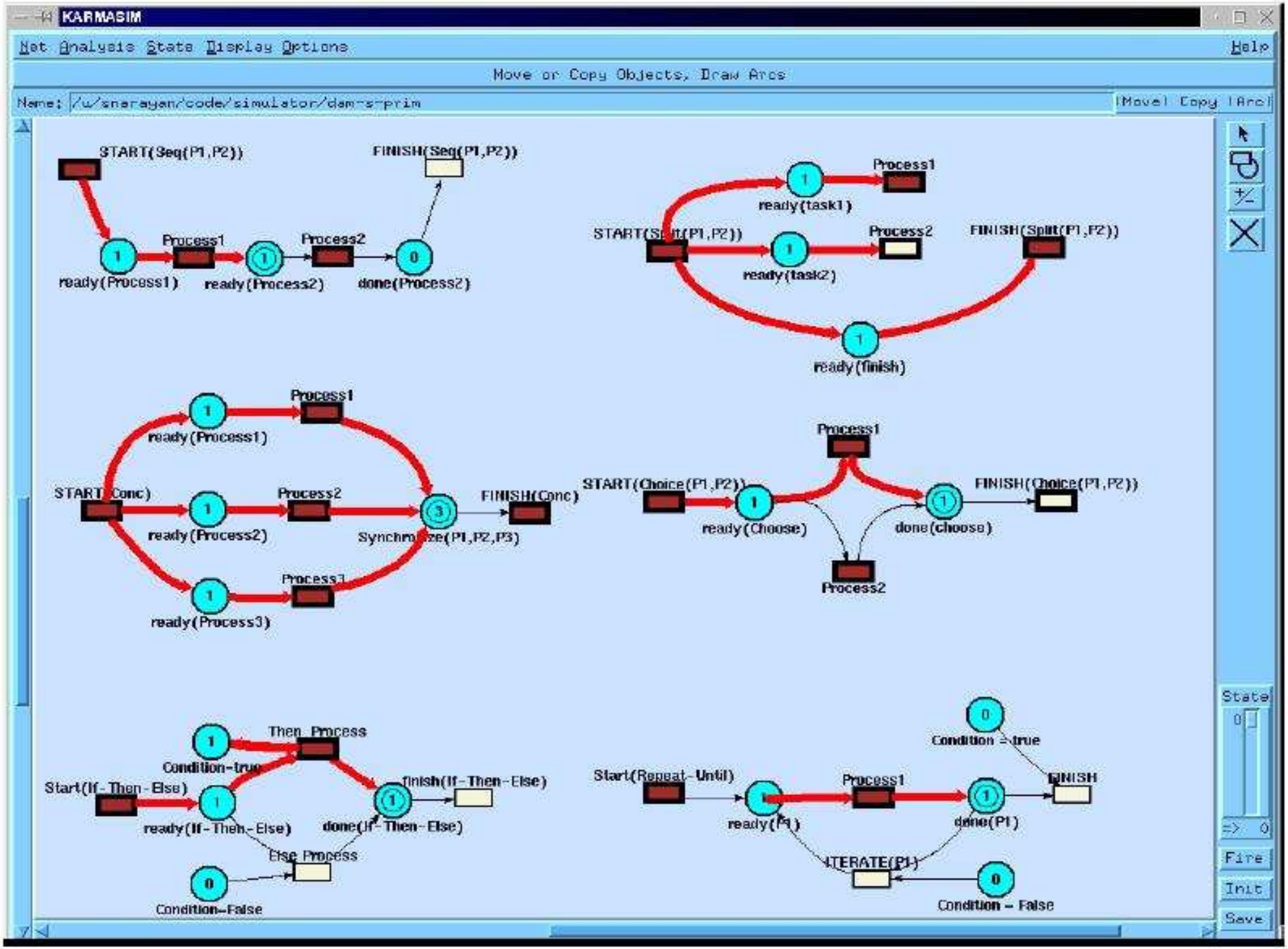


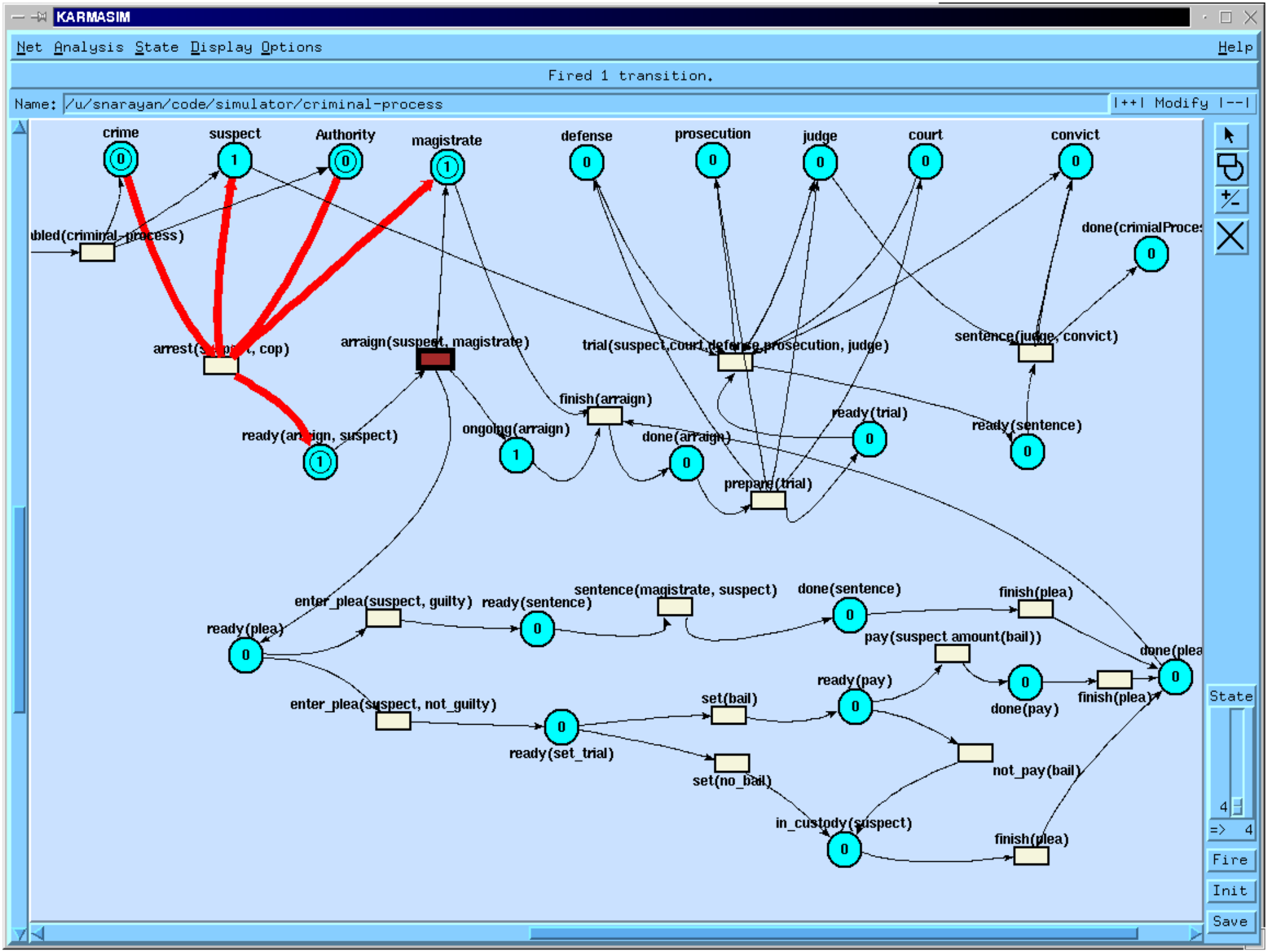
Petrinetze

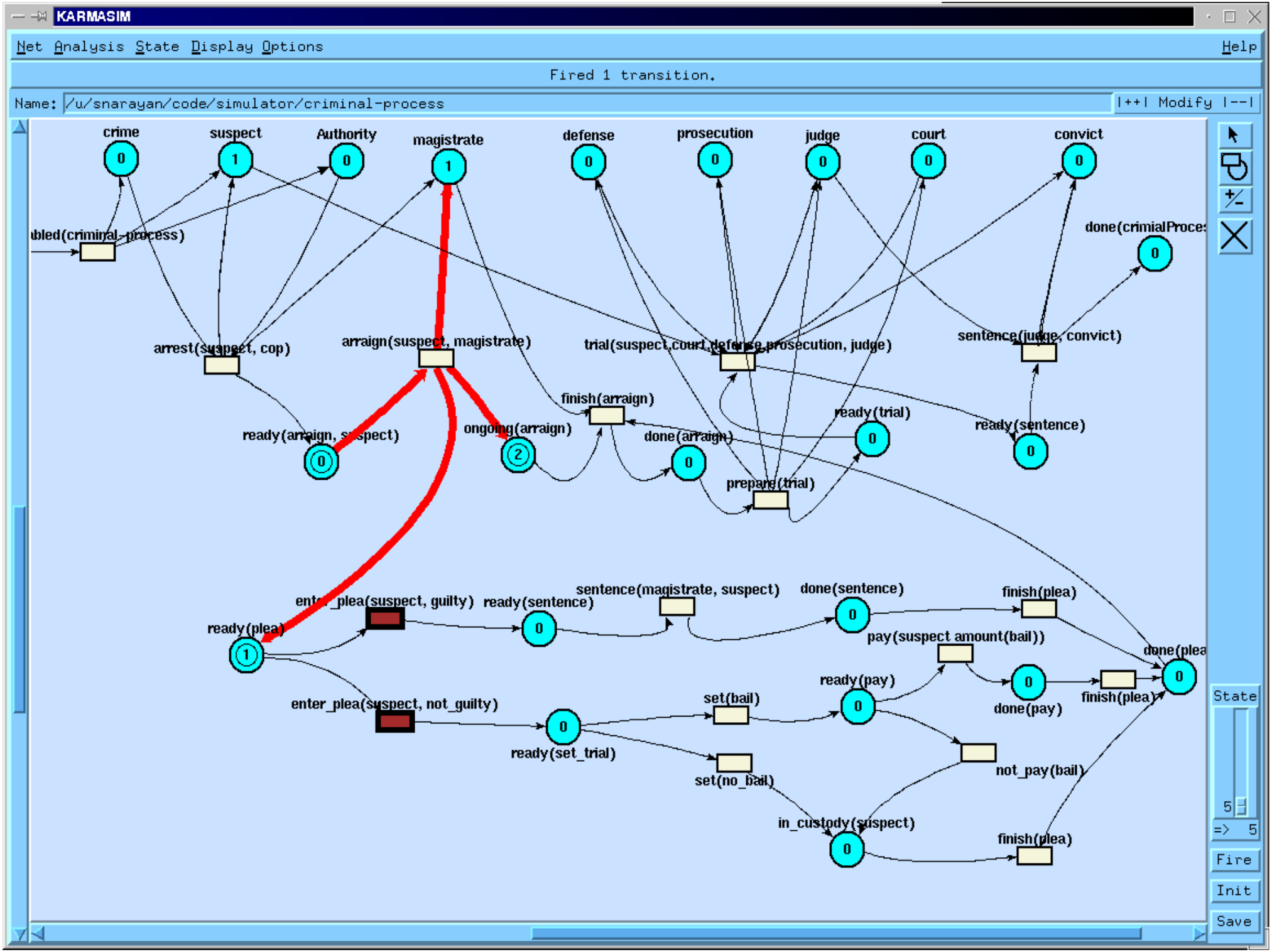

π_1
π_2
π_n
World
^
$Poss_w(a,s)$
Kref
Kref
Kref
Input
^
$Poss_k(a,s)$
Action a
Effect

Petrinetze mit Marken

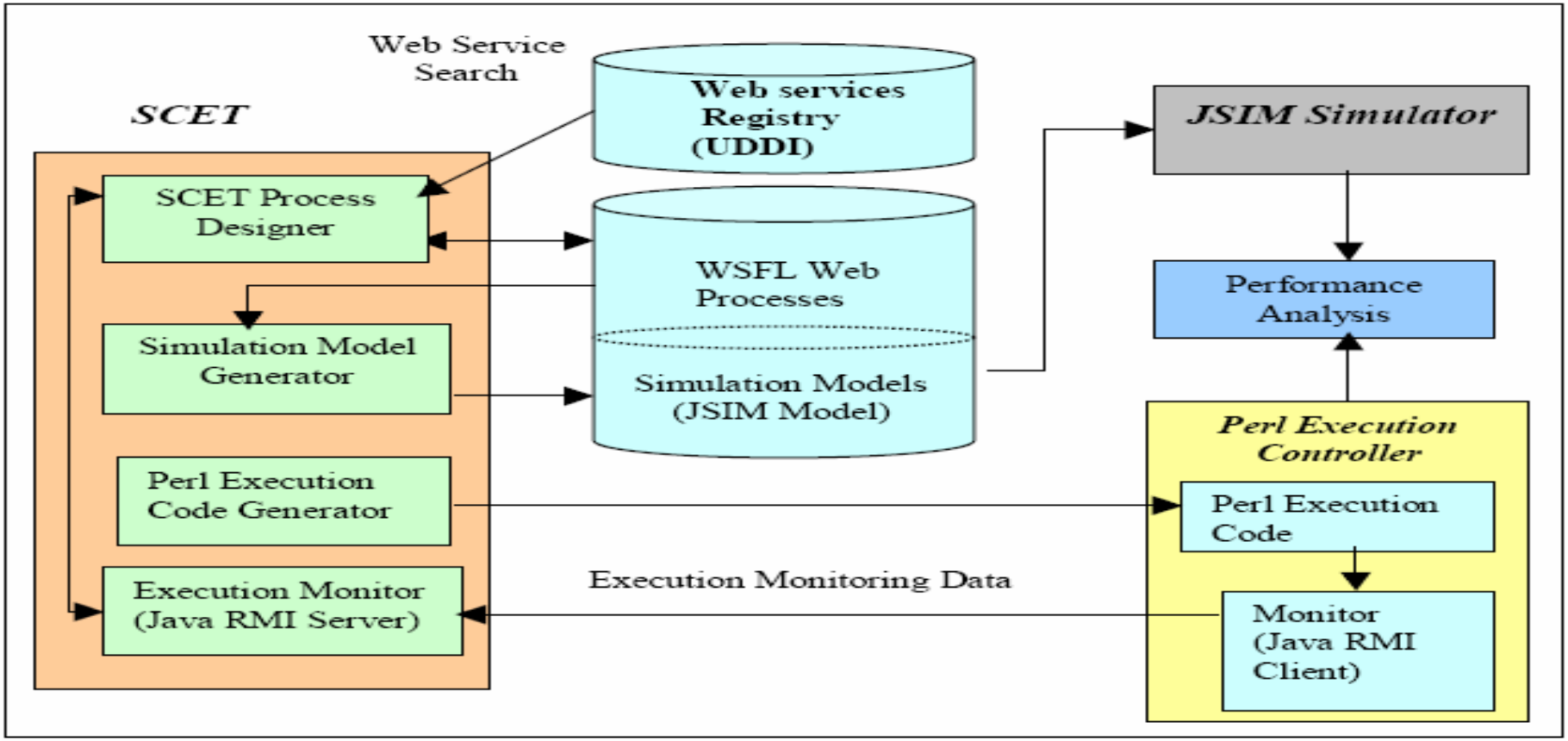
- Dynamische Eigenschaften werden durch Markierungen (Tokens) dargestellt.
- Mehrere Tokens an einer Stelle werden durch Zahlen repräsentiert.
- Schaltregeln (Firing rules) bestimmen den Übergang zwischen 2 oder mehr Stellen und den Transitionen und Tokens.
  - z.B. könnte bei 2 Input-Transitionen verlangt werden, dass beide ausgeführt wurden, oder es müssen mindestens 3 Anfragen gestellt werden.


KARMASIM
Net Analysis State Display Options
Help
Move or Copy Objects, Draw Arcs
Name: /u/sneragan/code/simulator/dam-s-prim
[Move] Copy [Arc]
START(Seq(P1,P2))
FINISH(Seq(P1,P2))
Process1
Process2
ready(Process1)
ready(Process2)
done(Process2)
Process1
ready(task1)
START(Split(P1,P2))
Process2
FINISH(Split(P1,P2))
ready(task2)
ready(finish)
Process1
ready(Process1)
START(Conc)
Process2
FINISH(Conc)
ready(Process2)
Synchronize(P1,P2,P3)
Process3
ready(Process3)
Process1
START(Choice(P1,P2))
FINISH(Choice(P1,P2))
ready(Choose)
done(choose)
Process2
Then_Process
Condition-true
Start(If-Then-Else)
finish(If-Then-Else)
ready(If-Then-Else)
done(If-Then-Else)
Else_Process
Condition-False
Condition = true
Start(Repeat-Until)
Process1
FINISH
ready(P1)
done(P1)
ITERATE(P1)
Condition = False
State
Fire
Init
Save


KARMASIM
Net Analysis State Display Options
Help
Fired 1 transition.
Name: /u/snarayan/code/simulator/criminal-process
|++| Modify |--|
crime
suspect
Authority
magistrate
defense
prosecution
judge
court
convict
bled(criminal_process)
done(crimialProce
arrest(
, cop)
arraign(suspect, magistrate)
trial(suspect,court,defense,prosecution, judge)
sentence(judge, convict)
finish(arraign)
ready(trial)
ready(sentence)
ready(
ign, suspect)
ongoing(arraign)
done(arraign)
prepare(trial)
enter_plea(suspect, guilty)
ready(sentence)
sentence(magistrate, suspect)
done(sentence)
finish(plea)
ready(plea)
pay(suspect amount(bail))
done(plea
ready(pay)
enter_plea(suspect, not_guilty)
set(bail)
done(pay)
finish(plea)
ready(set_trial)
not_pay(bail)
set(no_bail)
in_custody(suspect)
finish(plea)
State
4
=> 4
Fire
Init
Save

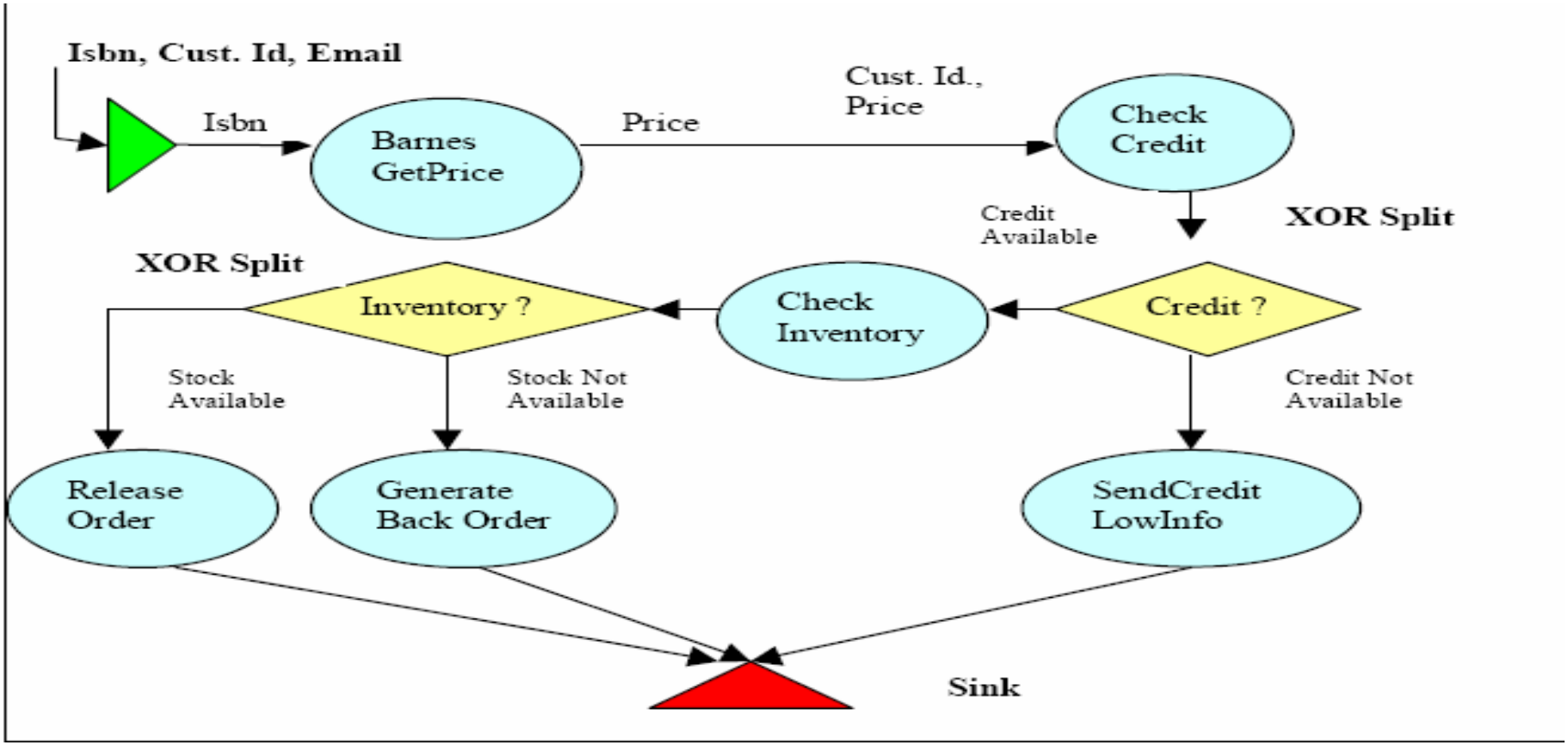

KARMASIM
Net Analysis State Display Options
Help
Fired 1 transition.
Name: /u/snarayan/code/simulator/criminal-process
|++| Modify |--|
crime
suspect
Authority
magistrate
defense
prosecution
judge
court
convict
bled(criminal-process)
done(crimialProces
arrest(suspect, cop)
arraign(suspect, magistrate)
trial(suspect,court,defense,prosecution, judge)
sentence(judge, convict)
finish(arraign)
ready(arraign, suspect)
ongoing(arraign)
done(arraign)
ready(trial)
ready(sentence)
prepare(trial)
sentence(magistrate, suspect)
done(sentence)
finish(plea)
enter_plea(suspect, guilty)
ready(sentence)
ready(plea)
pay(suspect amount(bail))
done(plea
ready(pay)
enter_plea(suspect, not_guilty)
set(bail)
done(pay)
finish(plea)
ready(set_trial)
not_pay(bail)
set(no_bail)
in_custody(suspect)
finish(plea)
State
5
=> 5
Fire
Init
Save

Architekturüberblick



Service Composition and Execution Tool (*SCET*)

Beispielszenario



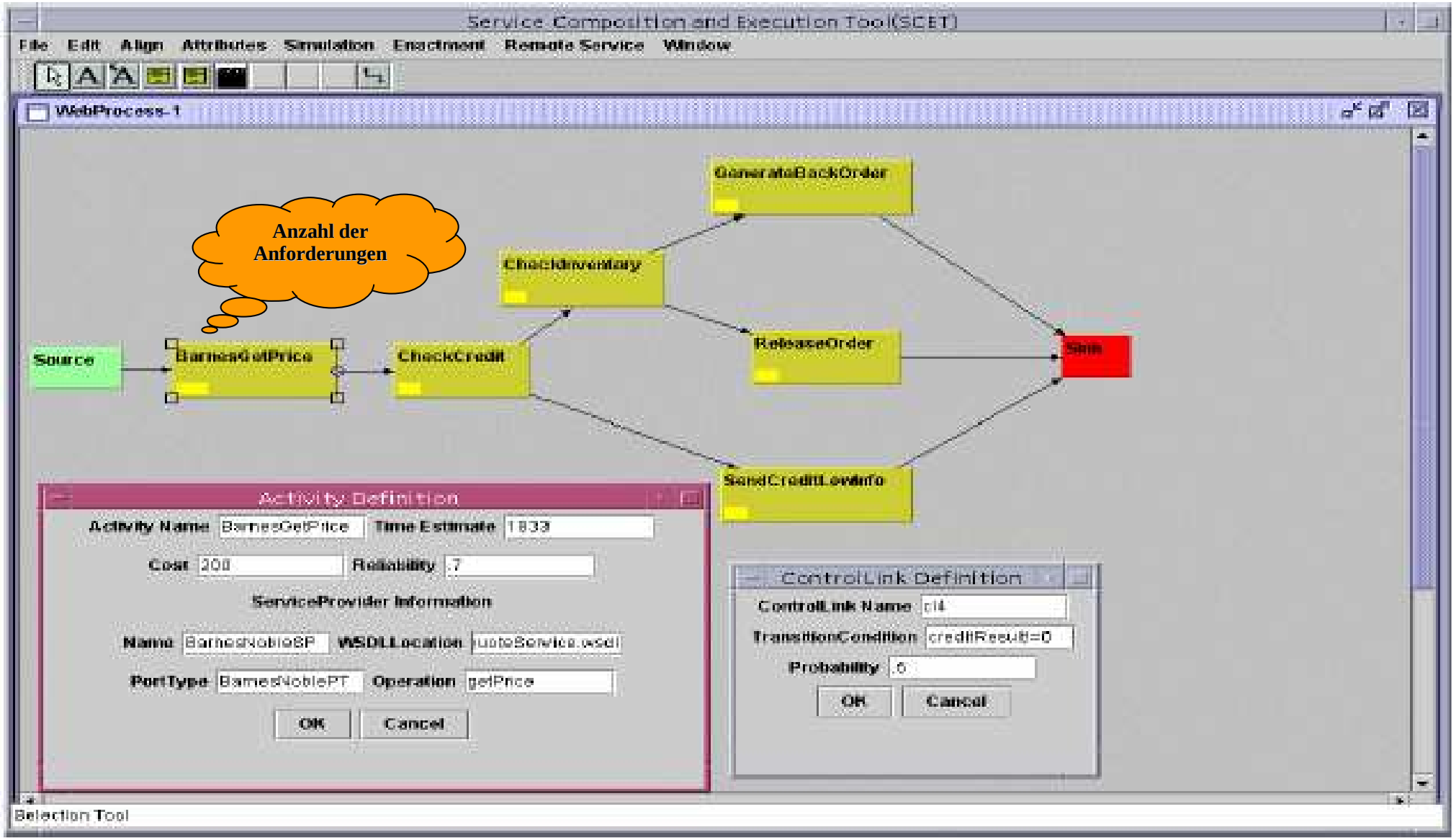
SCET Prozess Designer

- Ein grafisches Tool, welches dem Benutzer eine statische Darstellung des Prozesses erlaubt.
- Die notwendigen Informationen des Prozesses werden durch Knoten & Links dargestellt.
- Der Entwerfer kann den Prozess als WSFL-basiertes Spezifikationsfile oder in einem XML Dokument speichern.
  - *The Web Services Flow Language (WSFL) is an XML language for the description of Web Services compositions as part of a business process definition.”*

Perl Execution Code Generator

- *SCET ist zur automatischen Erzeugung eines Perl Execution Codes aus **WSFL**-basierten Prozessbeschreibungen fähig.*
- Der Code-Generator liest die WSFL Spezifikation des Prozesses und *erzeugt* ein Perl Programm bzw. Code.
- Ein Perl-Kontroller verwaltet gesamtes Projekt
- Alternative Sprachen für die Ausführung:
  - Java
  - Python

SCET Process Composition



WSFL Based Specification Generation

- Direkte Umwandlung vom Prozess-Design auf WSFL Spezifikation.
  - Source/Sink Knoten-> Source/Sink Elemente
  - Aktivität Knoten->Aktivität Elemente
  - Input/Output Inf.-> Input/Output Messages
  - Control/Data Links-> Control/Data Elemente

WSFL Based Specification Generation

```
<defintions>

.......

<controlLink name="cl6" source="CheckInventory"
    target="GenerateBackOrder"
condition="CheckInventoryResult==0" probability=".2" />

.......

<dataLink name="dl2" source="Search Amazon"
target="CheckInventory">

    <mapInfo sourcePart="bookIsbn"

    targetPart="bookIsbn" />

</dataLink>

</definitions>
```

Performance Evaluation

- *Aus der Perspektive des Users ist die Gesamtausführungszeit eines Prozesses ein (Kosten-) **Maß** seiner Leistungsfähigkeit.*

- *Da die Erstellung eines einzelnen Web Services die Leistung des gesamten Web-Prozesses beeinflusst, werden seperate Leistungstests vor der kommerziellen Anwendung ausgeführt.*

Zeitanalyse

- Die Gesamtausführungszeit setzt sich aus Delay-, Waiting- und Service Time zusammen.

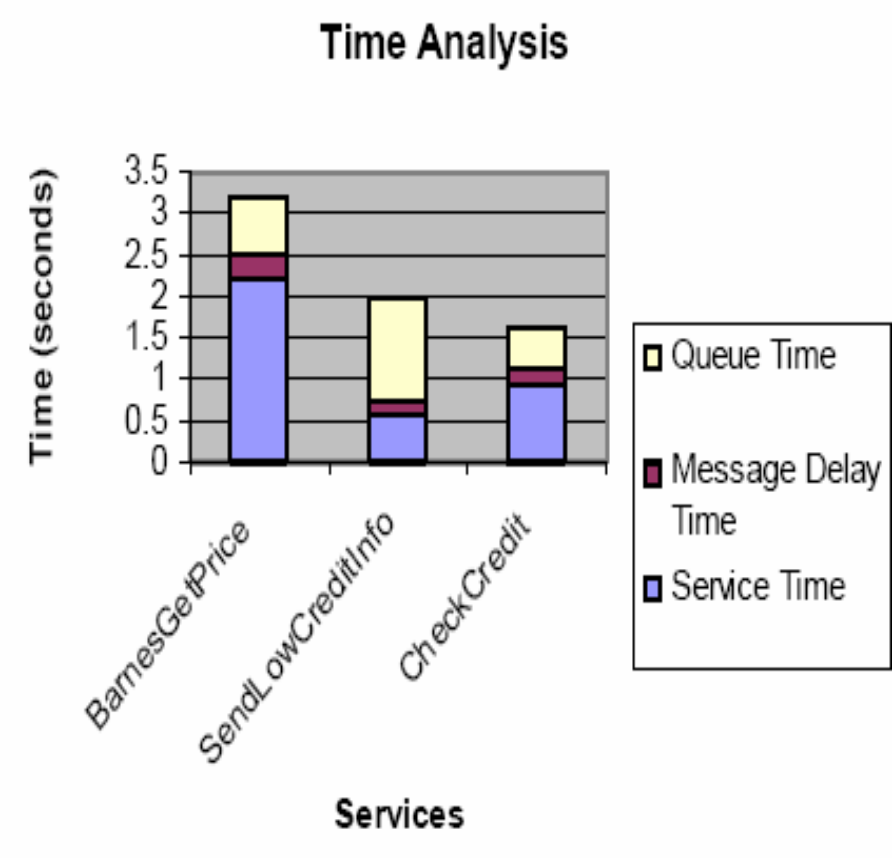
$$T_{gesamt} = T_{delay} + T_{service} + T_{wait}$$

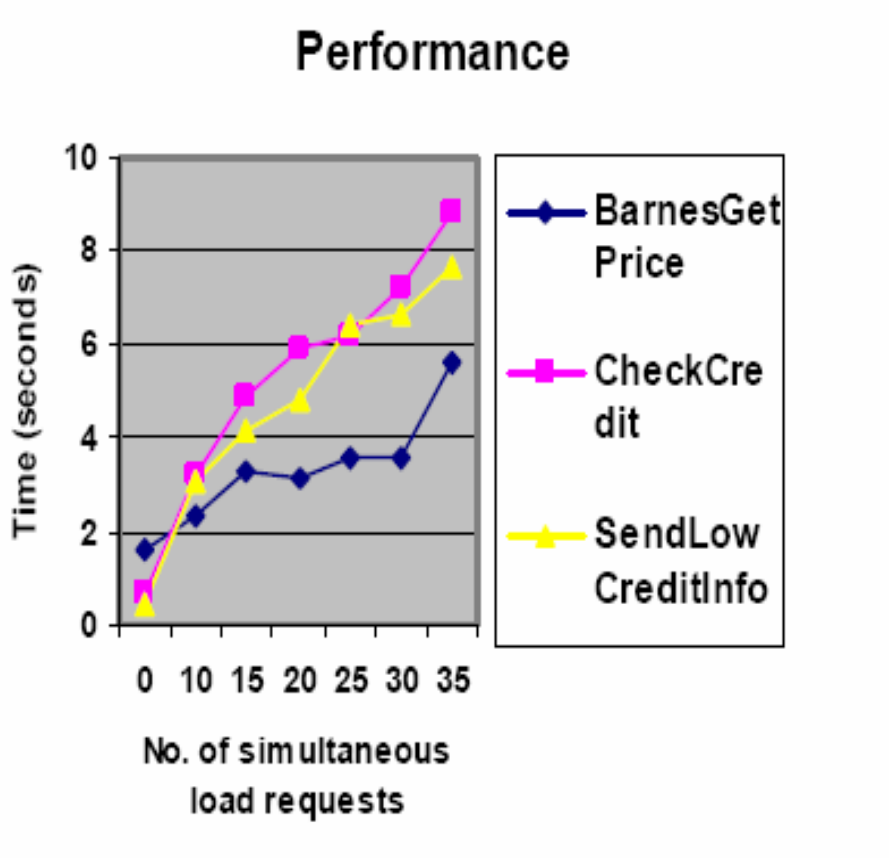
- **Delay time** wird geschätzt, in dem die Ping-Funktion für jeden Webservice aufgerufen wird. (Verzögerungszeit)
- Service time wird geschätzt, in dem man Tests für einen Web Service durchführt, bei welchen die Zustände Load und Waiting kontrolliert werden.
- Waiting time wird geschätzt, indem man einen Test laufen lässt, bei dem der Web Service mit bestimmten Quantitäten von Anfragen systematisch aufgerufen wird.

Test Results for Total Invocation Time (*T*)

10: Num of Iterations			
	:BarnesGetPrice	:CheckCredit	:SendcreditLowInfo
	:1.913394	:1.342986	:0.759598
	:1.714697	:0.682717	:0.486978
	:1.562682	:1.42475	:0.692787
	:1.886854	:0.643781	:0.473613
	:1.828123	:0.816607	:0.525161
	:1.765729	:0.876123	:0.486001
	:1.502503	:0.93461	:0.534121
	:1.617242	:0.644926	:0.464647
	:2.485304	:0.82555	:0.691154
	:2.126191	:0.855064	:0.472463
	:1.84127	:0.918563	:0.57291
Total	:18.330595	:8.622691	:5.399835
Average	:1.8330595	:0.8622691	:0.5399835
Minimum	:1.502503	:0.643781	:0.464647
Maximum	:2.4853	:1.42475	:0.692787
STD	:0.276012	:0.2134661	:0.0824032

Zeitanalyse





Zeitanalyse

- Die Ergebnisse helfen dem Entwickler dabei,
  - *Overloaded-Web-Service-Hosts zu kennzeichnen*, indem sie die **Waiting Time** für jede Service-Anforderung schätzen;
  - *Message Kommunikation overheads zu finden*, indem sie die **Message Delay Time** für jede Service-Anforderung abschätzen.
  - **Load** zu schätzen und den *effektivsten Service* für die Loadtime zu finden.

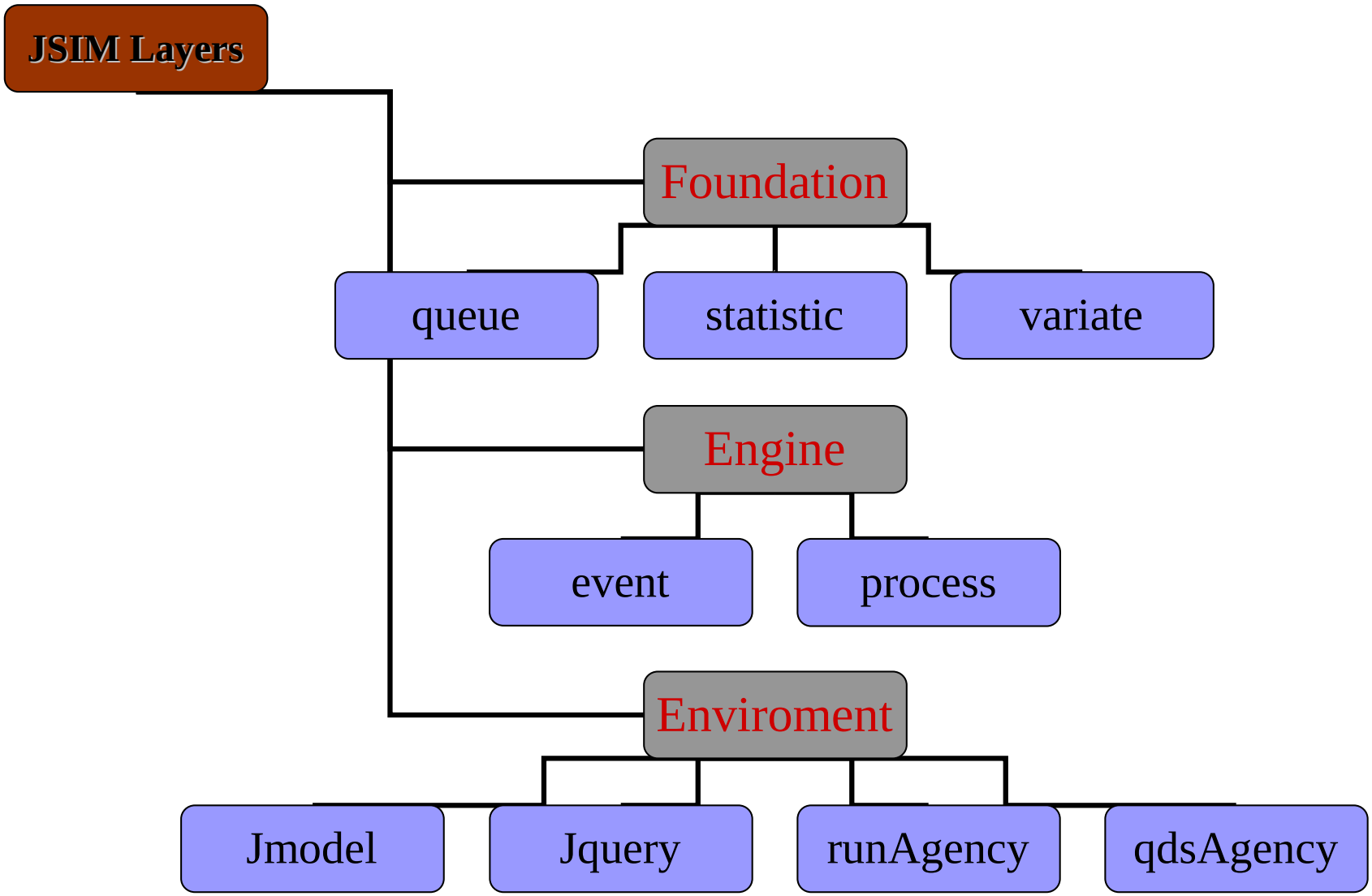
Nachteile solcher Tests

- Man kann gut schätzen, wenn die betroffenen Web Services unter unserer Kontrolle sind.
- Die Verteilung und die Autonomie der Web Services macht diese Anforderung schwierig.
  - Web Service-Over
- Der Server hat Schwierigkeiten mit dem Laden
  - Service-Zeit wird in der Analyse falsch eingeschätzt.

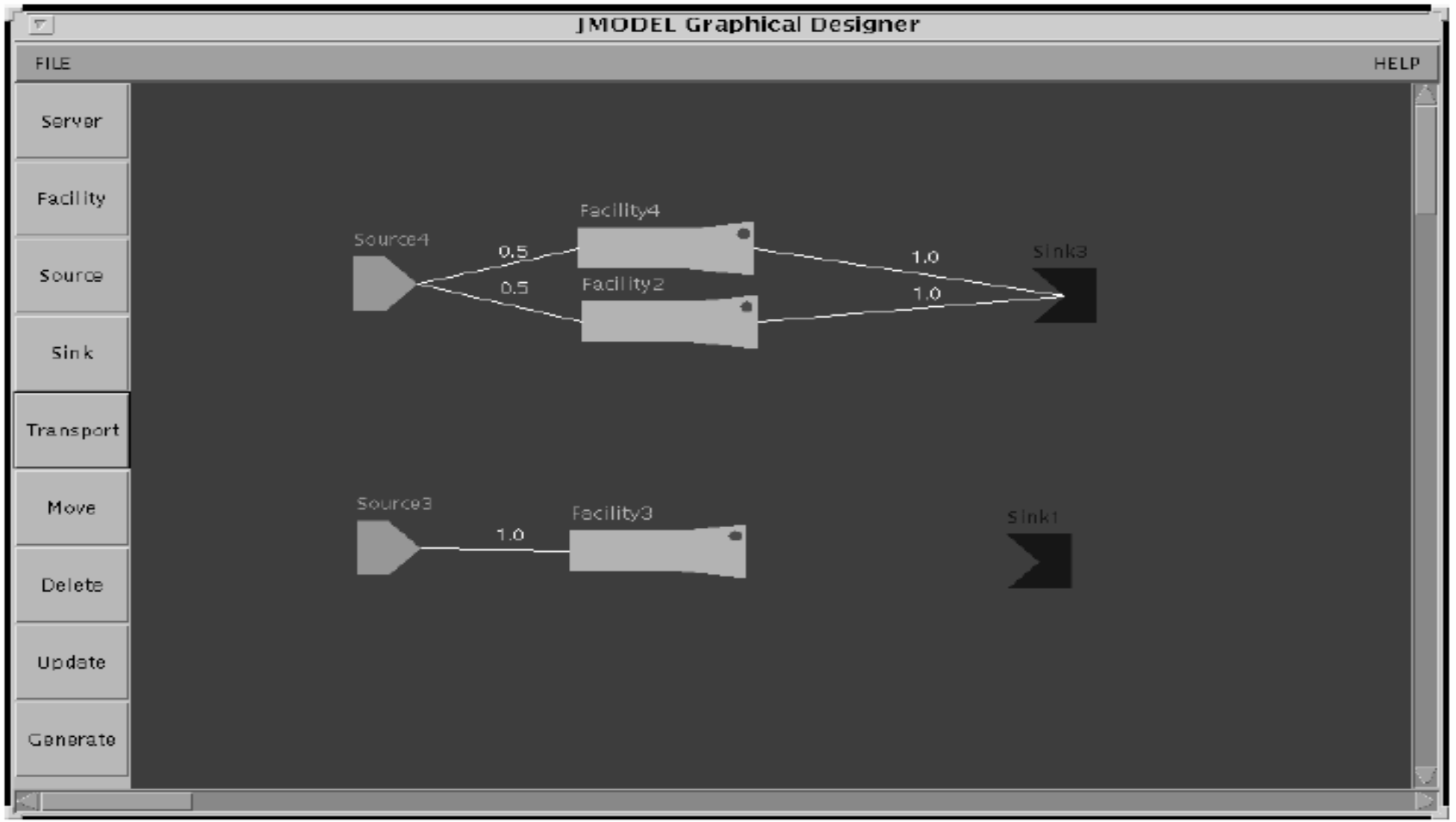
JSIM

- JSIM ist ein *Java-basiertes Simulations- und Animationswerkzeug*, welches die web-basierte Simulation unterstützt.
- Die „service time distribution functions“ dienen als Eingangsdaten
- Für jeden control link erfordert JSIM einen verbundenen Wahrscheinlichkeitswert für das Simulieren der ProzessExecution.
- Nach seinem Simulationsablauf erzeugt JSIM statistische Informationen.

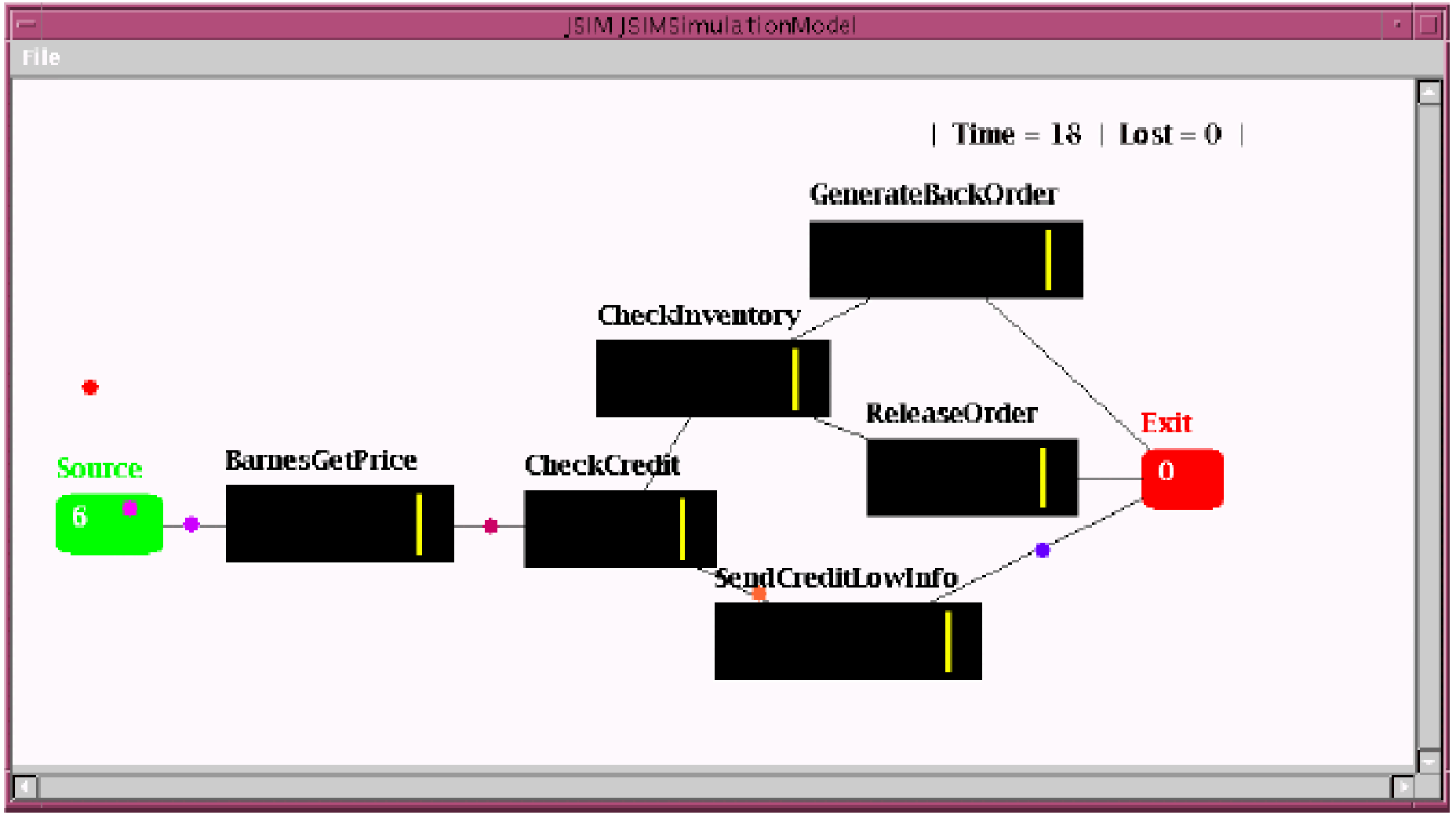
JSIM Framework und Pakete



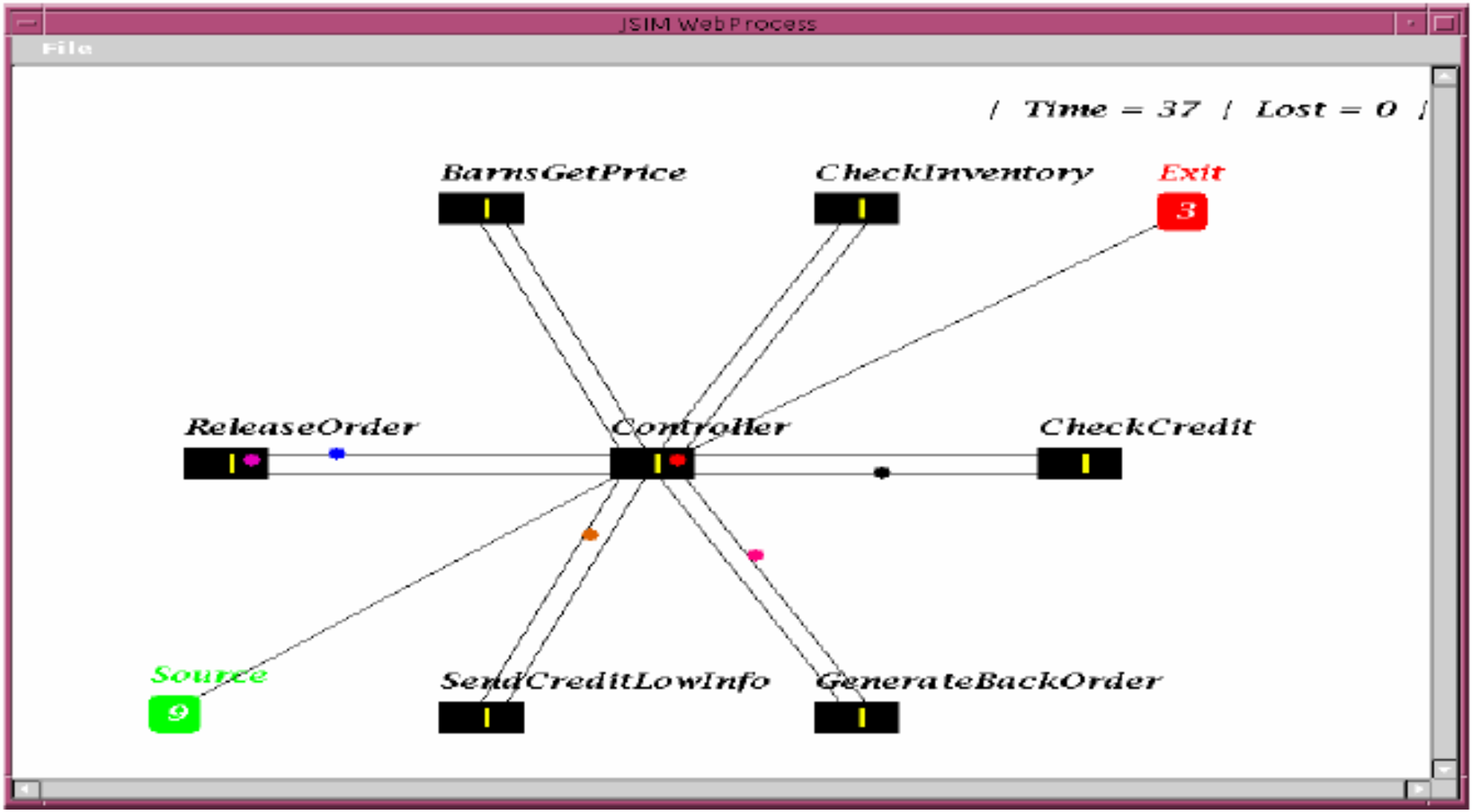
JMODEL


JMODEL Graphical Designer
FILE
HELP
Server
Facility
Source
Sink
Transport
Move
Delete
Update
Generate
Source4
0.5
0.5
Facility4
Facility2
1.0
1.0
Sink3
Source3
1.0
Facility3
Sink1

JSIM Distributed Modell



JSIM Centralized Model



Vergleich zwischen Simulation- und Realzeitanforderung

- *Dynamische* Zahl durch JSIM
- *Tatsächliche Zah*l durch SCET Designer
  - z.B. in unserem Test

Methode	*JSIM*	*SCET Designer*
BarnesGetPrice	1,833 s	1,833 s
CheckCredit	0,861 s	0,862 s
SendCreditLowInfo	0,538 s	0,539 s

JSIM Statistik Tabelle

Statistics

NoSamples	MinValue	MaxValue	MeanValue	Deviation	Interval	Precision	StatName
10.0	3000.0	3000.0	3000.0	0.0	0.0	0.0	Source (dur)
10.0	0.0	0.0	0.0	0.0	0.0		Exit (dur)
10.0	1832.938	1833.028	1833.002	0.025	0.019	0.0	BarnesGetPrice (du
36033.0	0.0	1.0	0.509	0.5	0.005	0.010	BarnesGetPrice (oc
10.0	861.947	862.058	861.997	0.034	0.026	0.0	CheckCredit (dur)
38496.0	0.0	1.0	0.224	0.417	0.004	0.019	CheckCredit (occ)
0.0	0.0	0.0	0.0	0.0	0.0	0.0	CheckInventory (d
0.0	0.0	0.0	0.0	0.0	0.0	0.0	CheckInventory (o
10.0	538.951	539.032	538.991	0.029	0.022	0.0	SendCreditLowInf
41035.0	0.0	1.0	0.131	0.338	0.003	0.025	SendCreditLowInf
0.0	0.0	0.0	0.0	0.0	0.0	0.0	GenerateBackOrde
0.0	0.0	0.0	0.0	0.0	0.0	0.0	GenerateBackOrde
0.0	0.0	0.0	0.0	0.0	0.0	0.0	ReleaseOrder (dur)
0.0	0.0	0.0	0.0	0.0	0.0	0.0	ReleaseOrder (occ)
30.0	400.0	400.0	400.0	0.0	0.0	0.0	path1 (dur)
0.0	0.0	0.0	0.0	0.0	0.0	0.0	path2 (dur)
0.0	0.0	0.0	0.0	0.0	0.0	0.0	path3 (dur)
0.0	0.0	0.0	0.0	0.0	0.0	0.0	path4 (dur)
0.0	0.0	0.0	0.0	0.0	0.0	0.0	path5 (dur)
0.0	0.0	0.0	0.0	0.0	0.0	0.0	path6 (dur)
140.0	400.0	400.0	400.0	0.0	0.0	0.0	path7 (dur)
50.0	400.0	400.0	400.0	0.0	0.0	0.0	path8 (dur)
40.0	400.0	400.000	400.000	0.0	0.0	0.0	path9 (dur)

Zusammenfassung

- Simulationen helfen Kosten zu sparen.
- Simulationen decken jeweils nur einen oder nur wenige Teilaspekte der QoS-Anforderungen ab.
- Jede Web-Service-Sprache braucht einen eigenen Simulator.
- Dynamik von Web-Services lässt sich simulieren.

Zukunftsaussichten

- Zurzeit SCET unterstützt nur WSFL Spz.
  - Erweiterungen sind geplant, BPEL4WS
- Simulation von Webservices durch Webservices
- Kommerzielle Anbieter warten noch Standardisierungs-verfahren ab