Fraunhofer Resource Grid

Fraunhofer Institut Rechnerarchitektur und Softwaretechnik

Fraunhofer Resource Grid

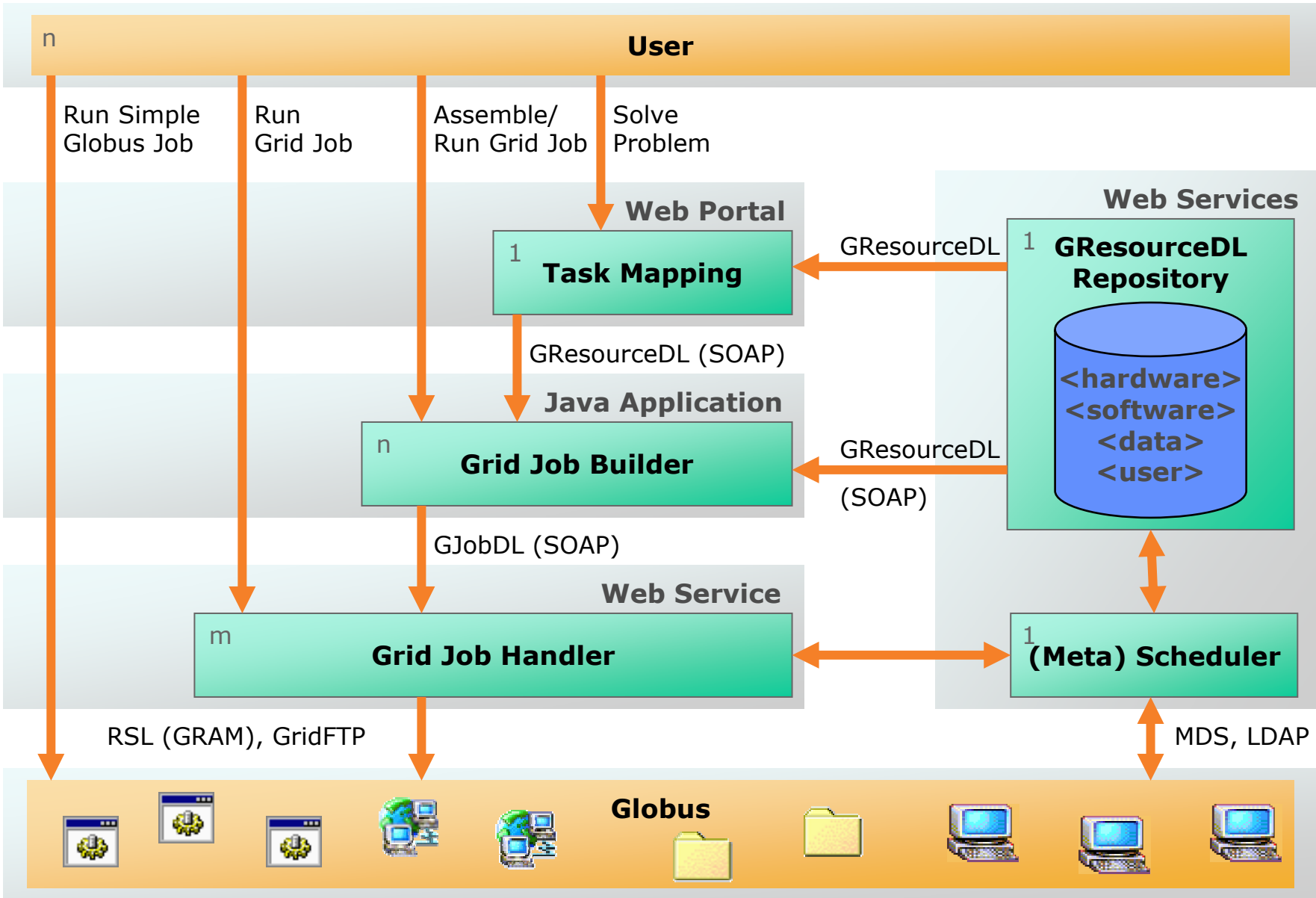
Grid Job Handler auf Basis von Petrinetzen

Andreas Hoheisel
(andreas.hoheisel@first.fraunhofer.de)

Uwe Der
(uwe.der@first.fraunhofer.de)

FhRG
Fraunhofer Resource Grid

Fraunhofer FIRST Institut
Rechnerarchitektur
und Softwaretechnik


n
User
Run Simple Globus Job
Run Grid Job
Assemble/ Run Grid Job
Solve Problem
Web Portal
1
Task Mapping
GResourceDL
Web Services
1
GResourceDL Repository
<hardware>
<software>
<data>
<user>
GResourceDL (SOAP)
Java Application
n
Grid Job Builder
GResourceDL
(SOAP)
GJobDL (SOAP)
Web Service
m
Grid Job Handler
1
(Meta) Scheduler
RSL (GRAM), GridFTP
MDS, LDAP
Globus

Bausteine des FhRG: Grid Job Handler (FIRST)

Grid Job Handler	Software zur Ausführung und Steuerung von gekoppelten Grid-Anwendungen
	Prozessmodellierung
	Verteilung des Grid-Jobs auf die geeigneten Ressourcen
	Abarbeiten des Jobs unter Verwendung von Grid-Middleware (z.B. Globus → Java Commodity Grid Kit)
Komponentenumgebung	lose Kopplung von Komponenten über Dateieingabe/Ausgabe
Monitoring	Job-Status (Pending, Active, Failed, Done)
Integration ins FhRG	Kommunikation mit anderen FhRG-Diensten per SOAP (→ WebService)

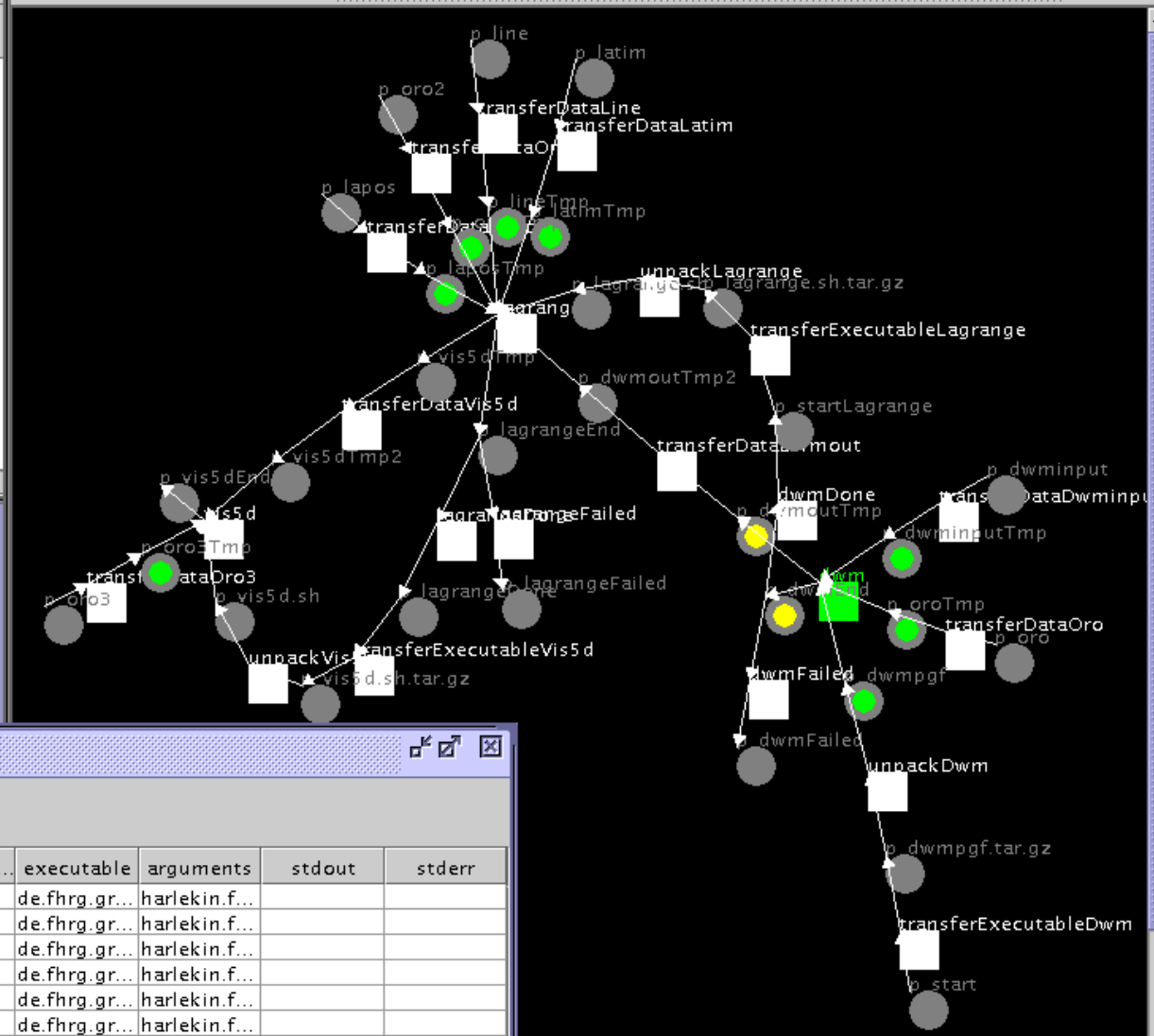
File Grid Job Security Help

GJobDL: /export/demeter/hoheisel/I-Lab/executo

Create Grid Job

```
        <transitionRef id = "transferDataOro3"/>
    </arc>
    <arc id = "arc16b" type = "T2P">
        <transitionRef id = "transferDataOro3"/>
        <placeRef id = "p_oro3Tmp"/>
    </arc>
    <arc id = "arc16c" type = "P2T">
        <placeRef id = "p_oro3Tmp"/>
        <transitionRef id = "vis5d">
            <inputRef id = "oro" type = "file"/>
        </transitionRef>
```

Petri Net -- Grid Job #1



Grid Job #1

Run | Show Workflow | Stop

status	start time	end time	ID	resourceM...	executable	arguments	stdout	stderr
DONE	Feb 05 11...	Feb 05 11...	1	local	de.fhrg.gr...	harlekin.f...		
DONE	Feb 05 11...	Feb 05 11...	2	local	de.fhrg.gr...	harlekin.f...		
DONE	Feb 05 11...	Feb 05 11...	3	local	de.fhrg.gr...	harlekin.f...		
DONE	Feb 05 11...	Feb 05 11...	4	local	de.fhrg.gr...	harlekin.f...		
DONE	Feb 05 11...	Feb 05 11...	5	local	de.fhrg.gr...	harlekin.f...		
DONE	Feb 05 11...	Feb 05 11...	6	local	de.fhrg.gr...	harlekin.f...		
DONE	Feb 05 11...	Feb 05 11...	7	local	de.fhrg.gr...	harlekin.f...		
DONE	Feb 05 11...	Feb 05 11...	8	local	de.fhrg.gr...	harlekin.f...		
DONE	Feb 05 11...	Feb 05 11...	9	p1.itwm.f...	/bin/tar	-xzf tmp...	tmp1044...	tmp1044...
ACTIVE	Feb 05 11...	N/A	10	p1.itwm.f...	dwmpgf	-output d...	dwmpgf.s...	dwmpgf.s...

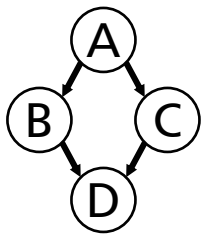
Wieso Petrinetze?

Motivation

Problem

Beschreibung komplexer Prozessabläufe von Grid-Anwendungen

DAG

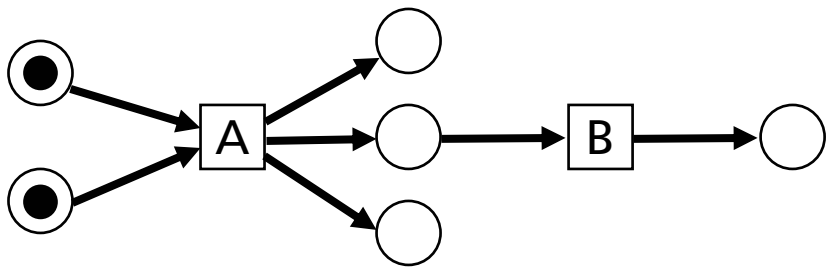


Directed Acyclic Graph (siehe Condor, Cactus, UNICORE)
keine bidirektionale Kopplung (Interaktion)
keine Schleifen

```
PARENT A CHILD B C
PARENT B C CHILD D
```

Petrinetze

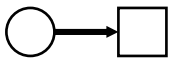
Grafische Ablaufsteuerung diskreter Systeme



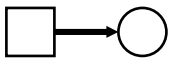
Petrinetze

Stellen Dateien, Kontrollstellen

Transitionen Softwarekomponenten, Kontrolltransitionen



Pfeile von Stellen nach Transitionen (Stelle ist Eingabestelle der Transition)



Pfeile von Transitionen nach Stellen (Stelle ist Ausgabestelle der Transition)

Marken Daten, Zustand (failed, done, ...)

Regeln Eine Transition ist aktiviert, wenn alle ihre Eingabestellen belegt sind und alle Ausgabestellen noch nicht ihre maximale Kapazität erreicht haben

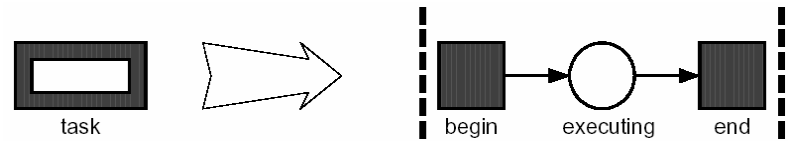
Verfeinerung Eine Transition kann durch ein Petrinetz ersetzt werden

Beschreibung des Zustands Petrinetz beschreibt nicht nur Prozessablauf, sondern auch Zustand des Systems

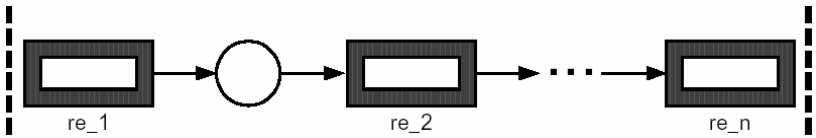
Petri Nets

(from *van der Aalst und Kumar, 2000*)

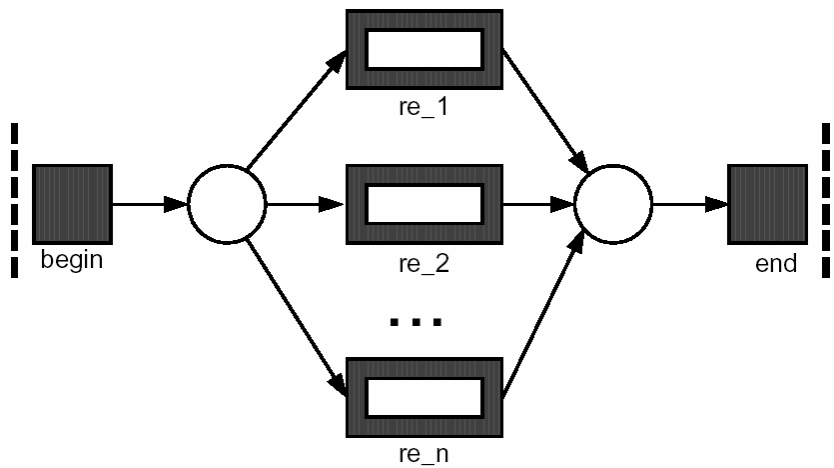
Task



Sequence

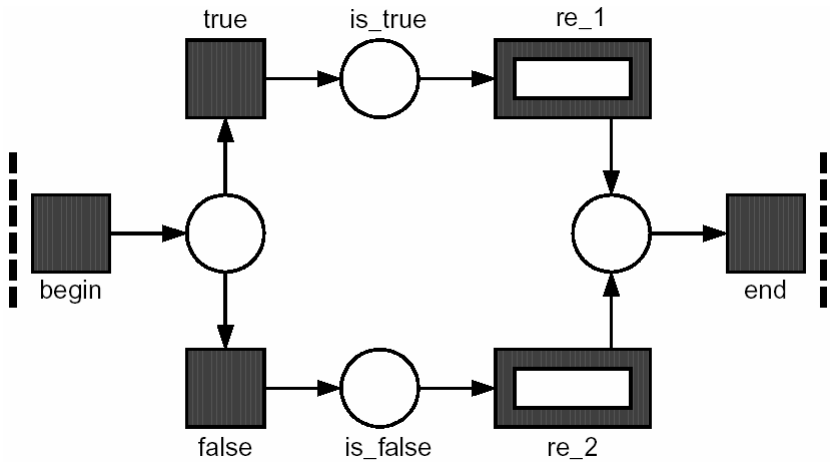


Choice



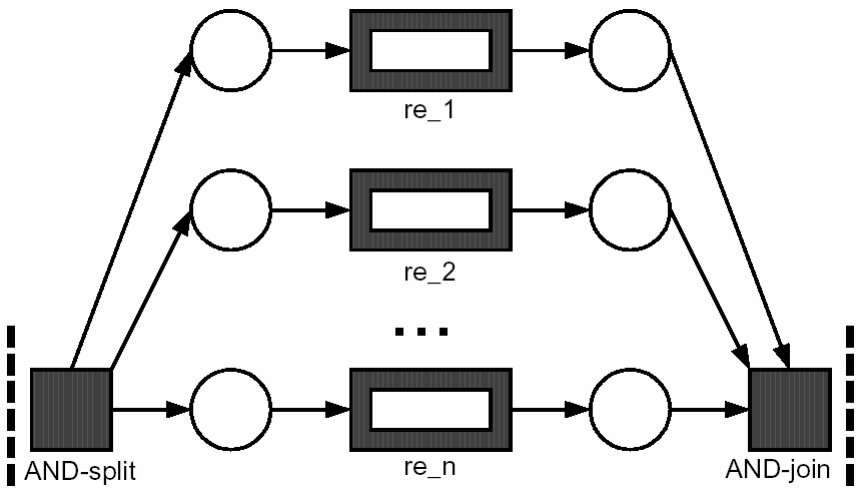
Petri Nets

Condition



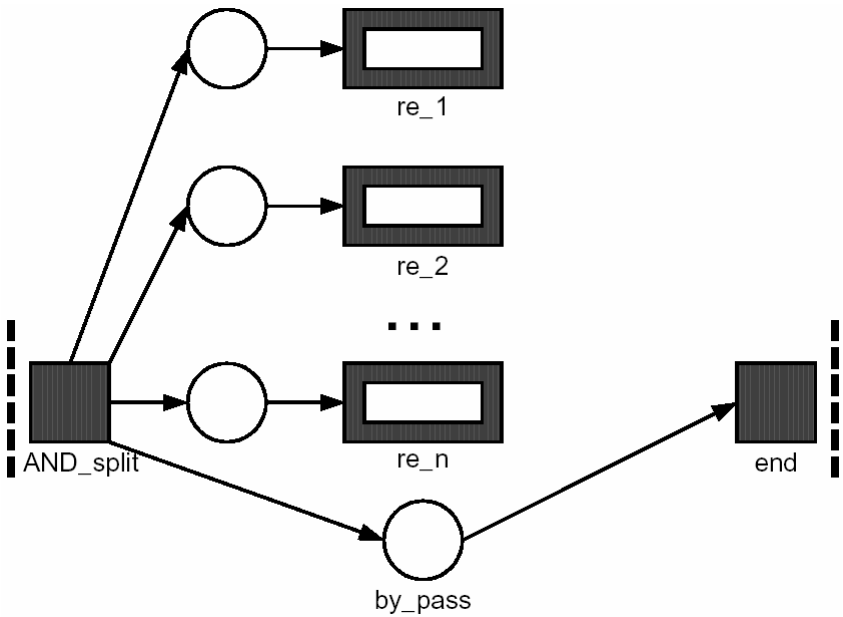
Petri Nets

Parallel execution with synchronization



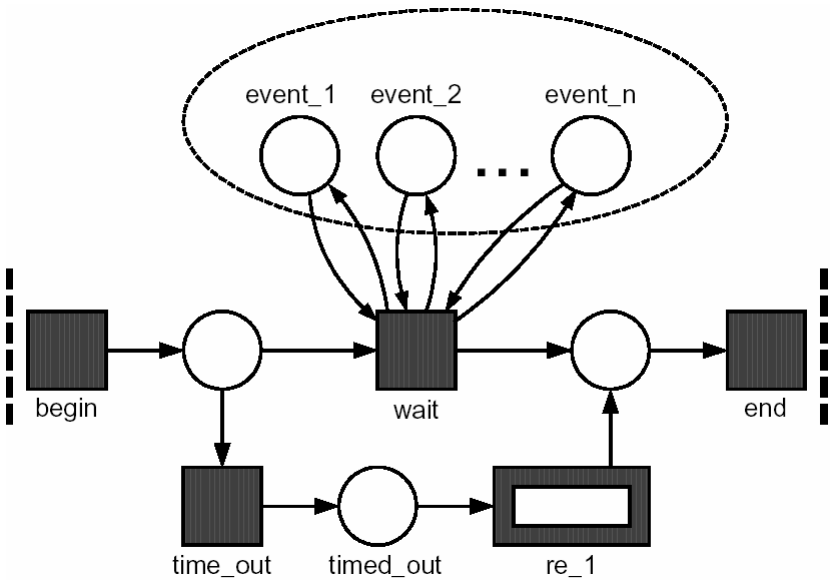
Petri Nets

Parallel execution without synchronization



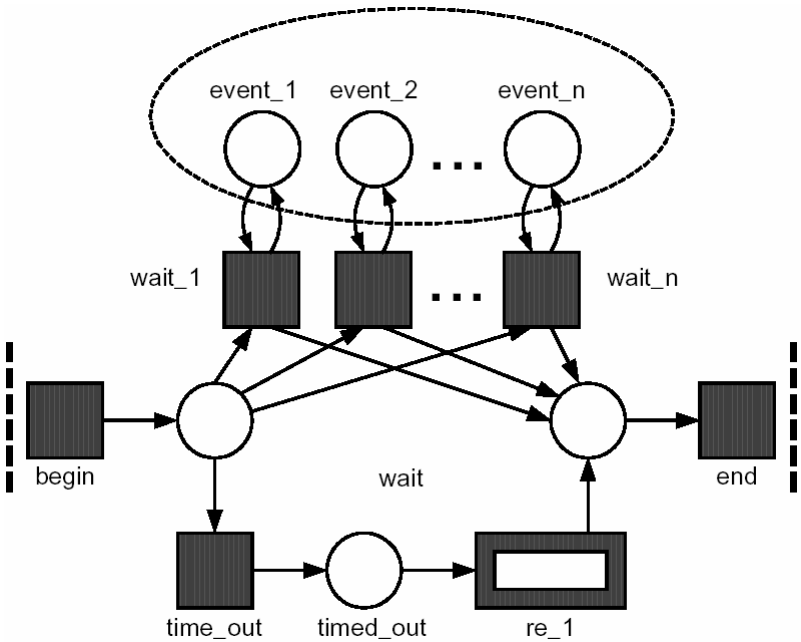
Petri Nets

Wait all with time out



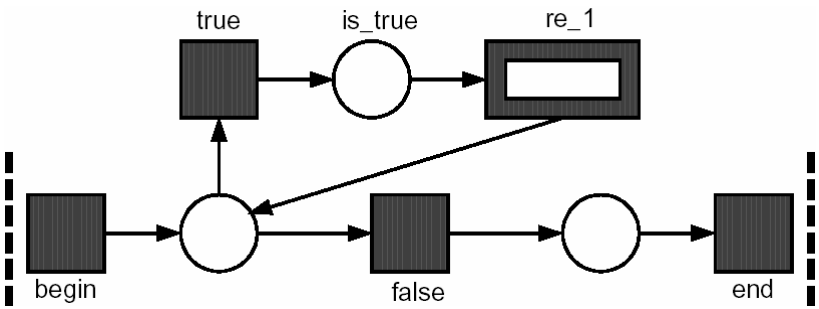
Petri Nets

Wait any with time out



Petri Nets

While … do



Spezielle Transitionen

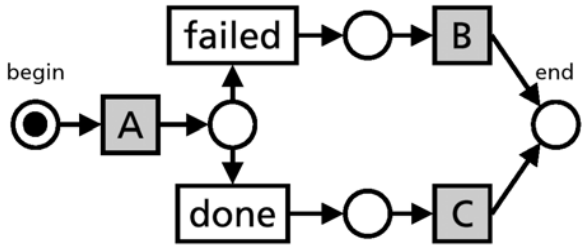
JobHandler: Transitionen vom Typ „MethodCall“

```
<transition id = "unpack_data">
  <methodCall>unpack()</methodCall>
</transition>
```

- **unpack():** Entpacken von tar.gz-Dateien
- **transferData():** Übertragen von Dateien
- **transferExecutable(String softwareTransitionId):** Übertragen der ausführbaren Dateien auf den Zielrechner
- **transferData(String softwareTransitionId):** Übertragen von Eingabedateien auf den Zielrechner

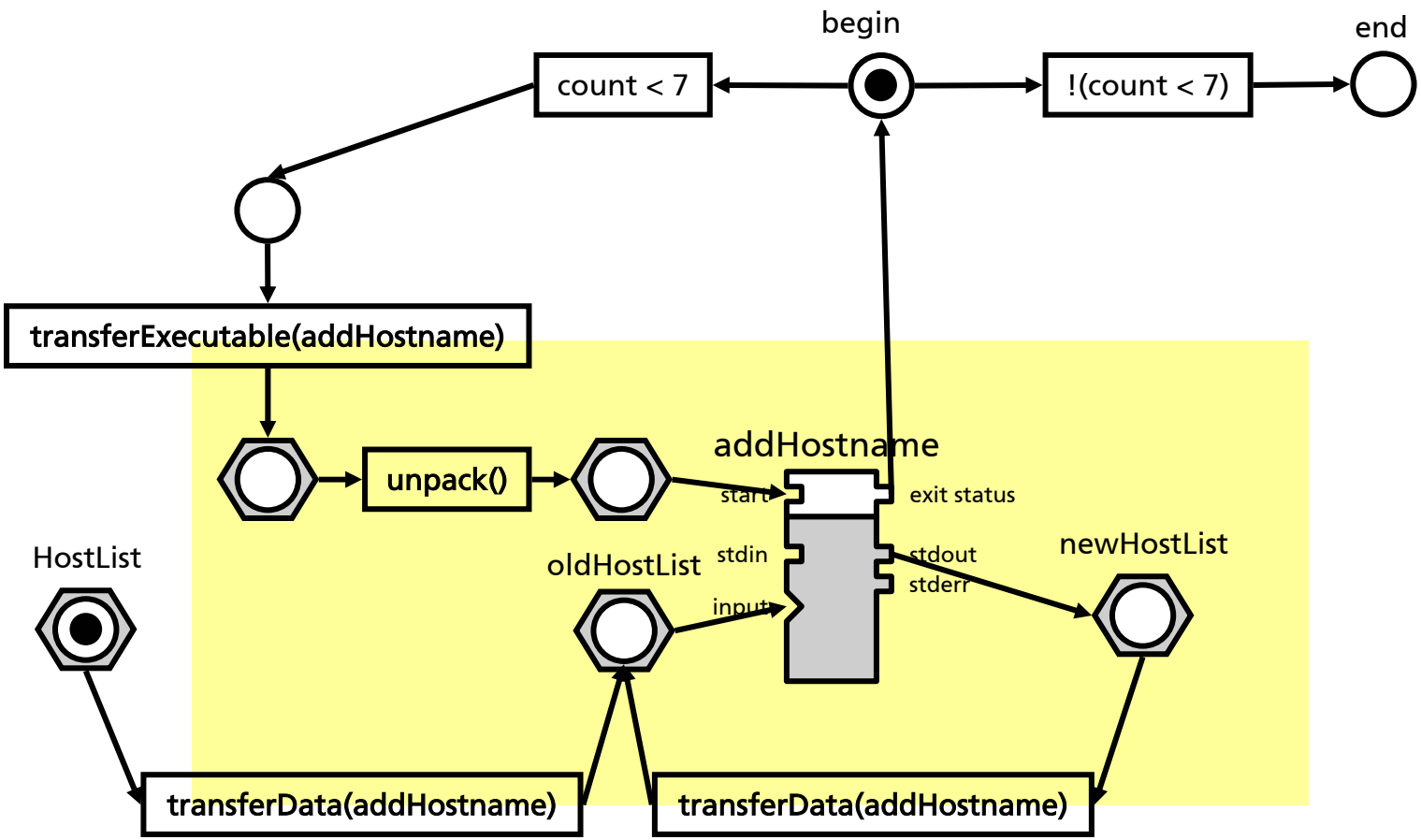
JobHandler: Transitionen vom Typ „Condition“

```
<transition id = "A_done">
  <condition>isDone()</condition>
</transition>
```



- **isDone():** Wahr, wenn alle Eingabestellen „true“
- **isFailed():** Wahr, wenn mindestens eine Eingabestelle „false“
- **countLT(int maxCount):** Zählt die Anzahl, wie oft eine Transition bisher geschaltet hat. Wahr, falls Anzahl kleiner als maxCount
- **negate(String transitionID):** Negierung der Bedingung einer anderen Transition (entspricht ELSE bei IF THEN), z.B. negate(count.lt.7) schaltet nur dann, wenn die Bedingung der Transition „count.lt.7“ nicht erfüllt ist

FhRG Loop



XML-Beschreibung von Petrinetzen

Beschreibung von Petrinetzen in XML

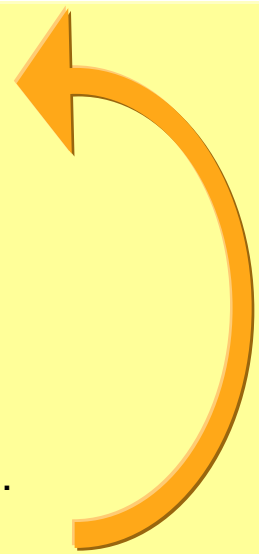
PNML — Petri Net Markup Language
(modified from *Jüngel, Kindler, Weber; HU-Berlin*)

```
<place id="start">
  <initialMarking>
    <value type="boolean" op="eq">true</value>
  </initialMarking>
</place>
<place id="output"/>
<transition id="program"/>
<arc type="P2T" id="arc1">
  <placeRef id="start" />
  <transitionRef id="program" />
</arc>
<arc type="T2P" id="arc2">
  <transitionRef id="program"/>
  <placeRef id="output" />
</arc>
```

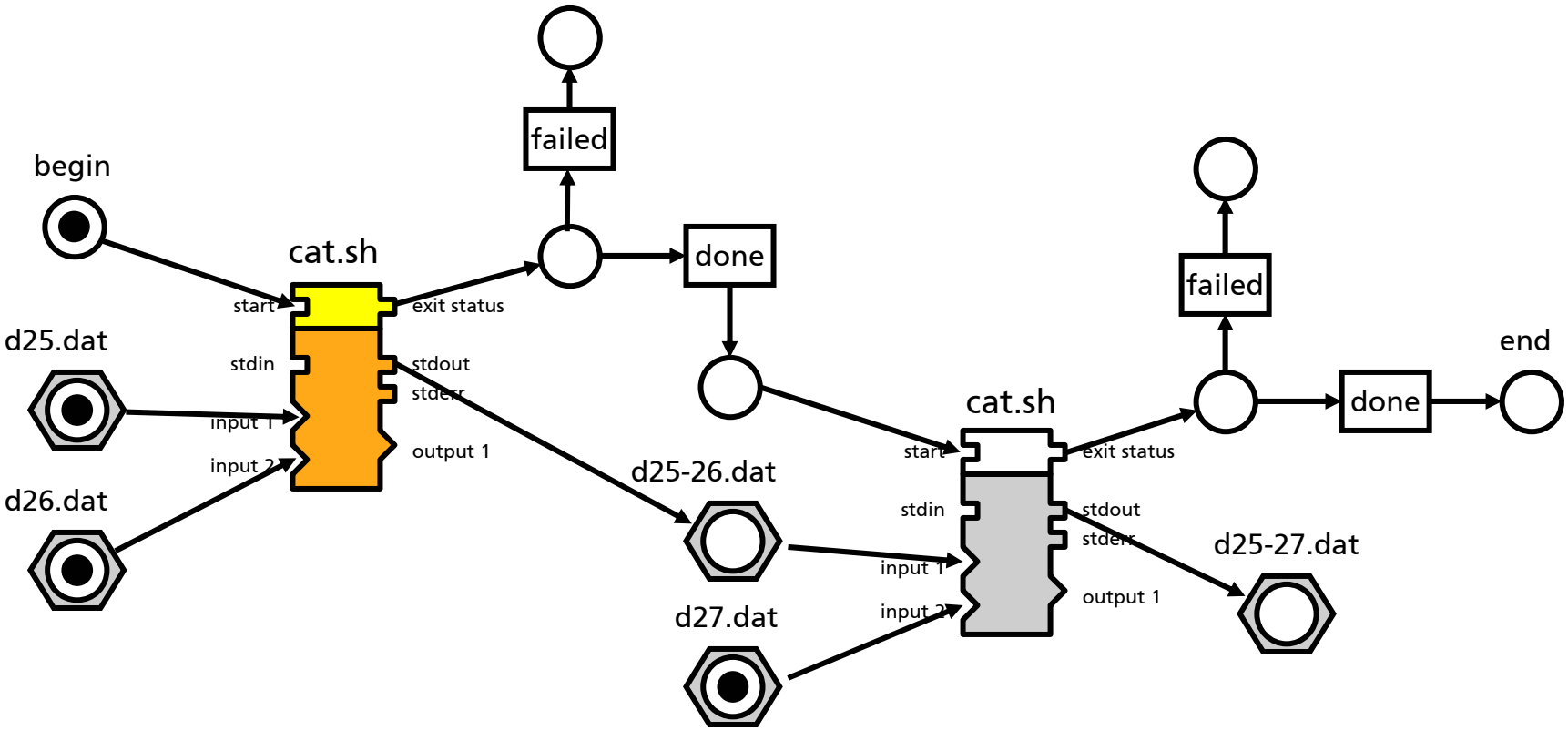
Ablauf GridJobHandler

Ablauf JobHandler

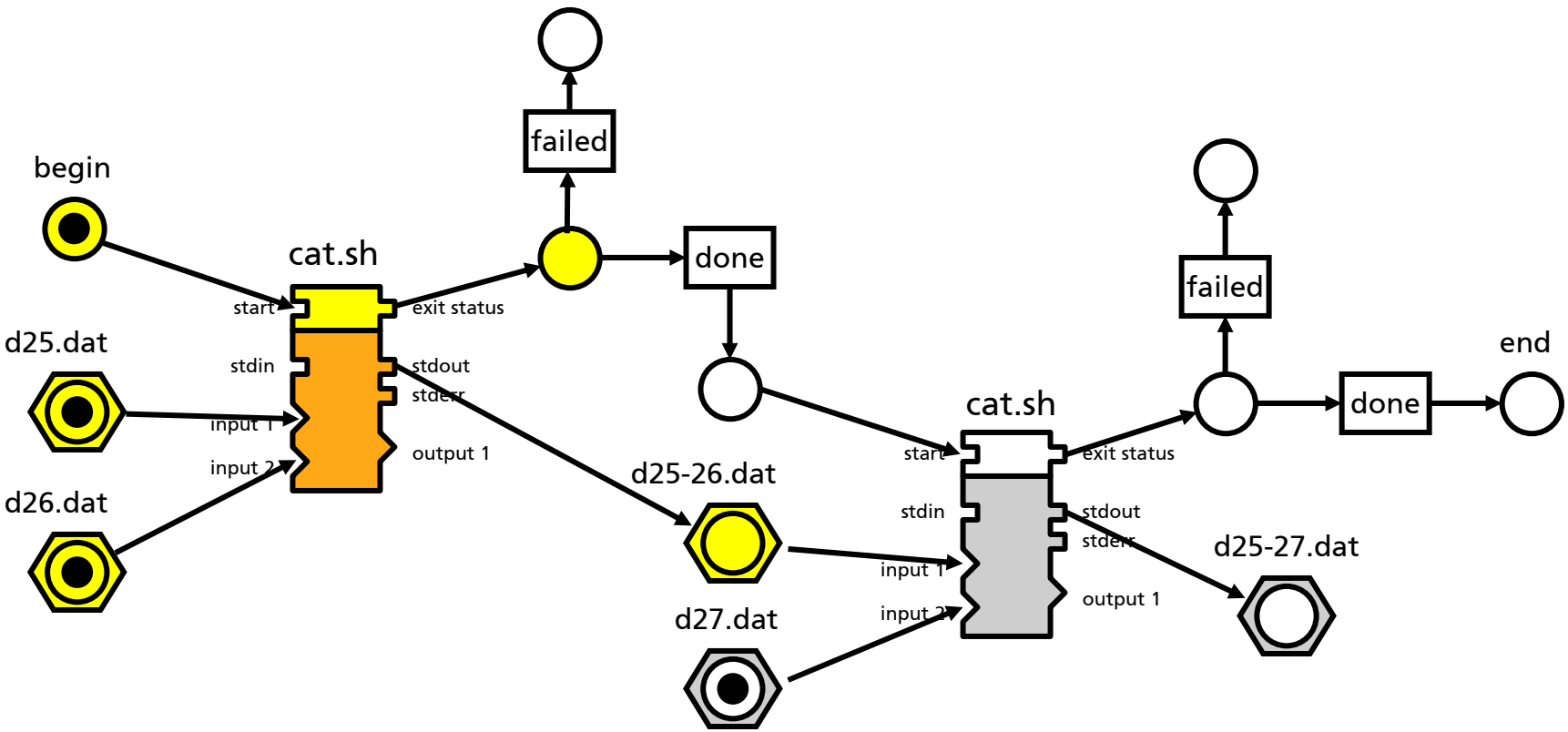
- ✓ Einlesen des GJobDL-Dokuments per validierenden DOM-Parser
- ✓ Erzeugen des Petrinetzes aus der Job-Beschreibung
- • Prüfen des Petrinetzes (Wohlgeformtheit, Lebendigkeit, Deadlocks, Fallen, ...)
- ✓ Start des GridJobs (eigener Thread)
- ✓ Sammeln aller aktivierten Transitionen
- ✓ Ggf. Auswerten von Bedingungen (conditions)
- ✓ Sperren der Eingabe- und Ausgabestellen (lock)
- ✓ Ggf. Resource Mapping → Repository, (Meta-)Scheduler
- ✓ Ggf. Verfeinern des Petrinetzes → Einfügen eines GridFTPs
- ✓ Ggf. Erzeugen und Submitten des AtomicJobs per GRAM
- ✓ Transition schaltet, wenn Atomic Job „done“ oder „failed“. Eingabe- und Ausgabestellen werden freigegeben (unlock)



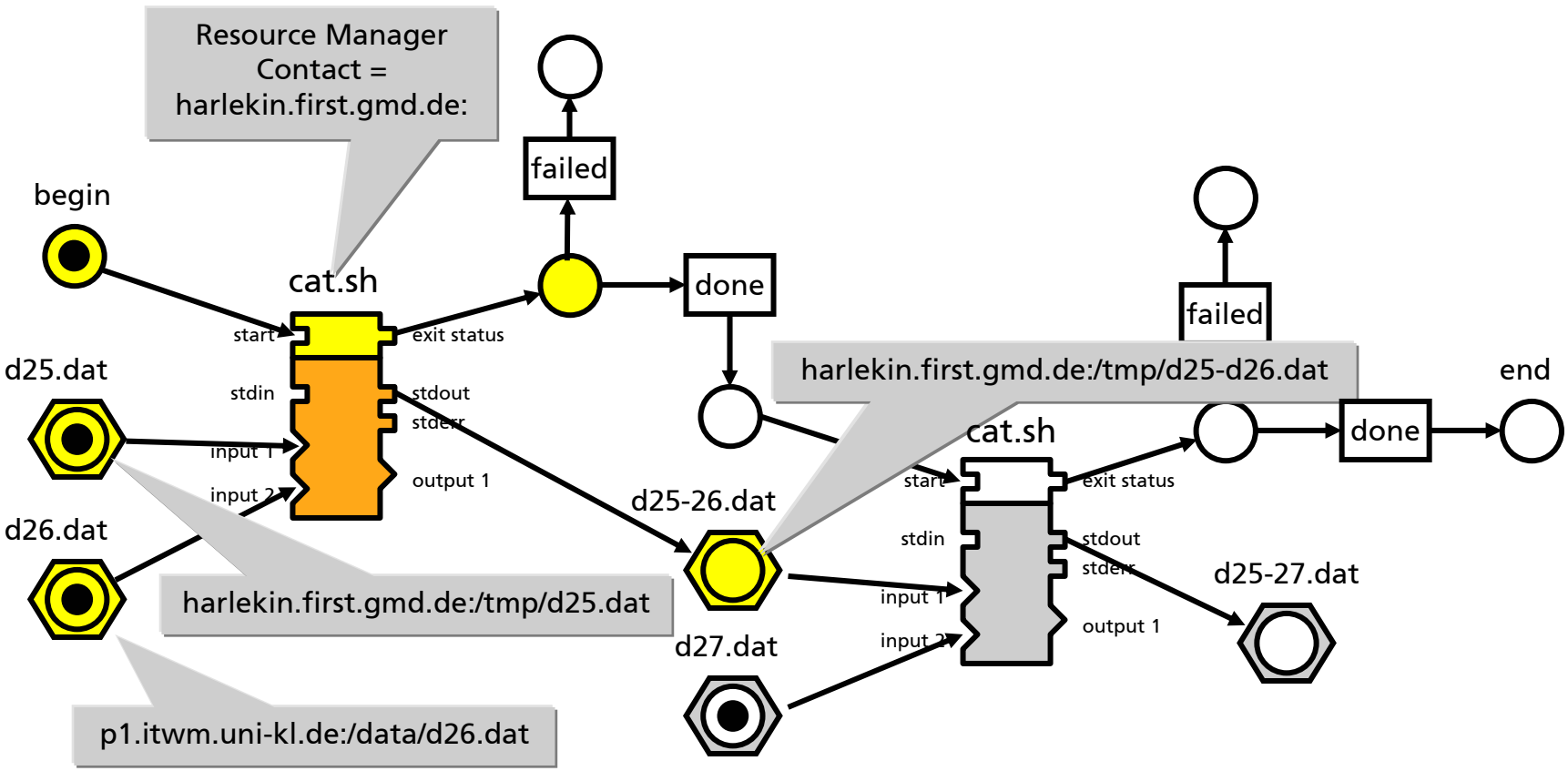
Sammeln aller aktivierten Transitionen



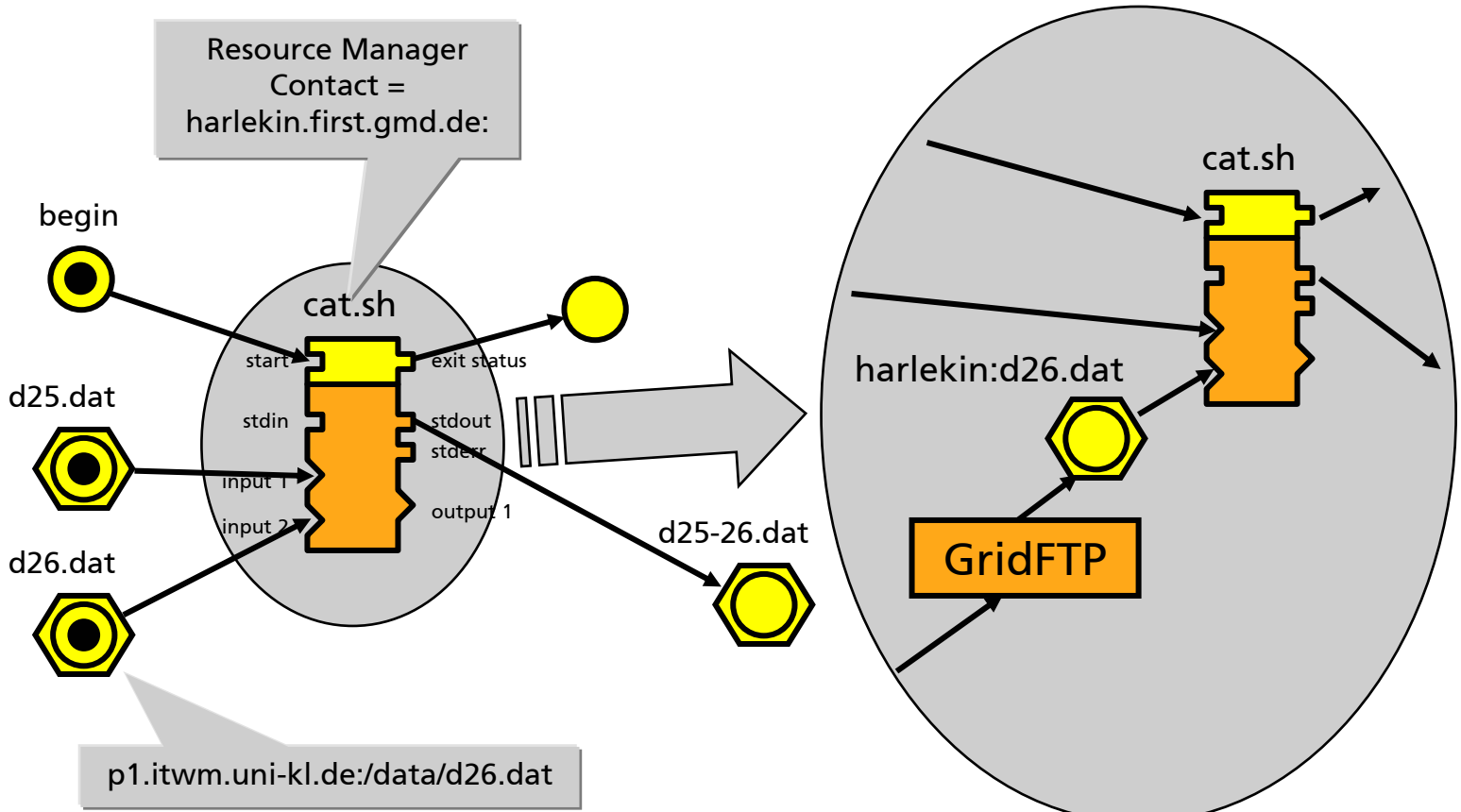
Sperren der Eingabe- und Ausgabestellen



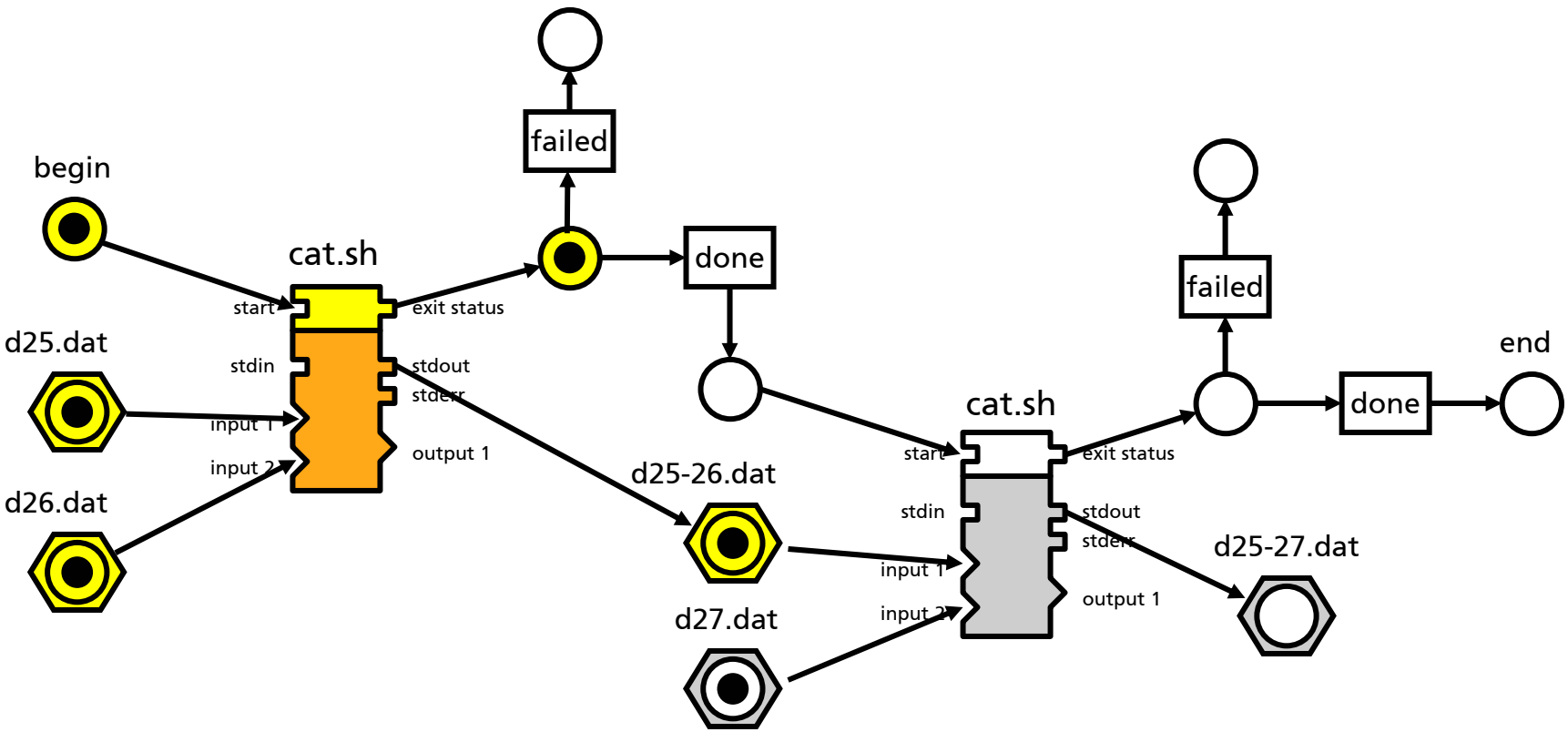
Resource Mapping



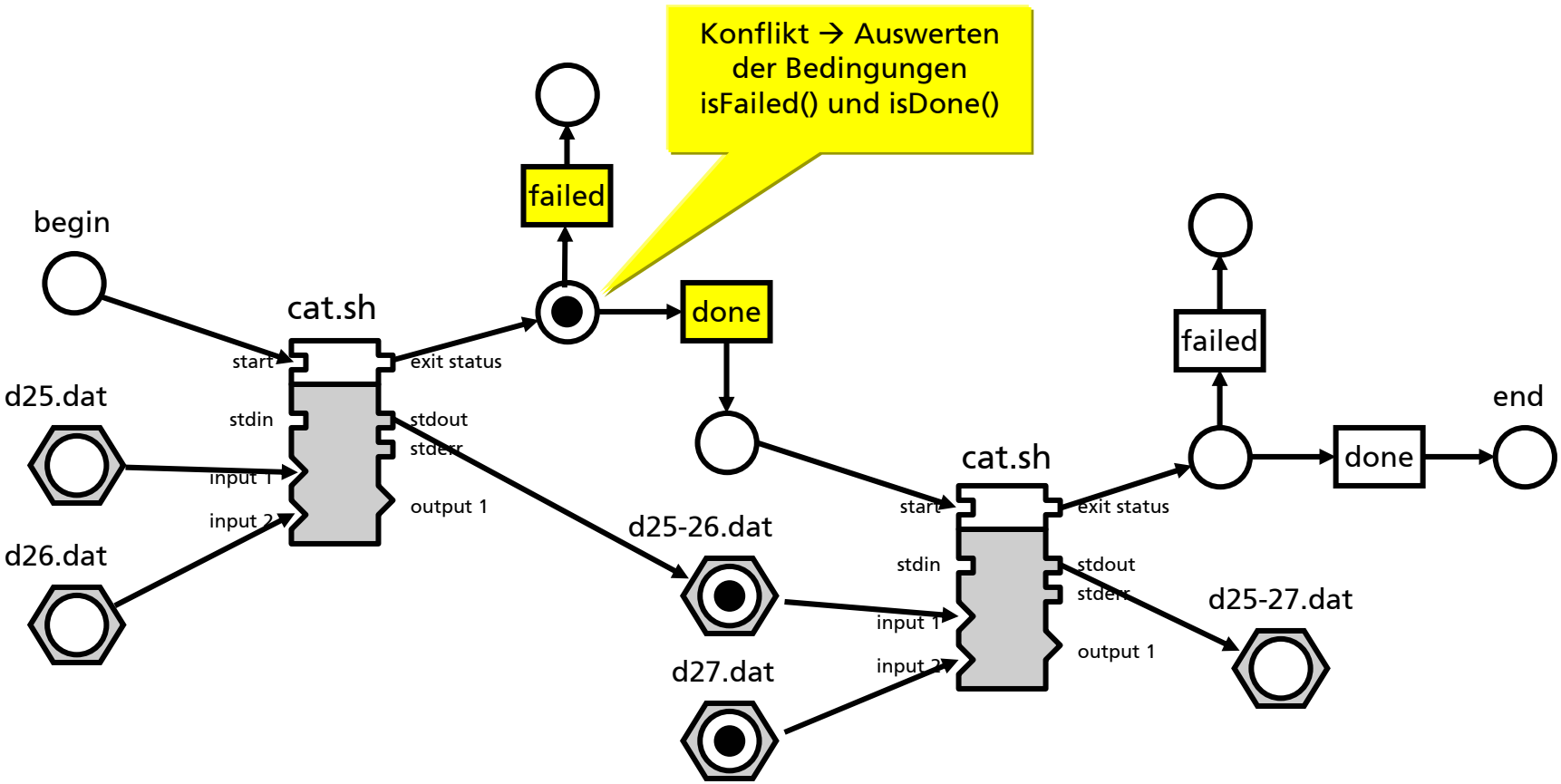
Verfeinern des Petrinetzes → Einfügen eines GridFTPs



... Schalten der Transition



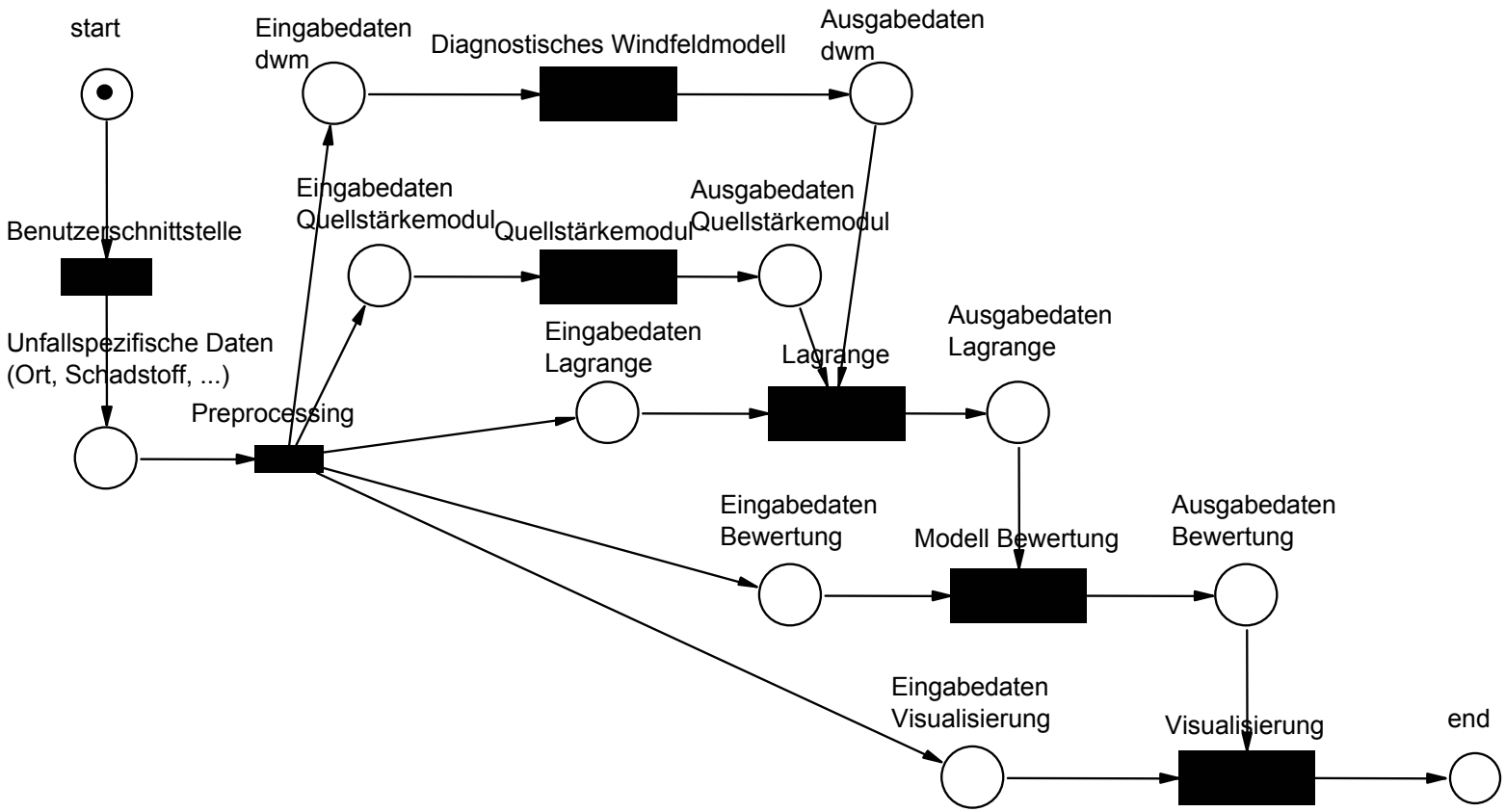
Sammeln aller aktivierten Transitionen...



Beispiel einer Grid-Anwendung: ERAMAS

ERAMAS: Schadstofftransport in der Atmosphäre:

Unfall → Quellstärkemodul → Atmosphärenmodell → Expositionsmodell



Ausblick

Ausblick

Petrinetze und OGSA?	Wie gliedert sich Prozessmodellierung mit Petrinetzen in OGSA ein?
Feingranulare Kopplung?	Bisher: Eine Transition → eine Softwarekomponente
	Zukunft: Eine Transition → eine spezielle Funktionalität einer Softwarekomponente (Methodenaufruf)
	➔ Komponentenarchitektur a la CORBA, WebService
Simulation von Petrinetzen	Prognose für „advanced reservation“ von Ressourcen (→ Scheduler) auf Basis von Software und Hardware-Benchmarks
Weitere Informationen	http://www.fhrg.fhg.de
	http://www.andreas-hoheisel.de