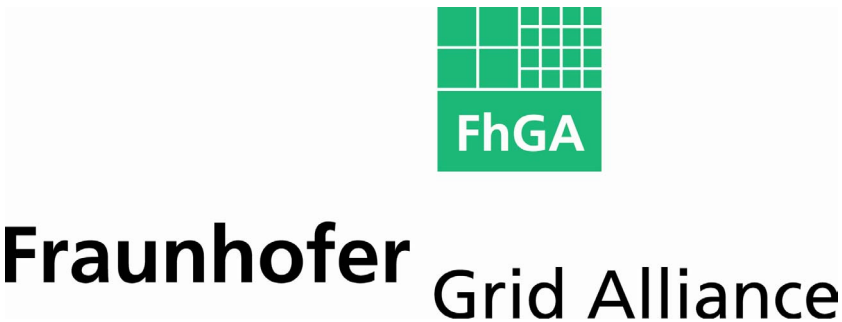
Workflow Specification in the Fraunhofer Resource Grid



Workflow Specification in the Fraunhofer Resource Grid

Global Grid Forum 10 – Berlin, Germany

Workflow Management RG

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

Franz-Josef Pfreundt
(pfreundt@itwm.fhg.de)

FhRG Fraunhofer Resource Grid

FhGA Fraunhofer Grid Alliance

Outline

Fraunhofer Resource Grid

Grid Application Definition Language
- GResourceDL
- GJobDL
- Petri Nets

Workflow Enactment
- Grid Job Handler

Our Contributions

Fraunhofer Resource Grid (FhRG)

Fraunhofer-Gesellschaft

58 Fraunhofer Institutes, focus: applied research
> 40 different locations in Germany
staff ≈ 12,700

Fraunhofer Resource Grid

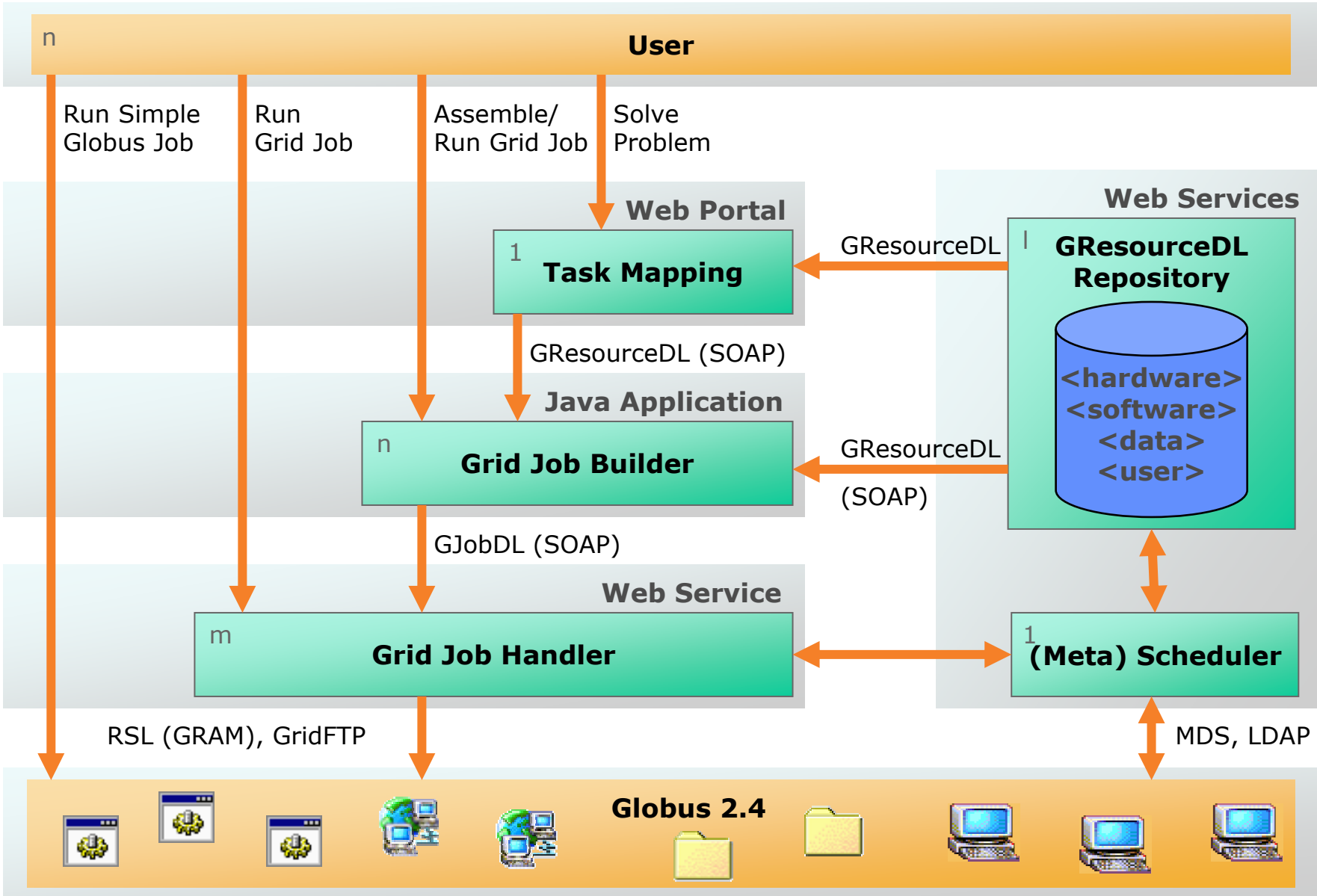
5 Fraunhofer Institutes
Development and implementation of a computing Grid
Focus: Computational science and engineering applications

eXeGrid
execution grid

Open Source Software:
http://www.eXeGrid.net/

FhRG Fraunhofer Resource Grid

FhGA Fraunhofer Grid Alliance


n
User
Run Simple Globus Job
Run Grid Job
Assemble/ Run Grid Job
Solve Problem
Web Portal
1
Task Mapping
GResourceDL
l
GResourceDL Repository
<hardware>
<software>
<data>
<user>
Web Services
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 2.4

How to define a Grid application?

Grid Application Definition Language (GADL)

should enable us:

- to describe complex workflows
- to hide hardware resources
- to execute workflows on the Grid
- to monitor the job status
- to integrate it as Grid service

Grid Application Definition Language (GADL)

GADL	Set of XML-based description languages needed to define and to execute Grid applications

The GADL consists of:

GResourceDL	Description of resources
GJobDL	Description of Grid jobs → Set of resources + workflow
GInterfaceDL	Interface definition of software components
GDataDL	Description of data

GResourceDL example that depends on other resources

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE fhrgResources (View Source for full doctype...)>
<fhrgResources>
 <resource id="sleep" type="software">
 ...
 <dependencies type="depends">
  <resourceRef id="linux" type="softwareClass" />
  <resourceRef id="glibc-2-3" type="softwareClass" />
  <resourceRef id="x86" type="hardwareClass" />
 </dependencies>
 ...
</fhrgResources>
```

depends, conflicts, provides, suggests

GResourceDL example that provides other resources

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE fhrgResources (View Source for full doctype...)>
<fhrgResources>
 ...
 <resource id="harlekin.first.fraunhofer.de" type="hardware">
  <dependencies type="provides">
   <resourceRef id="x86" type="hardwareClass" />
   <resourceRef id="network-ethernet-100" type="hardwareClass" />
   <resourceRef id="linux-kernel-2-4-18" type="softwareClass" />
   <resourceRef id="glibc-2-3" type="softwareClass" />
  </dependencies>
  <authorization>
   <userGroup id="all" read="true" write="false" />
   <userGroup id="first" read="true" write="true" execute="true" />
  </authorization>
 </resource>
</fhrgResources>
```

What is a Grid Job?

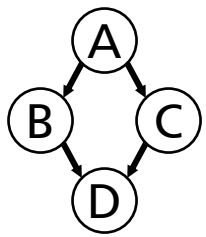
Grid job	Composition of Grid resources forming Grid applications with specific workflow
Grid resource	Software, hardware, data, (people)
Atomic job	Single task, indivisible component of a Grid job Now: Execution of a software component Future: Invocation of a Web Service method call
GJobDL	Description of Grid jobs on an abstract level Independent from Grid infrastructure Connecting software components and data Based on Petri Nets XML

Why Petri Nets?

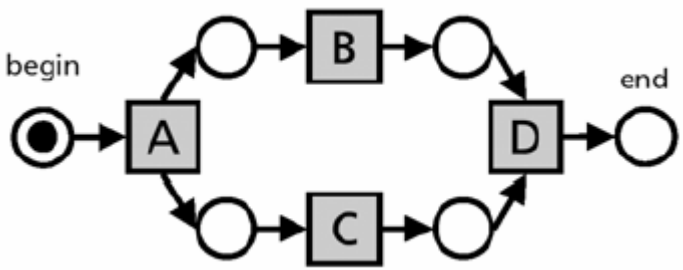
Why Petri Nets?

Problem Description of complex workflows of grid jobs

DAG Directed Acyclic Graph
loops not explicitly defined by graph



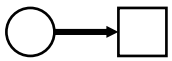
Petri Nets Directed graph, can be made Turing complete



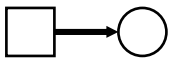
Petri Nets

Places Files, buffers, control places

Transitions Software components, control transitions



Arcs from places to transitions (Place is input place of transition)



Arcs from transitions to places (Place is output place of transition)

Tokens Data, State (done, failed)

Rule A transition is activated if all input places are filled with tokens and all output places have not reached their maximum capacity of tokens

Refinement A single part of a Petri Net can be replaced by a sub Petri Net

Description of state A Petri Net describes workflow and state of a system

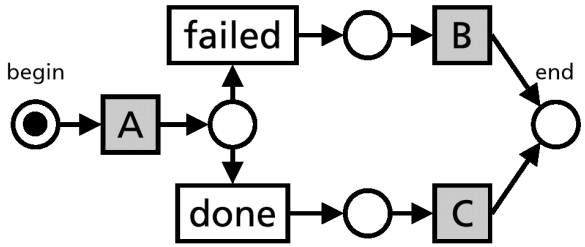
Transitions of type “MethodCall“

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

- unpack()
- transferData()
- transferExecutable(String softwareTransitionId)
- transferData(String softwareTransitionId)
- clear()

Transitions of type “Condition“

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

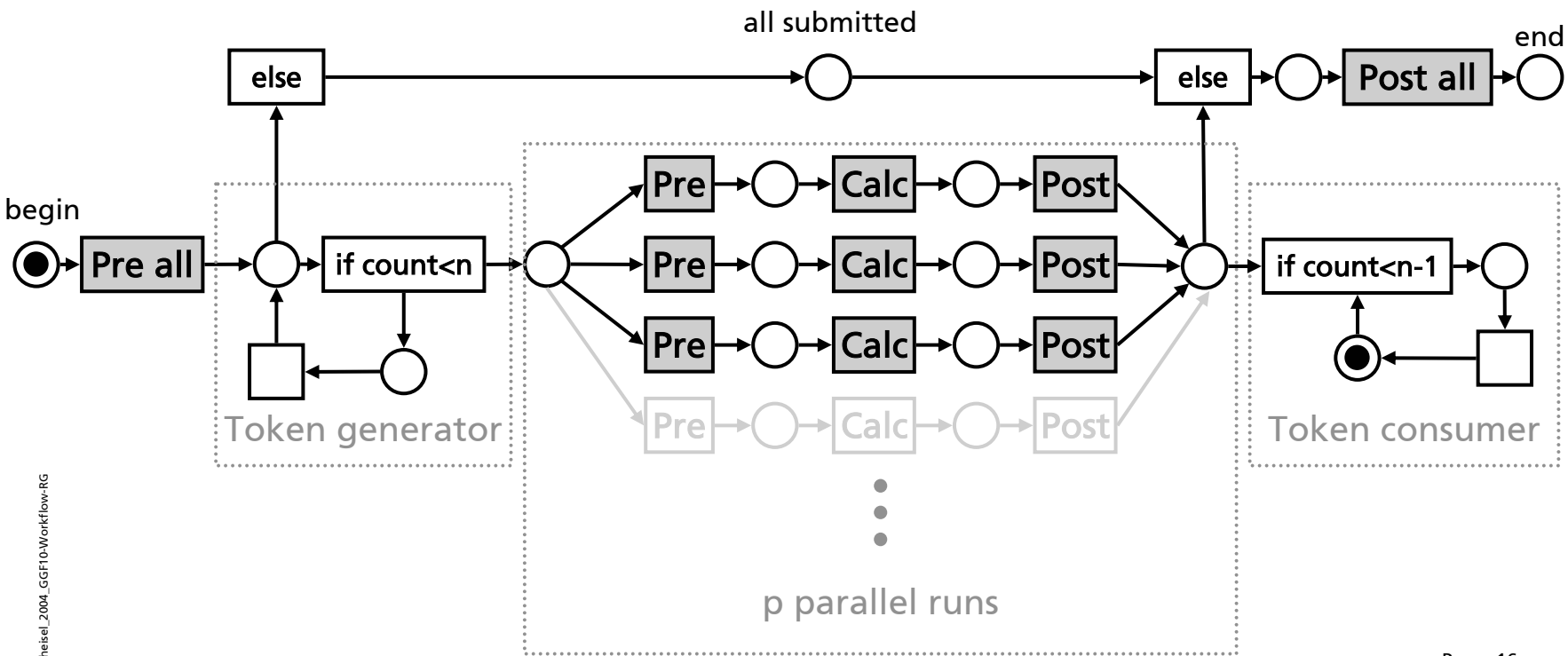


- isDone()
- isFailed()
- countLT(int maxCount)
- negate(String transitionID)
- timeGE(long msSince1970)
- idleTimeGE(long milliseconds)

Parameter Study

n = total number of runs (e.g. 10,000)

p = number of parallel runs (e.g. 128)

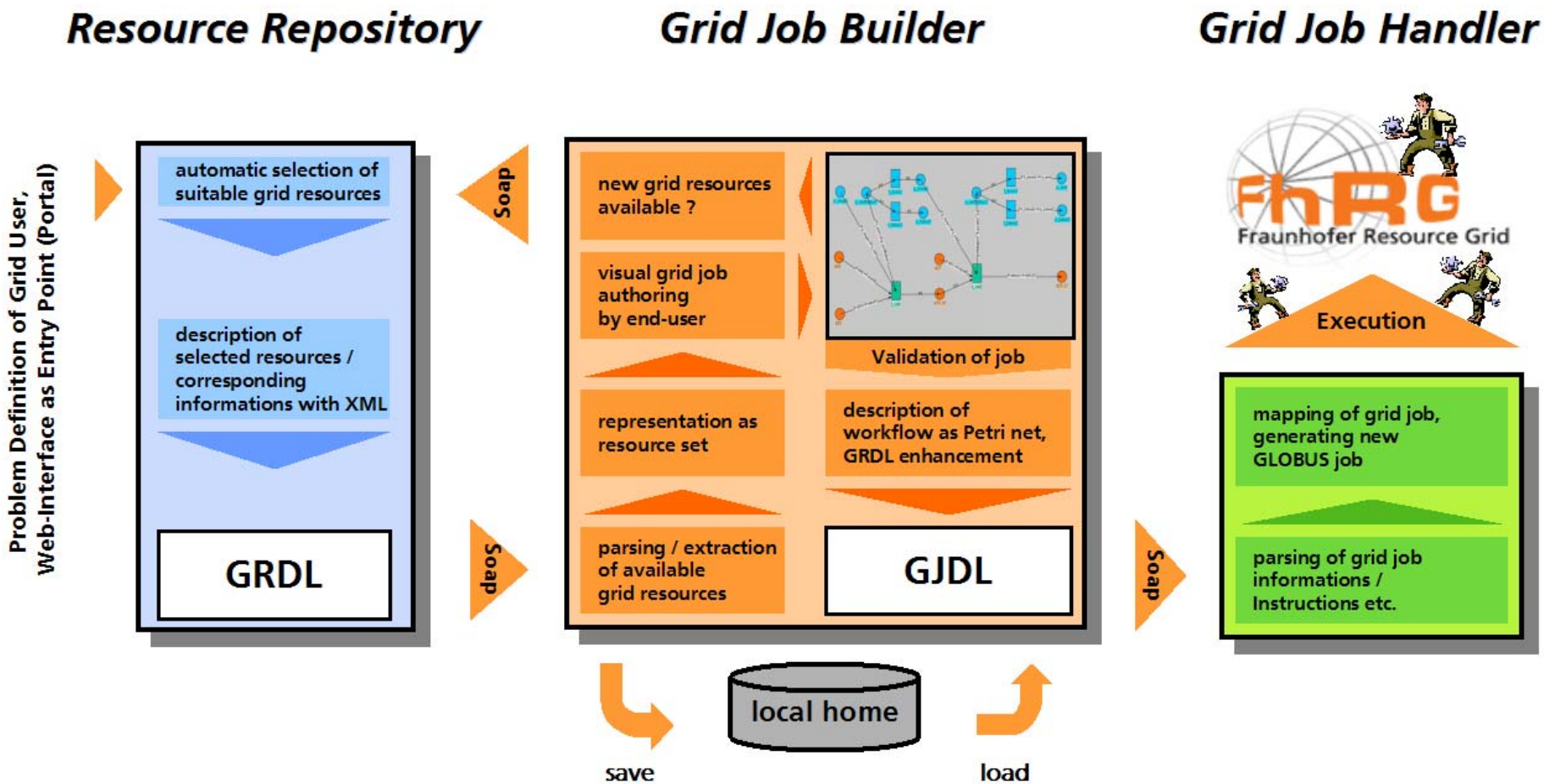


Description of Petri Nets in XML

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>
```

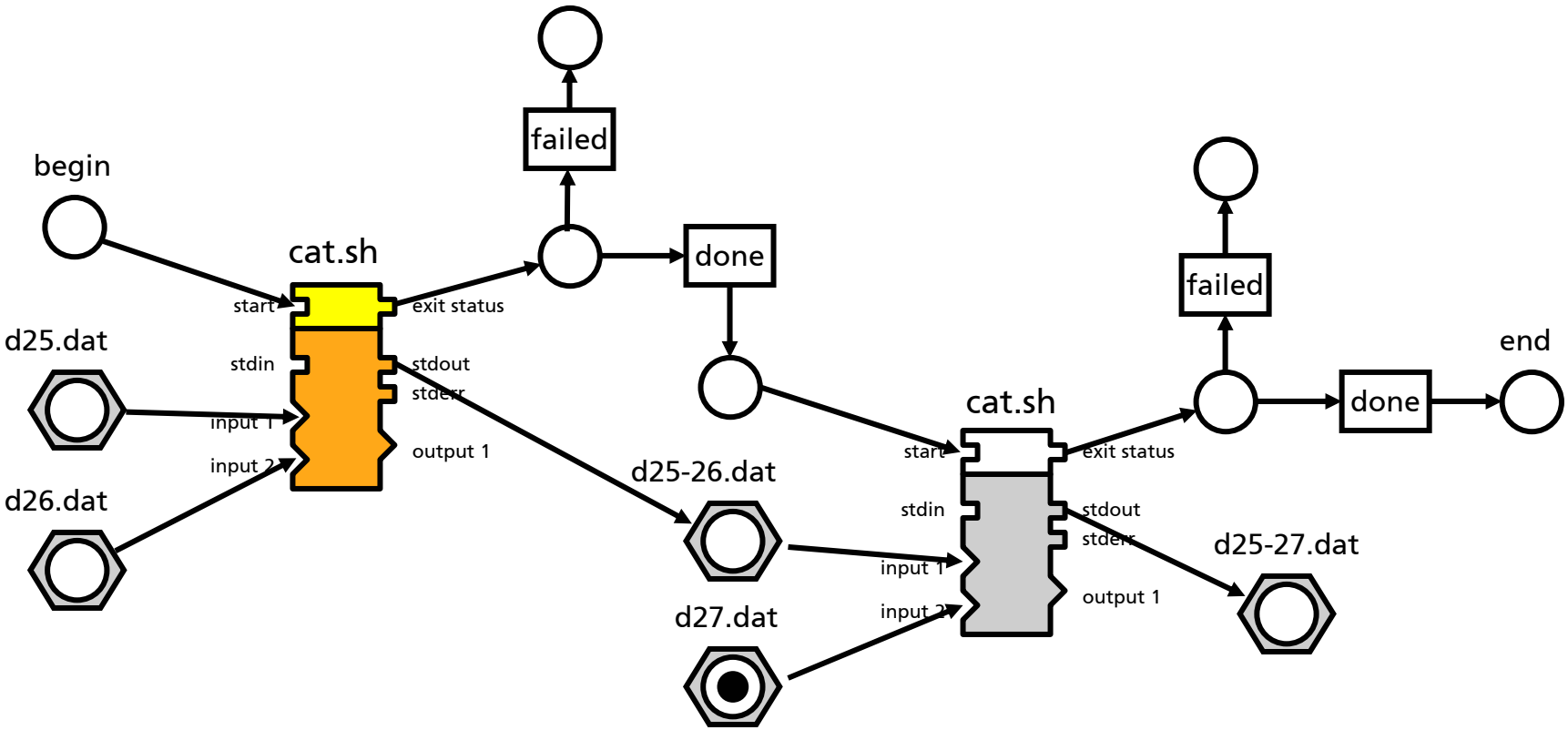
Workflow Enactment



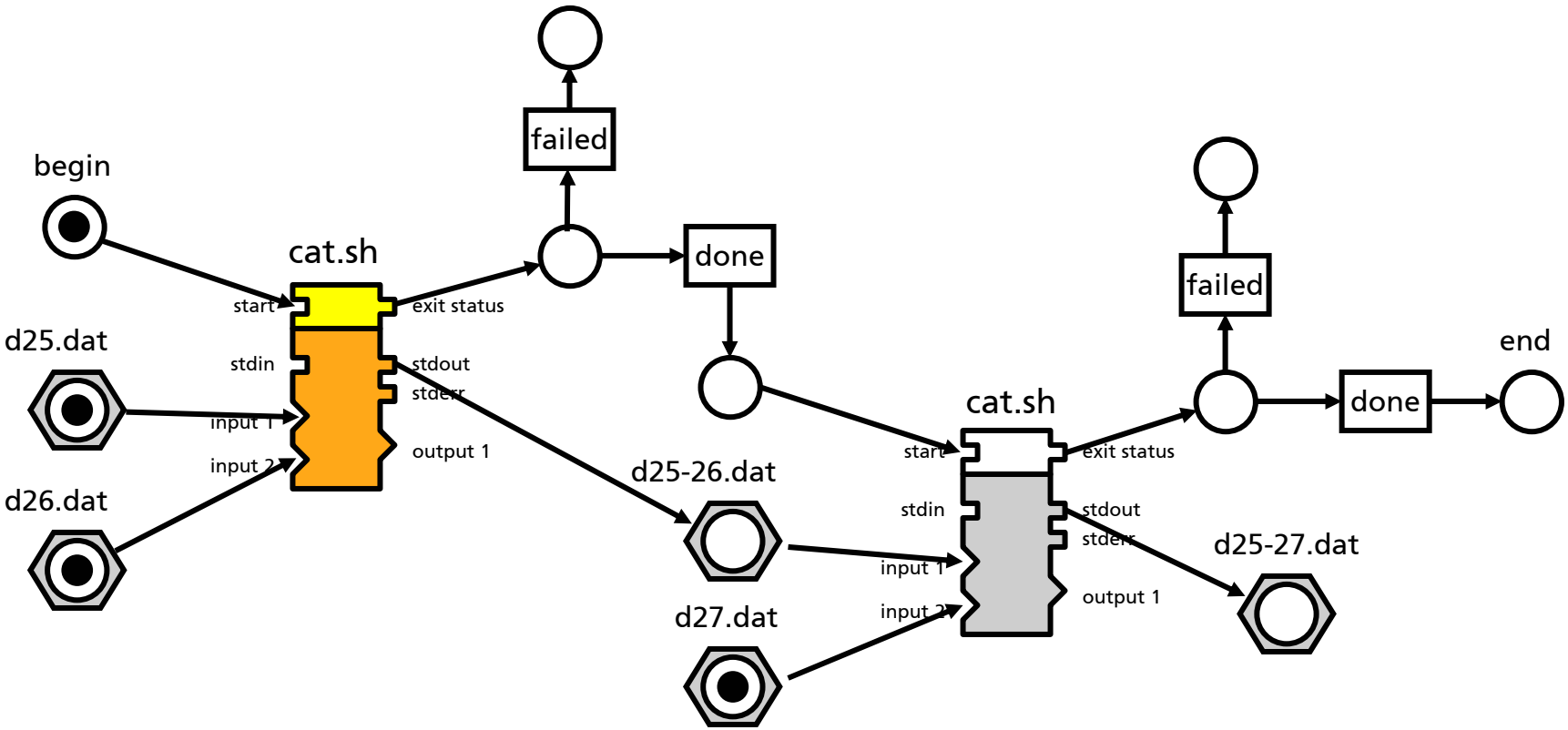
Application flow of Grid Job Handler

- Read the GJobDL document
- Create Petri Net from this job description
- Verify the Petri Net (well-formedness, liveliness, deadlocks, pits, ...)
- Start the Grid Job (own thread)
- Collect all activated transitions
- Evaluate conditions
- Invoke resource mapping → repository, (meta-)scheduler
- **Refine the Petri Net**
  → insert GridFTPs, fault management, etc. if necessary
- Create and submit atomic jobs using grid middleware (e.g. GRAM)
- The transition fires, if atomic job is “done“ or has “failed“.

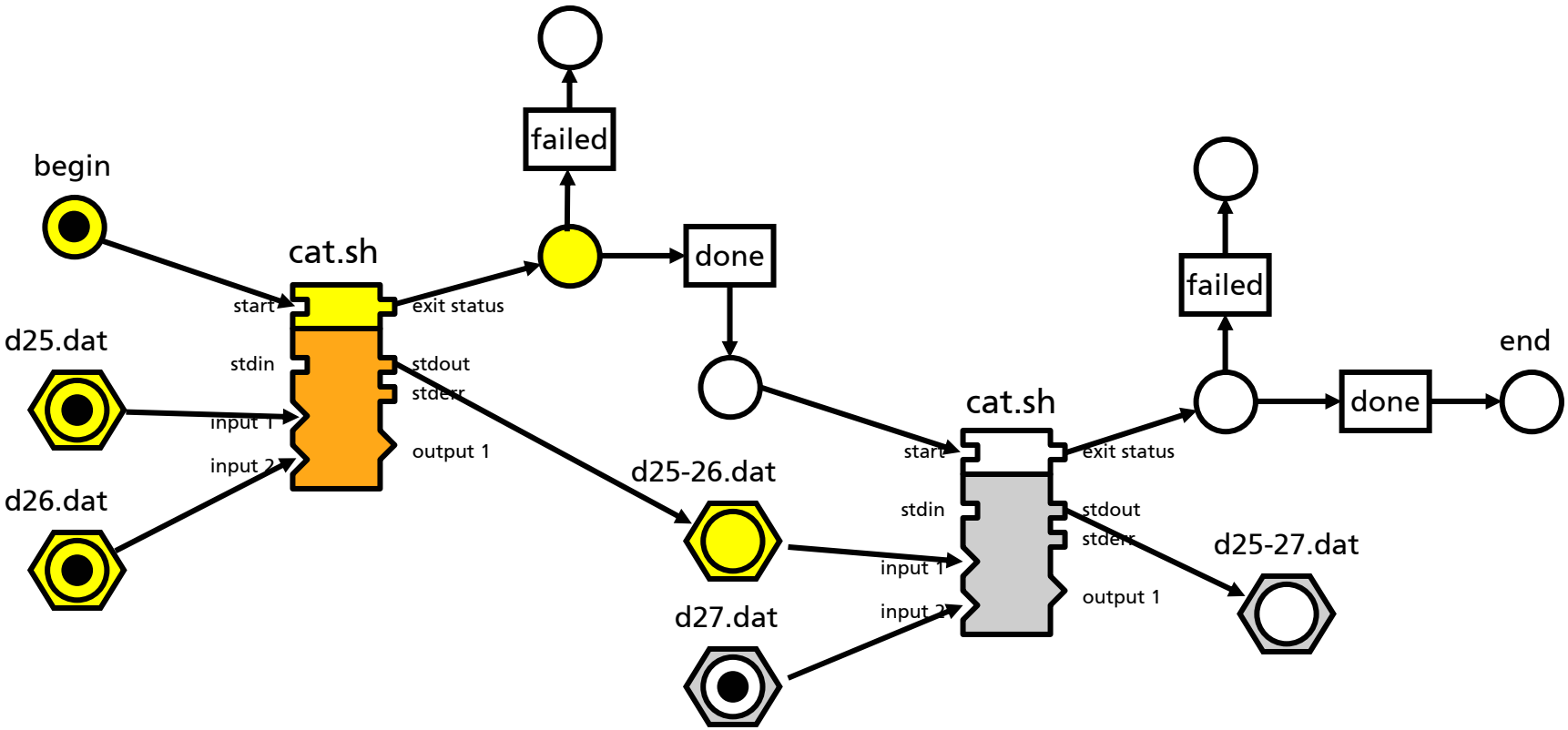
Grid Job Handler



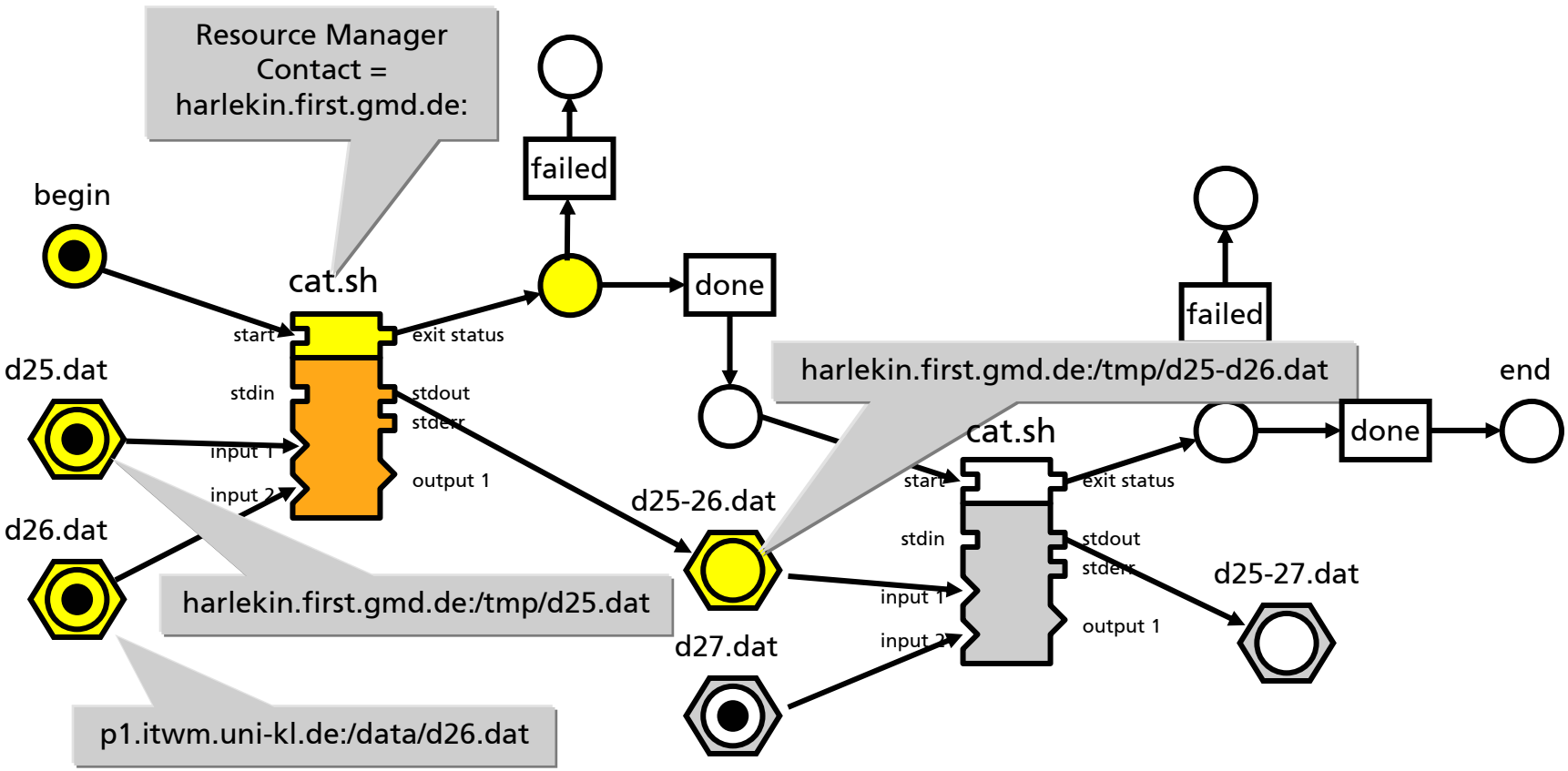
Collect all enabled transitions …



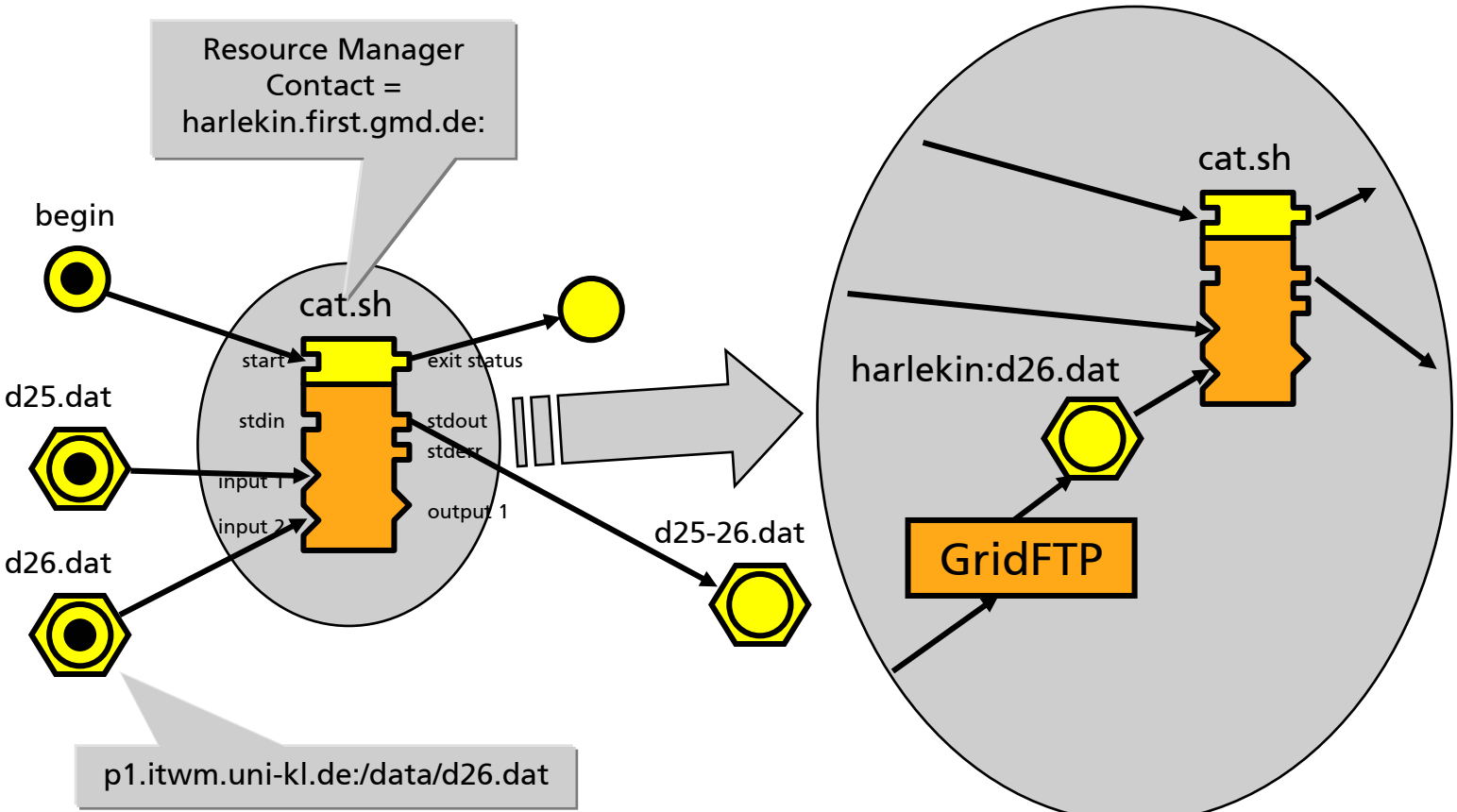
Lock input and output places



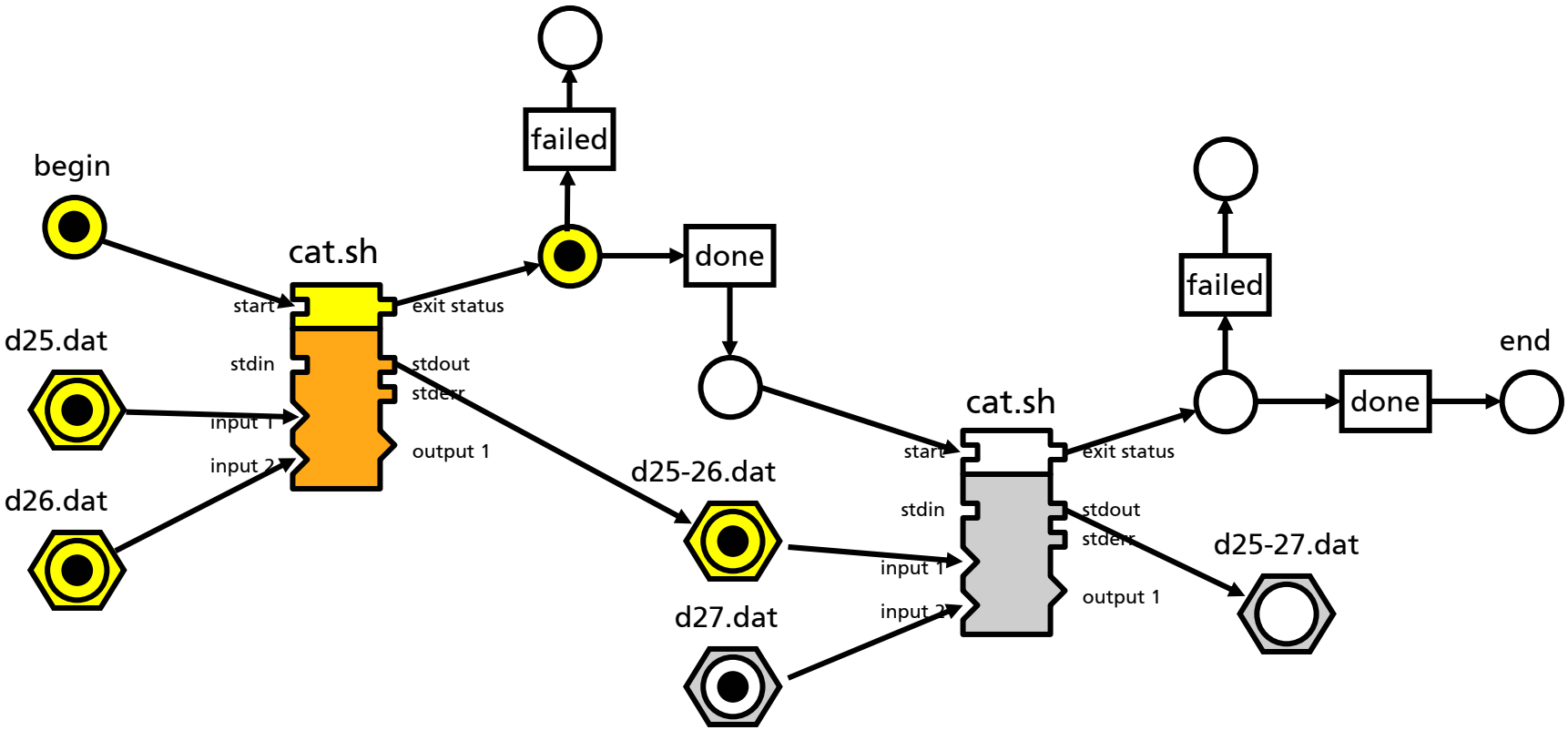
Resource mapping



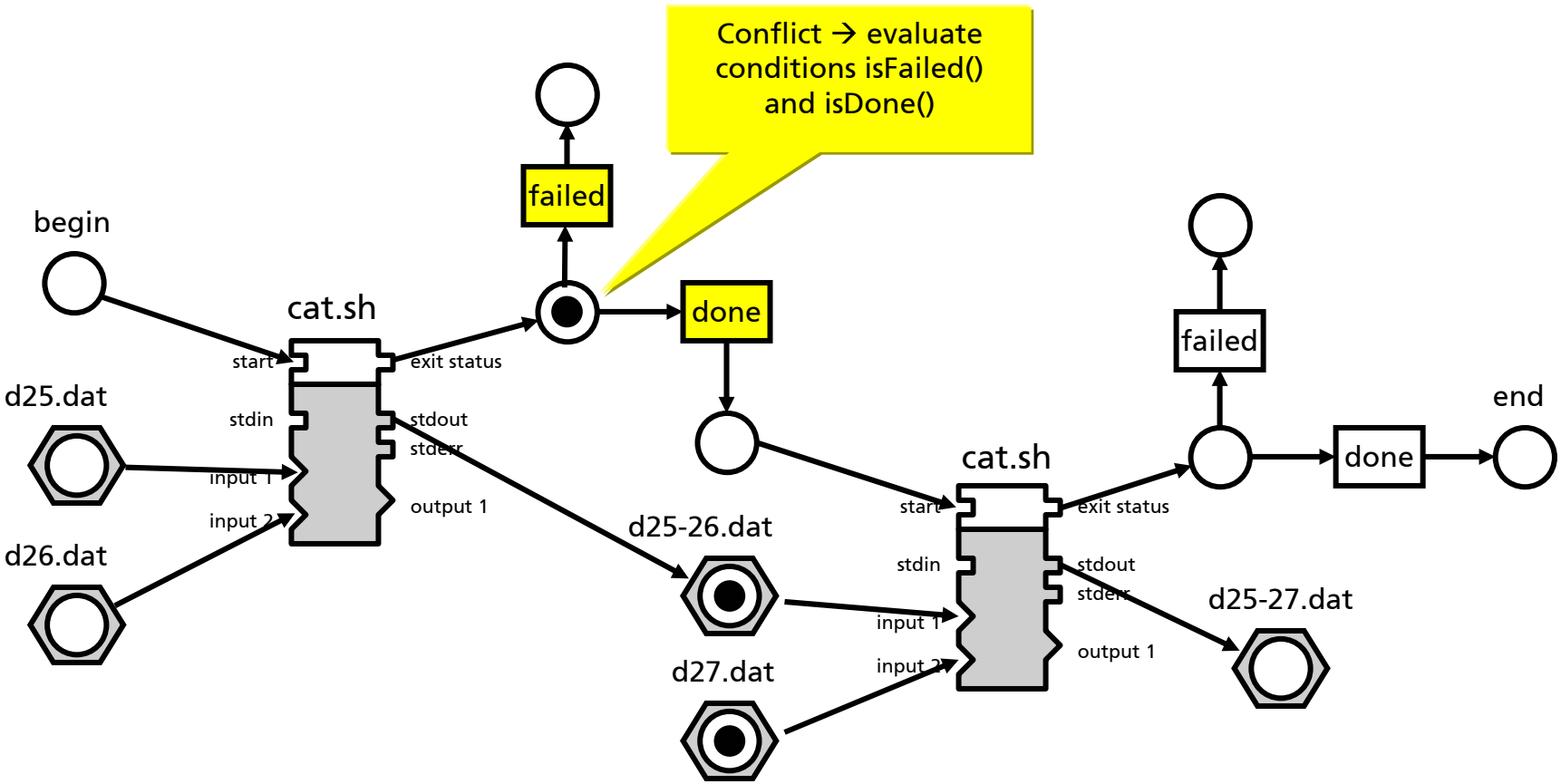
Refine the Petri Net → insert GridFTP



... The transition fires



Collect all enabled transitions …



Fraunhofer Resource Grid Control Desktop

File Grid Job Security Help

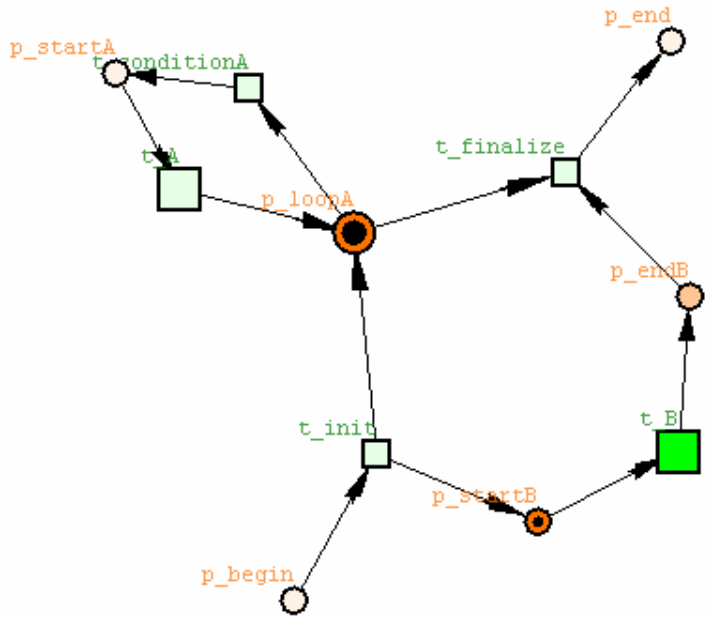
GJobDL: C:\hoheisel\I-Lab\executor\examples\cycleAWhileExecutingB.xml

Create Grid Job

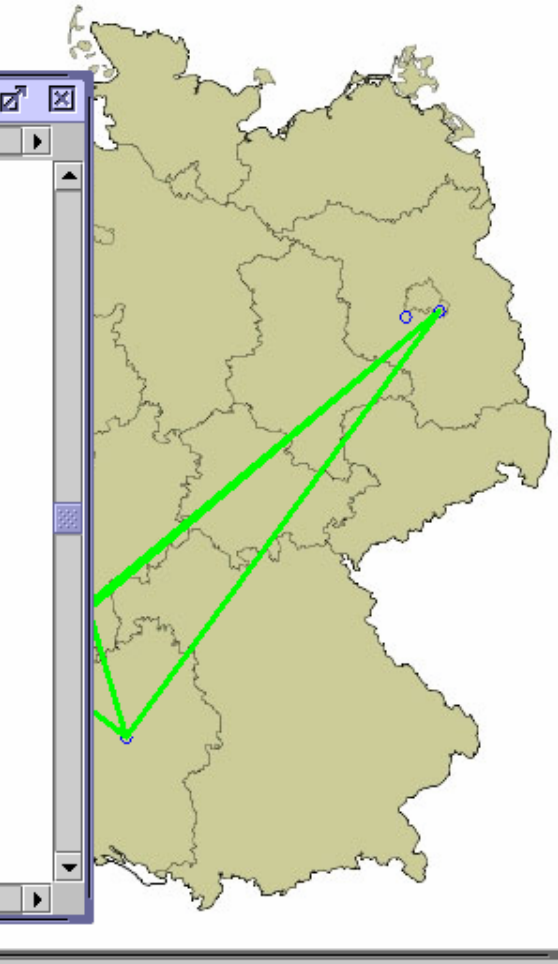
```
      <transitionRef id="t_conditionA"/>
      <placeRef id="p_startA"/>
    </arc>
    <arc id="arc10" type="P2T">
      <placeRef id="p_startA"/>
      <transitionRef id="t_A"/>
    </arc>
    <arc id="arc11" type="T2P">
      <transitionRef id="t_A">
        <outputRef id="stdout" type="std
      </transitionRef>
      <placeRef id="p_loopA"/>
    </arc>
    <arc id="arc12" type="P2T">
      <placeRef id="p_loopA"/>
      <transitionRef id="t_finalize"/>
    </arc>
  </job>
</fhrgJob>
```

touchgraphed Petri-Net for GridJob #1

Zoom



Karte



Grid Job #1

Run Stop Show Workflo

status	start time	end time					
ACTIVE	Feb 27 15:5...	N/A	1	harlekin.firs...	./sleep	20	
DONE	Feb 27 15:5...	Feb 27 15:5...	2	harlekin.firs...	./date		$(HOME)/re...
DONE	Feb 27 15:5...	Feb 27 15:5...	3	harlekin.firs...	./date		$(HOME)/re...

Our Contribution

Our Contribution

GResourceDL	Grid Resource Definition Language to describe arbitrary dependencies (software, hardware)
GJobDL	Grid Job Definition Language based on Petri Nets
Implementation	Working implementation of workflow enactment Web Service available as Open Source → http://www.eXeGrid.net/

More Information: http://www.fhrg.fhg.de/
http://www.eXeGrid.net/
http://www.andreas-hoheisel.de/
andreas.hoheisel@first.fraunhofer.de

FhRG Fraunhofer Resource Grid
FhGA Fraunhofer Grid Alliance