the globus alliance
www.globus.org/cog

Java CoG Kit Workflow

Gregor von Laszewski

Argonne National Laboratory

University of Chicago

gregor@mcs.anl.gov

http://www.cogkit.org

Updated slides will be available on the CoG Kit web site

Funding sources & Acknowledgement

- The Java CoG Kit receives funding from the following sponsors
  - DOE MICS
  - NSF NMI
- Previous versions of the CoG Kit also received funding from
  - NCSA Alliance
- Please, contact gregor@mcs.anl.gov in case you like to work with us more closely.
- Acknowledgement:
  - CoG Team, Globus Team, Globus Alliance, many others as listed on www.cogkit.org

Community

- Call on the community to help us with extending and improving the CoG Kit

Outline

- Related to the Java CoG Kit …

- What is the CoG Kit?
- History of workflow
- Concepts of Workflow
  - API
  - GridAnt/Karajan
  - GridShell
- Graphical Interfaces
- Conlusion

Observation

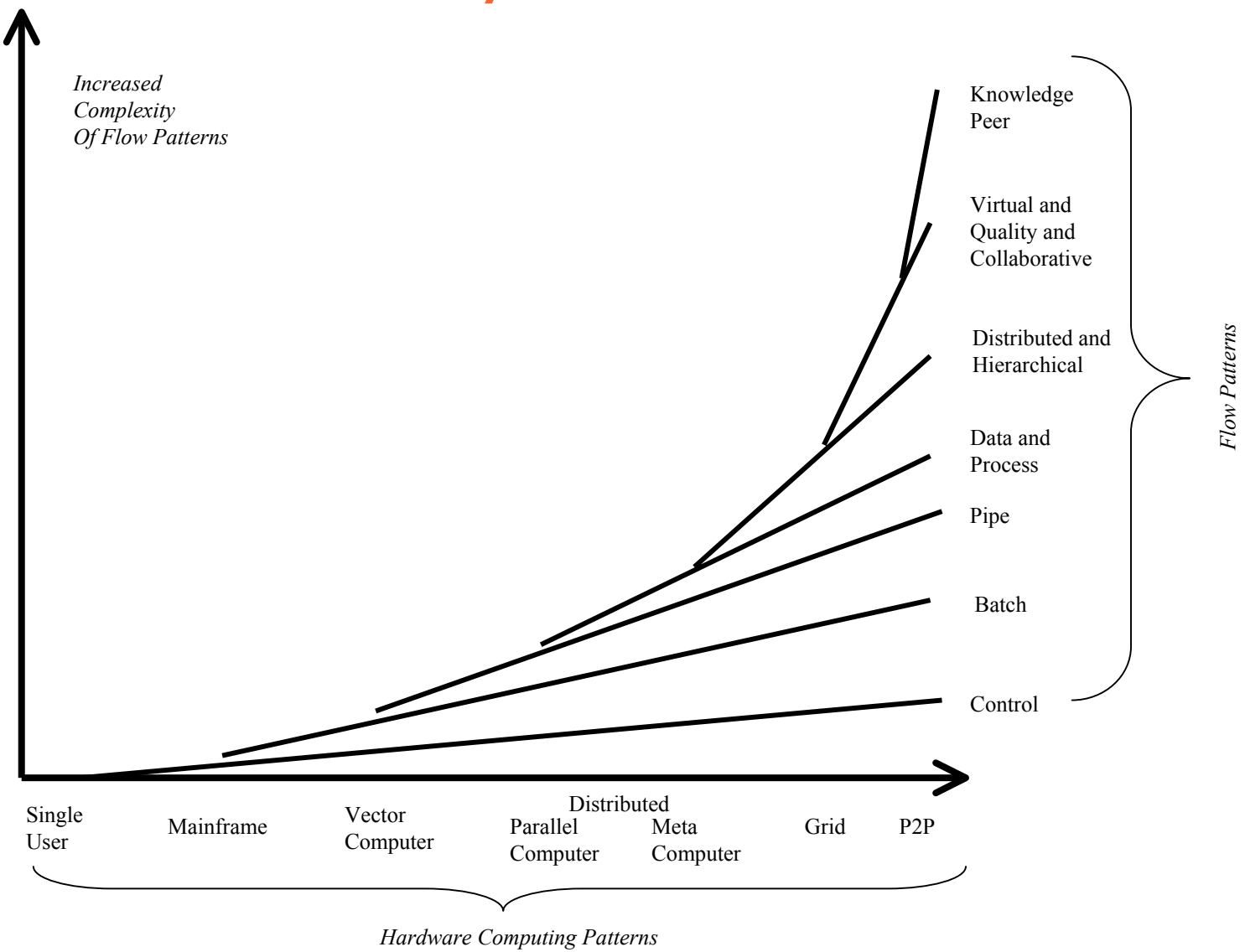
- **Problem**
  - Many application developers desire to program the Grid in familiar higher level frameworks that allow rapid prototyping.
- **Solution**
  - We propose to reuse a variety of commodity tools, protocols, approaches, methodologies, while integrating Grid software based on the Globus Toolkit
    - Easier development of advanced Grid services
    - Easier and more rapid application development
    - Easier deployment of Grid services
    - Code reuse and use of component repositories
    - Use of Web services as part of the Grids
    - Widespread use of the Grid
    - Use of commodity technology is not limited to the client!

What is the Java CoG Kit ?

- (Presentation earlier this week)
- Make Grid programming and use easier
- Includes Workflow abstractions:
  - Language based on XML/ant
  - Workflow engine called karajan (German conductor)
  - Workflow viewer and monitor
  - Portal components (to be completed)

Lets focus on workflow

History: Flow Patterns



von Laszewski,
et al.
"Grid Middleware"

von Laszewski

History of Workflow in CoG Kit

1994	von Laszewski dataflow and job control Metacomputing system in tcl/tk
1996	von Laszewski data and workflow Metacomputing system in Java
1996	Start of the Globus Project
1997	a) Fault tolerant high throughput broker in Java for SnB application b) Less capable broker in sh as Globus does only use sh
1998	Improved GUI for the Java CoG Kit high throughput broker
1998	Development of Condor-G (C based)
…	A long period of inactivity as Globus toolkit is only C not Java
2000	Official start of the Java CoG Kit
2001	Reimplementation of parts of the features of the system from 1994 and 1997
2002	GSFL, Java CoG Kit jglobus is essential part of GT3 and in 2005 in GT4
2003	Java CoG Kit GridAnt & Karajan, Workflow with GT2, GT3, SSH
2004	Java CoG Kit GridDesktop and GridShell, Workflow with GT2, GT3, SSH
2005	New integrated Release, Workflow with GT2, GT3, GT4, SSH, Condor

*** If a date is wrong please help us correcting it.**

Current status

Concepts of Workflow (in Java CoG Kit)

- Three ways to do workflow
  - Workflow with CoG Kit abstractions (API)
  - Workflow with GridAnt/Karajan (XML)
  - Workflow with the GridShell (in future)
- Which to chose depends on your requirements

Features

	Abstraction API *	GridAnt/ Karajan	GridShell
Java API	yes	undocumented	undocumented
XML	no	yes	no
If, While, For	implicit	yes	not yet
Caching	no	yes	no
Checkpointing	no	yes	no
Logging	no	not yet	yes
Viewer	no	yes	yes
Batch	no	yes	yes
Uses Karajan	no	yes	yes
Uses Abstr. API	yes	yes	yes

*** The features marked with * can be implemented by the programmer**

Abstractions

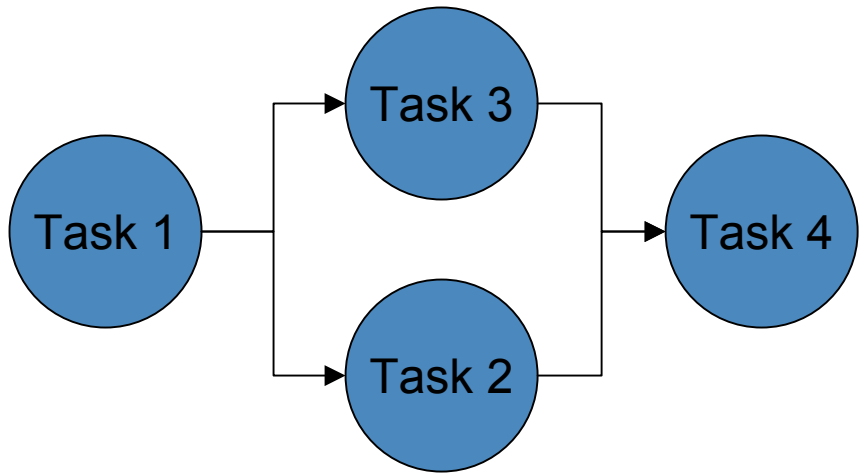
- Hypothesis:
  - With rapidly changing technologies it may be beneficial to have an abstraction that can be assisting in this technical challenge.
- Solution:
  - CoG Kit abstractions are defined for precisely that reason.

Java CoG Kit Abstractions

Java CoG Kit Abstractions

- We gave a talk earlier this week …

```
TaskGraph tg = new TaskGraphImpl();

public void create () {
  // define tasks
  .....
  /* Add the tasks to the TaskGraph */
  tg.add(task1);
  tg.add(task2);
  tg.add(task3);
  tg.add(task4);
  tg.addDependency(task1, task2);
  tg.addDependency(task1, task3);
  tg.addDependency(task2, task4);
  tg.addDependency(task3, task4);
}

public void submit() {
  TaskGraphHandler handler =  new TaskGraphHandlerImpl();
      try {
        handler.submit(tg);
      } catch (Exception e) {
        logger.error(``Some Error occured'', e);
        System.exit(1);
      }
}
```



Java CoG Kit GridAnt/Karajan

GridAnt

- One day …
  - Why not use ant as workflow engine?
  - We already have a Grid interface through the Java CoG Kit …
  - Prototype finished in less than a week
  - Worked great but …
    - Scalability, some language constructs were missing
  - Advantages uses ant which is used by millions of people

GridAnt/Karajan

- Goal:
  - improve what first version of GridAnt lacked and come up with a new version of GridAnt.
  - Be still able to call ant from GridAnt
  - Have more enhanced language features
  - Have an easy syntax

  - At the time we started BEPL was not available/ready
  - Be easier than BEPL

Control structures

Loops

```
<for name="iteration" from="1" to="4">
  <echo message="Iteration {iteration}"/>
</for>

<foreach name="iteration" in="one, two, three, four">
  <echo message="Iteration {iteration}"/>
</for>

<while>
  <condition>
    <false/>
  </condition>
  <echo message="You will never see this message"/>
</while>
```

Condition

```
<if>
  <condition>
    <condition1/>
  </condition>
  <then>
    ...
  </then>
    ...
  <condition>
    <conditionN/>
  </condition>
  <elseif>
    ...
  </elseif>
  <else>
    ...
  </else>
</if>
```

Variables

assignment

```
<set name="var" value="1"/>
```

Lists

```
<list:append>
  <argument value="1"/>
  <argument value="2"/>
  <argument value="3"/>
</list:append>
```

Operators

```
<math:sum>
  <argument value="1"/>
  <argument value="2"/>
  <argument value="3"/>
</math:sum>
```

Hashtable/map

```
<set name="map">
  <map>
    <map:entry key="name" value="John"/>
    <map:entry key="age" value="99"/>
  </map>
</set>
<echo message="Name: {1}, age: {2}">
  <map:get map="{map}" key="name"/>
  <map:get map="{map}" key="age"/>
</echo>
```

Templates

Definition

```
<templatedef name="sample">
  <default name="arg1" value="default1"/>
  <default name="arg2" value="default2"/>
  <echo message="arg1 is {arg1}"/>
  <echo message="arg2 is {arg2}"/>
  <echo message="arg3 is {arg3}"/>
</templatedef>
```

Using

```
<template name="sample"
  arg1="value1"
  arg2="value2"
  arg3="value3"/>
\end{lstlisting}
```

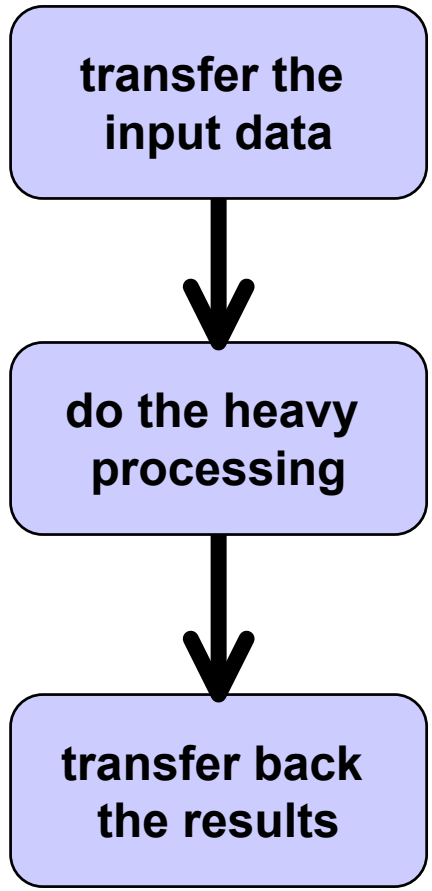
Recursion

Recursion

```
<element name="fibonacci" arguments="n">
  <if>
    <math:le value1="{n}" value2="2"/>
    <then>
      <argument value="1"/>
    </then>
    <else>
      <math:sum>
        <fibonacci>
          <math:subtraction from="{n}" value="1"/>
        </fibonacci>
        <fibonacci>
          <math:subtraction from="{n}" value="2"/>
        </fibonacci>
      </math:sum>
    </else>
  </if>
</element>
```

A simple Grid job

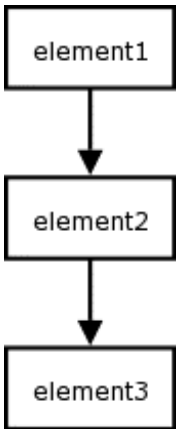
```
<gridTransfer
        srchost="localhost"  srcdir="/tmp" srcfile="in"
        desthost="{remote}"  destdir="/tmp"/>


<gridExecute host="{remote}"
                    executable="/usr/bin/climate"
                    args="512 512"
                    stdin="/tmp/in"
                    stdout="/tmp/out"/>


<gridTransfer  srchost="{remote}"  srcdir="/tmp"
                    srcfile="out"
                    desthost="localhost"  destdir="/tmp"/>
```



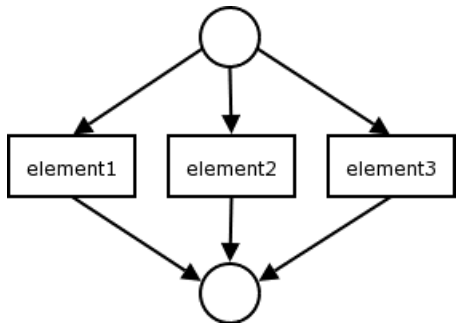
Using Loops

```
<grid name="default">
  <for name="index" from="1" to="20">
    <host name="lg0n{index}.pts.uml.mov" cpus="1">
    <gridExecute host="{host}"
                    executable="/usr/bin/climate"
                    args="512 512"
                    stdin="/tmp/in"
                    stdout="/tmp/out"/>

    </host>
  </for>
</grid>
```

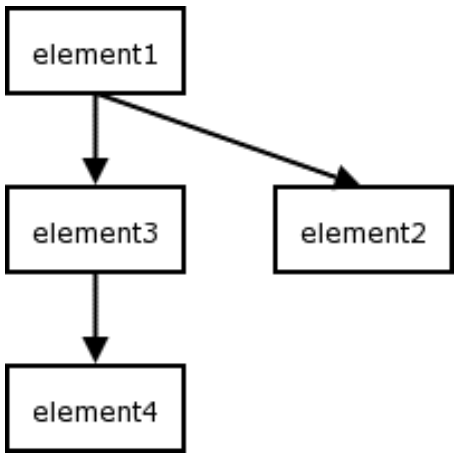
Parallelism




```
<sequential>
   <element1/>
   <element2/>
   <element3/>
</sequential>
```




```
<parallel>
   <element1/>
   <element2/>
   <element3/>
</parallel>
```




```
<parallel>
   <element1/>
   <element2 sync="false"/>
   <element3/>
</parallel>
```

Simple Types

Arithmetic

math:sum ::= a + b
math:product ::= a * b
math:subtraction ::= a - b
math:quotient ::= a/b
math:remainder
math:square ::= a*2
math:sqrt
math:equals ::= a =b
math:gt ::= a > b
math:lt ::= a < b
math:ge ::= a >= b
math:le ::= a <= b

Booleans

and, or, not, true, false

Lists

list:append, list:prepend, list:concat,
list:first, list:last, list:butFirst, list:butLast,
list:size, list:isEmpty

Map/Hashtable

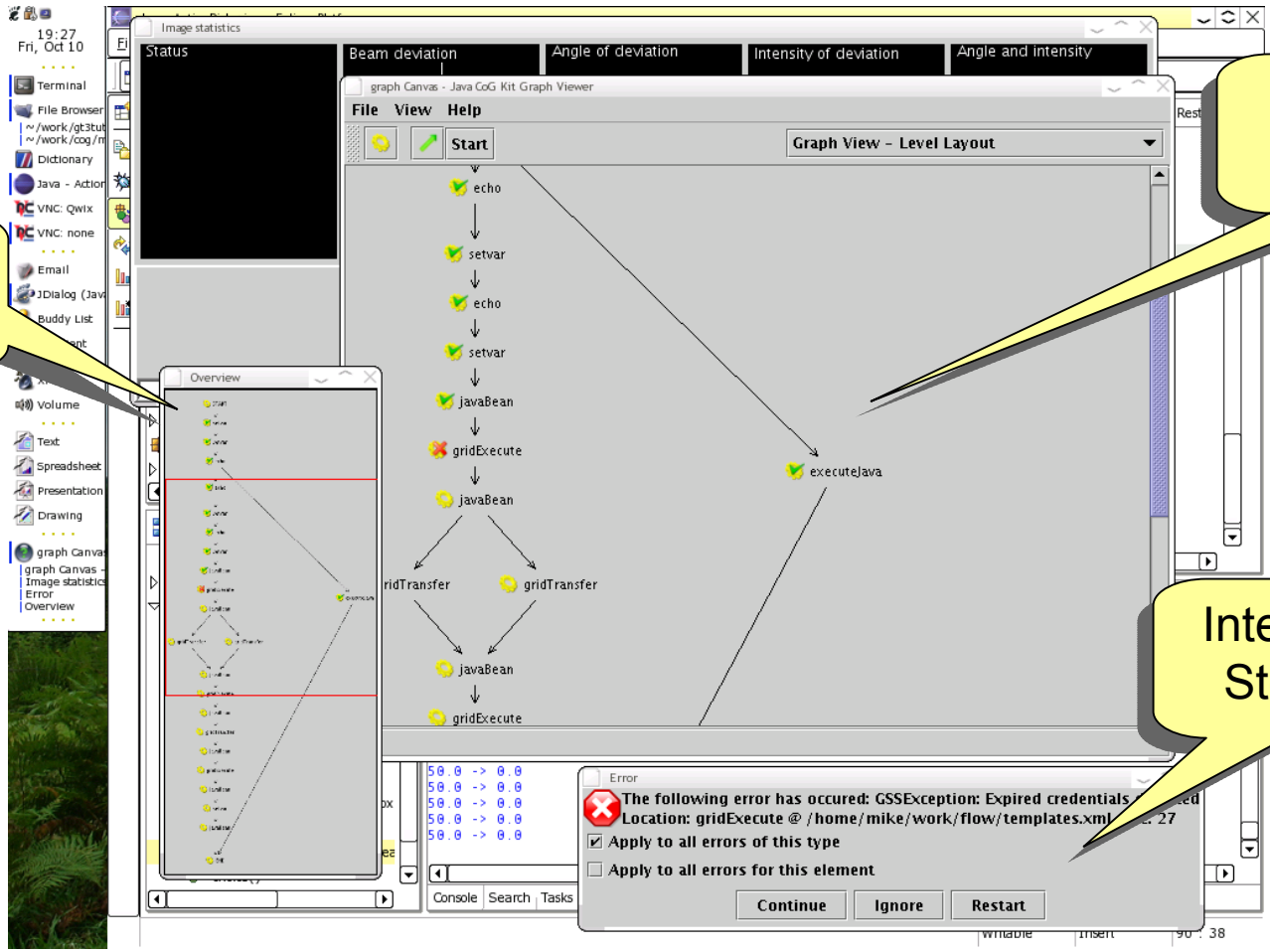
map:put, map:entry, map:get, map:contains,
map:delete, map:size

Error Handeling

```
<onError match="(.*Expired credentials detected.*)|(.*Proxy file.*not found.*)">
   <if>
      <condition>                                          <!-- avoid recursive errors -->
         <math:equals value1="{errorcount}" value2="1"/>
      </condition>
      <then>
         <!-- pop up a proxy init window -->
         <echo message="Invalid GSI credentials detected. Executing proxy init..."
         <executeJava mainClass="org.globus.cog.karajan.util.ProxyInitWrapper"/
         <!-- re-execute the element -->
         <echo message="Restarting failed element"/>
         <executeElement element="{element}"/>
      </then>
      <else>
         <!-- "Error count > 1" -->
         <!-- error produced by generateError are never intercepted -->
         <generateError message="{error}"/>
      </else>
   </if>
</onError>
```

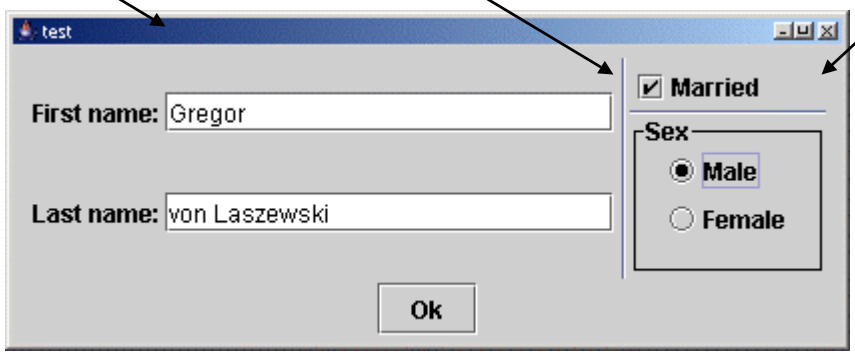
the globus alliance
www.globus.org/cog

Java CoG Kit
GridAnt Workflow

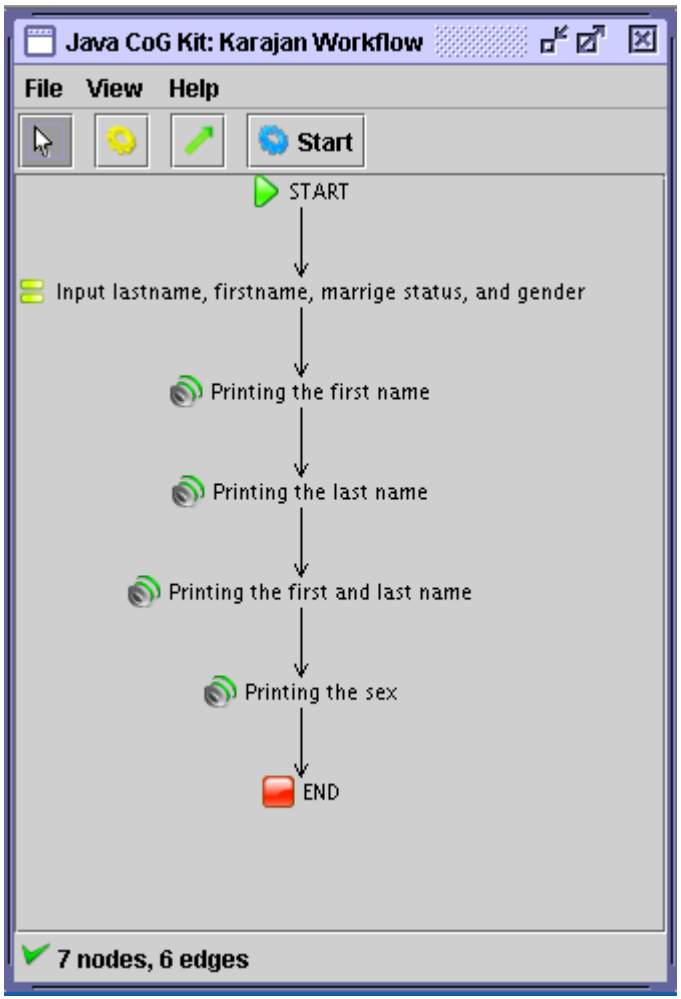


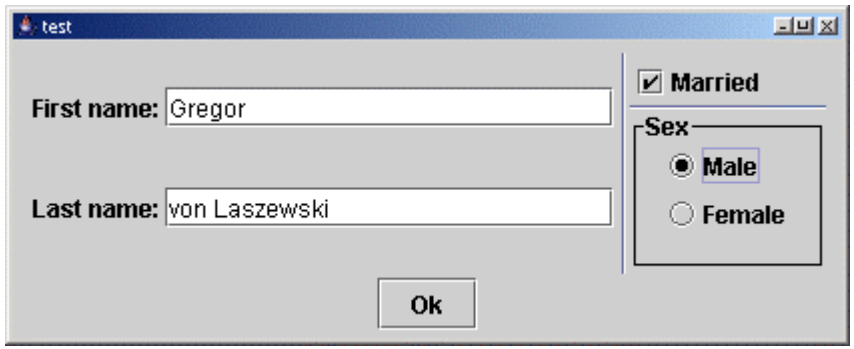
XML GUI Forms

```
<form:form title="test" id="form" waitOn="IDOk">

 <form:vbox>
  <form:hbox>
   <form:vbox>
    <form:hbox>
     <form:label text="First name: "/>
     <form:textField id="IDFirst" columns="20"/>
    </form:hbox>
    <form:hbox>
     <form:label text="Last name: "/>
     <form:textField id="IDLast" columns="20"/>
    </form:hbox>
   </form:vbox>

                    <form:VSeparator/>

   <form:vbox>
    <form:checkBox caption="Married"
                  id="IDMarried" halign="0"/>
    <form:HSeparator/>
    <form:radioBox caption="Sex" id="IDSex">
     <form:radioButton caption="Male“
                  id="IDMale"/>
     <form:radioButton caption="Female"
                  id="IDFemale"/>
    </form:radioBox>
   </form:vbox>
  </form:hbox>
  <form:button id="IDOk" caption="Ok"/>
 </form:vbox>

</form:form>
```



XML GUI Forms





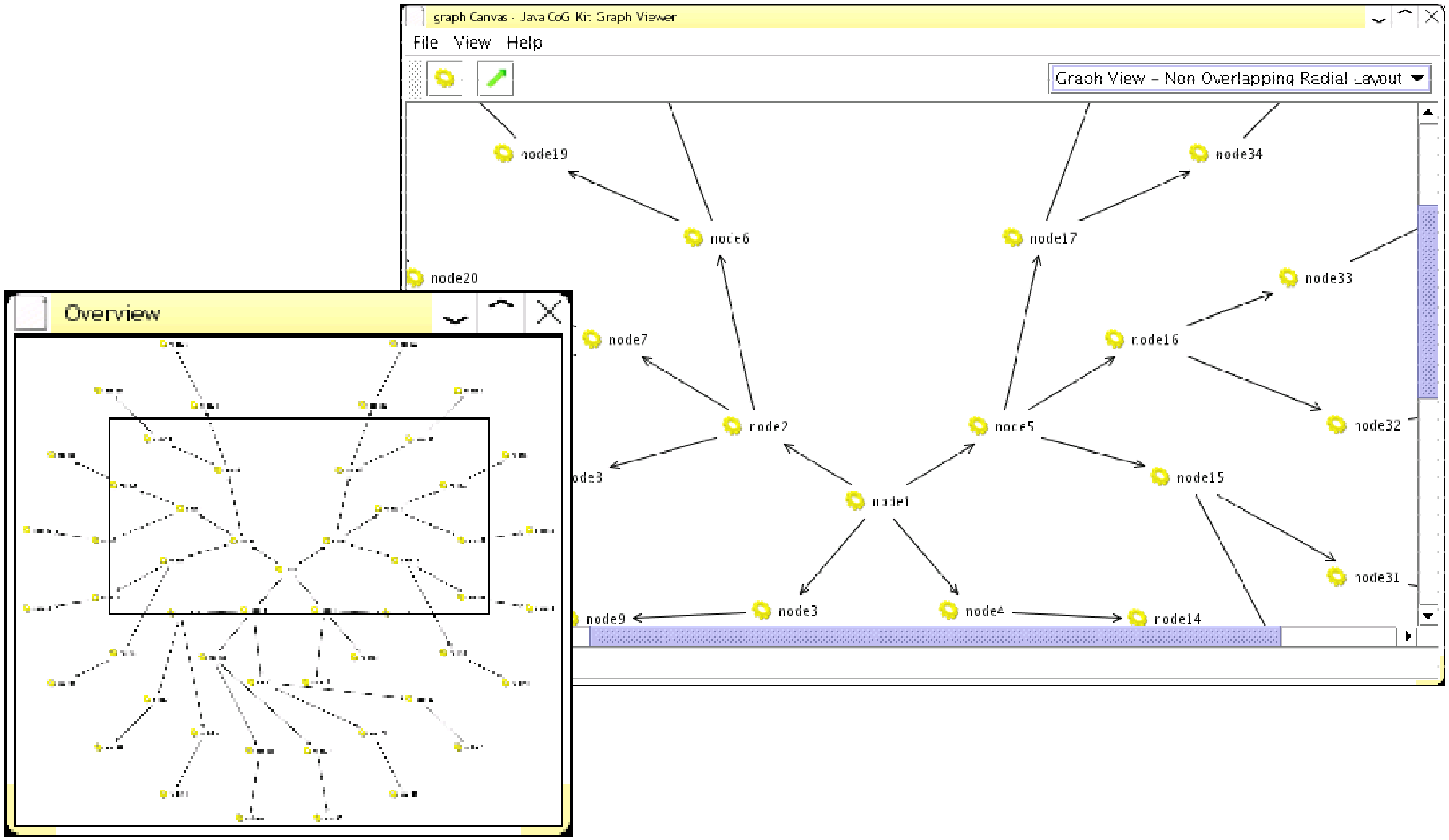
Applications of Workflow

With the Java CoG Kit

Graph in trivial XML

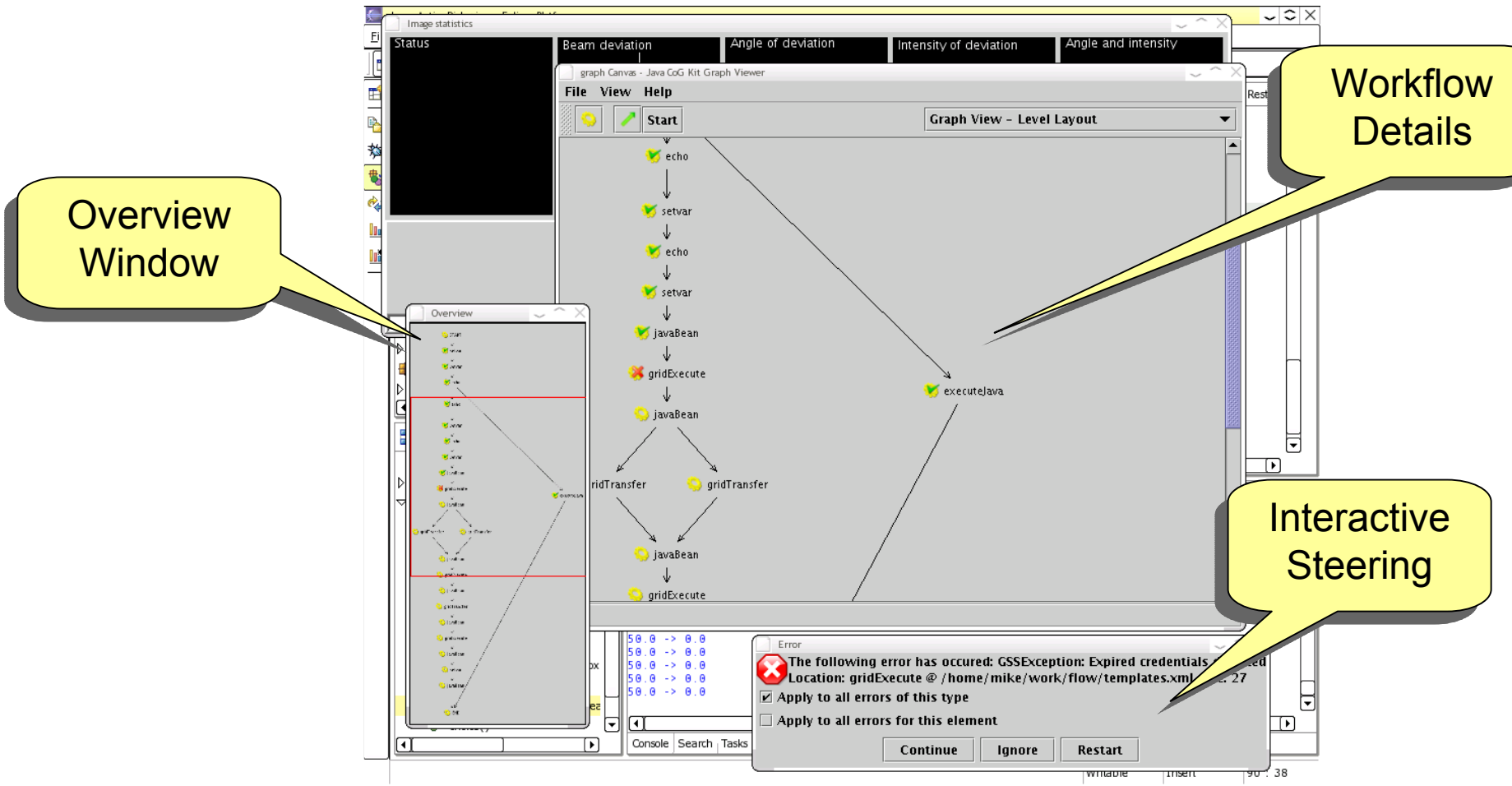
```
<graph id="simple-graph" name="Hierarchical Graph">
  <node nodeid="node1" name="node1"/>
  <node nodeid="node2" name="node2"/>
  <node nodeid="node3" name="node3">
        <node nodeid="nodeA" name="nodeA"/>
        <node nodeid="nodeB" name="nodeB"/>
        <edge from="nodeA" to="nodeB"/>
  </node>
  <node nodeid="node4" name="node4"/>
  <node nodeid="node5" name="node5"/>
  <edge from="node1" to="node3"/>
  <edge from="node2" to="node3"/>
  <edge from="node3" to="node4"/>
  <edge from="node3" to="node5"/>
</graph>
```

Workflow Layout

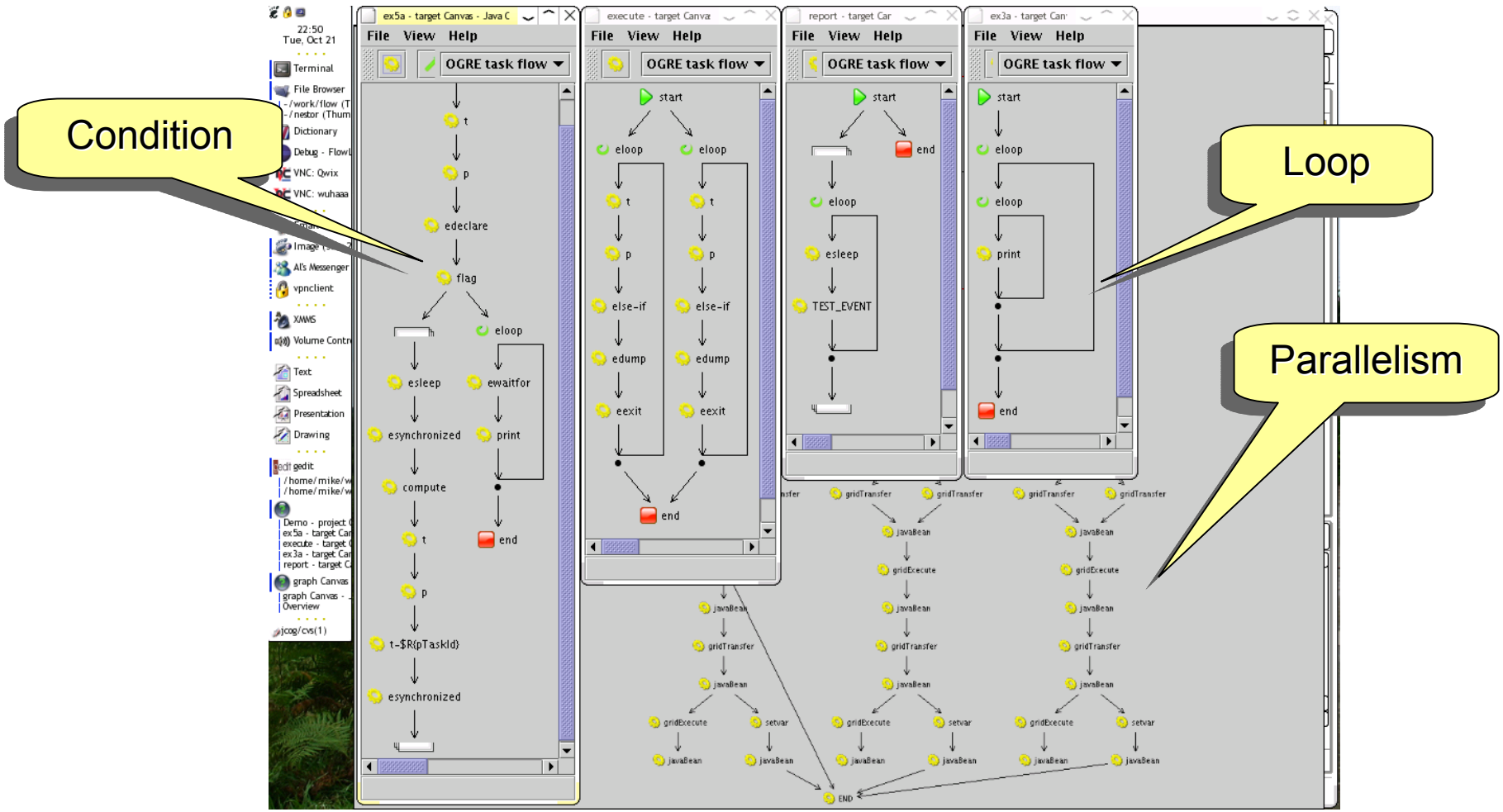


the globus alliance
www.globus.org/cog

Java CoG Kit GridAnt Workflow

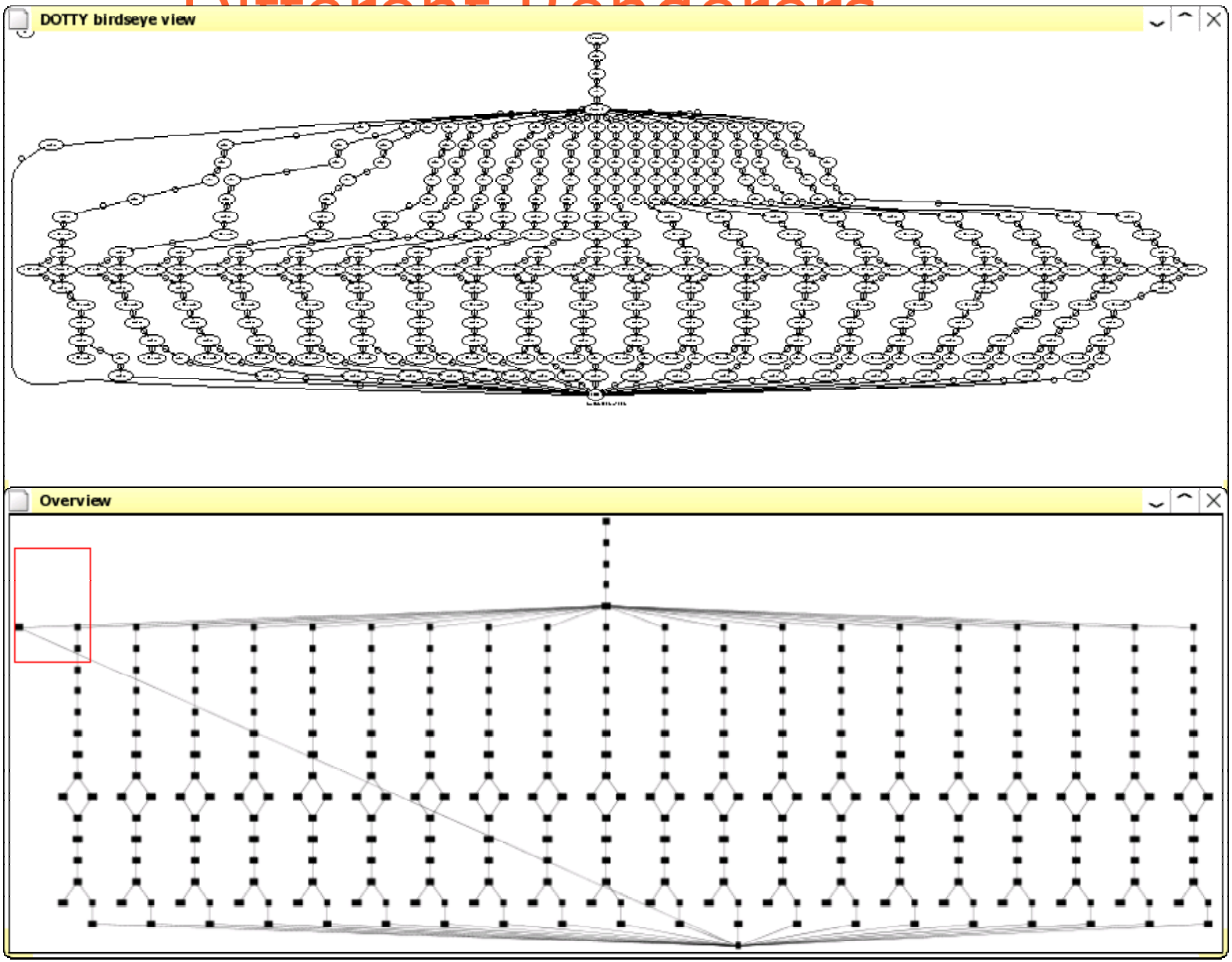


Java CoG Kit Workflow

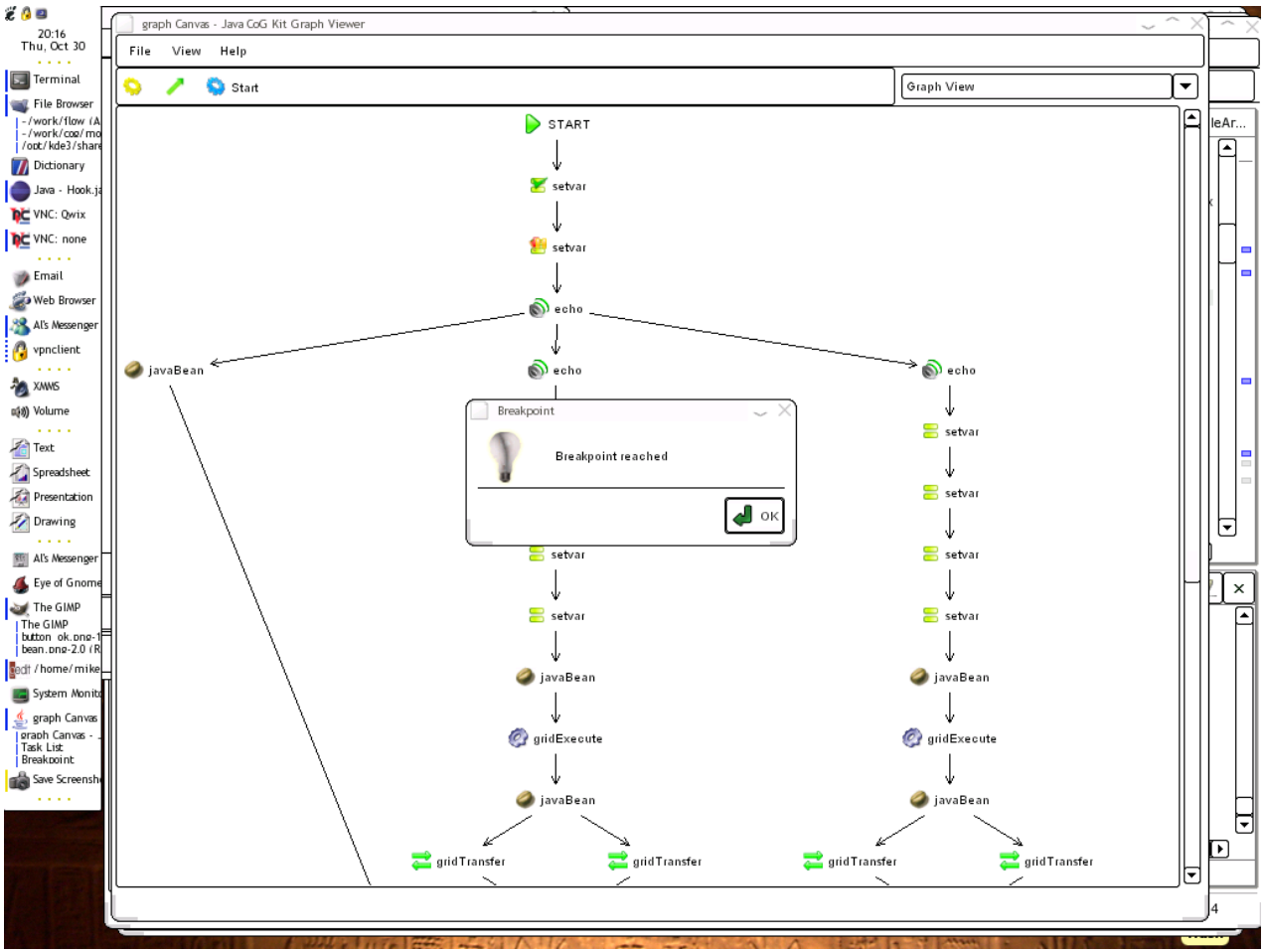


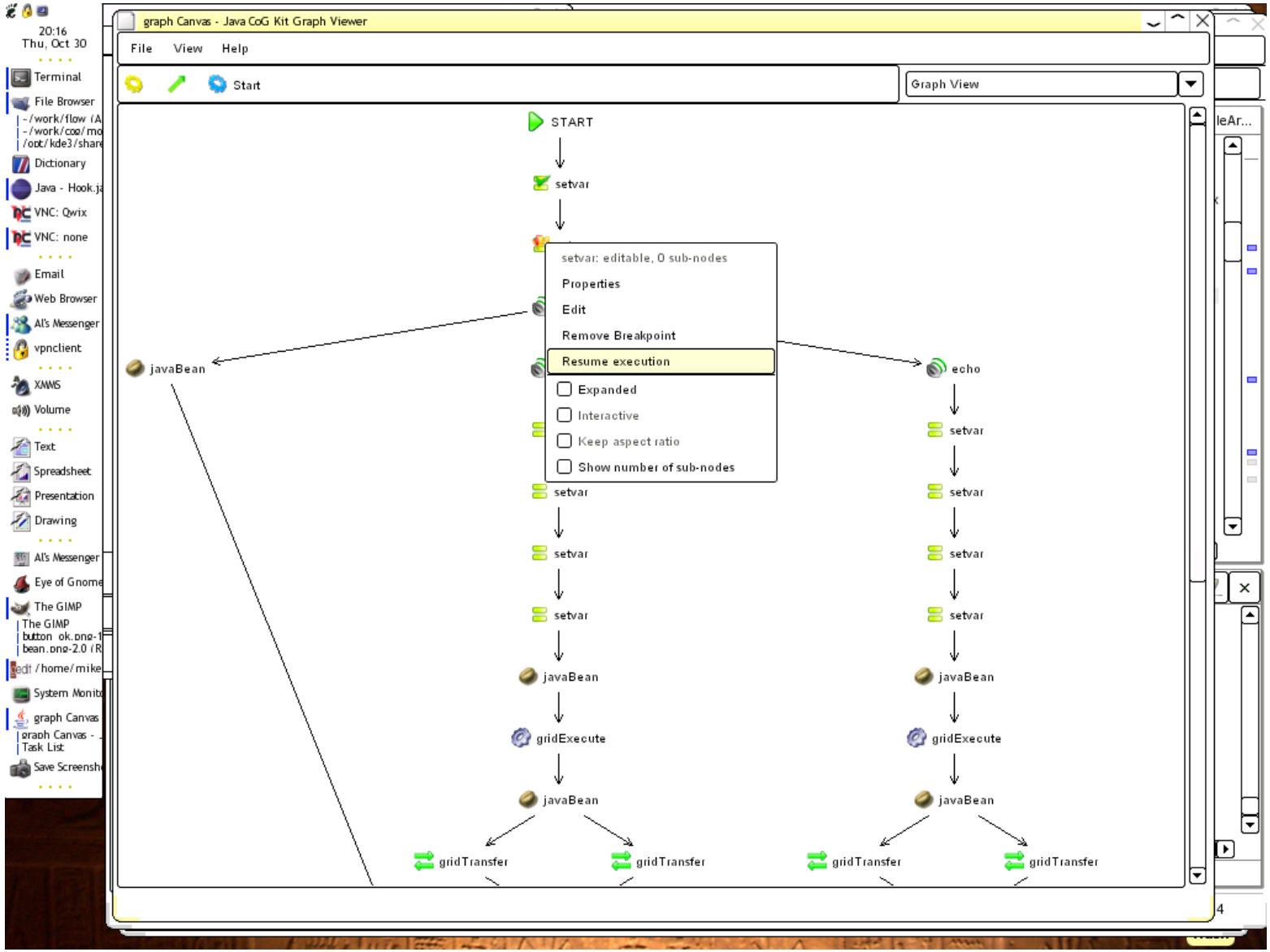
Different Renderers

- Dot

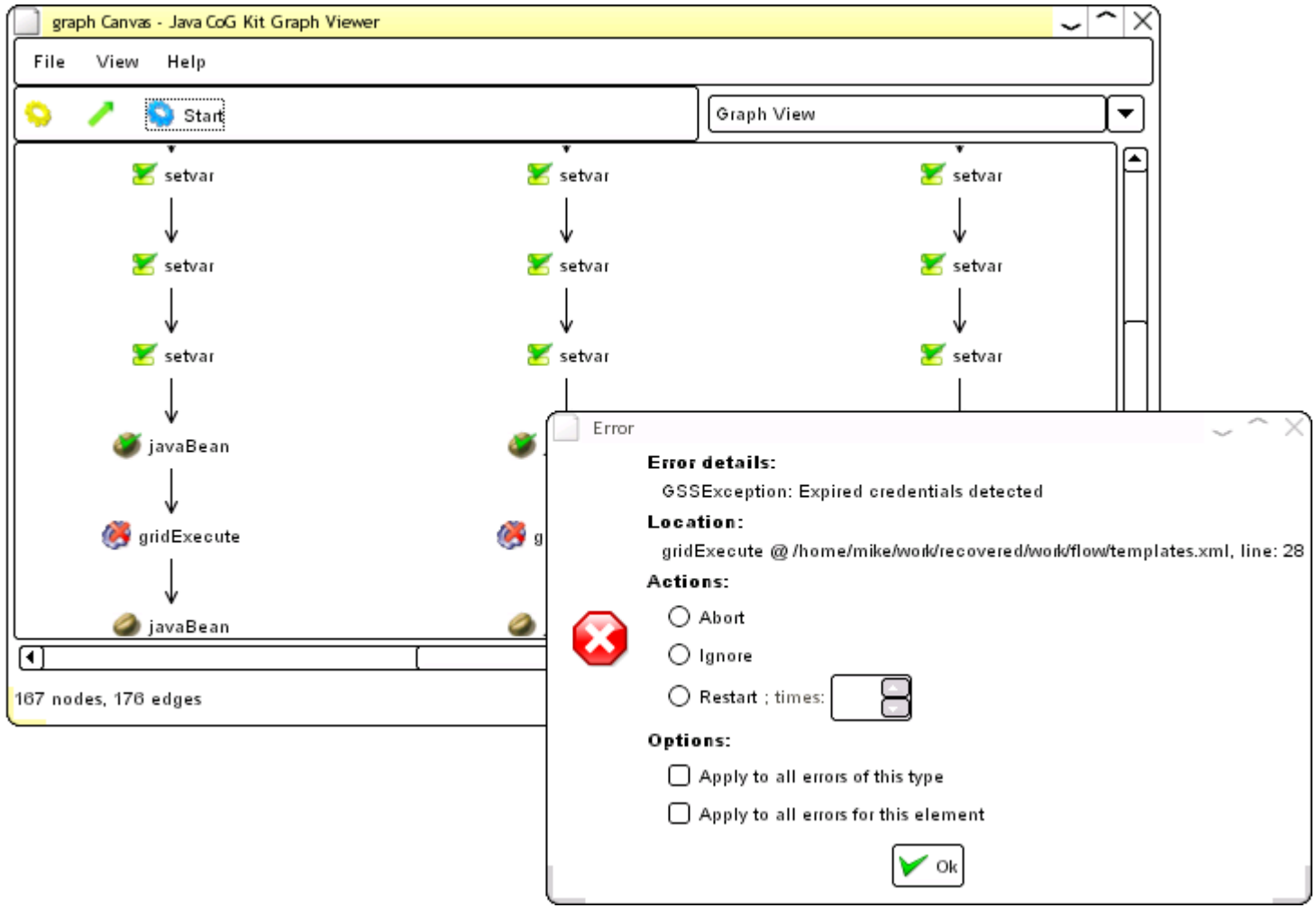


- CoG


graph Canvas - Java CoG Kit Graph Viewer
File
View
Help
Start
Graph View
START
setvar
setvar
echo
javaBean
echo
echo
Breakpoint
Breakpoint reached
OK
setvar
setvar
setvar
setvar
setvar
setvar
javaBean
javaBean
gridExecute
gridExecute
javaBean
javaBean
gridTransfer
gridTransfer
gridTransfer
gridTransfer


graph Canvas - Java CoG Kit Graph Viewer
File
View
Help
Start
Graph View
START
setvar
setvar: editable, 0 sub-nodes
Properties
Edit
Remove Breakpoint
Resume execution
Expanded
Interactive
Keep aspect ratio
Show number of sub-nodes
javaBean
echo
setvar
gridExecute
gridTransfer

Fault tolerance



Integration

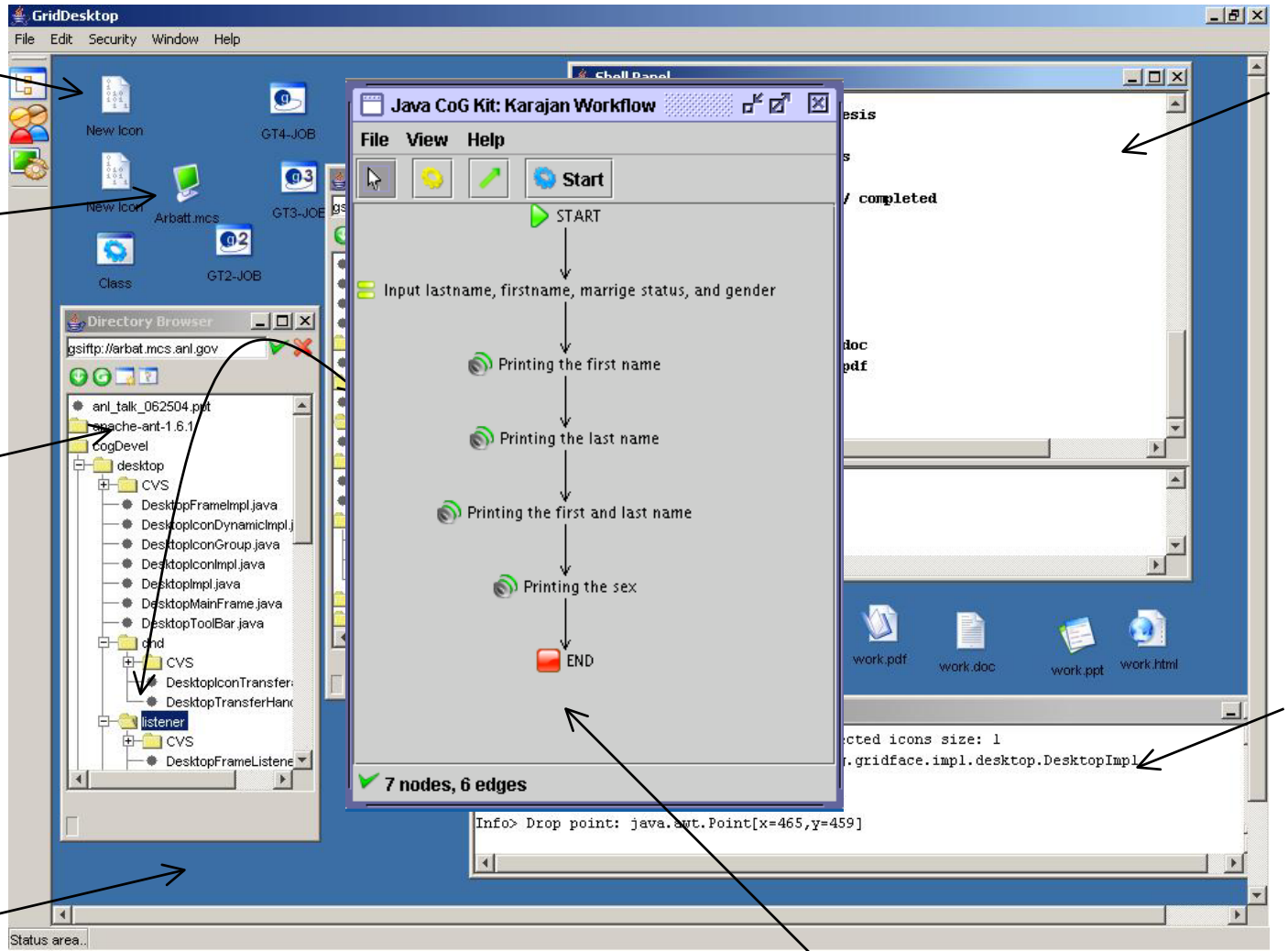
CoG Kit Desktop (in progress)

Job Icons

Machine Icons

File Transfer GUI

Native Icons



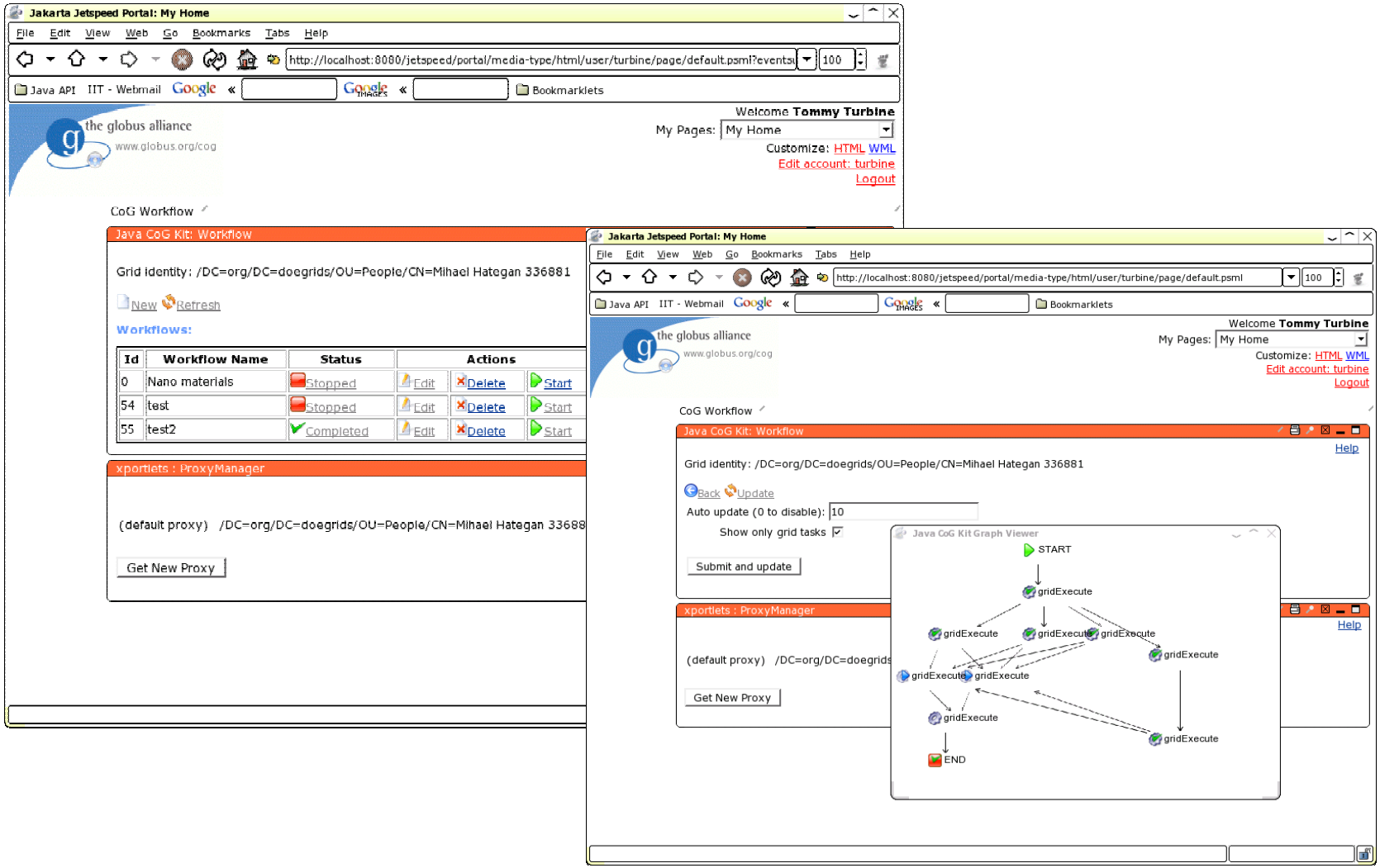
Grid Shell

Grid Log

GridAnt/Karajan

the globus alliance
www.globus.org/cog

Portlets: OGCE.org



Nanomaterials Application

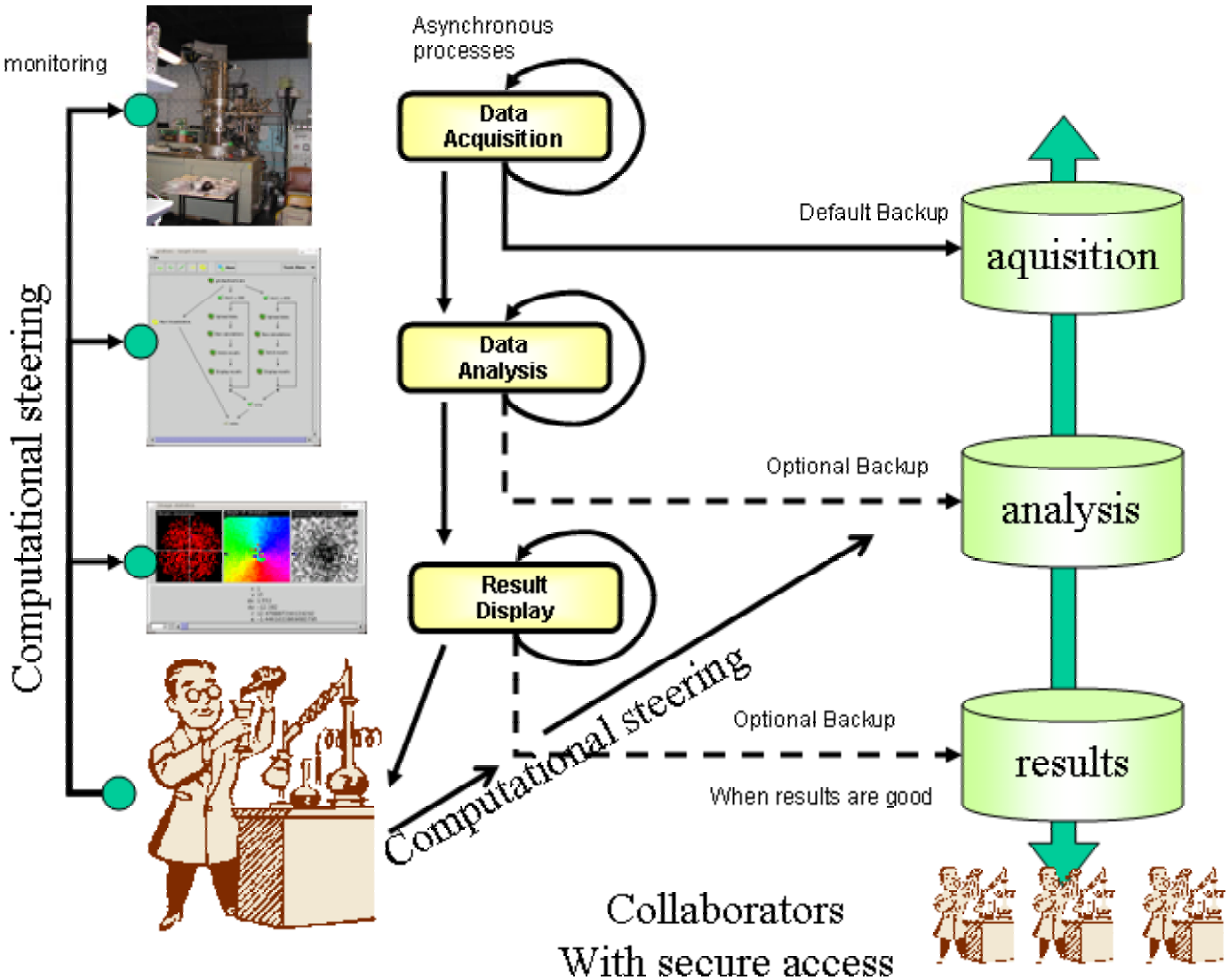
With the Java CoG Kit

F

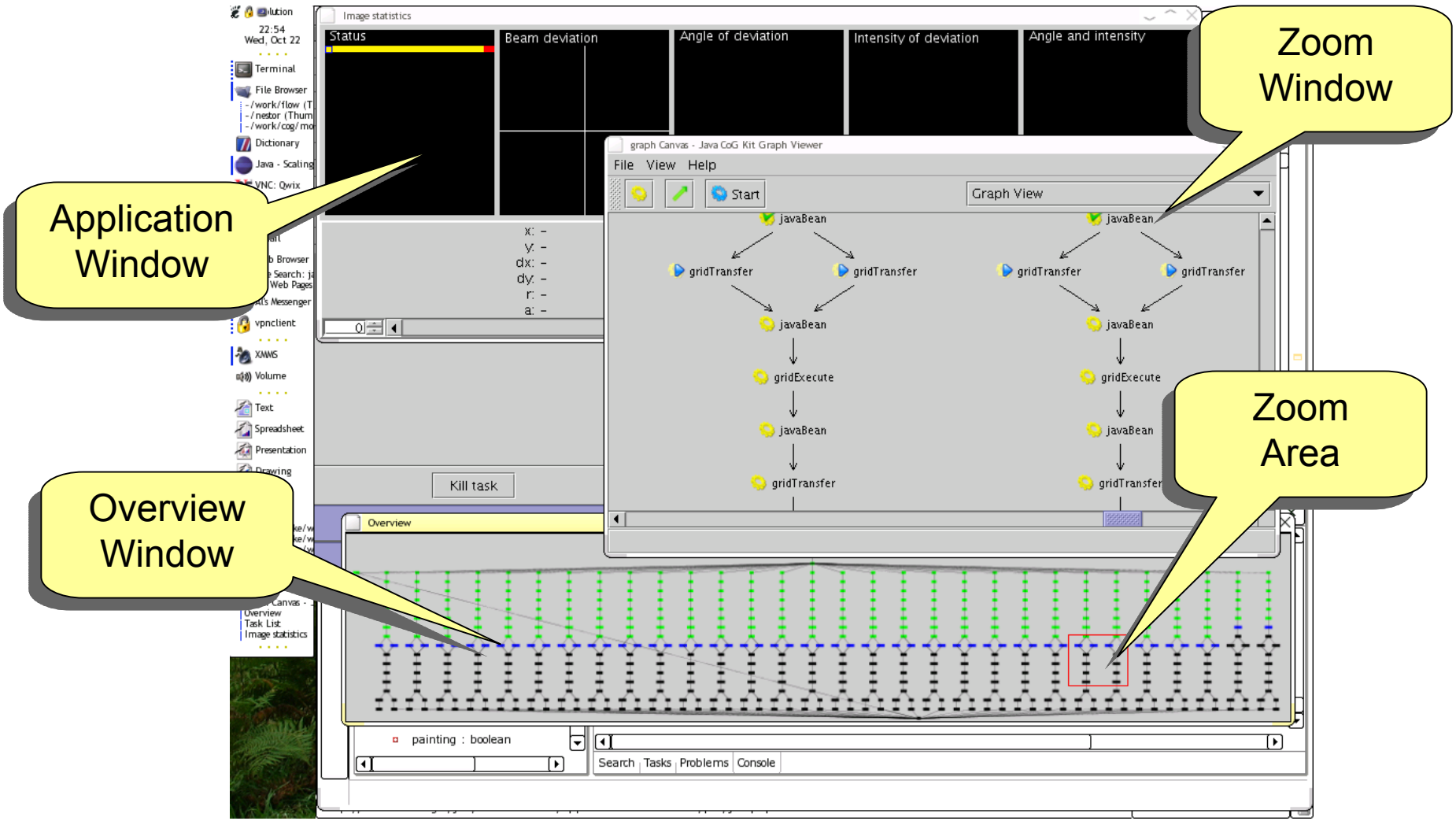
1)

2)

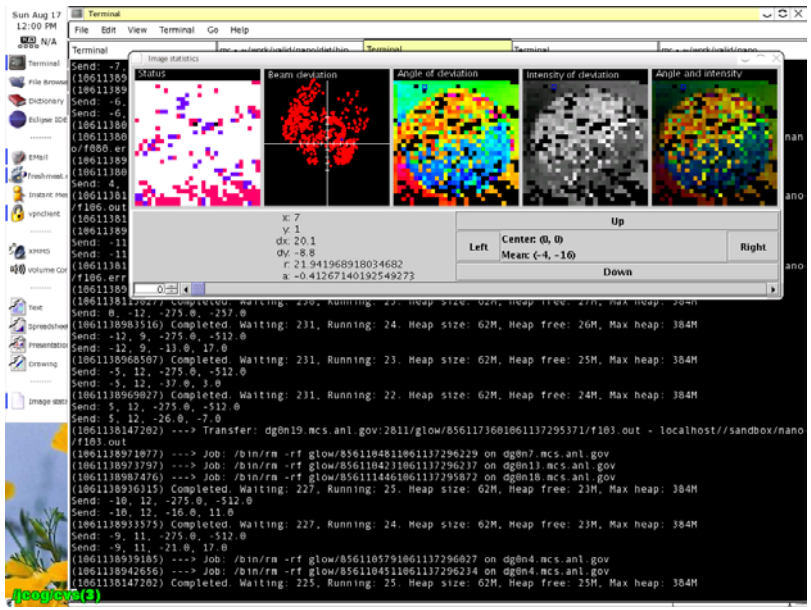
3)


monitoring
Asynchronous processes
Data Acquisition
Default Backup
aquisition
Data Analysis
Optional Backup
analysis
Result Display
Computational steering
Optional Backup
results
When results are good
Computational steering
Collaborators
With secure access

Nanomaterials

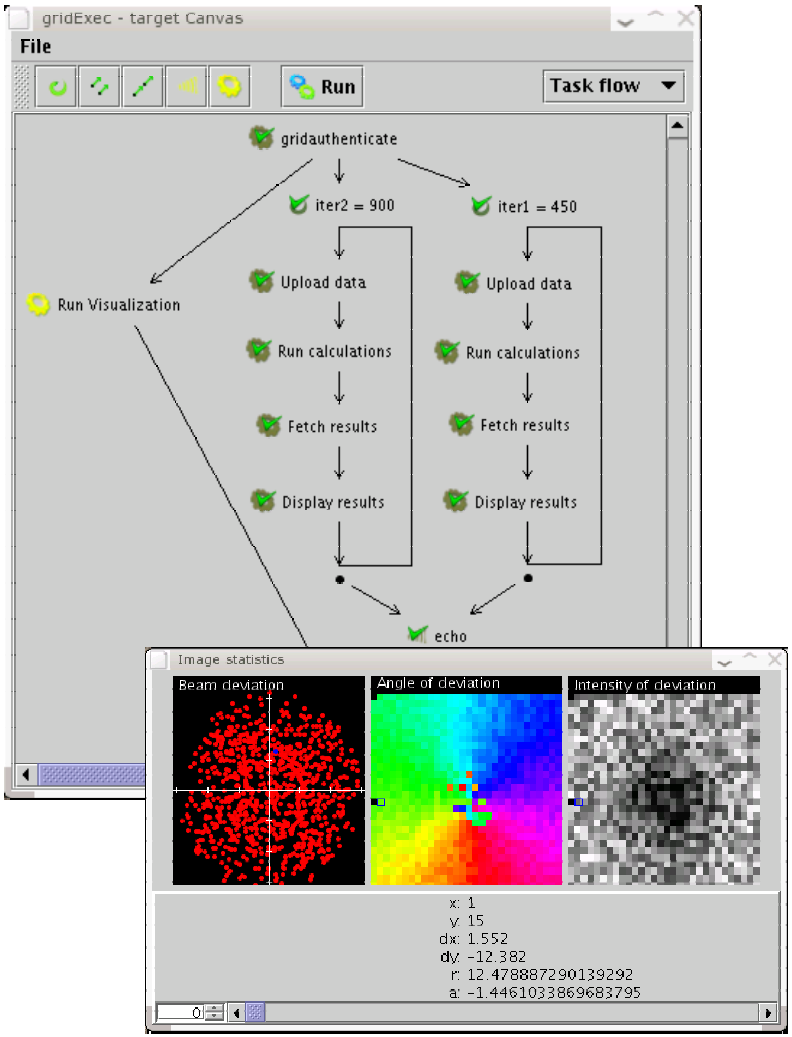


Nanomaterials in Adhoc Fashion



- 900 images taken in novel apparatus
- 6 hours analysis in 45 minutes
- Injection of new filters during runtime
- Step toward Adhoc Grid

Instrument


gridExec - target Canvas
File
Run
Task flow
gridauthenticate
iter2 = 900
iter1 = 450
Run Visualization
Upload data
Upload data
Run calculations
Run calculations
Fetch results
Fetch results
Display results
Display results
echo
Image statistics
Beam deviation
Angle of deviation
Intensity of deviation
x: 1
y: 15
dx: 1.552
dy: -12.382
r: 12.478887290139292
a: -1.4461033869683795

Contributing

- You can contribute
- We have a module concept allowing components to be integrated in the distribution easily

Conclusion

- Workflow is possible in Java CoG Kit
- Allows integration of other commodity technology
- Open source
- Expandable
- Support of current and future Grid middleware
- We can customize!

Conclusion

- Programming with CoG workflows is simple
- We envision multiple programming models in CoG
- We envision multiple backend services
- We can support multiple protocols

- We like to engage the community
- Contributions:
  - CA management, Unicore provider, gsissh
  - These contributions are being integrated.

- We did not talk about the GridShell which is a topic by its own.