GridNexus and JXPL

A Grid Services Workflow System

UNCW Grid Group

Presented by
Jeff Brown, Clayton Ferner, Bill Shipman

Outline

- GridNexus Overview
- Graphical User Interface
- Grid and Web Service Clients
- JXPL
- OGSA-DAI Interactions
- Conclusions

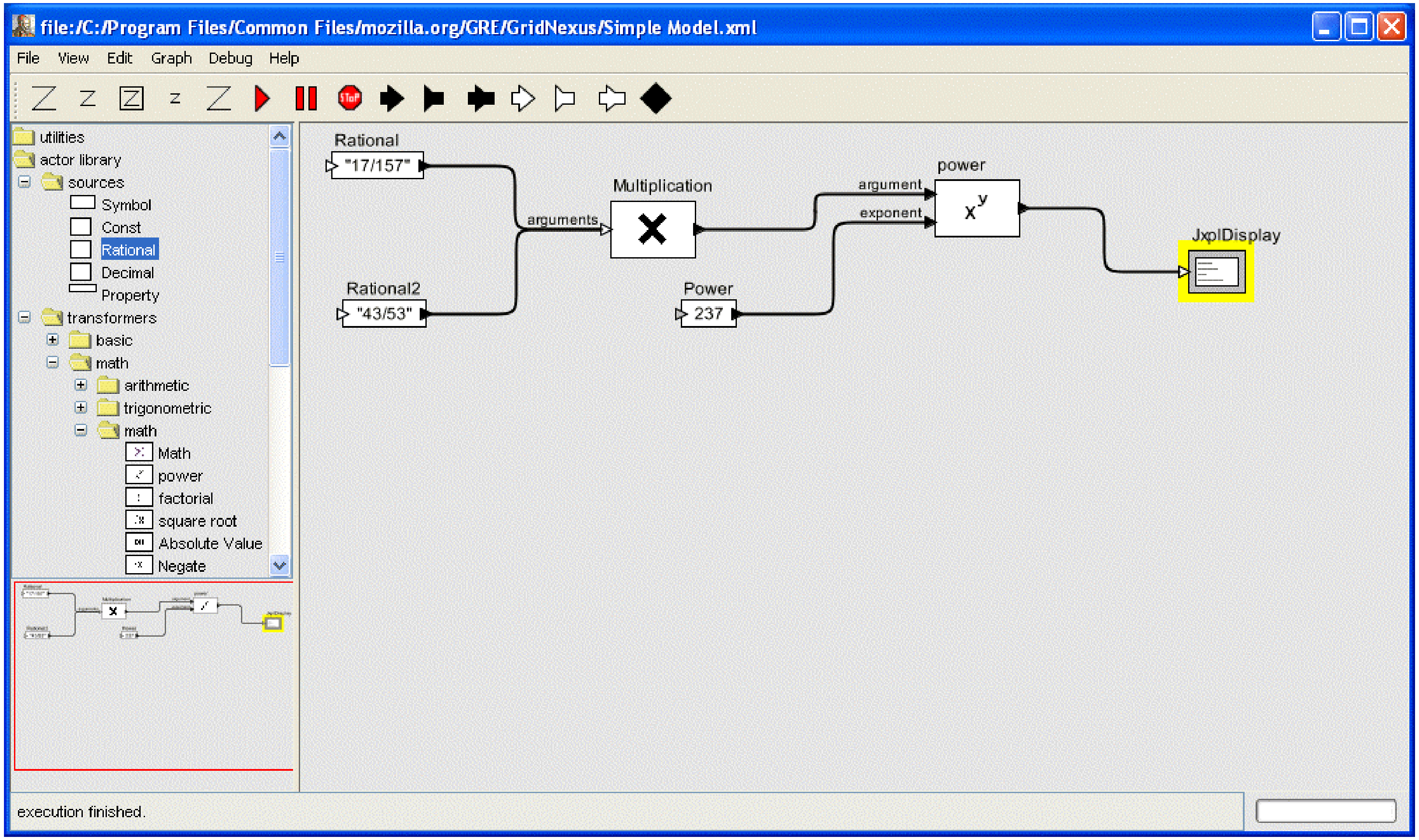
GridNexus

- Workflow construction and execution
- “Drag and Drop” environment
- Generic Grid and Web service clients
- OGSA-DAI support
- Separation of GUI from execution

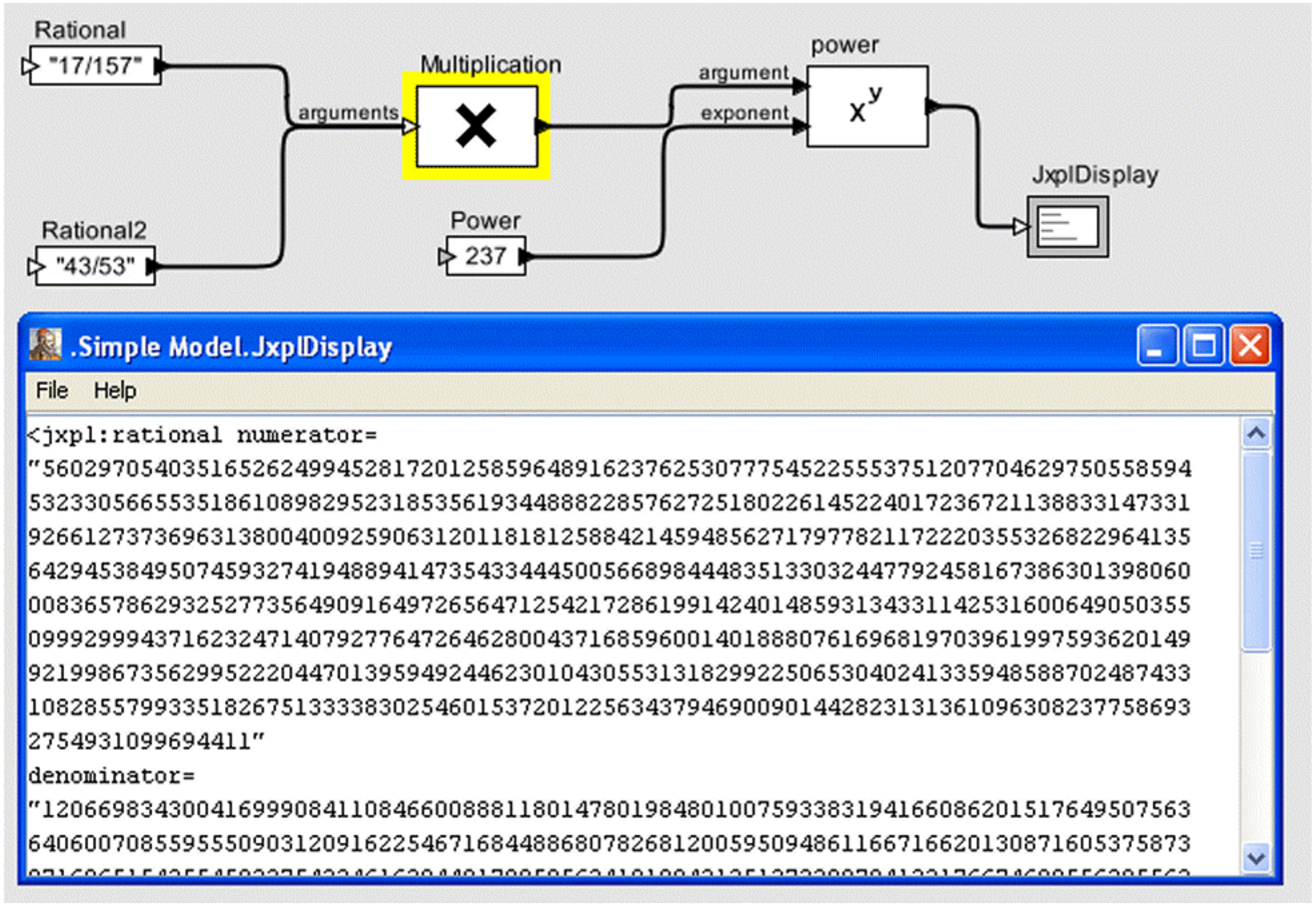
GUI Features

- Based on Ptolemy II
  - http://ptolemy.eecs.berkeley.edu/ptolemyII
- XML configuration makes it easy to offer specialized tools sets
- Composite models let you bundle complex structures inside a single graphical unit

Simple Example

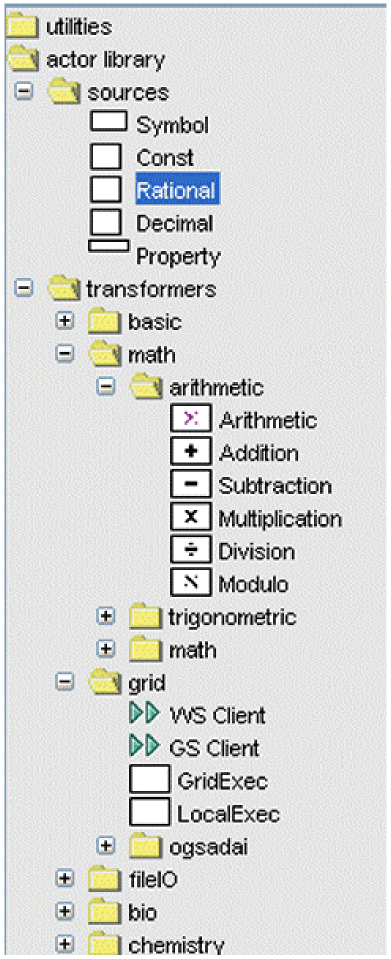


Simple Example Evaluated



```
<jxpl:rational numerator=
"5602970540351652624994528172012585964891623762530777545225553751207704629750558594
5323305665535186108982952318535619344888228576272518022614522401723672113883314 7331
9266127373696313800400925906312011818125884214594856271797782117222035532682296 4135
6429453849507459327419488941473543344450056689844483513303244779245816738630139 8060
0083657862932527735649091649726564712542172861991424014859313433114253160064905 0355
0999299943716232471407927764726462800437168596001401888076169681970396199759362 0149
9219986735629952220447013959492446230104305531318299225065304024133594858870248 7433
1082855799335182675133338302546015372012256343794690090144282313136109630823775 8693
275493109969441l"
denominator=
"1206698343004169990841108466008881180147801984801007593383194166086201517649507563
6406007085595550903120916225467168448868078268120059509486116671662013087160537 5873
```

XML Configuration

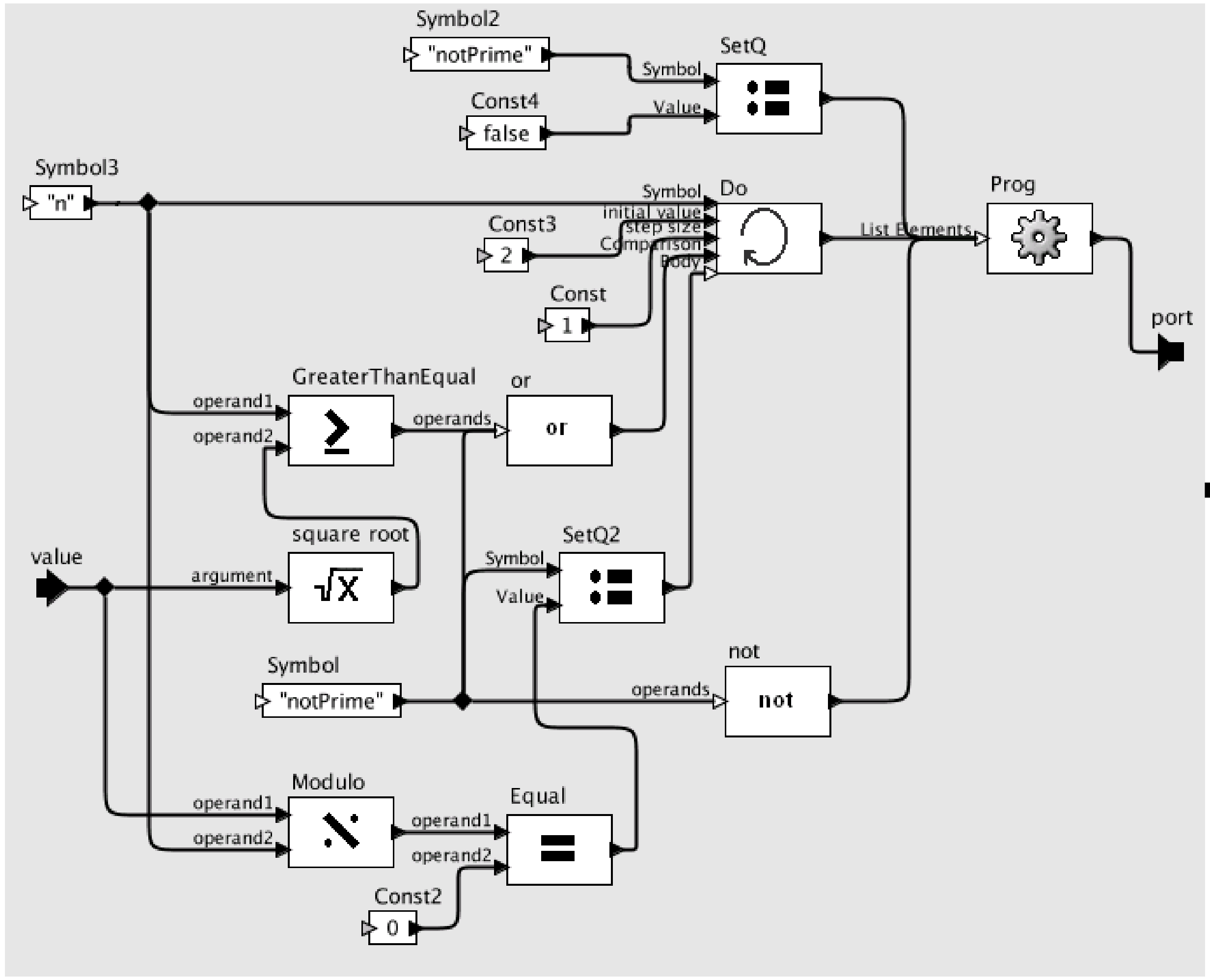


- Hierarchical tool library
- Easy to customize
- Structure determined by XML configuration files

Composites

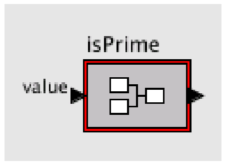
- Workflows can be complicated
- Complex structures can be stored in a Composite object which appears as a single module in other workflows.
- Composites can be stored in the user library

Composites



Complex Object

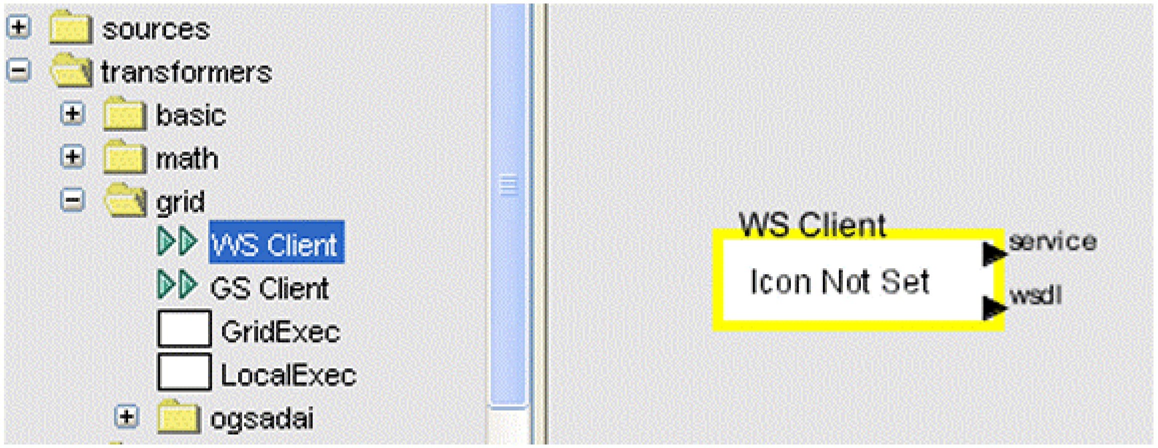
Composite



Generic Service Clients

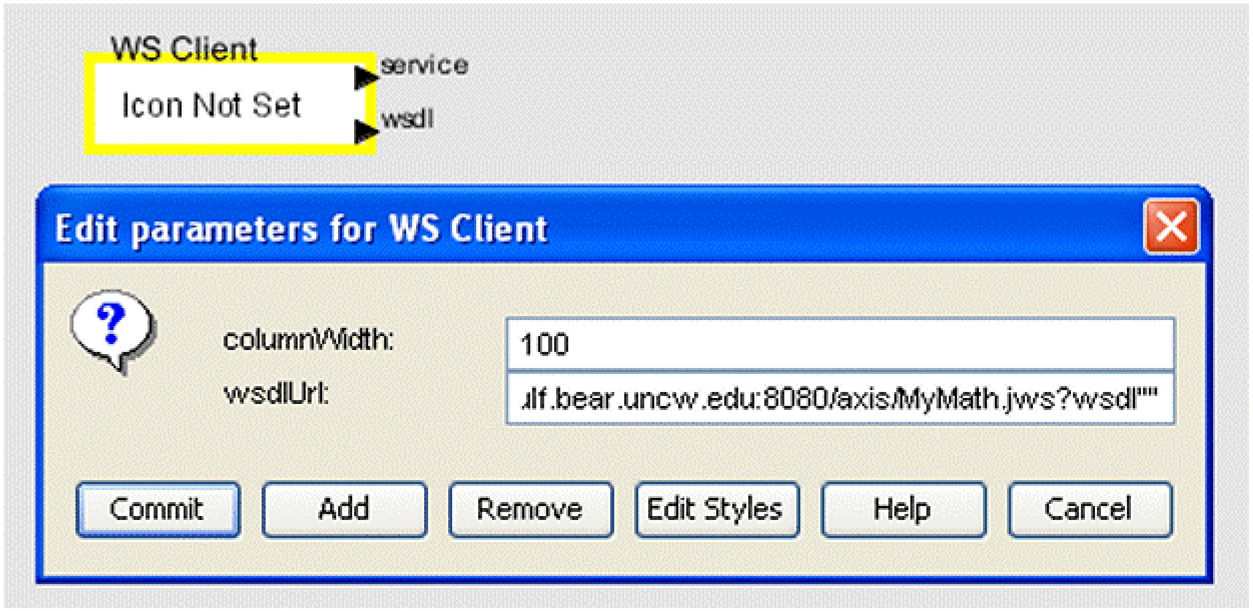
- Web Service
  - Provide URL of WSDL
- Grid Service
  - Provide factory URL and port type class
- Simple input and output types

Dynamic Configuration of Web Service Client



- Drag WSClient into work area
- No input port

Dynamic Configuration of Web Service Client

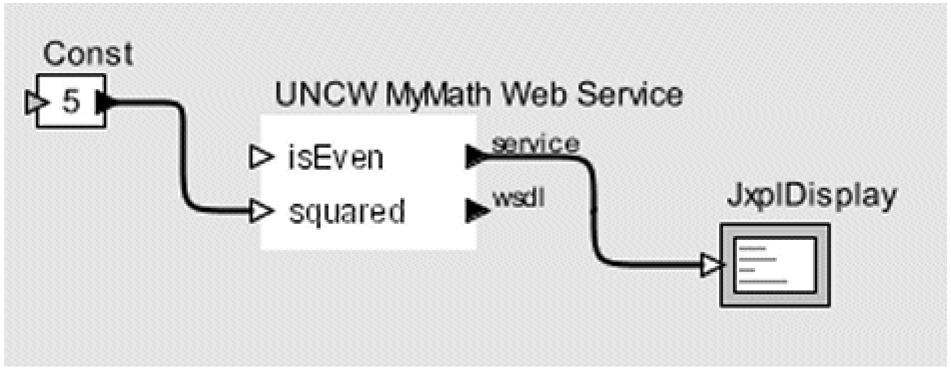


columnWidth: 100

wsdlUrl: .lf.bear.uncw.edu:8080/axis/MyMath.jws?wsdl""

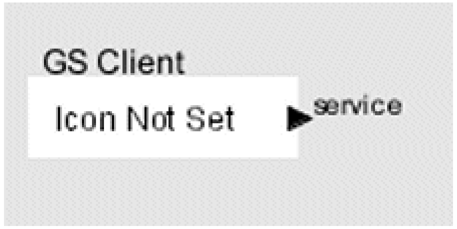
Enter WSDL URL

Dynamic Configuration of Web Service Client



- •WSDL retrieved and parsed
- •Input ports and labels created

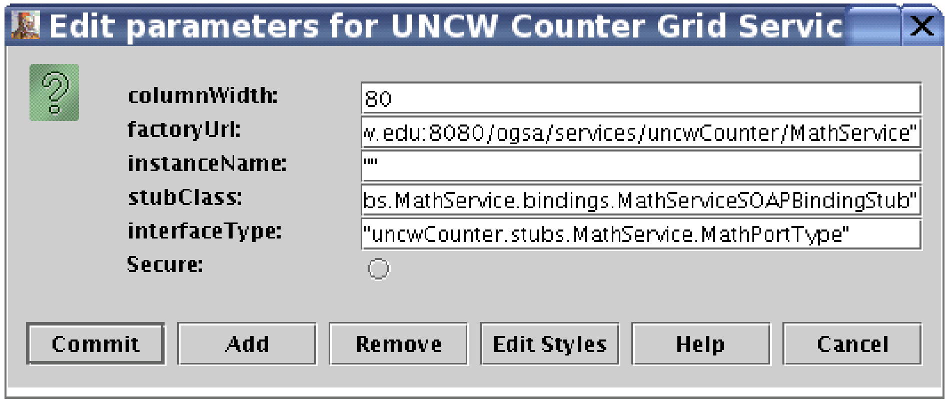
Dynamic Configuration of Grid Service Client



- Drag GSClient into work area
- No input port

Dynamic Configuration of Grid Service Client

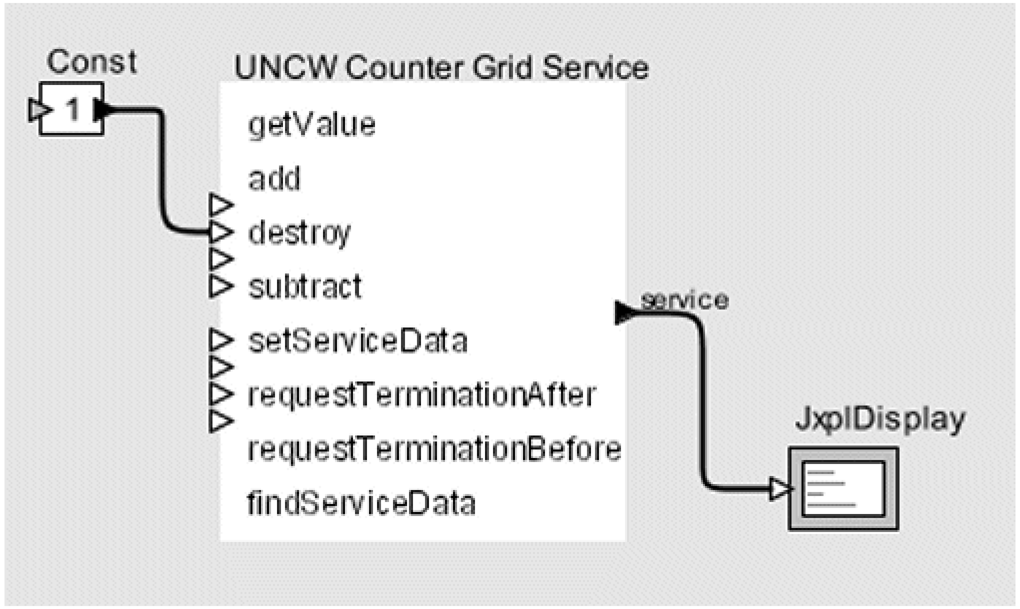
- Enter:
  - factory URL
  - Service instance
  - Stub class name
  - Interface type



Edit parameters for UNCW Counter Grid Servic

columnWidth:	80
factoryUrl:	w.edu:8080/ogsa/services/uncwCounter/MathService"
instanceName:	""
stubClass:	bs.MathService.bindings.MathServiceSOAPBindingStub"
interfaceType:	"uncwCounter.stubs.MathService.MathPortType"
Secure:	○

Dynamic Configuration of Grid Service Client



- Reflection API on interface
- Input ports and labels created

Workflow Execution

- GUI modules do not execute tasks
- GUI produces JXPL script
- Script is sent to JXPL processor
- Processor
  - Run on client machine
  - Remote OGSA service
  - RMI service
  - HTTP service

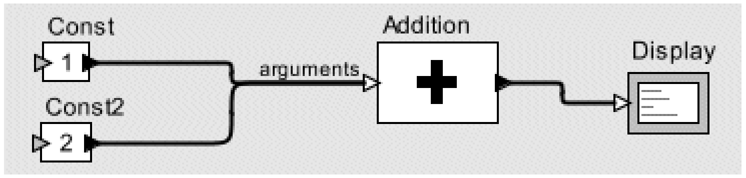
JXPL Overview

- Scripts written in XML
- Similar to LISP
- User defined functions
- Local variables
- Supports recursion

XML for Scripting

- Parsers are already written
- Many applications can create or edit scripts
- XSLT and XPath now available in J2SE and allow easy manipulation
- XML Schemas for validation

JXPL Example



```
<jxpl:list>
    <jxpl:primitive name="Arithmetic">
      <jxpl:property name="operation" value="add"/>
    </jxpl:primitive>
    <jxpl:integer value="1"/>
    <jxpl:integer value="2"/>
</jxpl:list>
```

OGSA-DAI

- Data Access and Integration
- OGSA service: Grid Data Service (GDS)
- GDS
  - Querying data sources
  - Transforming data
  - Deliver data to other services or programs
- Integration

  GDS output -> GDS input

GDS Interaction

- A GDS is controlled by an XML script
- OGSA-DAI package has many helper classes to simplify the process of creating these scripts
- Managing the interaction of GDS is complicated

GDS Example

- Two services: *source* and *sink*
- Output of *source* will be input of *sink*
- Managing interaction:
  - Create *sink* instance and store handle
  - Run *sink* in separate thread to wait for input
  - Create script for *source*, incorporating handle of *sink*
  - Create *source* instance and run

Managing GDS

- Languages such as Java can be used to manage GDS interaction
  - Store variables
  - Spawn threads
  - Dynamic XML creation
- Interactive workflow environment is better suited to scripting language
- JXPL was designed for GDS compatibility

Remote JXPL Access

- JXPL scripts can be executed by remote processors accessed in server ways
  - OGSA service
  - Remote Method Invocation (RMI)
  - HTTP
- GridNexus can be configured to use either a local or remote JXPL processor

Remote JXPL Data

- A JXPL Symbol is the name of a variable
- The XML namespace of the symbol determines the location of the processor that holds the value
- Example

```
<… xmlns:foo=“ogsa://…/ogsa/serivces/Jxpl”
<jxpl:symbol name=“foo:myData”/>
```

Remote JXPL Execution

- A JXPL script can contain pieces of code that are executed on remote processors
- Location of the remote processor is determined by XML namespace

Conclusions

- GridNexus: intuitive workflow creation
- JXPL
  - Flexible XML scripting language
  - Remote data and execution
- Together they provide workflow system that separates GUI from execution