Service Oriented Computing EnviRonment (SORCER)

R. Kolonay
S. Burton
M. Sobolewski

Training Goals

- Create your own SORCER Workspace
- Become familiar with the Eclipse Development Environment
- Have SORCER Environment Running
- Publishing Providers & Running Requestors
- Creating your own Providers & Publishing
- Creating your own Requestors & executing
- Creating your own Models & Querying

Training Agenda

- Thursday
  - SORCER Overview
  - Creating your own workspace
  - Some SORCER Terminology
  - Running Hello World Examples
  - Creating Providers & Publishing
  - Creating Requestors (Job, Task, Context) & Executing calling providers

Training Agenda

- Friday
  - Variables, Filters, Evaluators
  - Variable fidelity of Variables & their Sensitivities
  - Creating Models & Publishing & Querying
  - Using the ModelClient Class from the command line & Matlab
  - Doing Optimization using Matlab and SORCER Modles

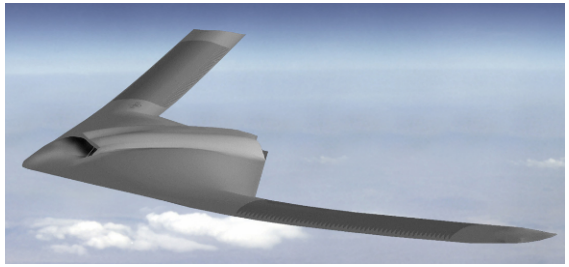
Training Agenda

- SORCER Overview
- Creating your own workspace
- Running Hello World Examples
- Creating Providers & Publishing
- Creating Requestors (Job, Task, Context) & Executing calling providers

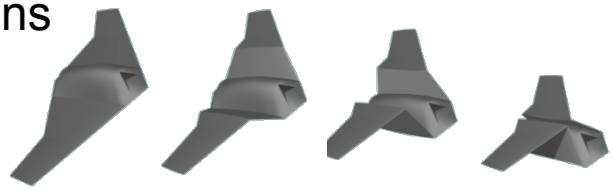
AFRL Air Vehicles Directorate
Revolutionary Aerospace Vehicles

AFRL Air Vehicles Directorate Multidisciplinary Sciences & Technology Center

Extreme Environments

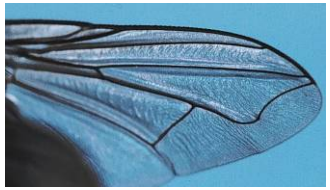


Large Deformations



Time Varying Shape

MAVs



Higher, Farther, Faster to
Lower, Closer, Slower

How to Capture the Physics Driving the Design Pre-Milestone A

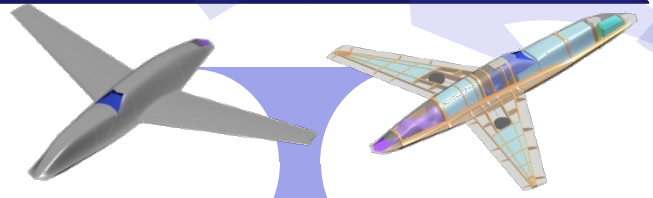
10000s configurations

Conceptual Design Studies

Technology Suite

Prototype Representation

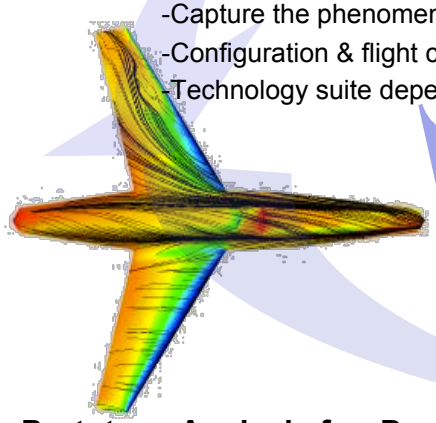
10s configuration



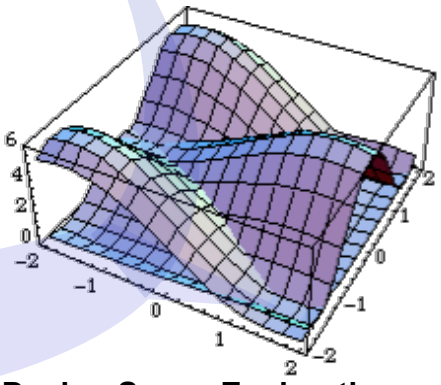
Dialable Fidelity

-Capture the phenomena driving the design
-Configuration & flight condition dependent
-Technology suite dependent

Prototype Experimental Validation



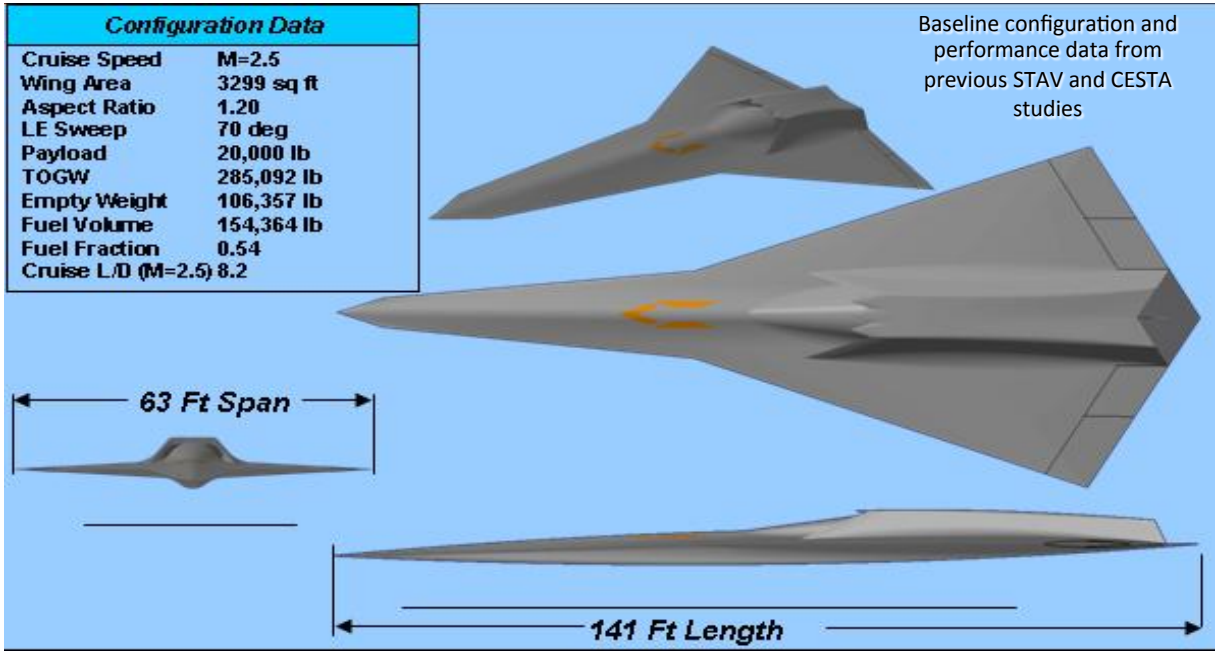
Prototype Analysis for Design



Prototype Design Space Exploration

Get more (and Better) Information … and get it Earlier

Typical Target Application LRS Configuration



SOO Requirements:

Range = 4000 nmi

Payload = 5-10 % weight fraction

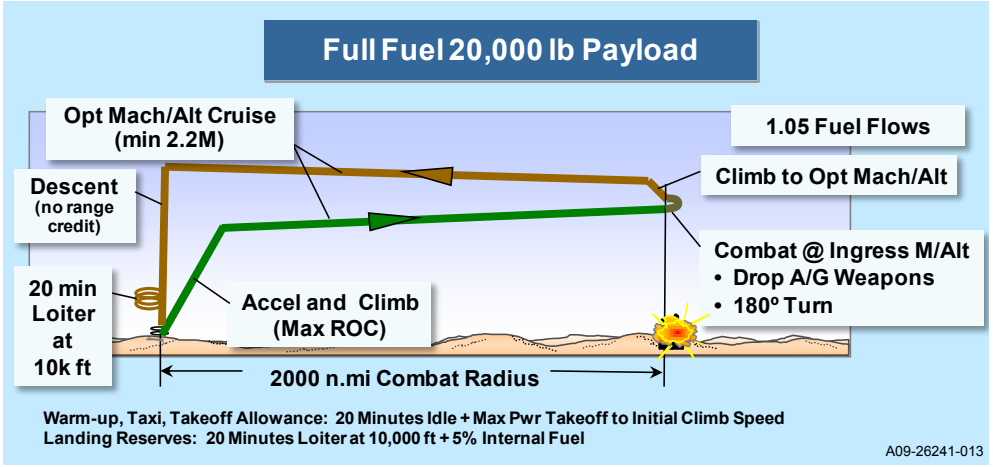
Maneuver Loads at Cruise – 2.5 g

Cruise speed: Mach 2.0

Cruise L/D: 8.5-9

Level 1 Flying Qualities

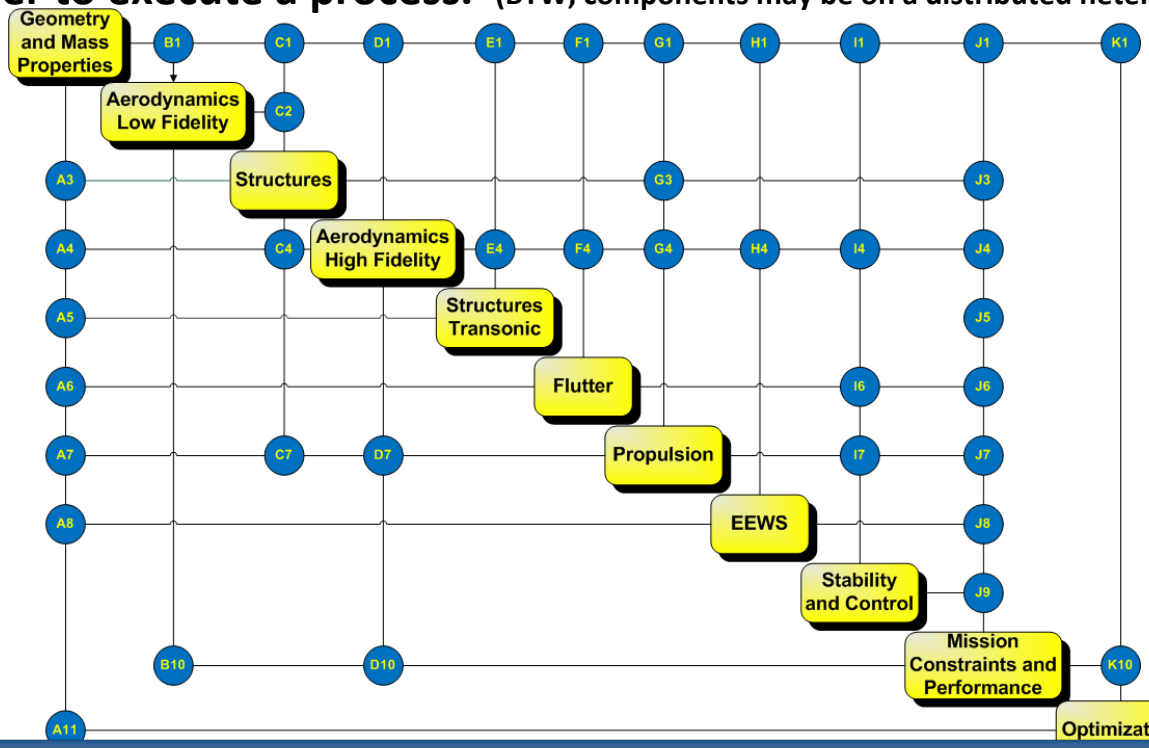
AF-specified mission will be "flown" through mission analysis to determine required fuel weight and TOGW



System Level MDMFMSAOwUQ

"Best in Class Approach (BCA)"

Components are one or more engineering computational applications that need to be "glued" together to execute a process. (BTW, components may be on a distributed heterogeneous network)



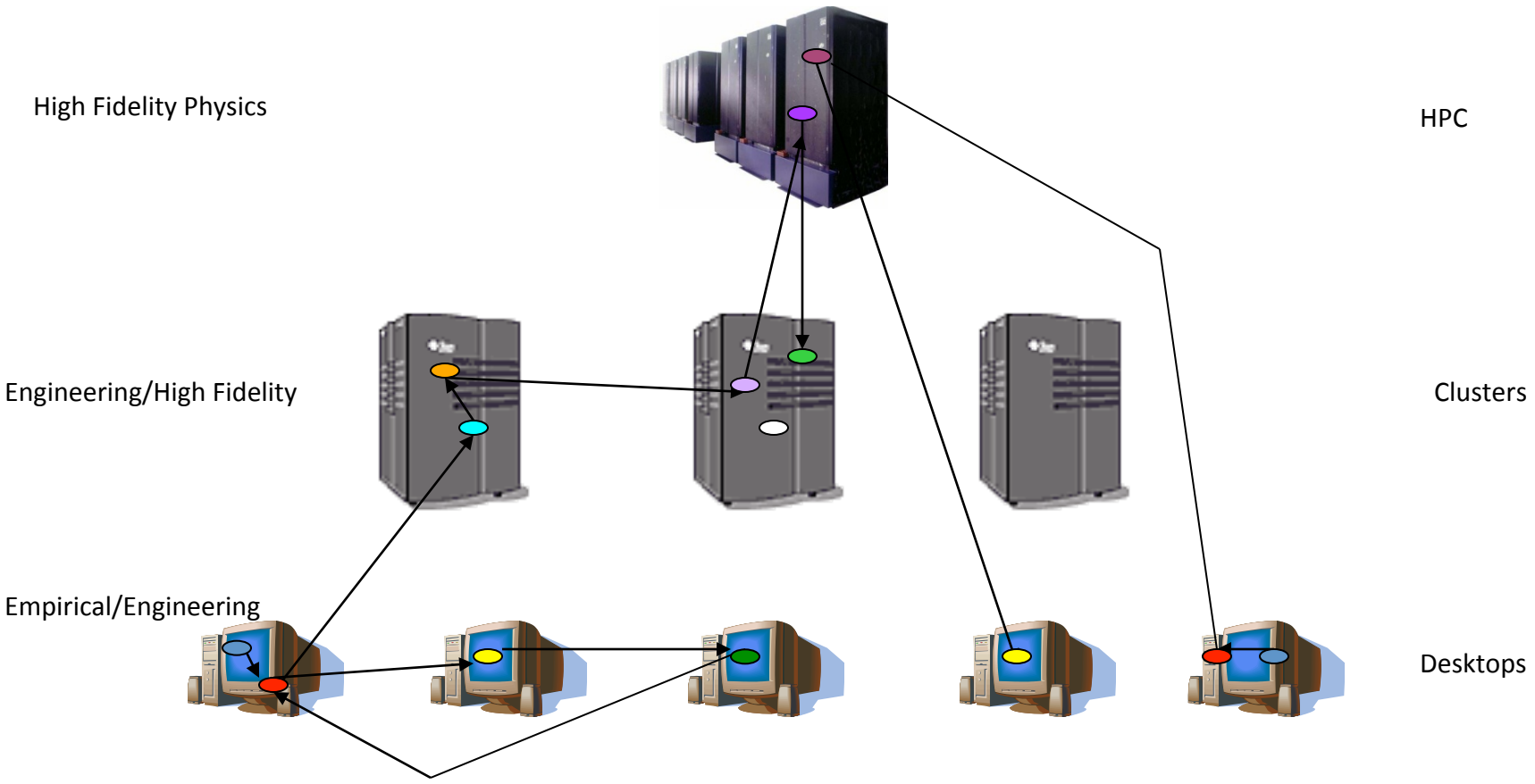
Need a Computational Environment to Achieve BCA MDMFMSAOwUQ

Need to Maximize Reuse

High Level Requirements for System Level MDMFMSAOwUQ

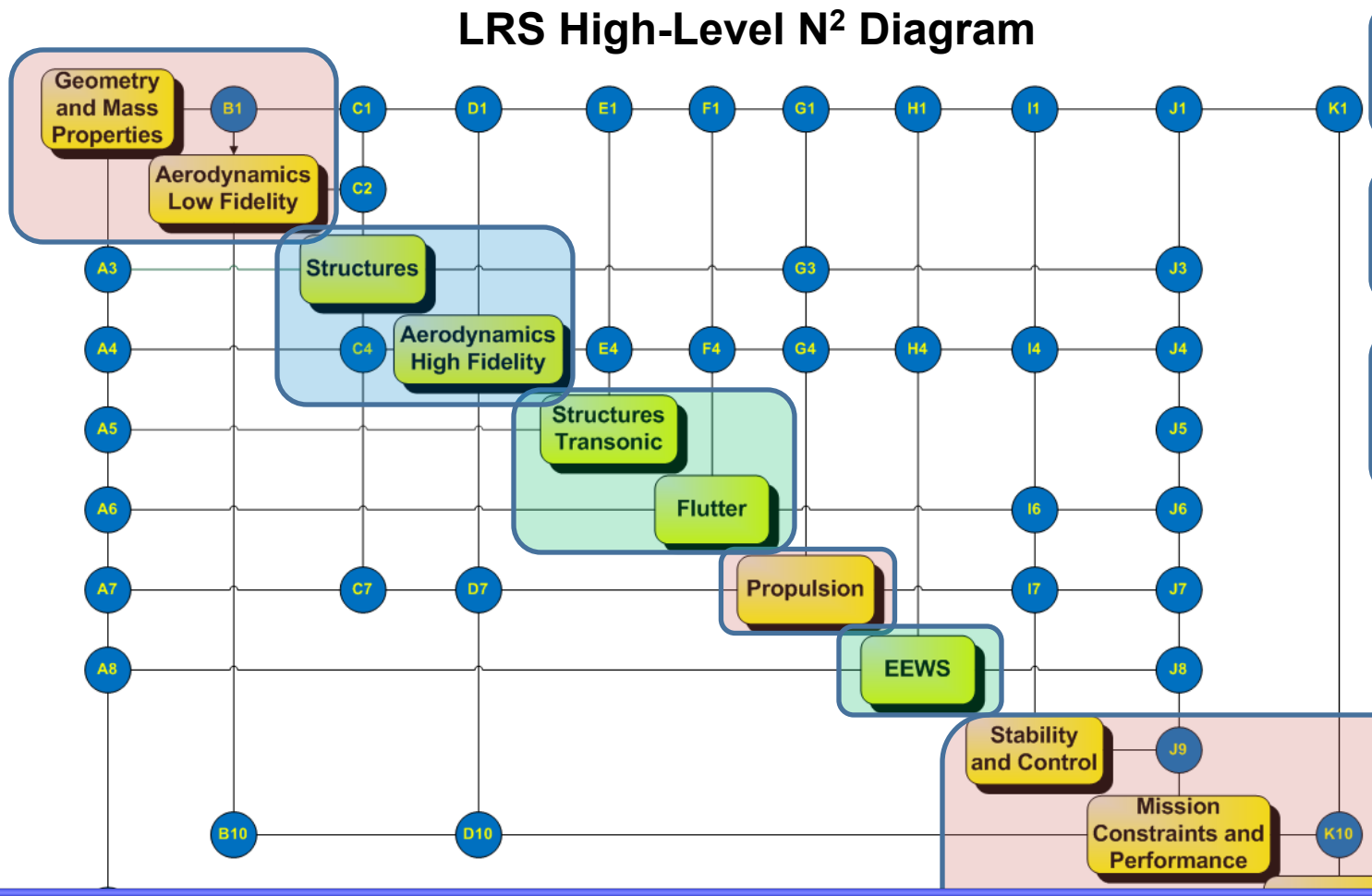
- # of components/applications/Services – 100's to 1000's
- Run times of services – secs to many days
- Data
  - kilobytes to terabytes
  - ascii, binary, databases
- Distributed (across organizational boundaries) heterogeneous computing environment
  - Hand held devices to HPC resources
  - Seamless access to data and services
  - Process representation with secure communications

Seamless Access to All Methods, Models, Data, and Computing Resources



Large Scale Distributed MDA/MDO

SORCER



VT (Blacksburg)

AFRL/RBSD (WPAFB)

WSU (Dayton)

Multi-fidelity Multidisciplinary Computations Require Real Time Seamless Access to Applications, Data & Compute Resources by ALL

SORCER is a Service-Object-Oriented Architecture

A service-oriented product development environment

- Provides an open flexible design environment which allows universal availability and incorporation of existing data, tools/ methods, processes and hardware as services.
- Provides a common way to model your analysis and design process in conjunction with your product data.

A network-based distributed framework

- Supports collaboration among geographically distributed engineering and business partners.

SORCER Philosophy – Accessibility, Flexibility & Reusability

SORCER is Research Code!

SORCER is not for the Faint at Heart

- *MDMFMSAOwUQ* *(MultiDisciplinary,MultiFidelity,MultiScale, Analysis/Optimizaiton)*
- Eclipse
- Ant
- Java
- JNA, JNI, SWIG
- Network Computing
- Object Oriented Concepts
- SORCER Concepts

Service Oriented Computing EnviRonment - SORCER

A service-oriented product development environment

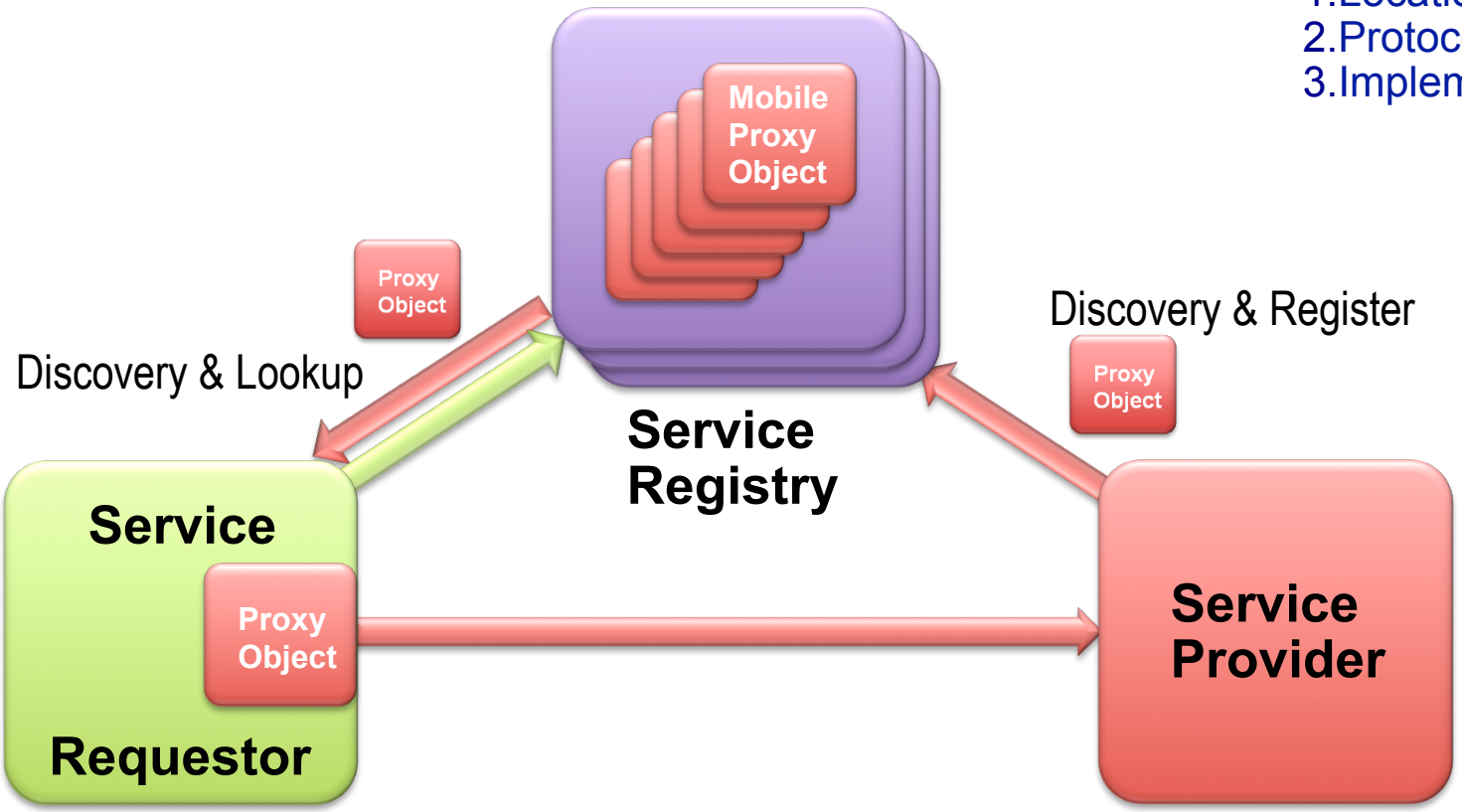
- **SORCER** is a programming and **computing environment** that enables one to perform large scale system level engineering analysis and design space exploration that may be in a distributed heterogeneous computing environment.
- SORCER federates a series of ***Service Providers*** (which may be distributed) in real time and **orchestrates the communication** between the ***Service Providers*** based on a ***Control Strategy*** **algorithm** defined in an ***Exertion****(process definition)* to perform a multi-disciplinary analysis and or design space exploration.

Service Oriented Computing EnviRonment - SORCER

A network-based distributed framework

Service Oriented

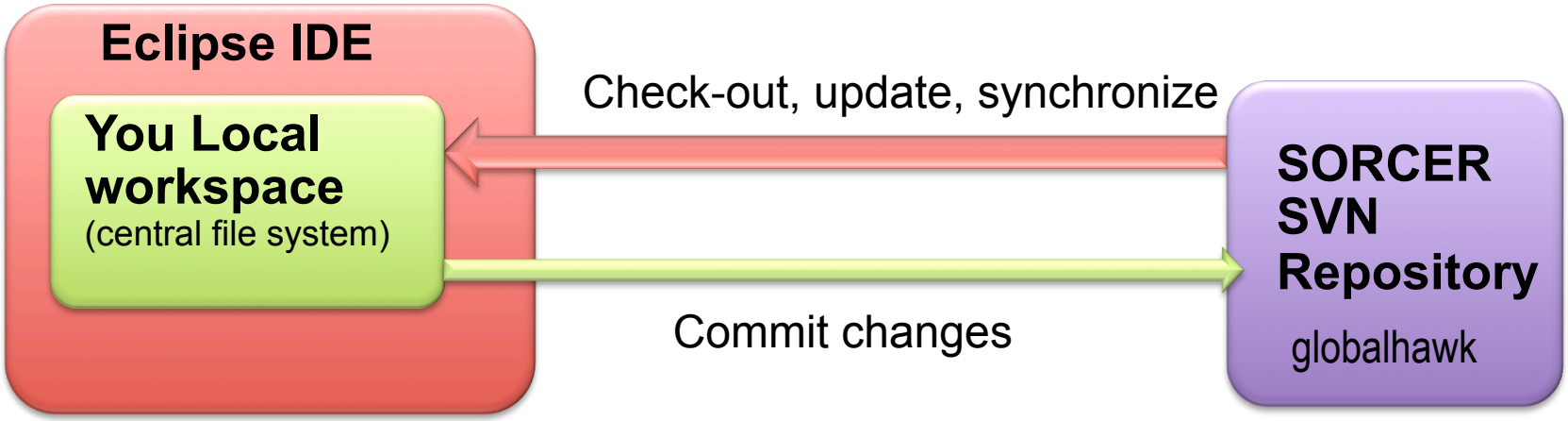
Neutrality:
1.Location
2.Protocol
3.Implementation



Training Agenda

- SORCER Overview
- Creating your own workspace
- Running Hello World Examples
- Creating Providers & Publishing
- Creating Requestors (Job, Task, Context) & Executing calling providers

SORCER SVN Repository



- Can do what ever you like in your workspace
- Once you want others to get your changes – “Commit” to repository

- To Access to svn commands from Eclipse navigator panel – right click- Team

Creating a workspace from an svn repository

- Notes/CreateSORCERSandbox.doc
- Make the Eclipse ***workspace*** directory <workspace dir>
- e.g.: mkdir /home/<username>/workspace
- Set JAVA_HOME to appropriate JDK directory (if necessary)
- Set ANT_HOME to appropriate Ant directory (if necessary)
- Set IGRID_HOME to <workspace dir>/iGrid-XX (Required)
- Run eclipse from your installation Eclipse Directory
- Check *Use this as the default and do not ask again*
- Select *Go to the workbench* at the top-right corner of the Eclipse window
- Click on the menu item *Project* and unselect *Build Automatically*
- Select *File => New => Project…*
- Select *SVN=> Checkout Projects from SVN*
- Click *Next*,
  - Check in *Create a new repository location* then click *Next*
  - Location URL:
- **svn+ssh://<username>@*needVTRepositoryIPaddress*/home/svn/sorcer**
- then click *Next*
  - *Select Folder: iGrid-XX* then click *Finalize*
  - Enter your password when asked, then Check in *Save Information*
  - In the *Check Out As* dialog make *Project Name* for your project in the workspace as needed, for example *iGrid-XX*.

Building/Compiling SORCER

1.Open (click on >) iGrid-XX in Eclipse Navigator (select the Eclipse's Perspective in the top-right corner as Resource)

2. Read and follow directions iGrid-XX/README and iGrid-XX/modules/sorcer/README and setup your additional environment variables for example for a class server (called webster) accordingly.

3.Double check if your Java compiler is at least Java 5 compatible.

a. Right-click on the iGrid-XX project in the Navigator view and select Properties.

b. In the Properties window select Java Compiler.

c. Check in Enable project specific settings and then click on Installed JREs at the bottom of Java Compiler view.

d. If required JDK or JRE is not shown in the Installed JREs list, click Add and append it to the list accordingly.

e. Check in the JDK/JRE for Java 5+ in the Installed JREs list and click OK to close the Installed JREs dialog.

4. Right click on iGrid-XX/iGrid-buid.xml

5. Select Run/1 Ant Build

If you have followed all the directions correctly, you should have a successfully compiled SORCER sources and built SORCER libraries (jar files in iGrid-XX/lib).

Configuring the SORCER Environment (sorcer.env)

Must have sorcer.env file in the folder iGrid-10/configs/

If it is not there copy a template fromiGrid-10/templates/sorcer.env to iGrid-10/configs/sorcer.env and edit it.

```
# SORCER environment properties

# The OS environment variable IGRID_HOME must be set

# Groups to register/lookup
provider.groups=sorcer.TEST
provider.space.group=sorcer.TEST
#provider.space.name=JavaSpace
#provider.space.name=Blitz JavaSpace
#provider.worker.transactional=true
#worker.transactiona.lease.time=50000

# Service discovery/lookup
# comma separated URLs
#provider.lookup.locators=jini://localhost
#   multicast and unicast discovery
provider.lookup.accessor=sorcer.util.ProviderAccessor
#   unicast or mixed discovery
#provider.lookup.accessor=sorcer.util.ProviderLocator
#   multicast  only
#provider.lookup.accessor=sorcer.util.ProviderLookup
#   unicast or mixed discovery with QoS capabilities
#provider.lookup.accessor=sorcer.util.QosProviderAccessor

# Code server configuration
provider.webster.interface=${localhost}
provider.webster.port=8000

# Data/file repository configuration
# Scratch directory format:
# ${data.root.dir}/${provider.data.dir}/${provider.scratch.dir}
# ${data.root.dir}/${requestor.data.dir}/${provider.scratch.dir}
# You can and normally do overwrite these properties in your provider or requestor properties
data.root.dir=${iGrid.home}/data
provider.data.dir=provider
requestor.data.dir=requestor
provider.scratch.dir=scratch
data.server.interface=${provider.webster.interface}
data.server.port=${provider.webster.port}
```

Configuring the SORCER Environment (sorcer.logging)

If logging information is desired (to the console and to .log files), a sorcer.logging file must be in the iGrid-10/configs folder. If it is not present copy the sorrcer.logging template found in the iGrid-10/templates folder to iGrid-10/configs/sorcer.logging and edit as necessary

```
# SORCER Logging Configuration File
# Default configuration for handlers associated with the root logger.
# To use this configuration file, specify the parameter
# java.util.logging.config.file on the command line, e.g.,
# java -Djava.util.logging.config.file=sorcer.logging

# Handlers installed for the root logger
handlers=java.util.logging.ConsoleHandler,
java.util.logging.FileHandler

# Level for the root logger - is used by any logger
# that does not have its level set
.level=INFO

# Initialization class - the public default constructor
# of this class is called by the Logging framework
#config=sorcer.core.util.Log

# Configure ConsoleHandler
java.util.logging.ConsoleHandler.level=ALL
java.util.logging.ConsoleHandler.formatter=java.util.logging.
SimpleFormatter

# Configure Plain Text FileHandler
#handlers=java.util.logging.FileHandler
java.util.logging.FileHandler.level=ALL
java.util.logging.FileHandler.formatter=java.util.logging.SimpleFormatt
er
java.util.logging.FileHandler.pattern=../logs/sorcer%g.log
java.util.logging.FileHandler.limit=10000000
java.util.logging.FileHandler.count=3

# Configure XML FileHandler (use it with Log Viewer)
#handlers=java.util.logging.FileHandler
#java.util.logging.FileHandler.level = ALL
#java.util.logging.FileHandler.encoding = UTF-8
# replace in the following line the text after the equal sign with your
log file destination
#java.util.logging.FileHandler.pattern = ../logs/sorcer%g.log
#java.util.logging.FileHandler.limit = 0
#java.util.logging.FileHandler.count = 0
#java.util.logging.FileHandler.formatter =
java.util.logging.XMLFormatter
 ....
```

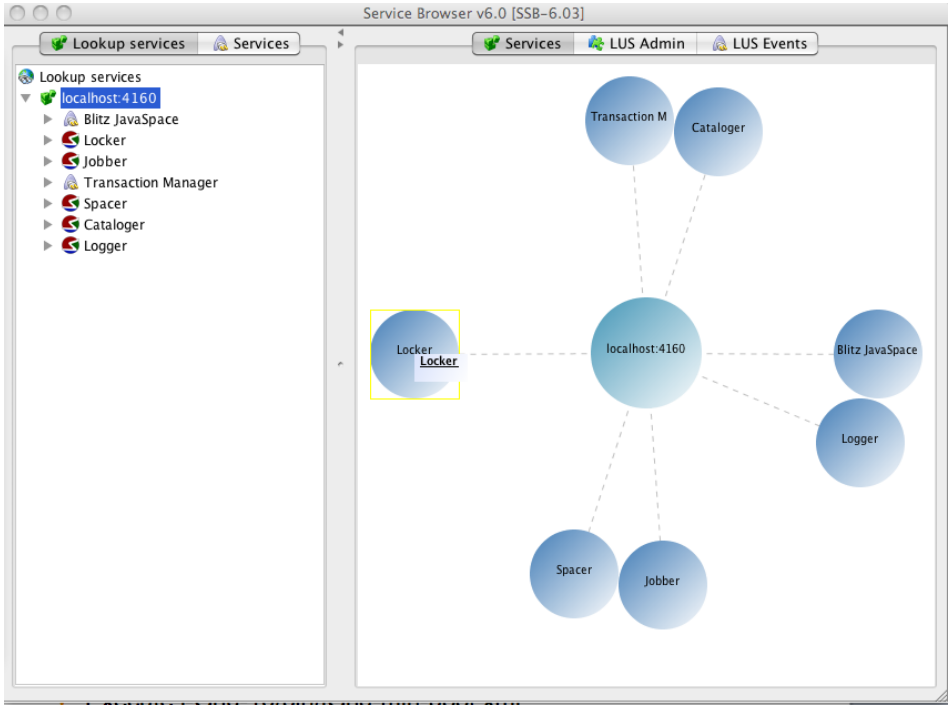
Configuring the SORCER Environment (sorcer browser configuration – ssb.config)

iGrid-10/browser/config/ssb.config

```
sorcer.sbb.browser.SorcerServiceBrowser {

// If 'discoveryManager' is set in the SSB config file then both discovery lookup management
// and discovery group management are used by SSB. If "initialLookupLocators' are set
// then only unicast discovery is used, otherwise multicast lookup discovery is used for
// the specified 'initialLookupGroups". However, in the latter
// additional runtime lookup locators can be added using 'Tools/Unicast discovery...'.

    initialLookupGroups = new String[] { "sorcer.RMK" };

    //initialLookupLocators = new LookupLocator[] { new LookupLocator("127.0.0.1", 4185) };

    //discoveryManager = new LookupDiscoveryManager(DiscoveryGroupManagement.NO_GROUPS, null, null);

    //listenerExporter = new BasicJeriExporter(TcpServerEndpoint.getInstance(4162), new BasicILFactory());

    //websterStartPort = 4176;
    //websterEndPort = 4180;

    //allowDestroy = true;

}
```

Starting the SORCER Environment

- Start your environment webster (if not running)
  - Execute iGrid-10/bin/webster/bin/webster-run.xml
- Start the i-Grid basic services (if not running)
  - Execute i-Grid-10/bin/iGrid-min-boot.xml
- Start the i-Grid service browser (if not running)
  - Execute i-Grid-10/bin/browser/bin/service-browser-run.xml
  - Should see the following

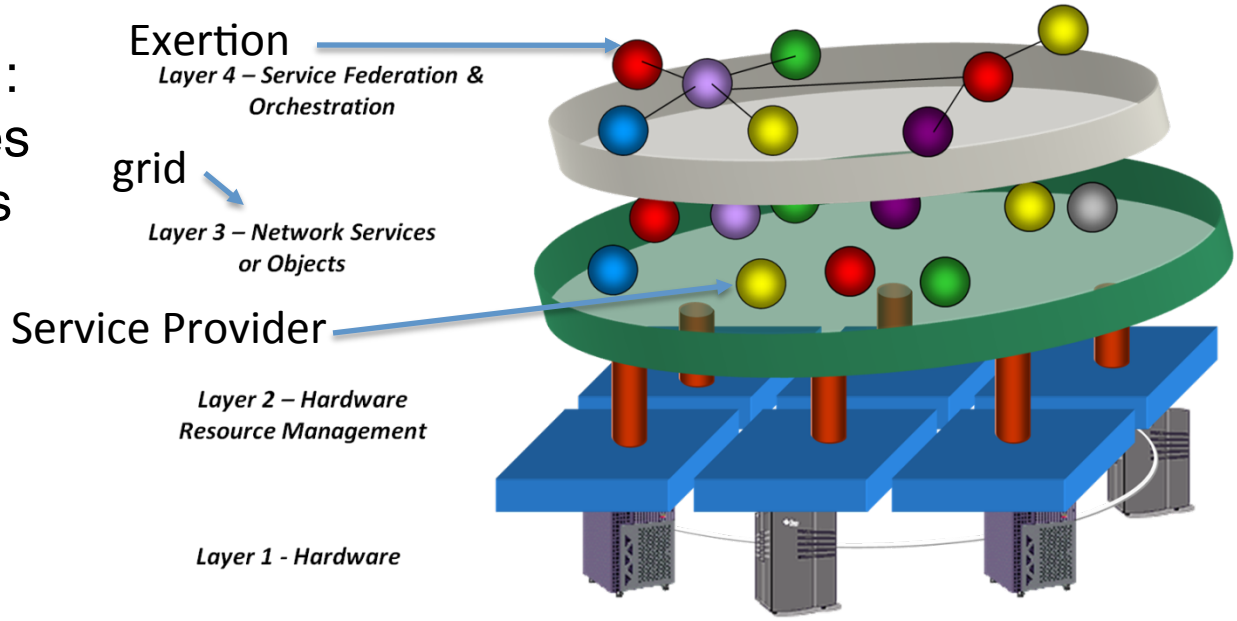


Training Agenda

- SORCER Overview
- Creating your own workspace
- Some SORCER Terminology
- Running Hello World Examples
- Creating Providers & Publishing
- Creating Requestors (Job, Task, Context) & Executing calling providers

SORCER Terminology

- ***Service Provider*** : a remote object accepting ***Exertions*** from service requestors and performs calculations. Can provide one or more services.
- The ***grid***: a collection of service providers on the network.
- ***Exertion*** *(think process representation or workflow):* defines collaborations - service-oriented programs. An object that represents a process by specifying the relationship between services and the information passed between them.
- ***Service Requestor*** : an object that creates ***Exertions*** & submits them to the grid.



Service Providers

- Wrap existing applications with java to expose one or more functionalities(services) of the application to the SORCER Environment

```
Public class xxxProvider extends ServiceProvider
implements xxxRemoteInterface, SorcerConstants{
    public Context servicexxx(Context
    context) throws RemoteException {
    }
}
```

- Tight integration with c, c++, fortran etc.. using jna, jni and swig.
- Note: Services Take a ***Context*** and return a ***Context***

Robustness & Reuse
Your Computational Environment is only as good as your Wrapping!

Service Requestor(Client)

- RequestorClass to construct exertions

```
Public class xxxRequestor  extends
SorcerRequestor implements, SorcerConstants{

    Code to create Exertions & submit to the
Sorcer environment

}
```

Anyone/thing can be a requestor – CAD, MATLAB, etc..

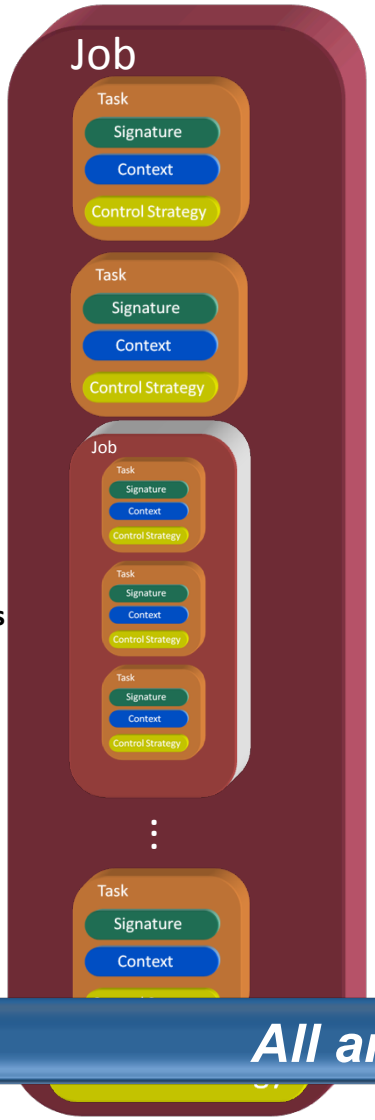
Exertion

SORCER

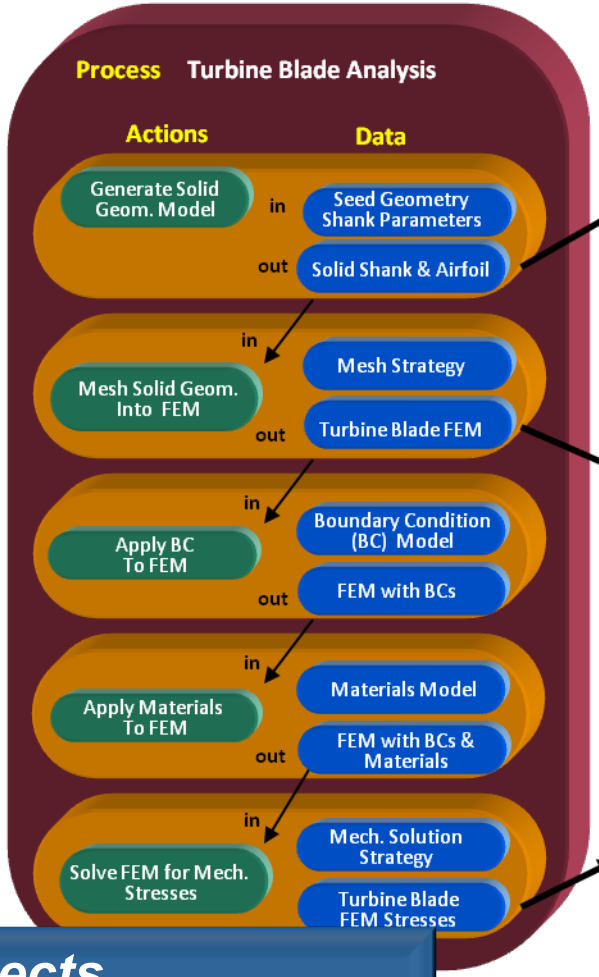
Elementary Exertion

Signature – identifies the Service
Context – contains input & output data/objects
Control Strategy – Guidance for Service

Composite Exertion



Engineering Example



All are Objects

Building an Elementary Exertion - ServiceTask

ServiceTask

Class: `sorcer.core.exertion.ServiceTask`

A Constructor: `public ServiceTask(String name, String description)`

An Example: `Exertion taskHi = new ServiceTask("hi", "Task to say hello”)`

Signature

Class: `sorcer.core.signature.ServiceSignature`

A Constructor:
```
public ServiceSignature(String methodName, Class<?>
serviceType, String providerName)
```

An Example:
```
Signature methodHi = new ServiceSignature("sayHi",
        HelloRemoteInterface.class, “Hello Mac1”);
```

ServiceContext

Class: `sorcer.core.context.ServiceContext`

A Constructor: `public ServiceContext(String name)`

An Example: `Context contextHi = new ServiceContext("sayHiContext”);`

Note:
Context is a hashtable.
(Name value pair).

ControlContext

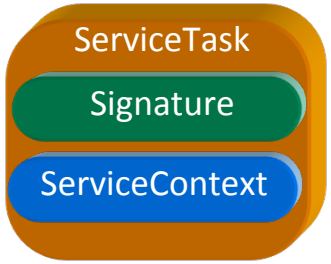
Class: `sorcer.core.context.ControlContext`

A Constructor: `public ControlContext()`

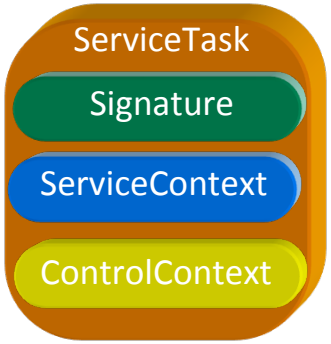
An Example: `Context controlContextHi = new ControlContext();`

Building an Elementary Exertion - ServiceTask

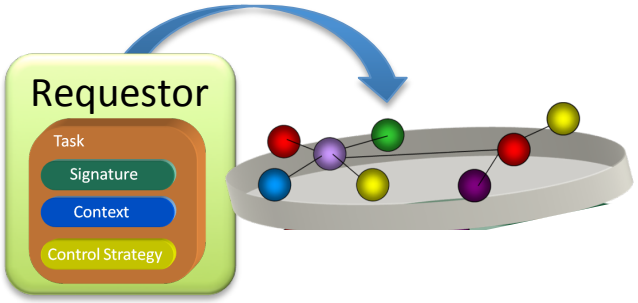

```
taskHi.addSignature(methodHi);
```




```
taskHi.setContext(contextHi);
```




```
taskHi.setControlContext(controlContextHi);
```



To run the task in the SORCER Environment: `taskHi.exert(null);`

Building a Composite Exertion - ServiceJob

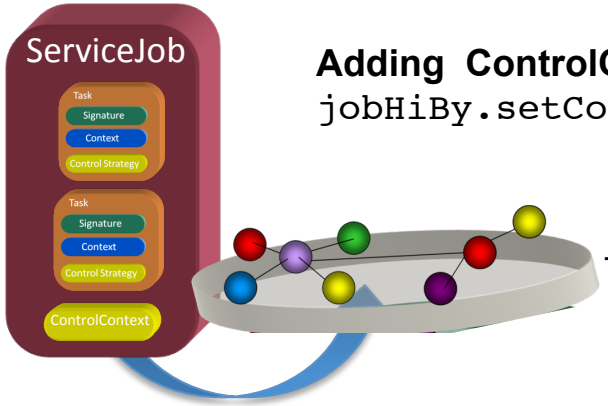
Class: `sorcer.core.exertion.ServiceJob`

A Constructor: `public ServiceJob(String name)`

An Example: `Exertion jobHiBy = new ServiceJob("HiBy");`

Adding tasks to the Job: `jobHiBy.addExertion(taskHi)`

Adding a 2nd tasks to the Job: `jobHiBy.addExertion(taskBy)`



Adding ControlContext to the Job:
`jobHiBy.setControlContext(controlContextHiBy);`

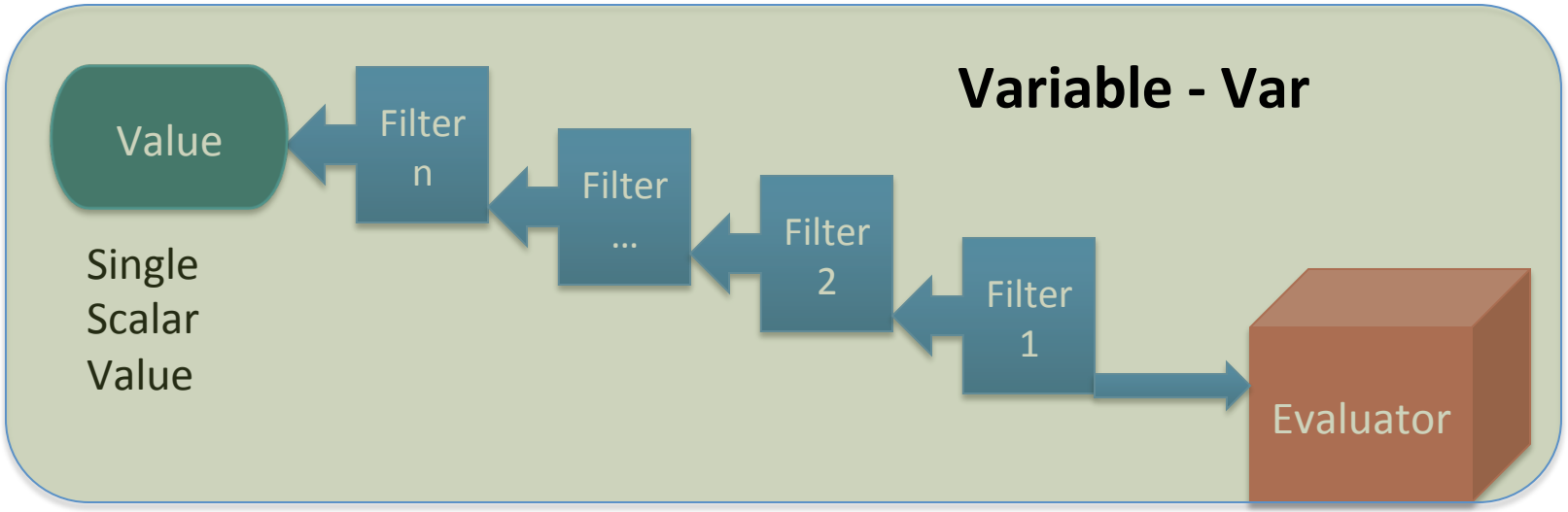
To run the Job in the SORCER Environment: `jobHiBy.exert(null);`

SORCER Model & Variables

- ***Models –*** consists of ***Variables, Filters*** and ***Evaluators***.
  - Current available models – ResponseModel(including sensitivities) , ParametricModel,, OptimizationModel
- ***Variables (Var) -*** can be dependent on other *Variables* enabling distributed **functional programming**. Variables can have multiple evaluators enabling **multi-fidelity calculations** for a specific variable's value.
- ***Evaluators –*** are used to determine the value of variables and their partial derivatives(chain rule works) with respect to their dependencies (Variables).
  - Current Evaluator Types – ModelEvaluator, ExertionEvaluator, ExpressionEvaluator, GroovyEvaluator, JepEvaluator, MethodEvaluator
- ***Filters -*** are used to map the results of *Evaluators* to *Variable* Values. Think unix shell piping. Filters can be concatenated *n* times.
  - Current Filter Types – BasicFileFilter, ContextFilter, FileFilter, GrepFilter, ListFilter, MapFilter, ObjetFilter, PatternFilter, TextFilter

All are Objects

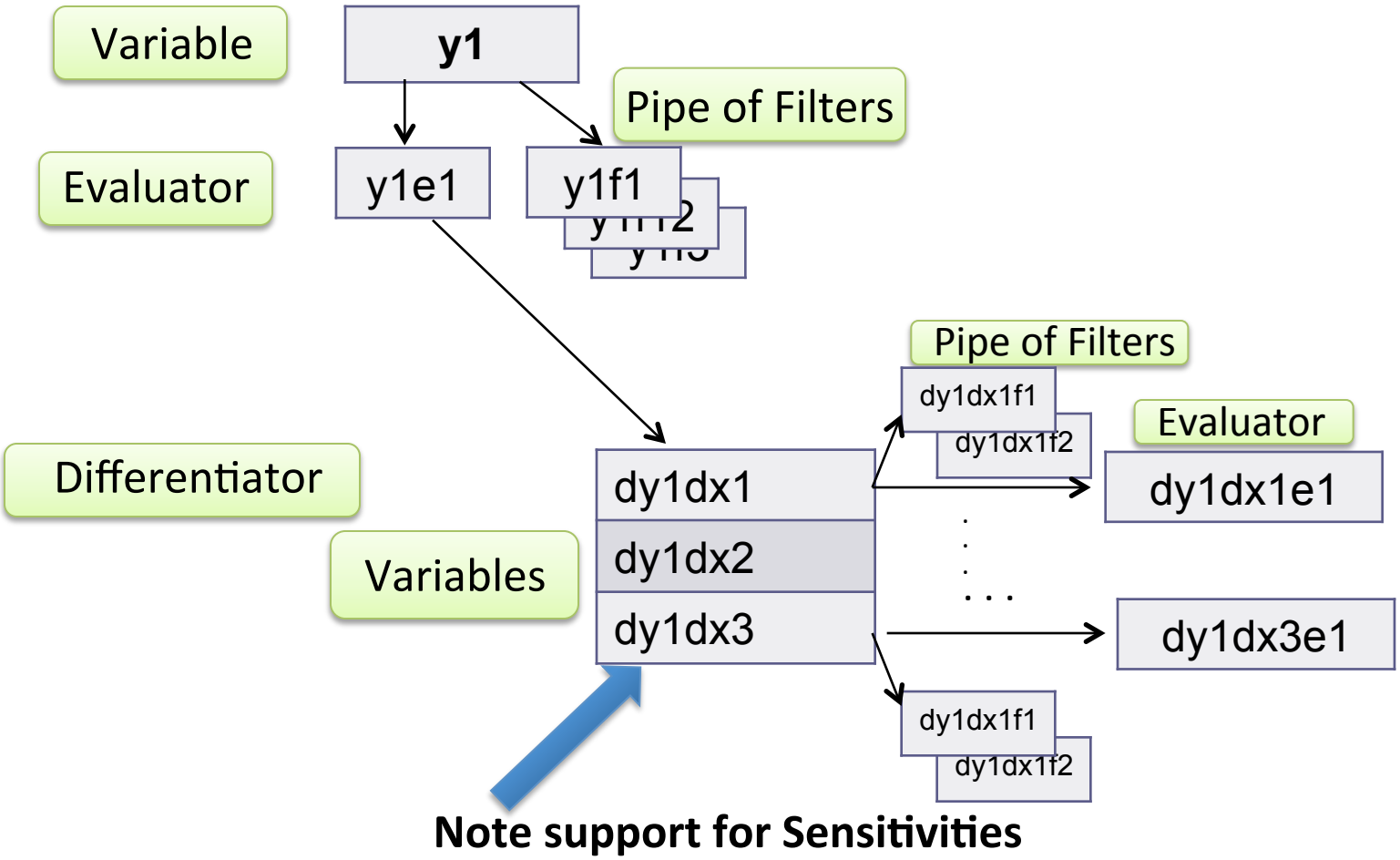
Variable - Value, Filters, Evaluators



Computational Entity. Creating Large Amounts of Data

Basic Variable Structure

$$y_1(x_1, x_2, x_3)$$

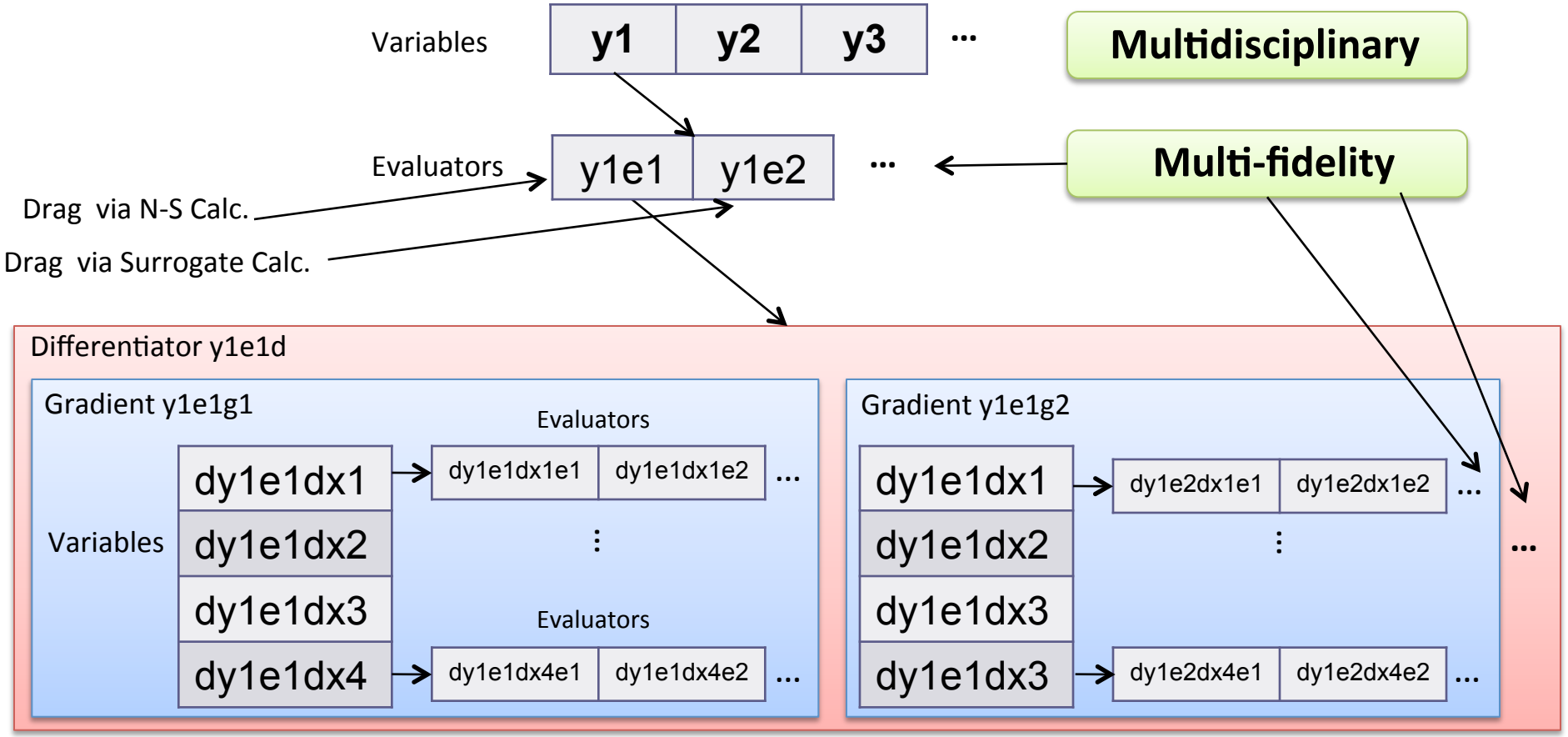


Advanced Variable Structure

SORCER

Functions of Functions

$$y_1(x_1, x_2, x_3), \quad y_2(x_4, x_5, x_6, y_1), \quad y_3(x_6, x_7, x_8, y_2)$$



Evaluators

- ***Evaluators –*** are used to determine the value of variables and their partial derivatives(chain rule works) with respect to their dependencies (Variables). Evaluators do not necessarily produce a scalar value. The result can be an entire database or file. (for a given set of input an application creates many outputs)
  - Current Evaluator Types – ModelEvaluator, ExertionEvaluator, ExpressionEvaluator, GroovyEvaluator, JepEvaluator, MethodEvaluator, SOAEvaluator, FDEvaluator

Filters

- ***Filters -*** are used to map the results of *Evaluators* to *Variable* Values. Think unix shell piping. Filters take the output of Evaluators and reduce it to a single scalar value necessary to define a *Variable* value. Filters can be concatenated *n* times.
  - Current Filter Types – BasicFileFilter, ContextFilter, FileFilter, GrepFilter, ListFilter, MapFilter, ObjetFilter, PatternFilter, TextFilter

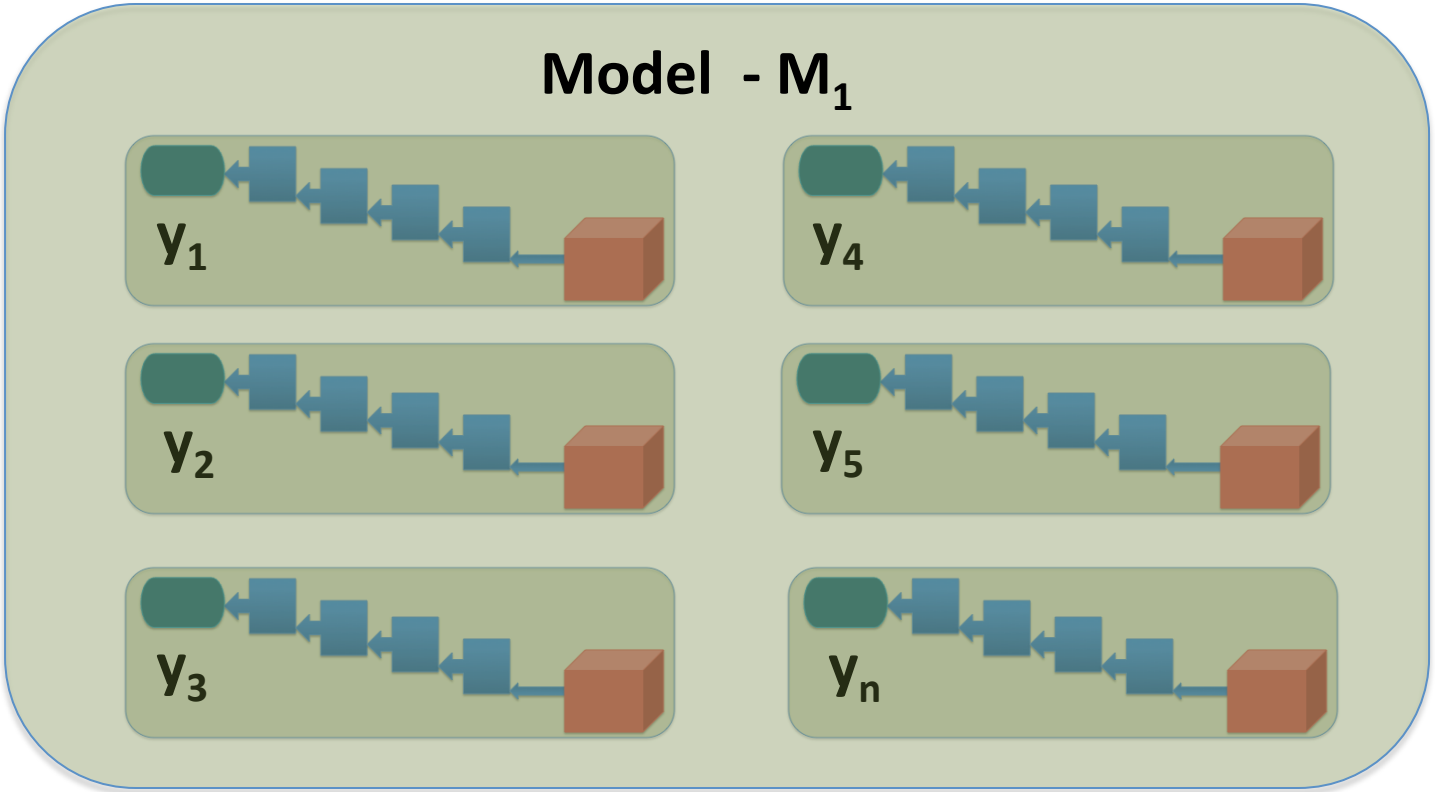
SORCER Models

SORCER Models support

- Analysis – (ResponsModel & ParametricModel)
  - multidisciplinary response analysis
  - multi-fidelity response analysis
  - multi-fidelity response sensitivities

- Design Space Exploration (OptimizationModel)
  - Optimization
  - Multidisciplinary Optimization

Model is a Collection of Variables



Training Agenda

- SORCER Overview
- Creating your own workspace
- Running Hello World Examples
- Creating Providers & Publishing
- Creating Requestors (Job, Task, Context) & Executing calling providers

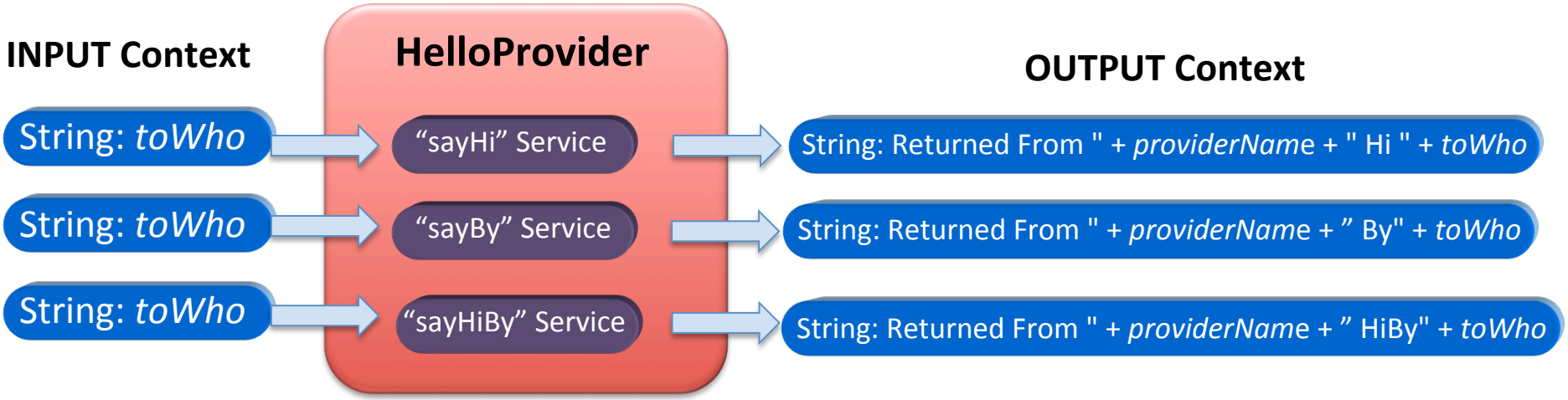
"Hello World" Example

iGrid-10/modules/examples/hello

- Create a single ServiceProvider that has three services: "sayHi", "sayBy", "sayHiBy".
- Example 1: Create a ServiceRequestor that creates a single ServiceTask to "call" the "sayHi" service.
- Example 2: Create a ServiceRequestor that creates a ServiceJob that calls the "sayHi" service, "sayBy" service and the "sayHiBy" service and pass a string into the service, modify it and return.

HelloProvider: “sayHi”, “sayBy”, “sayHiBy” Services

Create a single ServiceProvider that has three services: “sayHi”, “sayBy”, “sayHiBy”.



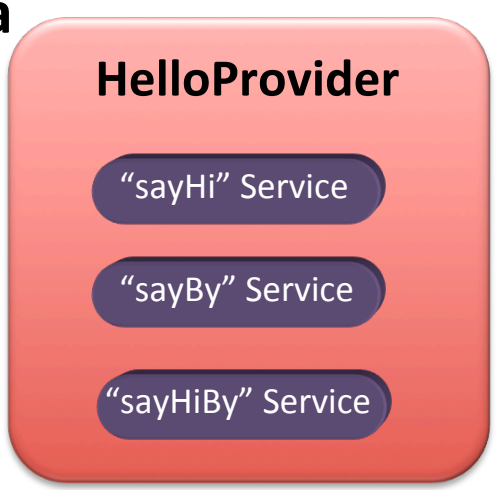
Steps to creating a service provider:

1. Create a generic Interface that has all the services (methods) that the provider will expose.
2. Create a Remote Interface that extends the generic Interface
3. Create a Provider class that extends ServiceProvider and implements the Remote Interface, and contains the business logic for each of the services defined in the Interface.

HelloInterface

engineering.provider.hello.HelloInterface.java

```
package engineering.provider.hello;

import java.rmi.Remote;
import java.rmi.RemoteException;
import sorcer.service.Context;

public interface HelloInterface {

    public Context sayHi(Context context)
        throws RemoteException;

    public Context sayBy(Context context)
        throws RemoteException;

    public Context sayHiBy(Context context)
        throws RemoteException;
}
```



"sayHi" Service

"sayBy" Service

"sayHiBy" Service

Generic Interface – Each Service takes a Context and returns a Context

HelloRemoteInterface

engineering.provider.hello.HelloRemoteInterface.java


```
package engineering.provider.hello;

import java.rmi.Remote;
import java.rmi.RemoteException;
import sorcer.service.Context;

public interface HelloRemoteInterface extends HelloInterface, Remote {

    public Context sayHi(Context context) throws RemoteException;

    public Context sayBy(Context context) throws RemoteException;

    public Context sayHiBy(Context context) throws RemoteException;
}
```

"sayHi" Service

"sayBy" Service

"sayHiBy" Service

Remote Interface enables calls from anywhere on the network

HelloProvider

engineering.provider.hello.HelloProvider.java


```
package engineering.provider.hello;
import java.rmi.RemoteException;
import sorcer.core.SorcerConstants;
import sorcer.core.provider.ServiceProvider;
import sorcer.service.Context;
import com.sun.jini.start.LifeCycle;

public class HelloProvider extends ServiceProvider implements
                HelloRemoteInterface, SorcerConstants {

        public HelloProvider(String[] args, LifeCycle lifeCycle)
         throws Exception {
                super(args, lifeCycle);
        }
```

Constructor must be present for Jini service

…

Source continued on next slide

Providers contain the business logic for the services

HelloProvider

SORCER

engineering.provider.hello.HelloProvider.java


```
public Context sayHi(Context context) throws RemoteException {
String providerName = getProviderName();
  try {
        String input = (String) context.getValue("in/value");
        String output = "Returned From " + providerName + " Hi " + input;
        context.putOutValue("out/value", output);
  } catch (Exception e) {
    e.printStackTrace();
  }
 return context;
}
public Context sayBy(Context context) throws RemoteException {
String providerName = getProviderName();
  try {
        String input = (String) context.getValue("in/value");
        String output = "Returned From " + providerName + " By " + input;
        context.putOutValue("out/value", output);
  } catch (Exception e) {
    e.printStackTrace();
  }
 return context;
}
```

"sayHi" Service

"sayBy" Service

…

Source continued on next slide

HelloProvider

engineering.provider.hello.HelloProvider.java

"sayHiBy" Service

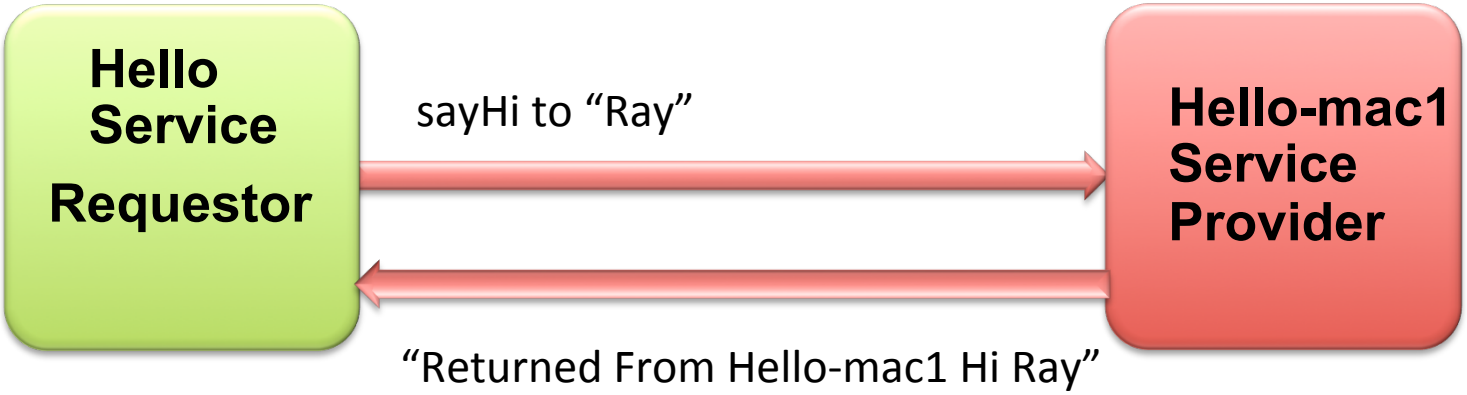
```
public Context sayHiBy(Context context) throws RemoteException {
String providerName = getProviderName();
  try {
        String input = (String) context.getValue("in/value");
        String output = "Returned From " + providerName + " HiBy " + input;
        context.putOutValue("out/value", output);
  } catch (Exception e) {
    e.printStackTrace();
  }
 return context;
}

}
```

"Hello World" Example

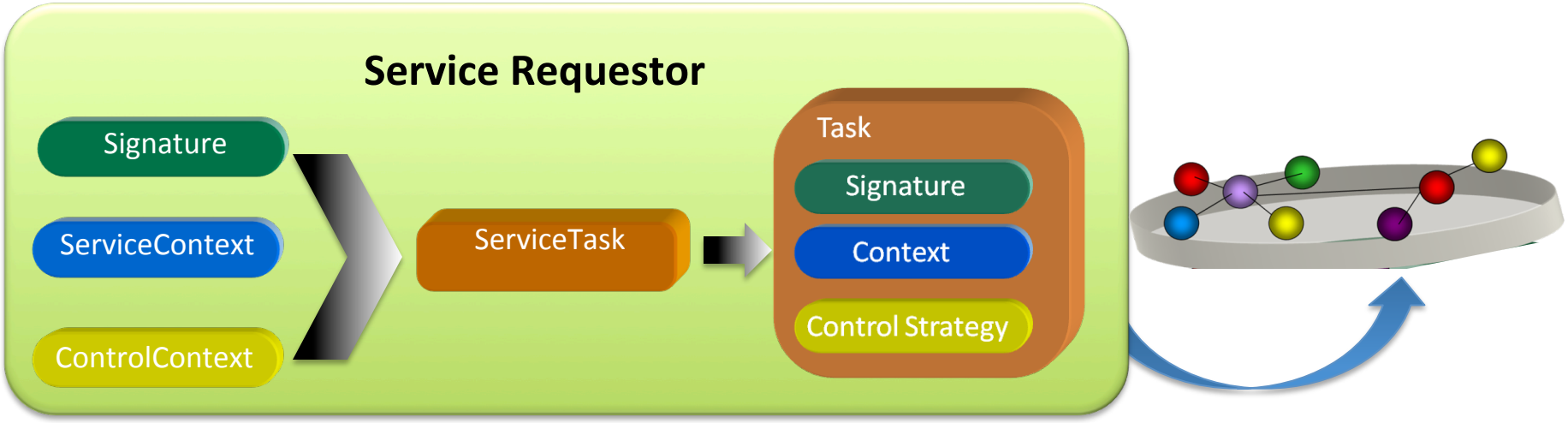
- Create a single ServiceProvider that has three services: "sayHi", "sayBy", "sayHiBy".
- Example 1: Create a ServiceRequestor that creates a single ServiceTask to "call" the "sayHi" service.
- Example 2: Create a ServiceRequestor that creates a ServiceJob that calls the "sayHi" service, "sayBy" service and the "sayHiBy" service and pass a string into the service, modify it and return.

"Hello World" Example

Conceptually what we would like to do



HelloTaskRequestor

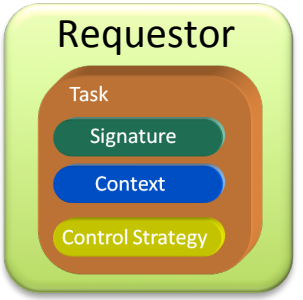


Steps to creating a service requestor:

1. Create a class that may extend SorcerRequestor
2. Create and set the System Security Manager
3. Create the task/job with each exertion containing a Signature, ServiceContext and an optional ControlContext
4. Submit the task/job to the SORCER grid for execution

HelloTaskRequestor

engineering.requestor.hello.java



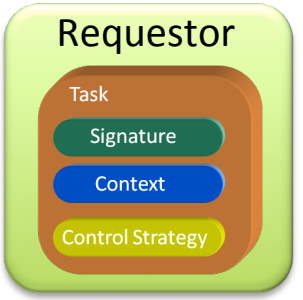

```
public class HelloTaskRequestor implements SorcerConstants {

public static void main(String[] args) throws Exception {

if (System.getSecurityManager() == null)
System.setSecurityManager(new RMISecurityManager());

String hiProviderName, toWho;

hiProviderName = args[0];
System.out.println("Hi Service: " + hiProviderName);

toWho = args[1];
System.out.println("Name of person you want to speak to: " + toWho);
```

…

Source continued on next slide

Note: args[] is a string array that are supplied on the java command line. They are set in the helloTask-req-run.xml file.

HelloTaskRequestor



engineering.requestor.hello.java

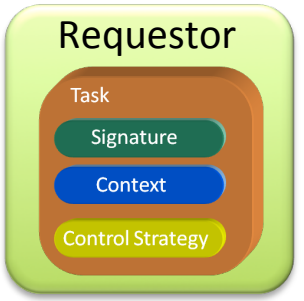
```
        // Create the task
ServiceTask   Exertion taskHi = new ServiceTask("hi", "Task to say hello");

        // Create signatures for  the Hi service and
        // set the provider name accordingly from the online args
Signature     Signature methodHi = new ServiceSignature("sayHi",
               engineering.provider.hello.HelloRemoteInterface.class, hiProviderName)

        // Create the Context for the input/output data
ServiceContext   Context contextHi = new ServiceContext("hiContext");

        //populate the context with the input data
            contextHi.putInValue("in/value", toWho);
        // create an empty node to hold the output - This is provider dependent
            contextHi.putValue("out/value", Context.EMPTY_LEAF);
            System.out.println("\ttaskHi context = " + contextHi + "\n");
```

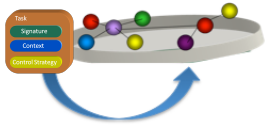
…

Source continued on next slide

HelloTaskRequestor

engineering.requestor.hello.java





```
    // Create the Control context - this is optional,
    // a default is created in the task
    Context controlContextHi = new ControlContext();

    // put the context into the task
    ((ServiceTask)taskHi).setContext(contextHi);
    // put the signature into the task
    ((ServiceTask)taskHi).addSignature(methodHi);
    // put the controlContext into the task
    ((ServiceTask)taskHi).setControlContext((ControlContext)controlContextHi);

    // submit the task to the sorcer grid for execution
    Exertion result = taskHi.exert(null);

    // print the results
    System.out.println("\n***************************");
    System.out.println("\tsayHi Method Response = "
    + result.getExertions().get(0).getContext().getValue("out/value"));
    System.out.println("\n***************************");
}
```

Running the HelloTask Example

- Start your environment webster (if not running)
  - Execute iGrid-10/bin/webster/bin/webster-run.xml
- Start the i-Grid basic services (if not running)
  - Execute i-Grid-10/bin/iGrid-min-boot.xml
- Start the i-Grid service browser (if not running)
  - Execute i-Grid-10/bin/browser/bin/service-browser-run.xml
- Start/publish the HelloProvider
  - i-Grid10/modules/examples/hello/hello/provider/bin/jeri-helloHi-run.xml
    - Should now see the HelloProvider in the Service Browser
- Execute the HelloTaskRequestor
  - i-Grid10/modules/examples/hello/hello/requestor/bin/helloTask-req-run.xml

Running the HelloTask Example

The following should print out to the console:

```
**************************
sayHi Method Response = Returned From Hello-mac1 Hi Ray
**************************
```

Running the HelloTask Example

- If you wish to change the service that is called or who to say hello to
  - Edit i-Grid10/modules/examples/hello/hello/requestor/bin/helloTask-req-run.xml

```
<target name="run.requestor">
<java classname="${requestor.class}" fork="yes">
<arg value="Hello-mac1" />
<arg value="Ray" />
<classpath refid="project.classpath" />
```

Service Provider Name

Say Hello to

- To change the name of your HelloProvider name
  - Edit i-Grid-10/bin/modules/examples/hello/hello/provider/bin/configs/helloHi-prv.config & republish your service

```
name="Hello-mac1";
```

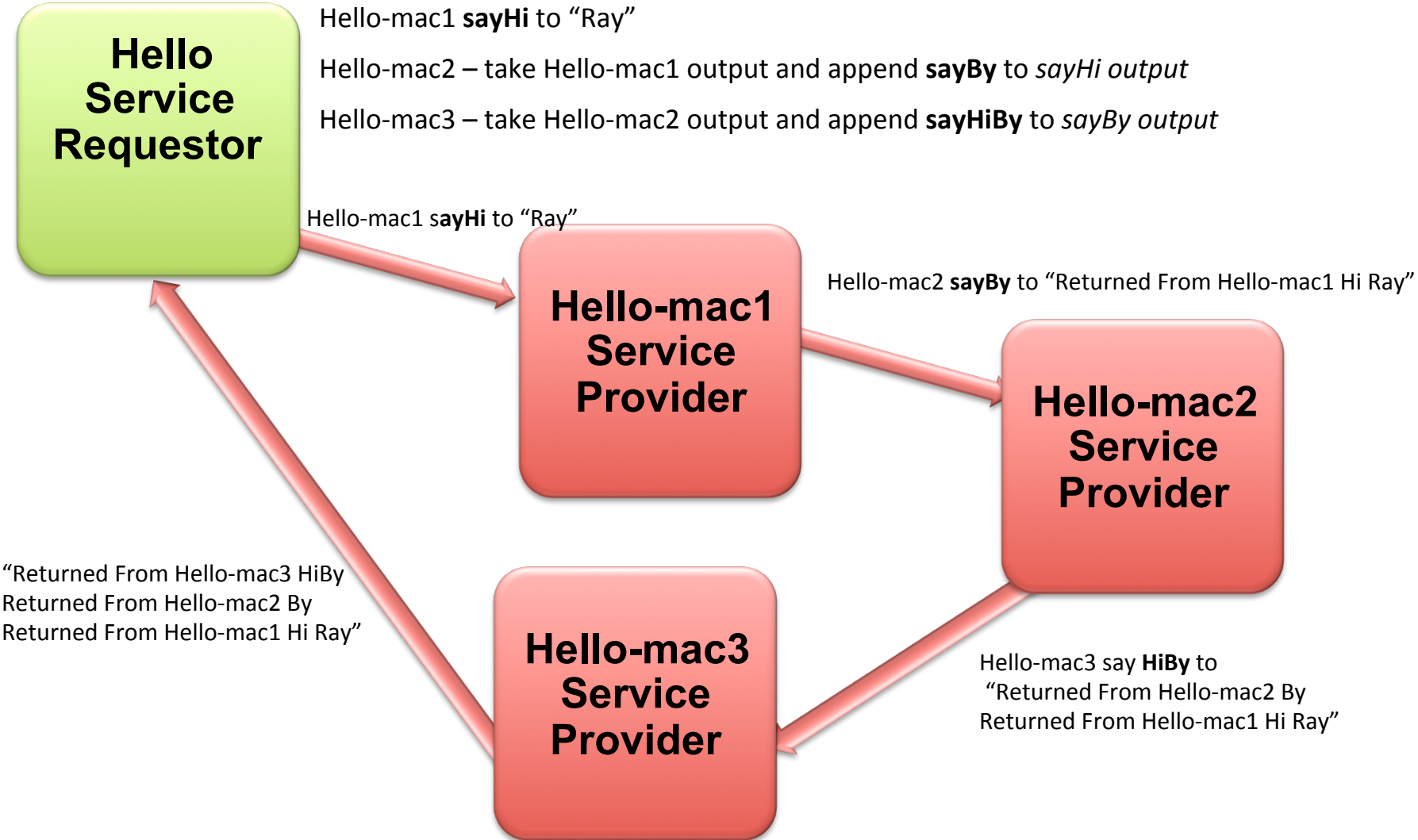
Your HelloProvider published name

"Hello World" Example

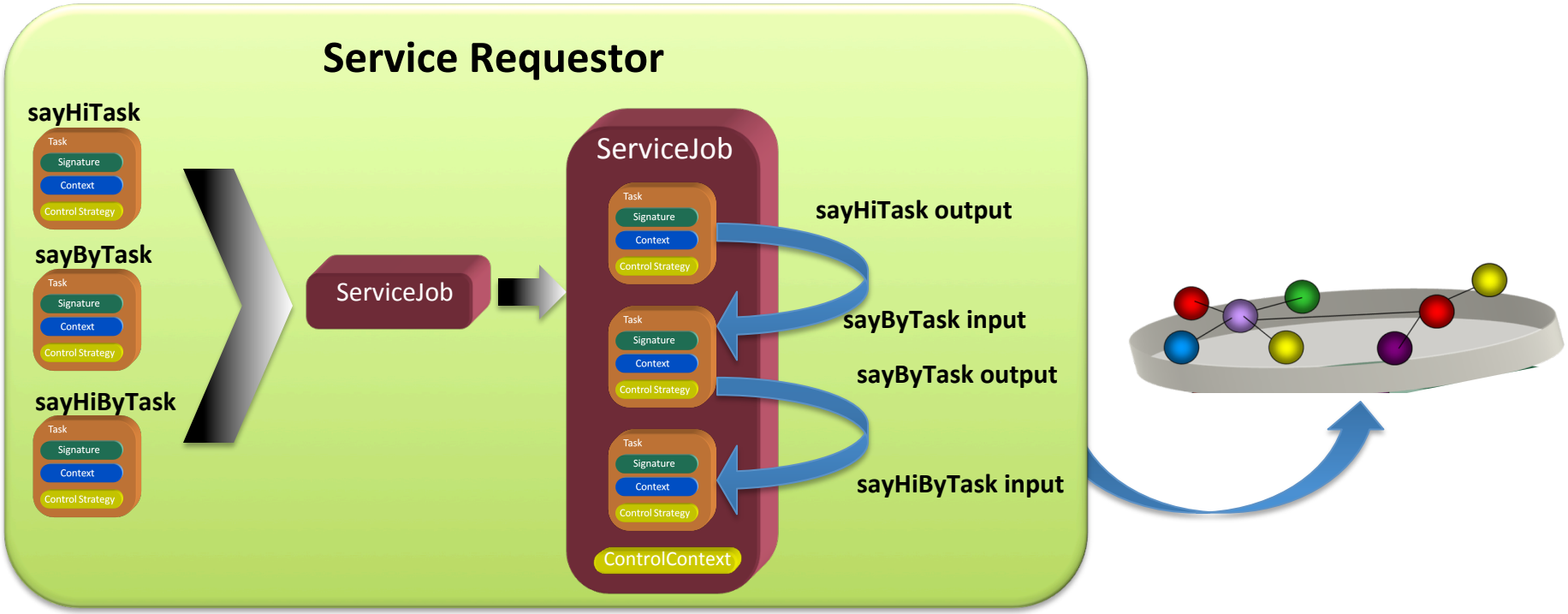
- Create a single ServiceProvider that has three services: "sayHi", "sayBy", "sayHiBy".
- Example 1: Create a ServiceRequestor that creates a single ServiceTask to "call" the "sayHi" service.
- Example 2: Create a ServiceRequestor that creates a ServiceJob that calls the "sayHi" service, "sayBy" service and the "sayHiBy" service and pass a string into the service, modify it and return.

"Hello World" Example

Conceptually what we would like to do



HelloJobRequestor

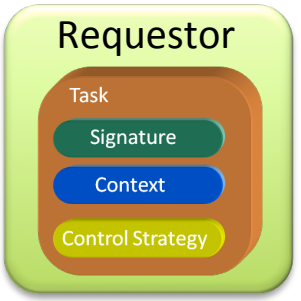


Steps to creating a service requestor:

1. Create a class that may extend SorcerRequestor
2. Create and set the System Security Manager
3. Create the job with each exertion containing a Signature, ServiceContext and an optional ControlContext for each task
4. Place each task into the Job
5. Map the output from the sayHiTask to the sayByTask, and the output from the sayByTask to the sayHiByTask
6. Submit the job to the SORCER grid for execution

HelloJobRequestor

engineering.requestor.hello.HelloJobRequestor.java




```
public class HelloJobRequestor implements SorcerConstants {

public static void main(String[] args) throws Exception {

if (System.getSecurityManager() == null)
System.setSecurityManager(new RMISecurityManager());

String hiProviderName, byProviderName, hiByProviderName, toWho;

hiProviderName = args[0];
System.out.println("Hi Service: " + hiProviderName);

byProviderName = args[1];
System.out.println("By Service: " + byProviderName);

hiByProviderName = args[2];
System.out.println("HiBy Service: " + hiByProviderName);

toWho = args[3];
System.out.println("Name of person you want to speak to: " + toWho);
```

…

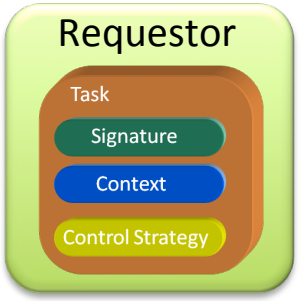
Source continued on next slide

Note: args[] is a string array that are supplied on the java command line. They are set in the helloJob-req-run.xml file.

HelloJobRequestor

SORCER

engineering.requestor.hello.HelloJobRequestor.java



sayHiTask
sayByTask
sayHiByTask

```
        // Create the component tasks, and put the service signature in
        // This constructor actually creates a default ControlContext
             Exertion taskHi = new ServiceTask("hi", "Task to say hello");
             Exertion taskBy = new ServiceTask("by", "Task to say by");
             Exertion taskHiBy = new ServiceTask("hiby", "Task to say hiby");

             // Create signatures for services the Hi, By, and HiBy and
             // set the provider name accordingly from the online args
```

sayHi Sig.
sayBySig.
sayHiBySig.

```
             Signature methodHi = new ServiceSignature("sayHi",  HelloRemoteInterface.class,
                                                       hiProviderName);
             Signature methodBy = new ServiceSignature("sayBy", HelloRemoteInterface.class,
                                                       byProviderName);
             Signature methodHiBy = new ServiceSignature("sayHiBy", HelloRemoteInterface.class,
                                    hiByProviderName);

        // Create & populate the context for the task Hi
```

sayHiContext

```
             Context contextHi = new ServiceContext("sayHiContext");
             contextHi.putInValue("in/value", toWho);
             contextHi.putOutValue("out/value", Context.EMPTY_LEAF);

        // Create & populate the context for the task By
```

sayByContext

```
             Context contextBy = new ServiceContext("sayByContext");
        // note the input for this task will come from the output from the Hi Task
             contextBy.putInValue("in/value", Context.EMPTY_LEAF);
             contextBy.putOutValue("out/value", Context.EMPTY_LEAF);
```

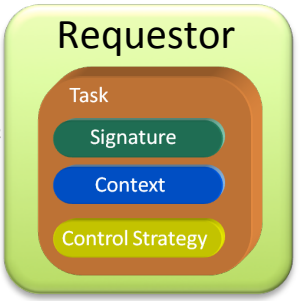
…

Source continued on next slide

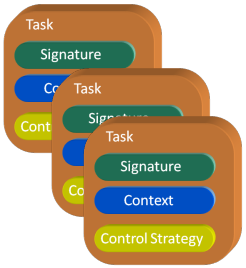
HelloJobRequestor

SORCER

engineering.requestor.hello.HelloJobRequestor.java




```
// Create & populate the context for the task HiBy
    Context contextHiBy = new ServiceContext("sayHiByContext");
ote the input for this task will come from the output from
//the By Task
    contextHiBy.putInValue("in/value", Context.EMPTY_LEAF);
```




```
// populate the tasks with the signatures and the context
((ServiceTask)taskHi).addSignature(methodHi);
((ServiceTask)taskHi).setContext(contextHi);
((ServiceTask)taskBy).addSignature(methodBy);
((ServiceTask)taskBy).setContext(contextBy);
((ServiceTask)taskHiBy).addSignature(methodHiBy);
((ServiceTask)taskHiBy).setContext(contextHiBy);
```

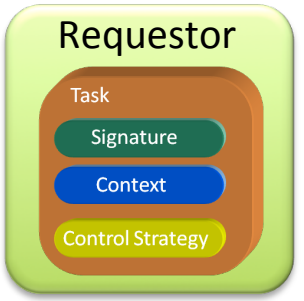

```
// create the exertion job composed of the above tasks
Job job = new ServiceJob("HelloBy");
```


```
job.addExertion(taskHi);
job.addExertion(taskBy);
job.addExertion(taskHiBy);
```

HelloJobRequestor

SORCER

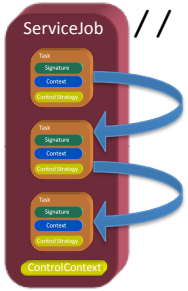
engineering.requestor.hello.HelloJobRequestor.java



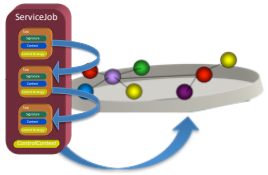

```
        // create the control context and set access to push, and
        // flow to sequential.
        // Finally request that the execution time for the job
```


```
        Context controlContextHiBy = new ControlContext();
```


```
        ((ControlContext)controlContextHiBy).setAccessType(Access.PUSH);
        ((ControlContext)controlContextHiBy).setFlowType(Flow.SEQ);
        ((ControlContext)controlContextHiBy).setExecTimeRequested(true);
    // set the job control strategy
        job.setControlContext((ControlContext)controlContextHiBy);
```




```
// create data context mappings from on task to the next
        contextBy.map("out/value", "in/value", contextHiBy);
        contextHi.map("out/value", "in/value", contextBy);
        contextBy.map("out/value", "in/value", contextHiBy);
```




```
        // execute the service job
         Exertion result = job.exert(null);
```

Running the HelloJob Example

- Start your environment webster (if not running)
  - Execute iGrid-10/bin/webster/bin/webster-run.xml
- Start the i-Grid basic services (if not running)
  - Execute i-Grid-10/bin/iGrid-min-boot.xml
- Start the i-Grid service browser (if not running)
  - Execute i-Grid-10/bin/browser/bin/service-browser-run.xml
- Start/publish the three HelloProviders, (Hi, By, HiBy)
  - i-Grid10/modules/examples/hello/hello/provider/bin/jeri-helloHi-run.xml
  - i-Grid10/modules/examples/hello/hello/provider/bin/jeri-helloBy-run.xml
  - i-Grid10/modules/examples/hello/hello/provider/bin/jeri-helloHiBy-run.xml
    - Should now see the three HelloProvider in the Service Browser
- Execute the HelloJobRequestor
  - i-Grid10/modules/examples/hello/hello/requestor/bin/helloJob-req-run.xml

Running the HelloJob Example

The following should print out to the console:

```
**************************
sayHi Method Response = Returned From Hello-mac1 Hi Ray
**************************
sayBy Method Response = Returned From Hello-mac2 By Returned From Hello-mac1 Hi Ray
**************************
sayHiBy Method Response = Returned From Hello-mac3 HiBy Returned From Hello-mac2 By Returned From Hello-mac1 Hi Ray
```

Running the HelloJob Example

- If you wish to change the service that is called or who to say hello to
  - Edit i-Grid10/modules/examples/hello/hello/requestor/bin/helloJob-req-run.xml

```
<target name="run.requestor">
<java classname="${requestor.class}" fork="yes">
<arg value="Hello-mac1" />
<arg value="Ray2" />
<classpath refid="project.classpath" />
```

Service Provider Names

Say Hello to

- To change the name of your HelloProviders name
  - Edit i-Grid-10/bin/modules/examples/hello/hello/provider/bin/configs/helloHi,By,HiBy-prv.config & republish your service name="Hello-mac1";

Your HelloProvider published name

Creating your Own Provider & Requestor

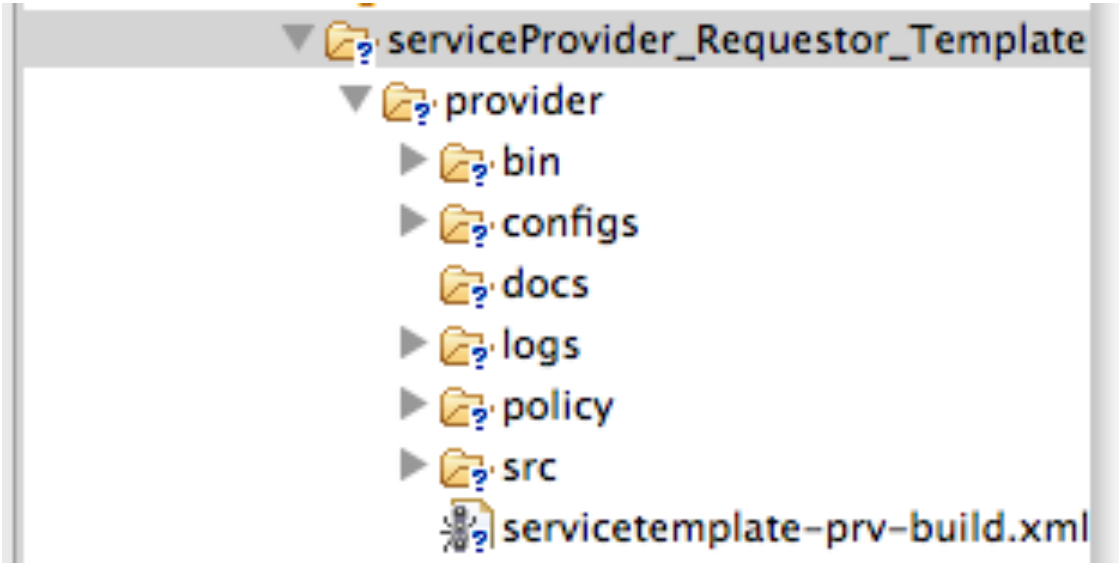
- Use existing files of providers & requestors as a template
- Engineering/apps/serviceProvider_Requestor_Template
- NEVER copy folders! Only Files (.svn hidden files will cause chaos)
- You can use the eclipse export/import facility to “copy” folder structures. (export does not include .svn files). If using export make sure the eclipse environment is in sync with the file system. (right click “refresh” on folder to export)
- Import – General-FileSystem

Creating your Own Provider & Requestor

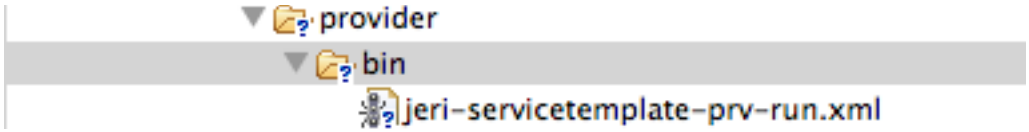
- Engineering/apps/serviceProvider_Requestor_Template
- Export serviceProvider_Requestor_Template folder
- Create new folder where the new service will be created
- Import the folder that was exported into the new location
- Change all the “servicetemplate” (file names and in files) to your provider package (example astros)
- Chage all “ServiceTemplate” (file name and in files) to your ProviderName (example AstrosProvider)

Creating your Own Provider & Requestor

- To create a new Sorcer provider & requestor for a given provider in Eclipse the following folders and files need to be created.
- Folders & hierarchy
  - provider - The provider folder will contain all the folders and files needed to create, compile and publish the provider. Below is the first level of folders and files required.



Creating your Own Provider & Requestor



- The provider bin folder contains at a minimum an ant xml file that is used to publish the serviceTempletProviderImpl (found in the provider/src folder) class service to the sorcer registry. It can also contain an xml files that tests the serviceTempleProviderImp class.
- The jeri-servicetemplate-prv-run.xml file is used to publish the service to the sorcer registry. It is essentially a java command with several java command line arguments being set. A typical set of arguments are:

```
<java jar="${jini.lib}/start.jar" fork="yes">
<sysproperty key="java.util.logging.config.file" value="${iGrid.home}/configs/
sorcer.logging" />
<sysproperty key="java.security.policy" value="../policy/${provider.name}-prv.policy" />
<sysproperty key="java.rmi.server.codebase" value="${j1} ${j2} ${j3}" />
<sysproperty key="sorcer.provider.codebase" value="${j1} ${j2} ${j3} ${j4} ${j5} ${j6} $
{j7}" />
<sysproperty key="sorcer.provider.classpath" value="${toString:project.classpath}" />
<sysproperty key="sorcer.provider.impl" value="${provider.class}" />
<sysproperty key="sorcer.provider.config" value="../configs/jeri-${provider.name}-
prv.config" />
<sysproperty key="iGrid.home" value="${iGrid.home}" />
<sysproperty key="provider.properties" value="../configs/jeri-servicetemplate-
prv.properties" />
```

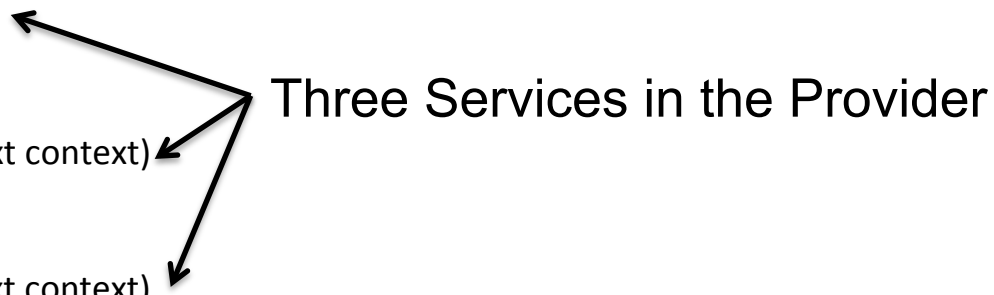
Engineering Provider - ASTROS

Automated **STR**uctural Optimization **S**ystem (ASTROS) – FEM for structures, panel code for aerodynamics. Performs static and dynamic aeroelastic optimization (structural sizing)

engineering.provider.astros.AstrosInterface.java

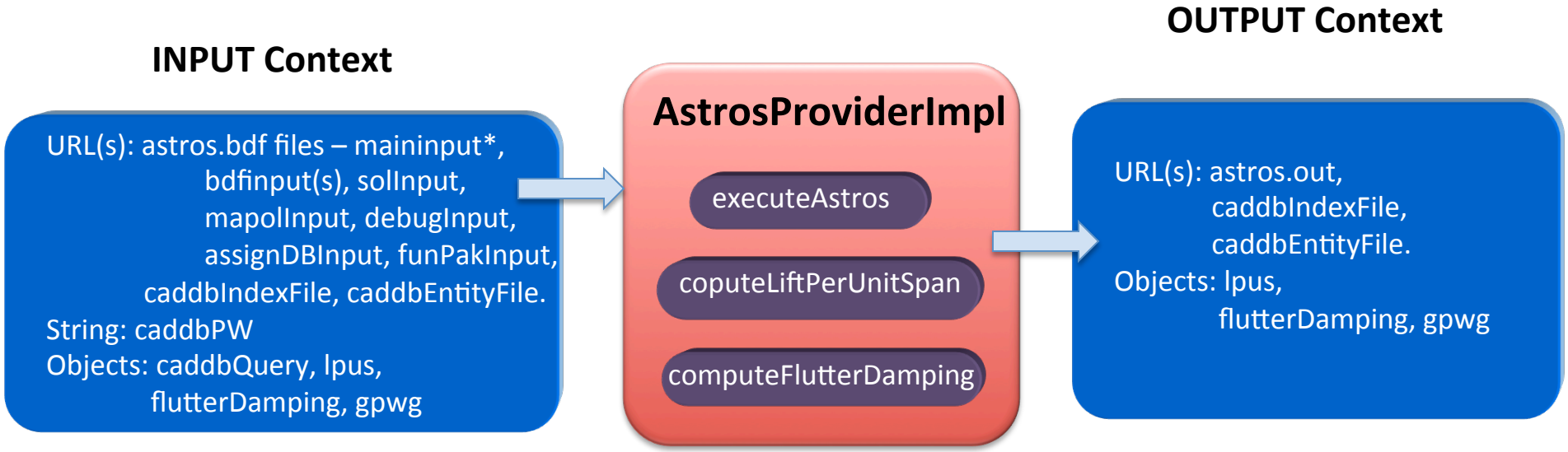
(iGrid-10/modules/engineering/apps/astros/provider/src/engineering/provider/astros/AstrosInterface.java)

```
public interface AstrosInterface {

   public Context executeAstros(Context context)
      throws RemoteException;

   public Context computeLiftPerUnitSpan(Context context)
   throws RemoteException;

   public Context computeFlutterDamping(Context context)
   throws RemoteException;
}
```



AstrosProviderImpl: "executeAstros", "computeLiftPerUnitSpan", "computeFlutterDamping" Services

engineering.provider.astros.AstrosProviderImpl.java in folder
iGrid-10/modules/engineering/apps/astros/provider/src/engineering/provider/astros/AstrosProviderImpl.java



Provider can run ASTROS in three ways: systemCall, slurm, JNA. All database queries are done using JNA. **No File Parsing!**

AstrosRequestor

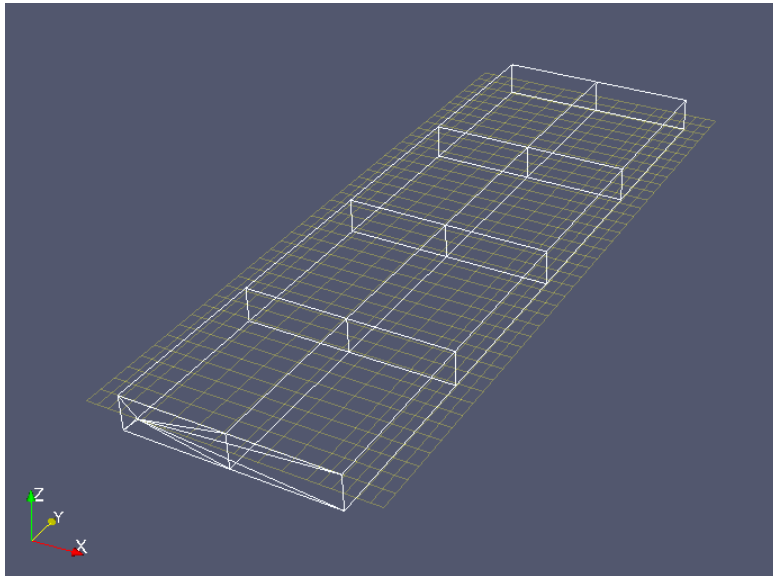
Engineering.requestor.astros.AstrosRequestor.java in folder
iGrid-10/modules/engineering/apps/astros/requestor/src/engineering/requestor/astros/AstrosRequestor.java

The Requestor has three cases that it can run to test the provider services:

```
protected void init(String... args) throws Exception {
        int test = new Integer(args[1]);
        switch (test) {
                case 1: astrosExecuteAstros(); break;
                case 2: astrosComputeLiftPerUnitSpan(); break;
                case 3: astrosComputeFlutterDamping(); break;
        }
}
```

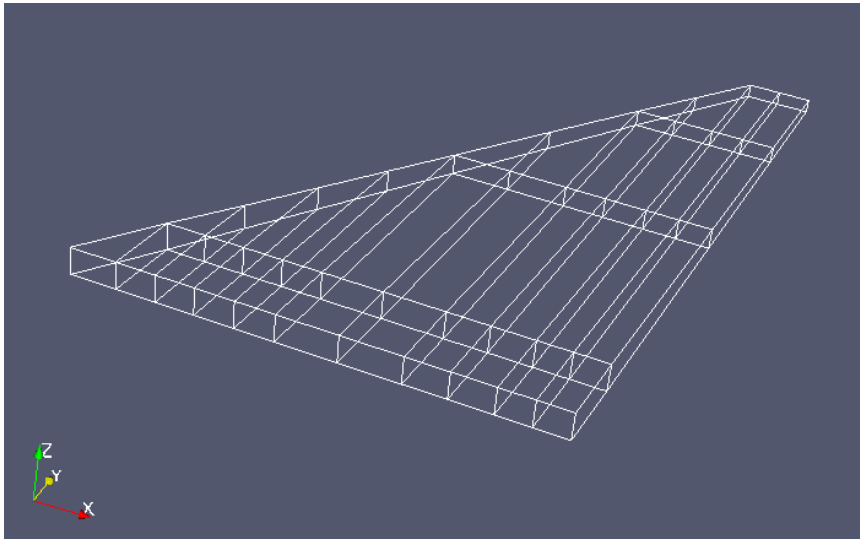
AstrosRequestor Example

Case 1 & 2: Rectangular Wing
Static Aeroelastic Analysis



M=0.7, q=5.0 psi
NZ=1g, Alpha=2.327
20 TE CS set

Case 3: Fighter Wing
Flutter Analysis



M=1.1
Sea Level

Running the AstrosRequestor Example

- Start your environment webster (if not running)
  - Execute iGrid-10/bin/webster/bin/webster-run.xml
- Start the i-Grid basic services (if not running)
  - Execute i-Grid-10/bin/iGrid-min-boot.xml
- Start the i-Grid service browser (if not running)
  - Execute i-Grid-10/bin/browser/bin/service-browser-run.xml
- Start/publish the AstrosProviderImpl (if not running)
  - i-Grid10/modules/engineering/apps/astros/provider/bin/jeri-astros-prv-run.xml
    - Should now see the AstrosProvider in the Service Browser
- Execute the AstrosRequestor
  - i-Grid10/modules/engineering/apps/astros/requestor/bin/astrosRequestor-runner-run.xml (this should run case 1)
  - To run case 2 or case 3 edit the i-Grid10/modules/engineering/apps/astros/requestor/bin/astrosRequestor-runner-run.xml

```
<target name="run.requestor">
<java classname="${requestor.class}" fork="yes">
<arg value="engineering.requestor.astros.AstrosRequestor" />
<arg value="1" />
```

Change this value to change cases (1,2,3)

AstrosRequestor:Case 1

i-Grid10/modules/engineering/apps/astros/requestor/bin/astrosRequestor-runner-run.xml

```
case 1: astrosExecuteAstros();
```

Runs the static aeroelastic analysis on rectangular wing & returns a URL to the output file & database files in the context.

Console Print of Context after analysis has completed

Subject = AstrosContext:
in/value/ASTROS/BDF1/INPUT = http://10.131.4.201:9000/astros/hw7/hw0SWMp7_grid.bdf
in/value/ASTROS/MAIN/INPUT = http://10.131.4.201:9000/astros/hw7/hw0SWMp7.d

out/value/ASTROS/DATABASE/PASSWORD = DNA
out/value/ASTROS/OUTPUT/D01FILE = http://10.131.4.201:9000/astros/provider/scratch/20111104-084630-1336e9dbc0b/ZSW.D01
out/value/ASTROS/OUTPUT/IDXFILE = http://10.131.4.201:9000/astros/provider/scratch/20111104-084630-1336e9dbc0b/ZSW.IDX
out/value/ASTROS/OUTPUT/OUTFILE = http://10.131.4.201:9000/astros/provider/scratch/20111104-084630-1336e9dbc0b/astrosResults.out

Paste this URL into a browser to view the astrosResults.out file (output file)

AstrosRequestor:Case 1

SORCER

i-Grid10/modules/engineering/apps/astros/requestor/bin/astrosRequestor-runner-run.xml

```
case 1: astrosExecuteAstros();
```

out/value/ASTROS/OUTPUT/OUTFILE = http://10.131.4.201:9000/astros/provider/scratch/20111104-084630-1336e9dbc0b/astrosResults.out

TRIM RESULTS FOR TRIM SET 100 OF TYPE LIFT

MACH NUMBER 7.00000E-01
DYNAMIC PRESSURE 5.00000E+00
TRIM PARAMETERS:

DEFINITION	LABEL	FLEXIBLE	RIGID		
ANGLE OF ATTACK	"ALPHA "	2.32700E+00	2.32700E+00	DEG	(USER INPUT)
LOAD FACTOR	"NZ "	3.86100E+02	3.86100E+02		(USER INPUT)
CONTROL SURFACE ROTATION	"CS1 "	-3.96100E+00	-3.96100E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS2 "	-1.56100E+00	-1.56100E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS3 "	-1.82800E+00	-1.82800E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS4 "	-2.35600E+00	-2.35600E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS5 "	-1.21700E+00	-1.21700E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS6 "	-3.76900E+00	-3.76900E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS7 "	4.09100E+00	4.09100E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS8 "	4.07500E+00	4.07500E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS9 "	-7.07000E-01	-7.07000E-01	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS10 "	-4.02000E-01	-4.02000E-01	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS11 "	-2.28600E+00	-2.28600E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS12 "	7.46000E-01	7.46000E-01	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS13 "	1.28600E+00	1.28600E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS14 "	9.67000E-01	9.67000E-01	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS15 "	-1.73600E+00	-1.73600E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS16 "	5.37000E-01	5.37000E-01	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS17 "	-1.46400E+00	-1.46400E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS18 "	-2.15800E+00	-2.15800E+00	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS19 "	5.03000E-01	5.03000E-01	DEG	(USER INPUT)
CONTROL SURFACE ROTATION	"CS20 "	6.51800E+00	6.51800E+00	DEG	(USER INPUT)

AstrosRequestor:Case 2

SORCER

i-Grid10/modules/engineering/apps/astros/requestor/bin/astrosRequestor-runner-run.xml

```
Case 2: astrosComputeLiftPerUnitSpan();
```

Runs the static aeroelastic analysis on rectangular wing & compute Lift per unit span and return in an object.

Console Print of Context after analysis has completed

Subject = AstrosContext:
in/value/ASTROS/BDF1/INPUT = http://10.131.4.201:9000/astros/hw7/hw0SWMp7_grid.bdf
in/value/ASTROS/CADDB/QUERY = engineering.provider.astros.CADDBQuery@f4d5bc9
in/value/ASTROS/MAIN/INPUT = http://10.131.4.201:9000/astros/hw7/hw0SWMp7.d
in/value/ASTROS/OUTPUT/LPUS = engineering.provider.astros.AstrosLpus@1fc4f0f8
in/value/ASTROS/SCRATCH/DIRECTORY = /Users/raymondkolonay/workspace/iGrid-10/data/astros/provider/scratch/20111104-085811-1336ea86dac

Empty AstrosLpus object was input. Object populated with data on output.

out/value/ASTROS/DATABASE/PASSWORD = DNA
out/value/ASTROS/OUTPUT/D01FILE = http://10.131.4.201:9000/astros/provider/scratch/20111104-085811-1336ea86dac/ZSW.D01
out/value/ASTROS/OUTPUT/IDXFILE = http://10.131.4.201:9000/astros/provider/scratch/20111104-085811-1336ea86dac/ZSW.IDX
out/value/ASTROS/OUTPUT/OUTFILE = http://10.131.4.201:9000/astros/provider/scratch/20111104-085811-1336ea86dac/astrosResults.out

>>>>>>>>>> Lift Per Unit Span Results from ASTROS Run <<<<<<<<<<<<

Spanwise cut	appliedLpus	rigidLpus	flexLpus
1	54.16099197169145	36.29533404111861	17.86565801501274
2	54.86817342042924	37.06625717878342	17.801917264858886
3	57.32152642806372	39.58923983573912	17.732287615537643
.			
.			
39	39.39830290277799	28.066914667685825	11.331388533270607
40	40	28.906596819559734	21.190397143363953	7.716199538883908

AstrosRequestor:Case 3

SORCER

i-Grid10/modules/engineering/apps/astros/requestor/bin/astrosRequestor-runner-run.xml

```
case 3: astrosComputeFlutterDamping();
```

Runs the flutter analysis on fighter wing and compute fluttering damping values and return in an object.

Console Print of Context after analysis has completed

in/value/ASTROS/CADDB/QUERY = [engineering.provider.astros.CADDBQuery@87e9ce2, ...
in/value/ASTROS/MAIN/INPUT = http://10.131.4.201:9000/astros/fighter/fighter_flutterM1p1.d
in/value/ASTROS/SCRATCH/DIRECTORY = /Users/raymondkolonay/workspace/iGrid-10/data/astros/provider/scratch/20111104-121730-1336f5ee5e2
out/value/ASTROS/DATABASE/PASSWORD = WING
out/value/ASTROS/OUTPUT/D01FILE = http://10.131.4.201:9000/astros/provider/scratch/20111104-121730-1336f5ee5e2/NLAAN.D01
out/value/ASTROS/OUTPUT/DAMPING = engineering.provider.astros.AstrosFlutterOutput@1acc0e01
out/value/ASTROS/OUTPUT/GPWG = engineering.provider.astros.AstrosGPWG@4aab7165
out/value/ASTROS/OUTPUT/IDXFILE = http://10.131.4.201:9000/astros/provider/scratch/20111104-121730-1336f5ee5e2/NLAAN.IDX
out/value/ASTROS/OUTPUT/OUTFILE = http://10.131.4.201:9000/astros/provider/scratch/20111104-121730-1336f5ee5e2/astrosResults.out
task/provider = ENGINEERING-Astros-RMac@macdna.rb.rad-e.wpafb.af.mil:10.131.4.201
Nov 4, 2011 12:18:19 PM engineering.requestor.astros.AstrosRequestor astrosComputeFlutterDamping

INFO: Flutter Damping :
mode number Velocity Damping
1 5000.0 -0.08188183605670929
1 10000.0 -0.11725959181785583
1 18000.0 -0.19990922510623932
1 20000.0 -0.20551984012126923
1 24000.0 -0.20518758893013
2 5000.0 -0.05098726227879524
2 10000.0 -0.08072353154420853
2 18000.0 0.3754781484603882
2 20000.0 0.629492461681366
2 24000.0 1.1414759159088135
3 5000.0 -0.09921614825725555
3 10000.0 -0.29697275161743164
...

AstrosFlutterOutput object is output and populated with damping data..

List of Available Services

Modules/engineering, modules/engineering/apps

CONMIN
DOT
ANSYS
ASTROS
AVUS
EXCEL
FLOPS
GRIDGEN
MAELSTROM
MDICE
MSTCGEOM
M3CT
NASTRAN
VORTEXLATTICE
MATLAB

NOT UPDATED to IGRID-10 AS OF 7 Nov 2011

SORCER Model & Variables

- ***Models –*** consists of ***Variables, Filters*** and ***Evaluators***.
  - Current available models – ResponseModel(including sensitivities) , ParametricModel,, OptimizationModel
- ***Variables (Var) -*** can be dependent on other *Variables* enabling distributed **functional programming**. Variables can have multiple evaluators enabling **multi-fidelity calculations** for a specific variable's value.
- ***Evaluators –*** are used to determine the value of variables and their partial derivatives(chain rule works) with respect to their dependencies (Variables).
  - Current Evaluator Types – ModelEvaluator, ExertionEvaluator, ExpressionEvaluator, GroovyEvaluator, JepEvaluator, MethodEvaluator
- ***Filters -*** are used to map the results of *Evaluators* to *Variable* Values. Think unix shell piping. Filters can be concatenated *n* times.
  - Current Filter Types – BasicFileFilter, ContextFilter, FileFilter, GrepFilter, ListFilter, MapFilter, ObjetFilter, PatternFilter, TextFilter

All are Objects

SORCER Variables

SORCER Variables are distinguished by four attributes: ***type, kind, valueType,*** and ***mathType***

SORCER Variable ***type*** (only one)

- DESIGN (DEFAULT)
- RESPONSE

SORCER Variable ***kind*** (one or more combinations)

- LINKED
- PARAMETER
- BOUNDED
- RANDOM
- CONSTRAINT
- OBJECTIVE
- INDEPENDENT

SORCER Variable ***ValueType*** (only one)

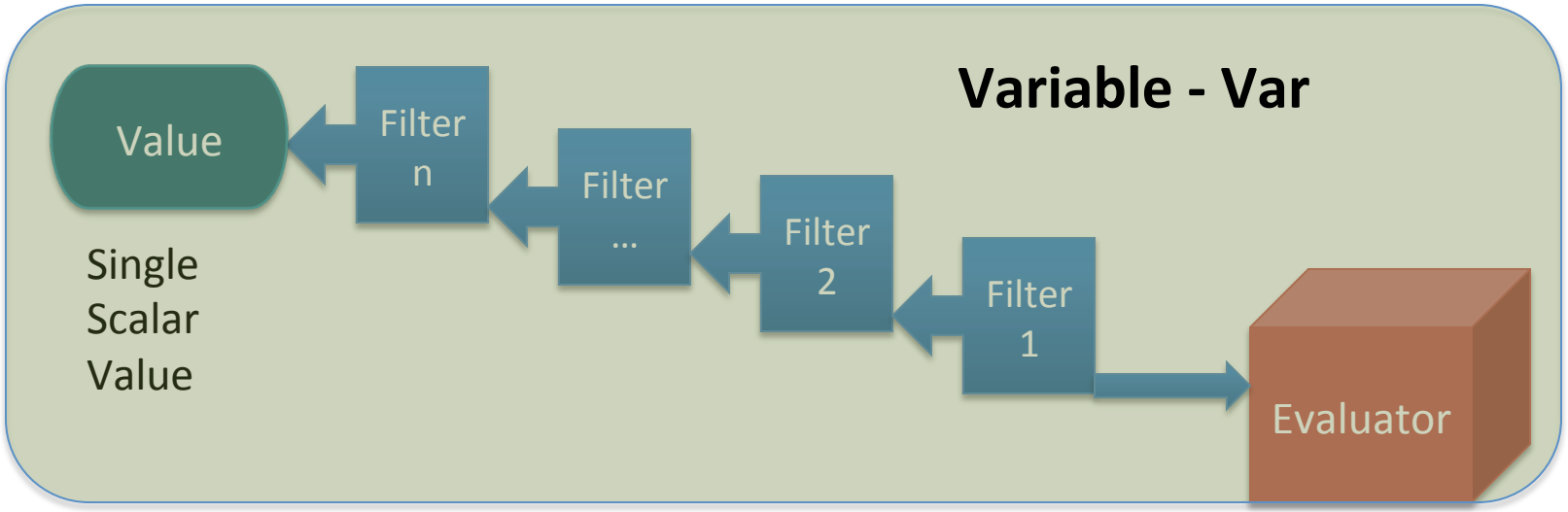
- INTEGER
- LONG
- DOUBLE (DEFAULT)
- FLOAT
- STRING
- OBJECT
- UNKNOWN
- UNDEFINED

SORCER Variable ***mathType*** (one or more combinations)

- CONTINUOUS (DEFAULT)
- DISCRETE
- DISCRETE_WITH_ORDER
- DISCRETE_WITH_MATH
- DISCRETE_NO_ORDER
- PROBLEM_PARAMETER
- REAL

Example: skinThickness: Type:DESIGN, Kinds:BOUNDED,RANDOM, ValueType:DOUBLE, MathType: REAL, CONTINUOUS

Variable - Value, Filters, Evaluators



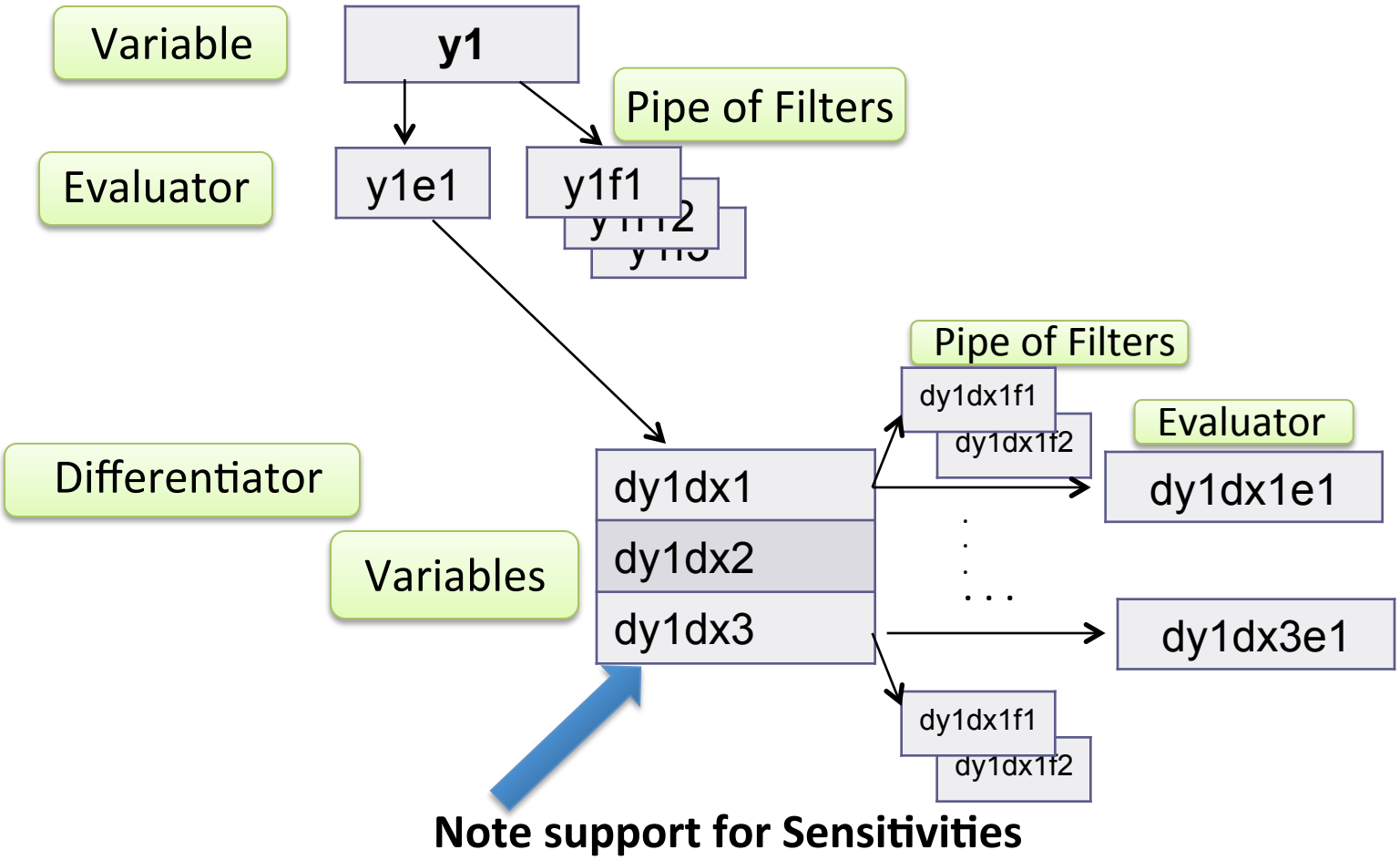
Computational Entity. Creating Large Amounts of Data

Variable, Evaluator, Filter (VEF) Design Patterns

- Lazy variable evaluation
  - getValue vs. evaluate (**demand driven calculations**)
- Evaluator – Decorator pattern
  - Evaluator: getValue/evaluate implements `EvaluationManagement`
  - Outer evaluator calls on its inner one
  - Evaluator is observable and observer (**functional programming**)
- Observer-Observable
  - setChanged
  - notify/update

Basic Variable Structure

$$y_1(x_1, x_2, x_3)$$

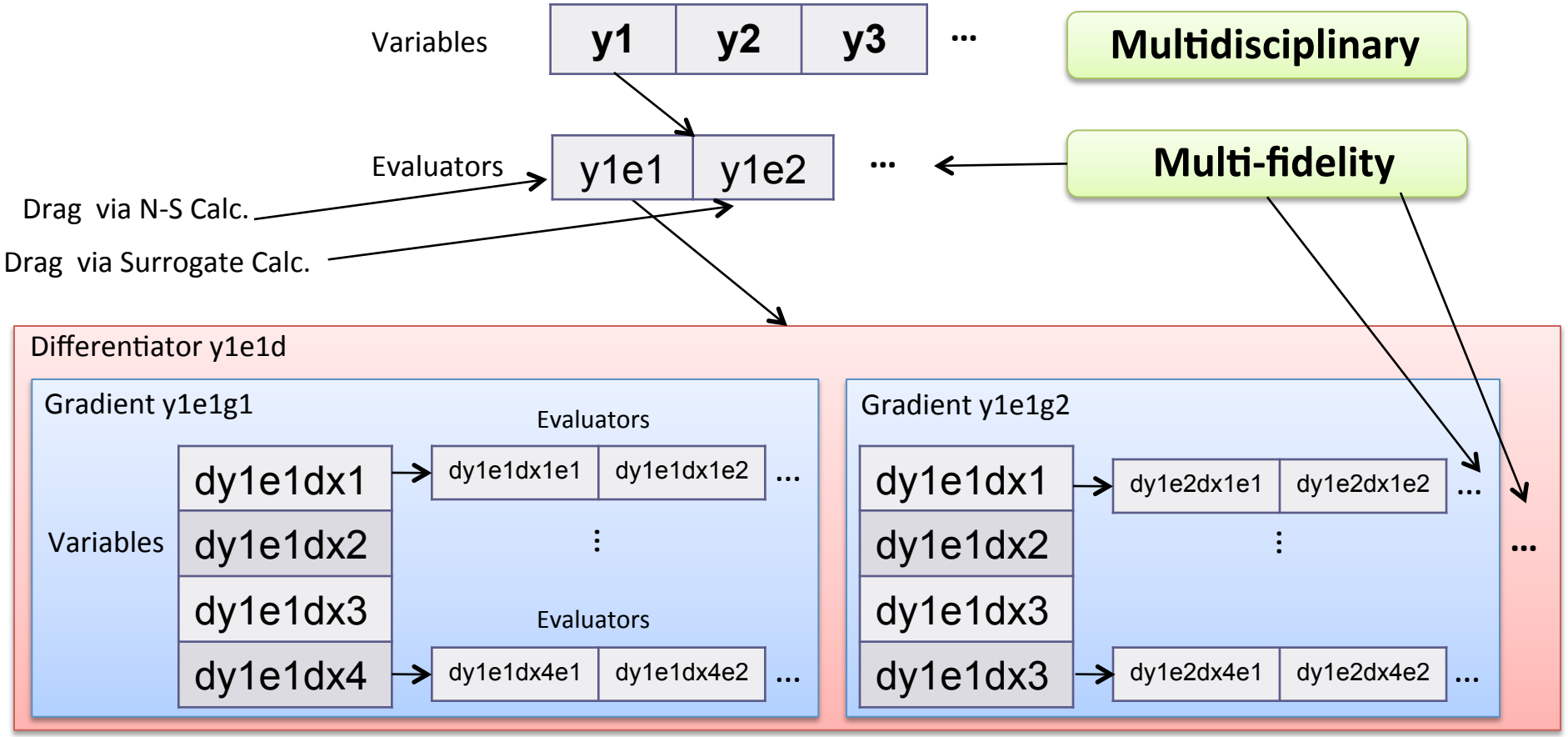


Advanced Variable Structure

SORCER

Functions of Functions

$$y_1(x_1, x_2, x_3), \quad y_2(x_4, x_5, x_6, y_1), \quad y_3(x_6, x_7, x_8, y_2)$$



Evaluators

- ***Evaluators –*** are used to determine the value of variables and their partial derivatives(chain rule works) with respect to their dependencies (Variables). Evaluators do not necessarily produce a scalar value. The result can be an entire database or file. (for a given set of input an application creates many outputs)
  - Current Evaluator Types – ModelEvaluator, ExertionEvaluator, ExpressionEvaluator, GroovyEvaluator, JepEvaluator, MethodEvaluator, SOAEvaluator, FDEvaluator

Filters

- ***Filters -*** are used to map the results of *Evaluators* to *Variable* Values. Think unix shell piping. Filters take the output of Evaluators and reduce it to a single scalar value necessary to define a *Variable* value. Filters can be concatenated *n* times.
  - Current Filter Types – BasicFileFilter, ContextFilter, FileFilter, GrepFilter, ListFilter, MapFilter, ObjetFilter, PatternFilter, TextFilter

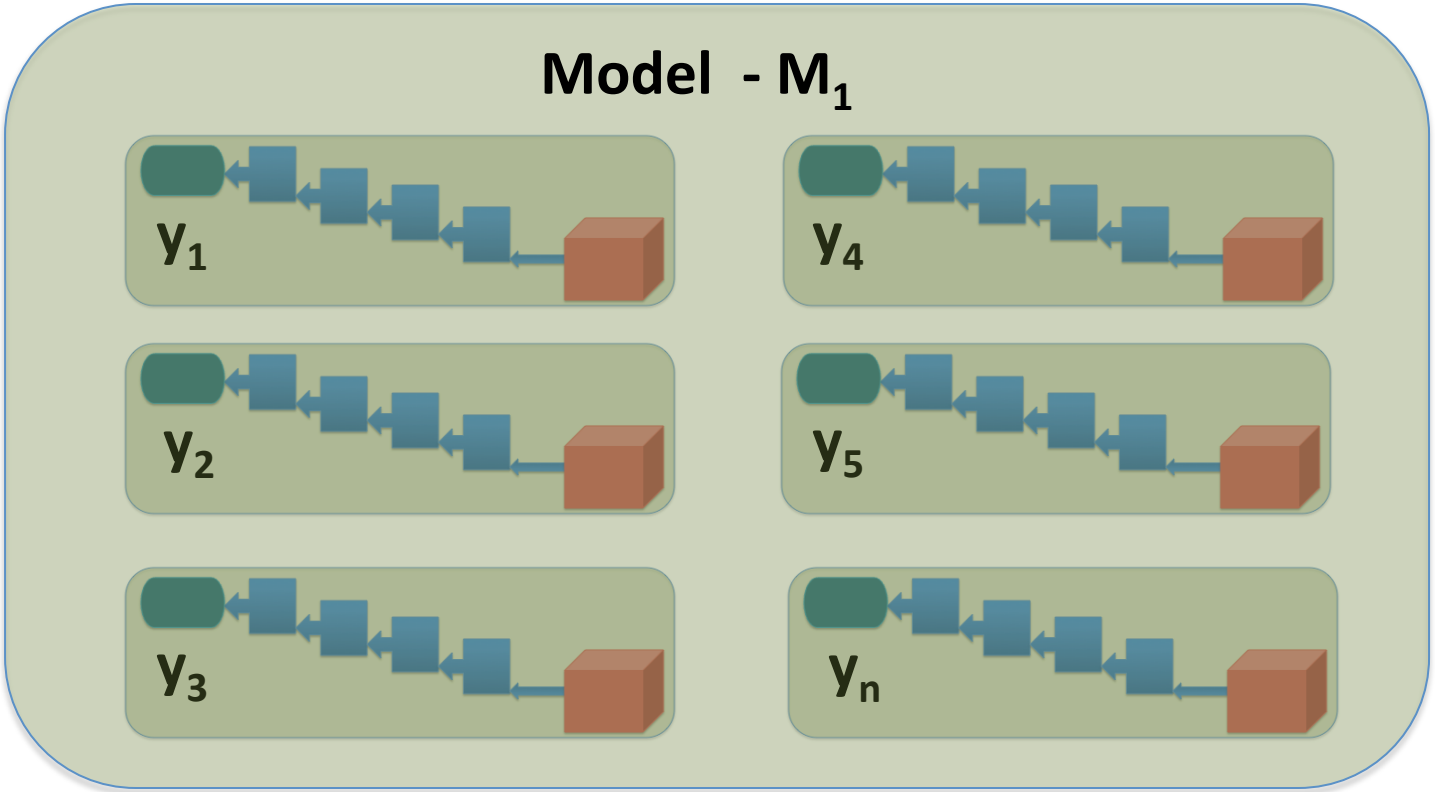
SORCER Models

SORCER Models support

- Analysis – (ResponsModel & ParametricModel)
  - multidisciplinary response analysis
  - multi-fidelity response analysis
  - multi-fidelity response sensitivities

- Design Space Exploration (OptimizationModel)
  - Optimization
  - Multidisciplinary Optimization

Model is a Collection of Variables



Model Construction

- Constructing a Model Consists of Two Steps
  - Model Definition (Skeleton)
    - Defines the model Variables (design, response, parameters, objectives, constraints.
    - Defines the Variable Realizations, Evaluations, and Differentiation
  - Model Configuration (Muscle)
    - Develops the evaluators and filters for all variables and derivative variables.

Models can be created programmatically with or without Operators (Functional Programming)

Model Construction Example
Rosen-Suzuki Functions Definition

Independent Varibles - $\{x_1, x_2, x_3, x_4\}$

Functions

$$f(x_1, x_2, x_3, x_4) = x_1^2 - 5x_1 + x_2^2 - 5x_2 + 2x_3^2 - 21x_3 + x_4^2 + 7x_4 + 50$$

$$g_1(x_1, x_2, x_3, x_4) = x_1^2 + x_1 + x_2^2 - x_2 + x_3^2 + x_3 + x_4^2 - x_4 - 8.0$$

$$g_2(x_1, x_2, x_3, x_4) = x_1^2 - x_1 + 2x_2^2 + x_3^2 + 2x_4^2 - x_4 - 10.0$$

$$g_3(x_1, x_2, x_3, x_4) = 2x_1^2 + 2x_1 + x_2^2 - x_2 + x_3^2 - x_4 - 5.0$$

Model Definition
(using functional programming operators)

`sorcer.rs.ex7.model.RsResponseModelBuilder` examples/ex7

Model Definition (using functional programming operators)

```
// define the model
model = responseModel("Rosen-Suzuki Response Model",
                designVars(vars(loop("i", 1,4),"x$i$")),
                responseVar("f",
                                realization(
                                                evaluation("FExacte", "fe",null,null))),
                responseVars(loop("i", 1, 3),"g$i$",
                                realization(
                                                evaluation("g$i$Exacte", "g$i$e",null,null)))
        );
```

The above creates
ResponseModel: “Rosen-Suzuki Response Model”, which contains -
Vars: with names: “x1”,”x2”,”x3”,”x4”, type: DESIGN
Var: with name: “f”, type: RESPONSE, evaluationName:”Fexacte” , evaluatorName:”fe”
Vars: with names: “g1”,”g2”,”g3”, type: RESPONSE, evaluationName:”g1Exacte”, evaluatorName:”g1e”
evaluationName:”g2Exacte”, evaluatorName:”g2e”
evaluationName:”g3Exacte”, evaluatorName:”g3e”

Rosen-Suzuki Functions

`sorcer.rs.ex7.model.RsResponseModelBuilder` Examples/ex7

Model Configuration (using operators and explicit programming)

```
fe = evaluator("fe","x1^2-5.0*x1+x2^2-5.0*x2+2.0*x3^2-21.0*x3+x4^2+7.0*x4+50.0");
fe.addArgs(model.getDesignVars("x1", "x2", "x3", "x4"));
model.setResponseEvaluator("f", fe);
// an alternative approach would be to do it declaratively
//var(model,"f","fe",evaluator("fe","x1^2-5.0*x1+x2^2-5.0*x2+2.0*x3^2-21.0*x3+x4^2+7.0*x4+50.0"),args("x1", "x2", "x3", "x4"));

Evaluator g1e = evaluator("g1e", "x1^2+x1+x2^2-x2+x3^2+x3+x4^2-x4-8.0");
g1e.addArgs(model.getDesignVars("x1", "x2", "x3", "x4"));
model.setResponseEvaluator("g1", g1e);

Evaluator g2e = evaluator("g2e", "x1^2-x1+2.0*x2^2+x3^2+2.0*x4^2-x4-10.0");
g2e.addArgs(model.getDesignVars("x1", "x2", "x3", "x4"));
model.setResponseEvaluator("g2", g2e);

Evaluator g3e = evaluator("g3e", "2.0*x1^2+2.0*x1+x2^2-x2+x3^2-x4-5.0");
g3e.addArgs(model.getDesignVars("x1", "x2", "x3", "x4"));
model.setResponseEvaluator("g3", g3e);
```

Steps for Configuration: 1. Create Evaluator, 2. add Arguments/Dependencies to Evaluator, 3. Create Filter(s) if required, 4. Associate Evaluator with Var

Rosen-Suzuki Functions

Model Use (using the methods in an instantiated class)

```
// you can set the value of the independent variables
rm.setDesignVarValues(set("x1", 2.0), set("x2", 3.0), set("x3", 4.0), set("x4", 200.0));

// get the values of the responses for the new x values
rm.getRepsonses();
Data Table [x1,x2,x3,x4,  f, g1,g2,g3]
           [2.0, 3.0, 4.0, 5.0,  46.,  44.0, 71.0, 24.0]

// additional examples of using the api
Var x4 = rm.getDesignVar("x4");
logger.info("\n\nx4 value: " + x4.getValue());
x4 value: 5.0

// examine the responseVar f
Var f = rm.getResponseVar("f");
ExpressionEvaluator ee = (ExpressionEvaluator) f.getEvaluator();
logger.info("\n\nexpression:\n" + ee.getName() + "/" + ee.getExpression());
fe/x1^2-5.0*x1+x2^2-5.0*x2+2.0*x3^2-21.0*x3+x4^2+7.0*x4+50.0
logger.info("\n\nresponse:\n" + ee.getName() + "/" + f.getValue());
fe/46.0
```

ModelContext

- The ***ModelContext*** is used to query the model, update the model or configure the model.
- If you want the values for variables or their derivatives. You use the ***ModelContext*** methods to set the appropriate path with a list of variables you desire values for. For example if you want a select set of response values from the model you would use
  - ***modelContext.setResponsesInfo(VarInfoList)***
  - where ***VarInfoList*** is a list of ***VarInfo*** variables that you desire values for. If you set ***VarInfoList = null***. The model will return the values for all ***ResponseVars*** in the model. This was demonstrated in the previous slide.

Rosen-Suzuki Function Optimization

Explorer Execution

Exact Function Optimization

```
outContext = (ExploreContext)explorer.explore(exploreContext);

********* CONMIN STATE **************

CONMIN Iteration # =  29

Objective Fuction fo = 6.00260795945281

Design Variable Values
   x1 = 2.5802964087086235E-4 x2 = 0.9995594642481355 x3 = 2.000313835134211 x4 = -0.9986692050113675

Constraint Values
   g1c = -0.002603585246998996 g2c = -1.0074147118087602 g3c = 4.948009193483927E-7
T
Termination Criterion
    ABS(OBJ(I)-OBJ(I-1))   LESS THAN DABFUN = 5.0E-5 FOR 3 ITERATIONS

Evaluation Statistics
    Number of Objective Evaluations = 88
    Number of Constraint Evaluations = 88
    Number of Objective Gradient Evaluations = 29
     Number of Constraint Gradient Evaluations = 29
```

Analytic Optimum Results
fo = 6.0
X1=0.0, x2=1.0, x3=2.0, x4=-1.0
g1c=0.0, g2c=-1.0, g3c=0.0

MSTC SORCER Engineering Applications to Date

- Hi-Fidelity Aeroelastic Analysis
  - Euler (AVUS), FEM (MSCNASTRAN), MDICE
- Hi-Fidelity Induced Drag minimization
  - Euler (AVUS), Optimization (CONMIN, MATLAB)
- 2-D large amplitude airfoil motion trajectory Optimization
  - Unsteady Vortex Lattice Method (in-house UVLM), Opti (DOT)
- Shape and sizing aeroelastic optimization
  - Euler (AVUS), FEM (MSCNASTRAN), MDICE, Opti (DOT) FD sensitivities

- Demonstrated tight integration with C, C++, and FORTRAN using JNA (can make direct calls to any shared object)
- Network configuration
  - Linux workstations
  - Linux cluster(2), Mac cluster(1)
  - SGI Irix
  - Windows
  - Mac Desktop
  - Mac laptop

Shape and sizing aeroelastic optimization

Minimize
Objective Function

$$f(\beta_i) = W_{Structural}$$

Subject to

Inequality Constraints

$$g_1(\beta_i) = 1.0 - \frac{C_{M_{roll}}}{C_{M_{roll},ref}}$$

$$g_j(\beta_i) = 1.0 - \frac{\sigma_{VonMises}}{\sigma_{MaxAllowable}} \leq 0 \qquad j = 2, N_{Constr.}$$

Side Constraints

$$\beta_i^L \leq \beta_i \leq \beta_i^U \qquad i = 1, N_{DesignVariables}$$

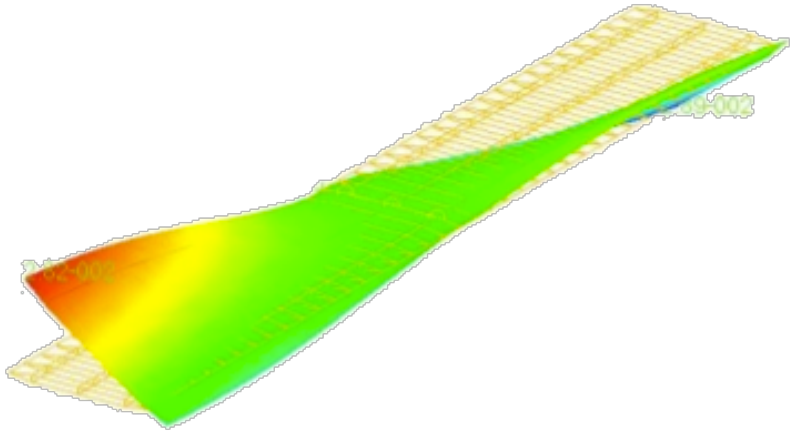
β_i - Structural sizes, and planform sweep

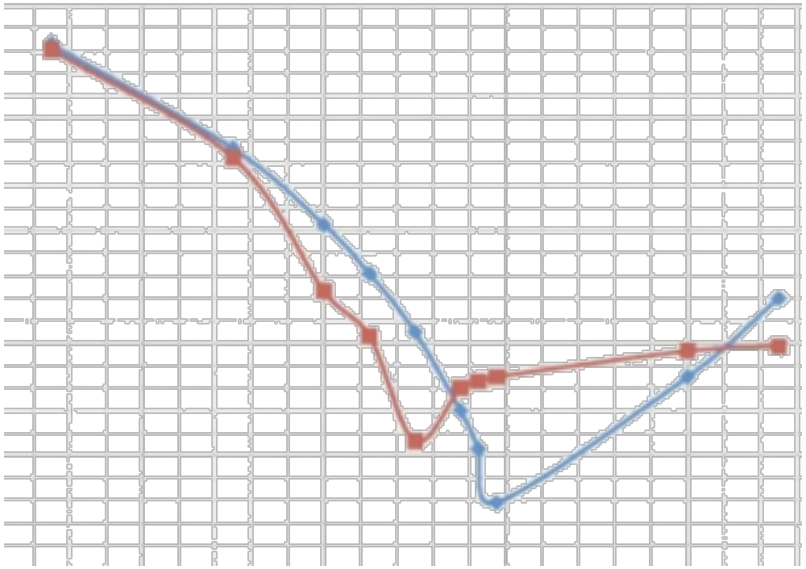
Ernest Thompson – AFRL/RBSD

Min weight shape & sizing opti

SORCER

- Euler Static Aeroelastic Calculations





Ernest Thompson – AFRL/RBSD

Sorcer network configuration for min weight (shape and sizing)

SORCER

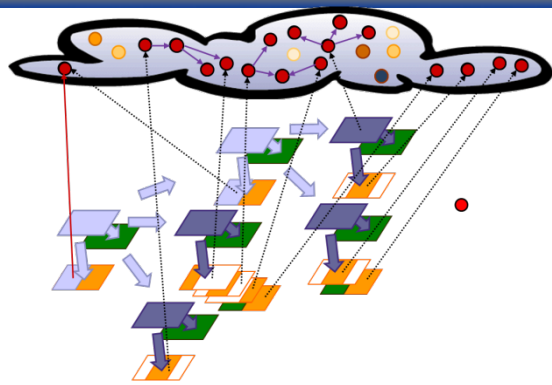
Coalescence

MDICE

Avus

RBSDZ0010143

Microsoft Windows



NMEDA

MDICE

Avus

Whittle

Gridgen

DOT

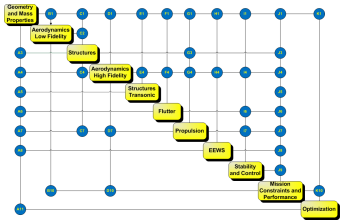
NASTRAN

Gladius

StructureTransform

NASTRAN

Client



Xserv2

MDICE

Avus

Gothmog

Java

JavaSpace

Ernest Thompson – AFRL/RBSD

Concluding Remarks

- MSTC is responsible for executing large scale system level MDA/MDO in a heterogenous computing environment
- SORCER exposes Sun's Jini connection Technology to create a service oriented programming and computing environment for large scale MDA/MDO
- SORCER Functionality
  - Create providers and expose them as jini services.
  - Exertion Oriented Programming – Ability to orchastrate Services
  - Models for analysis, sensitvitities, and design
  - VEF – Functional demand driven computations
  - Space Computing

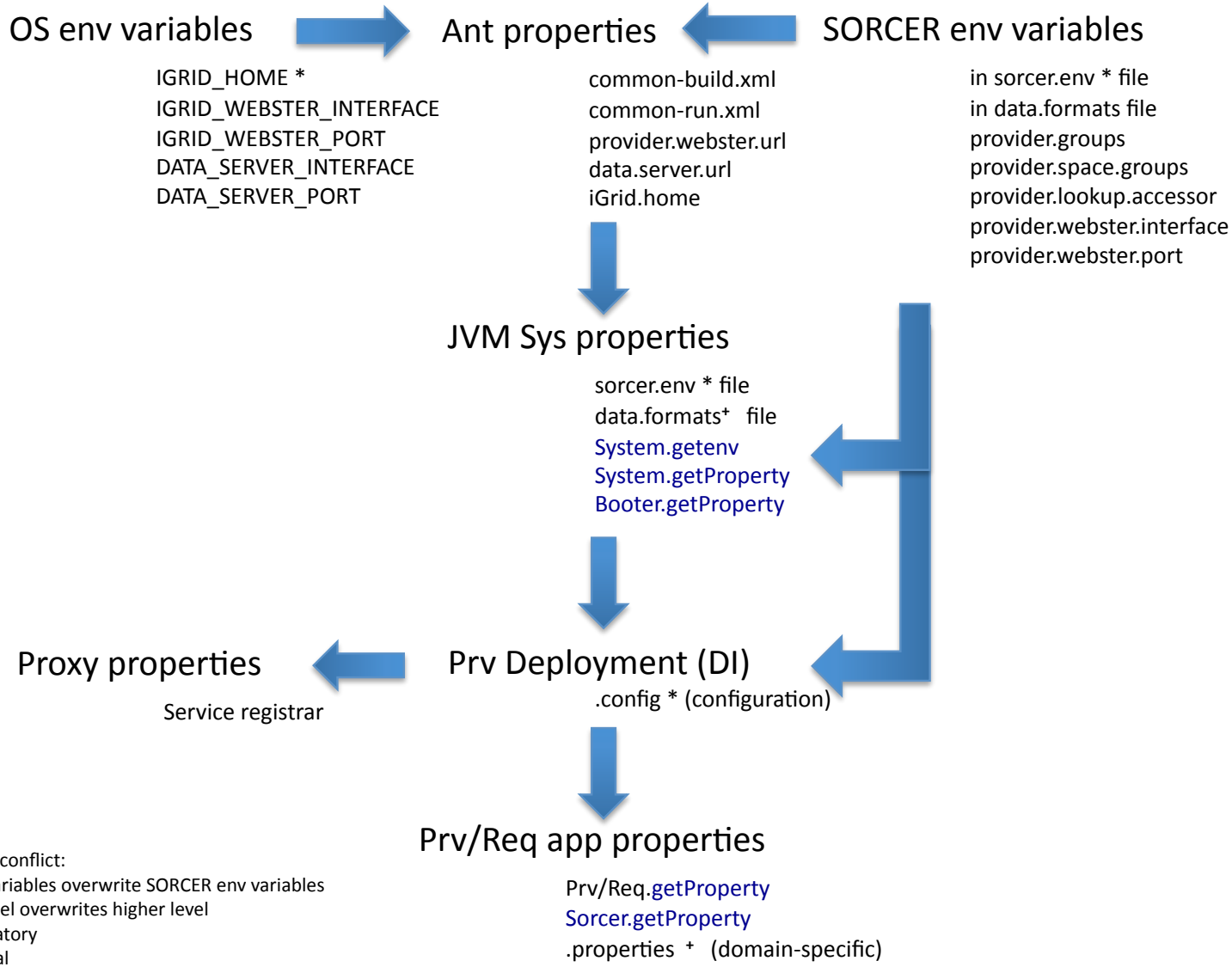
Appendix A

Configuring SORCER Env and Properties

Configuring SORCER Env and Providers

- Properties in SORCER
- OS Environment Variables
- SORCER Environment Variables
- SORCER Environment Properties File: sorcer.env
- Ant XML Scripts
- Provider Properties
- Reading Properties
- Scratch Directory
- Recommendations

Properties in SORCER



In the case of conflict:

1. OS env variables overwrite SORCER env variables
2. Lower level overwrites higher level
3. * - mandatory
4. + - optional

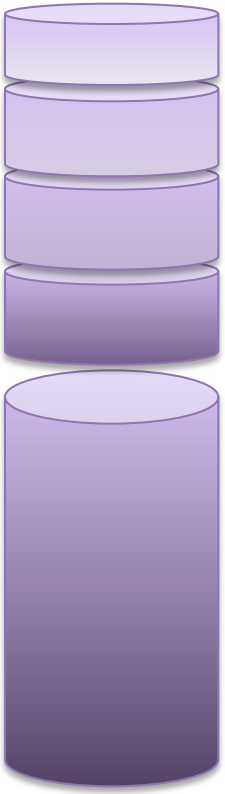
OS Environment Variables

- IGRID_HOME (required)
  the directory of your iGrid project
- IGRID_WEBSTER_INTERFACE and
  IGRID_WEBSTER_PORT (optional)
  hostname and port of your *code server* (webster)
- DATA_SERVER_INTERFACE and
  DATA_SERVER_PORT
  (optional)
- You can specify your webster in the sorcer.env file
- OS webster specification overrides sorcer.env

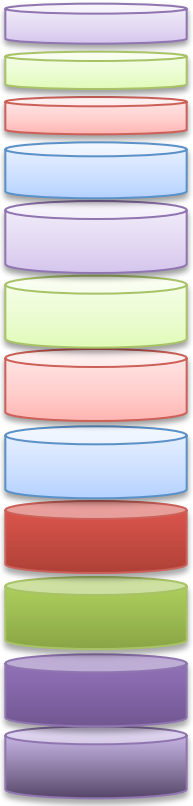
Shrinking Programs

One Large Executable

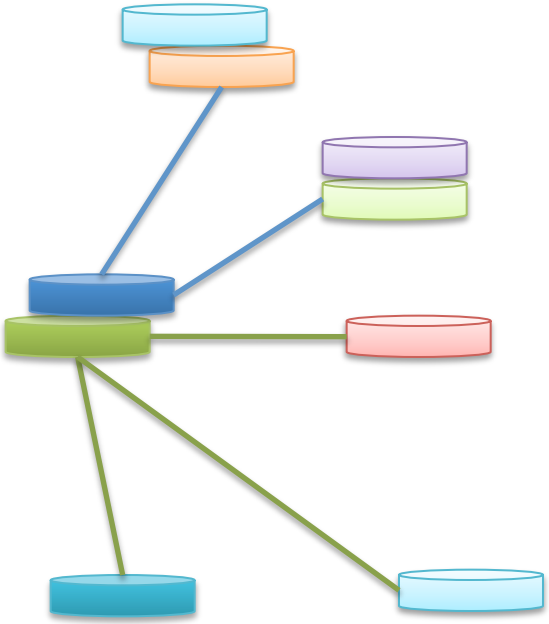
Shared Libraries



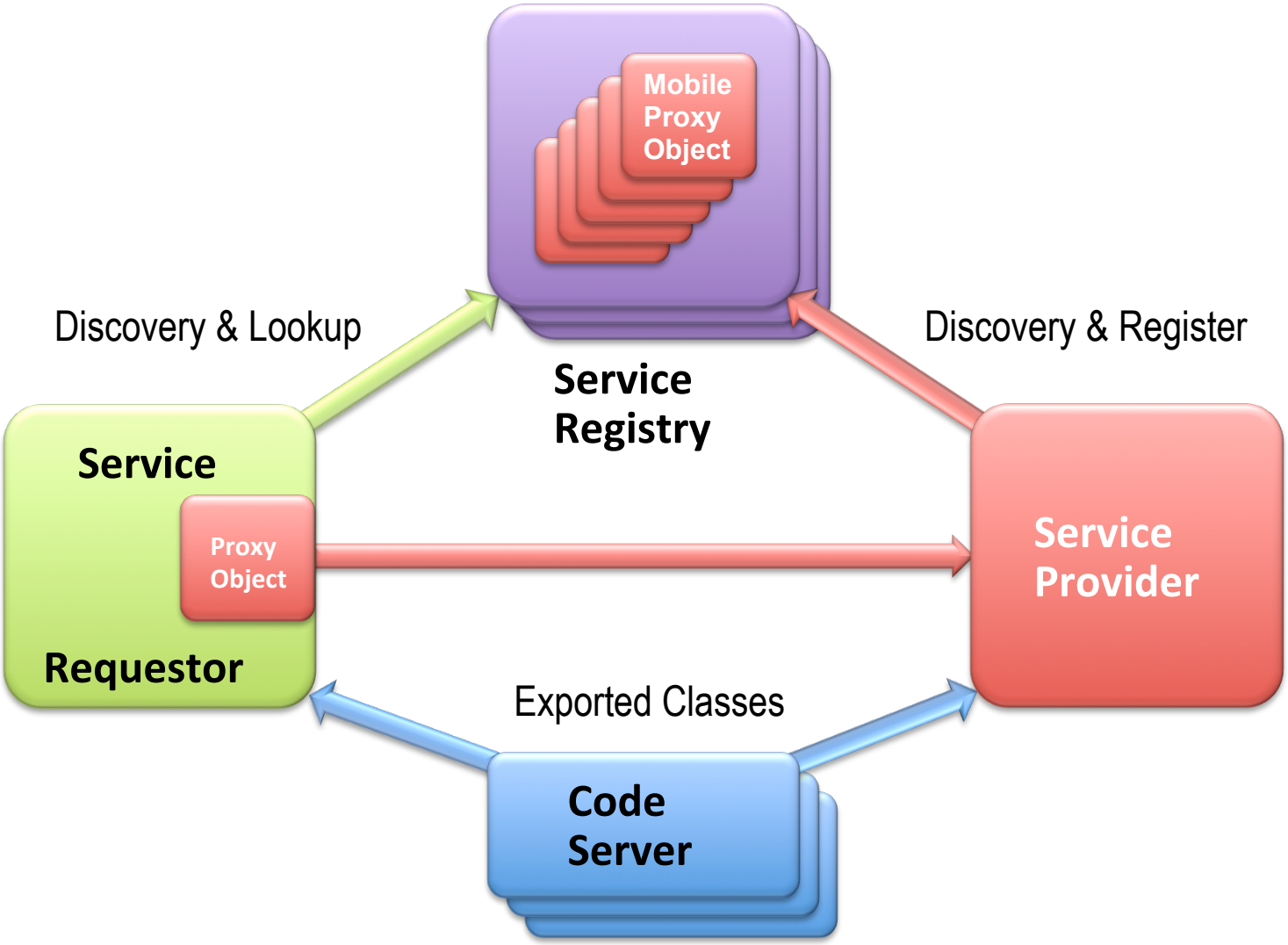
Shared Classes



Mobile Code



Three Neutralities & BT



SORCER Environment Properties

- SORCER environment variables specified in the file sorcer.env
- The full set of these variables is given in configs/templates/sorcer-all.env,
- In the beginning you can start with default values and set only those listed in configs/templates/sorcer.env
- Sorcer.getProperty(String property);

SORCER Environment Properties File
sorcer.env


```
# SORCER environment properties

# The OS environment variable IGRID_HOME must be set

# Groups to register/lookup
provider.groups=sorcer.TEST
provider.space.group=sorcer.TEST
#provider.space.name=JavaSpace
#provider.space.name=Blitz JavaSpace
#provider.worker.transactional=true
#worker.transactiona.lease.time=50000

# Service discovery/lookup
# comma separated URLs
#provider.lookup.locators=jini://localhost
#   multicast and unicast discovery
provider.lookup.accessor=sorcer.util.ProviderAccessor
#   unicast or mixed discovery
#provider.lookup.accessor=sorcer.util.ProviderLocator
#   multicast  only
#provider.lookup.accessor=sorcer.util.ProviderLookup
#   unicast or mixed discovery with QoS capabilities
#provider.lookup.accessor=sorcer.util.QosProviderAccessor
```

SORCER Environment Properties File
sorcer.env - continuation


```
# Code server configuration
provider.webster.interface=127.0.0.1
provider.webster.port=8000

# Data/file repository configuration
# Scratch directory format:
# ${data.root.dir}/${provider.data.dir}/${provider.scratch.dir}
# ${data.root.dir}/${requestor.data.dir}/${provider.scratch.dir}
# You can overwrite these properties in your provider properties
data.root.dir=${iGrid.home}/data
provider.data.dir=provider
requestor.data.dir=requestor
provider.scratch.dir=scratch
data.server.interface=${provider.webster.interface}
data.server.port=${provider.webster.port}
```

Ant XML Scripts

- Scripts use:
  - Local properties and inherited properties from imported `common-run.xml` and `common-build.xml` files
  - OS environment and SORCER environment properties from `sorcer.env` as JVM system properties
  - the command line args to the configurations files on provider/requestor startup

In examples/ex2/bin/worker1-prv-run.xml


```
<project name="Run Worker Provider" default="run.provider" basedir=".">

<!-- system environment variables -->
<property environment="env" />
<import file="${env.IGRID_HOME}/modules/common-run.xml" />

<echo message="IGRID_HOME: ${iGrid.home}" />
<echo message="WEBSTER: ${provider.webster.url}" />

<!-- set property values here -->
<property name="provider.name" value="worker" />
<property name="provider.class" value="sorcer.ex2.provider.WorkerProvider" />
<property name="webster" value="${provider.webster.url}" />

<!-- provider classpath -->
<path id="project.classpath">
<pathelement location="${sorcer.lib}/${provider.name}.jar" />
<pathelement location="${sorcer.lib}/sorcer-prv.jar" />
<pathelement location="${sorcer.lib}/sorcer-lib.jar" />
<pathelement location="${jini.lib}/jsk-lib.jar" />
<pathelement location="${jini.lib}/serviceui.jar" />
<pathelement location="${lib}/rio/rio.jar" />
</path>

<!-- provider codeabse jars -->
<property name="j1" value="${webster}/${provider.name}-dl.jar" />
<property name="j2" value="${webster}/sorcer-prv-dl.jar" />
<property name="j3" value="${webster}/jsk-dl.jar" />

…
</project>
```

In examples/ex2/bin/worker1-prv-run.xml

```
<target name="run.provider">
   <java jar="${jini.lib}/start.jar" fork="yes">
      <sysproperty key="java.security.manager" value="" />
      <sysproperty key="java.util.logging.config.file"
         value="${iGrid.home}/configs/sorcer.logging" />
      <sysproperty key="java.security.policy"
         value="../policy/${provider.name}-prv.policy" />
      <sysproperty key="sorcer.provider.codebase" value="${j1} ${j2} ${j3}" />
      <sysproperty key="sorcer.provider.classpath"
         value="${toString:project.classpath}" />
      <sysproperty key="sorcer.provider.impl" value="${provider.class}" />
      <sysproperty key="sorcer.provider.config"
         value="../configs/${provider.name}1-prv.config" />
      <sysproperty key="sorcer.env.file"
         value="${iGrid.home}/configs/sorcer.env" />
      <sysproperty key="iGrid.home" value="${iGrid.home}" />
      <arg value="${iGrid.home}/configs/startup-prv.config" />
      </java>
</target>

</project>
```

In examples/ex2/configs/worker1-prv.config


```
* A SORCER Provider Jini configuration for the WhoIsIt Provider example.
*/
import net.jini.jeri.*;
import net.jini.jeri.tcp.*;
import sorcer.core.*;
import net.jini.core.entry.Entry;
import net.jini.lookup.entry.*;
import sorcer.ex1.provider.*;

sorcer.core.provider.ServiceProvider {
   name="Worker1";
   publishedInterfaces = new Class[] { sorcer.ex2.provider.Worker.class };
   description = "SORCER Worker provider";
   location = "AFRL/WPAFB";
   entries = new Entry[] {
           new Comment("Implements Worker interface with three operations"),
           new Location("1", "218", "146-AFRL") };

   exporter = new BasicJeriExporter(TcpServerEndpoint.getInstance(0), new BasicILFactory());

   //application specific properties
   properties="../configs/worker-prv.properties";
}
```

Provider Properties (application specific)

- Provider loads application-specific properties defined in its local properties file.

- The location and name of application-specific properties file is given by an entry property :
  `properties="../configs/worker-prv.properties";`
  in your provider's component
  `sorcer.core.provider.ServiceProvider`
  in the corresponding provider's *.config file.

```
# Worker Provider properties
//delayed execution time of the Worker.doIt operation
provider.sleep.time=1000
```

Location of sorcer.env and data.formats

- The location of sorcer.env and data.formats files can be specified by the JVM system properties sorcer.env.file and sorcer.formats.file correspondingly

- Default loaction is ${iGrid.home}/configs

Reading Properties

- System.getenv
- System.getProperty
- Booter.getProperty
- Sorcer.getProperty
- myProvider.getConfiguration():Configuration
- myProvider.getProperty
- myRequestor.getProperty

Scratch Directory

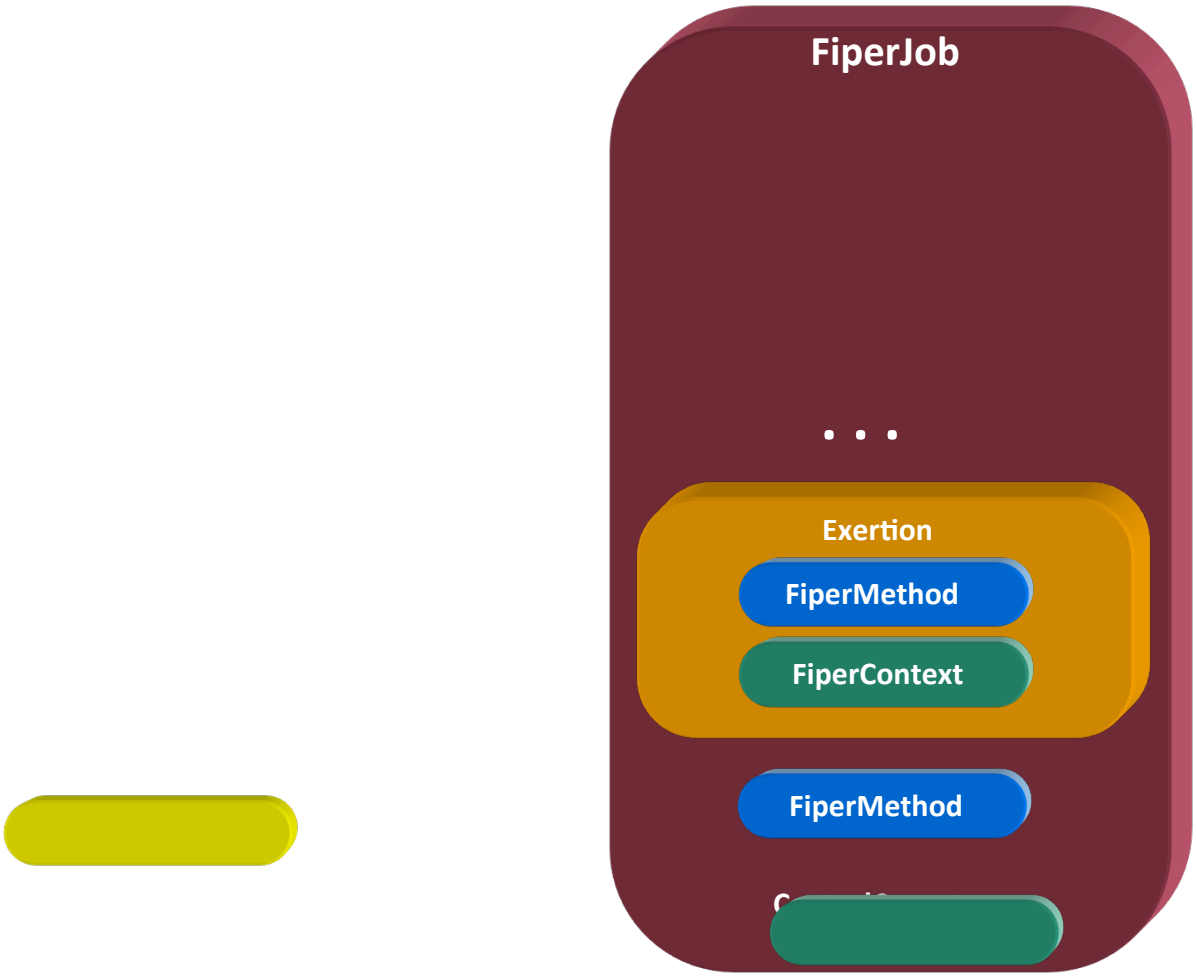
- myProvider.getScratchDirectory() : File
- myProvider.removeScratchDirectory(File) : boolean
- myProvider.getScratchURL(File) : URL

Based on properties:

```
# Data/file repository configuration
# Scratch directory format:
# ${data.root.dir}/${provider.data.dir}/${provider.scratch.dir}
# ${data.root.dir}/${requestor.data.dir}/${provider.scratch.dir}
# You can overwrite these properties in your provider properties
data.root.dir=${iGrid.home}/data
provider.data.dir=provider
requestor.data.dir=requestor
provider.scratch.dir=scratch
data.server.interface=${provider.webster.interface}
data.server.port=${provider.webster.port}
```

Recommendations

- For shared SORCER environment properties use `${iGrid.home}/configs/sorcer.env` copied from `${iGrid.home}/configs/templates`
- Alternatively you can copy `sorcer.env` to your provider `configs` directory and indicate it by a JVM system property `sorcer.env.file`
- The above applies to your `sorcer.formats` as well (`sorcer.formats.file`)
- For your provider configuration use template files in `${iGrid.home}/modules/examples/ex2` for the application `worker1` (in `bin`, `configs`, `policy`, and `*bulild.xml`)
- Keep all applications-specific properties in your provider/requestor properties file specified as a system property or passed as the argument to JVM
- Your scratch directory can be specified in `sorcer.env` or customized in your provider properties file with the same scratch directory properties. Local scratch specification takes precedence over `sorcer.env` specification
- Full set of the SORCER environment properties and provider configuration in `${iGrid.home}/configs/sorcer-all.env` and `${iGrid.home}/configs/provider-all-prv.config`


FiperJob
. . .
Exertion
FiperMethod
FiperContext
FiperMethod