# iGrid-10 DOT Implementation: Demonstration and Progress Report



Dean Bryson

14 Jun 2011 MSTC SORCER Group Meeting

# Outline



- Status update
- Simplistic but realistic example problem
  - Problem definition
  - Setting up and publishing a SORCER model
  - Publishing an explorer, dispatcher, and optimizer
  - Requesting a single optimization
- Behind the scenes of an optimization request
- Requesting multiple optimizations
- Real engineering problems

# Outline



- Status update
- Simplistic but realistic example problem
  - Problem definition
  - Setting up and publishing a SORCER model
  - Requesting a single optimization
- Behind the scenes of an optimization request
- Requesting multiple optimizations
- Real engineering problems

# Requirements Review (1 of 3)



- ✓ Utilize the DOT library
- ✓ Support iGrid-10 model-based paradigm
- ✓ Minimize a scalar-valued function
- ✓ Maximize a scalar-valued function
- ✓ Enforce scalar-valued inequality constraints
- (✓) Enforce scalar-valued equality constraints
- (✓) Perform unconstrained optimization
- ✓ Accept user configuration via ASCII file
- ✓ Accept user configuration via API
- ✓ Default to standard configuration in absence of user configuration

**(✓) Assumed based on normal DOT usage**

# Requirements Review (2 of 3)



- ✓ Return the identities of the objective and constraints to be evaluated
- ✓ Return the identities of gradients to be evaluated
- ✓ Return values of design variables at which to evaluate
- ✓ Accept values of objective and constraints requested
- ✓ Accept values of gradients requested
- ✘ Access run state via API
- ✘ Provide feedback mechanism to allow progress reporting (interface callback/Job Monitor tie-in/file URL)
- ✘ Report to a comma-delimited file
- ✘ Report to a Tecplot formatted file
- ✘ Report to a gnuplot formatted file
- ✘ Allow user selection of no, one, or multiple file formats

# Requirements Review (3 of 3)



- ✘ Return DOT library output files in entirety
- ✘ Organize output files in uniquely identifiable directory
- ✘ Report the optimized design back to the requestor
- ✓ Perform multiple optimizations in series without requiring service republishing
- ✘ Optionally provide diagnostic information
- ✘ Halt the optimization process if notification is received from an analysis provider that a design is inadmissible

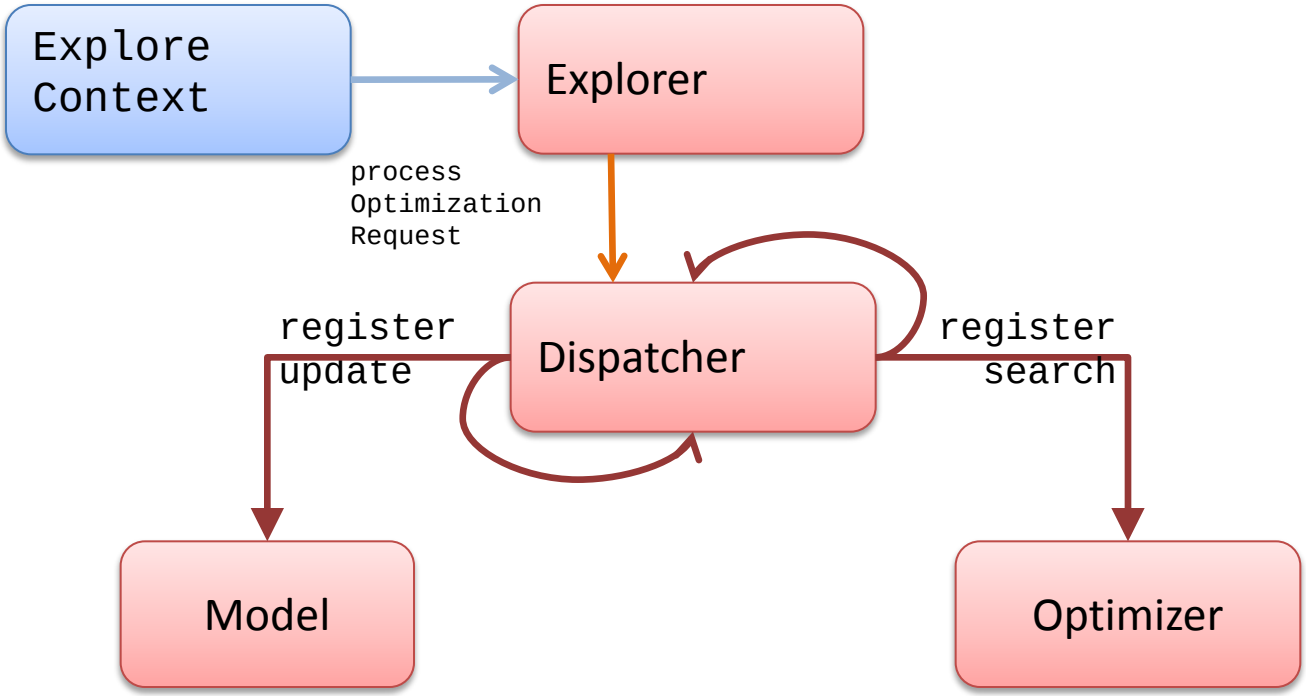

# Outline

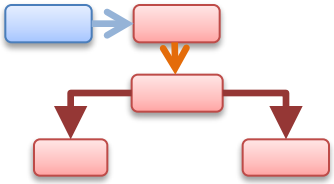



- Status update
- Simplistic but realistic example problem
  - Problem definition
  - Setting up and publishing a SORCER model
  - Publishing an explorer, dispatcher, and optimizer
  - Requesting a single optimization
- Behind the scenes of an optimization request
- Requesting multiple optimizations
- Real engineering problems

# Event-Driven Optimization: A Reminder





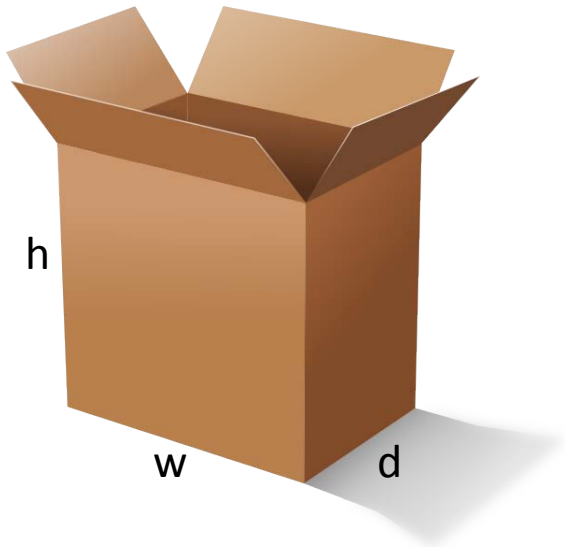



Reference SORCER documentation for details

# The Box Problem



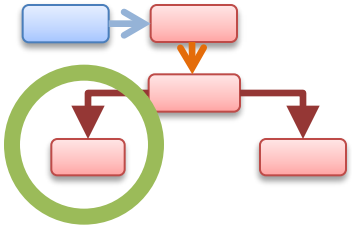
- Minimize material (surface area)
  - f = 2*(w*h + d*h + 2*w*d)
- Subject to 2 cu ft min volume
  - g = 2 – h*w*d ≤ 0
- Optimum:
  - h = 2, w = 1, d = 1
  - f = 12
  - g = 0



# Creating a SORCER Model: BoxModelBuilder Class



```
OptimizationModel createModel() throws ContextException, RemoteException {
   OptimizationModel om = optimizationModel(
         "Box Model",
         designVars(var("d", 1.0, 0.0, 5.0),
                  var("h", 2.0, 0.0, 5.0),
                  var("w", 1.0, 0.0, 5.0)),
         responseVars("f"),
         responseVars("g"),
         objectiveVars(var("area", "f", Target.min)),
         constraintVars(var("vol", "g", Relation.lte, 0.0)));

   om = configureAnalysisModel(om);
   om = configureSensitivityModel(om);
   om.initialize();
   return om;
}
```



…/dot/requestor/src/engineering/optimization/
dot/requestor/BoxModelBuilder.java

# Creating a SORCER Model: BoxModelBuilder Class

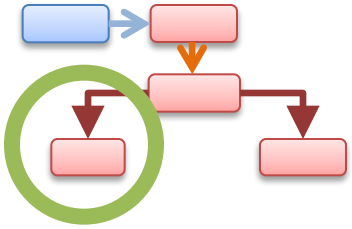


```
OptimizationModel createModel() throws ContextException {
    OptimizationModel om = optimizationModel(
        "Box Model",
        designVars(var("d", 1.0, 0.0, 5.0),
                   var("h", 2.0, 0.0, 5.0),
                   var("w", 1.0, 0.0, 5.0)),
        responseVars("f"),
        responseVars("g"),
        objectiveVars(var("area", "f", Target.min)),
        constraintVars(var("vol", "g", Relation.lte, 0.0));

    om = configureAnalysisModel(om);
    om = configureSensitivityModel(om);
    om.initialize();
    return om;
}
```

User calls this method to create a new model

Declare design variables with initial values (to be overwritten) and bounds

Declare response variables



…/dot/requestor/src/engineering/optimization/
dot/requestor/BoxModelBuilder.java

# Creating a SORCER Model: BoxModelBuilder Class

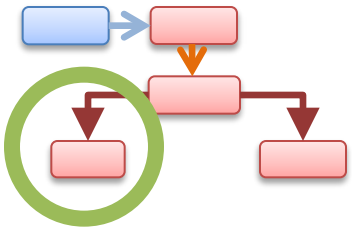


```
OptimizationModel createModel() throws Co                    teException {
    OptimizationModel om = optimizationMo
          "Box Model",
          designVars(var("d", 1.0, 0.0, 5.
                  var("h", 2.0, 0.0, 5.0),
                  var("w", 1.0, 0.0, 5.0)),
          responseVars("f"),
          responseVars("g"),
          objectiveVars(var("area", "f", Target.min)),
          constraintVars(var("vol", "g", Relation.lte, 0.0));

    om = configureAnalysisModel(om);
    om = configureSensitivityModel(om);
    om.initialize();
    return om;
}
```

Associate f with an objective to be minimized named area

Associate g with a constraint less than or equal to 0 named vol

Add evaluators to response variables

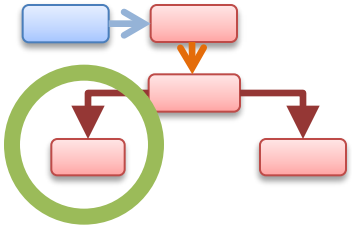


…/dot/requestor/src/engineering/optimization/dot/requestor/BoxModelBuilder.java

# Creating a SORCER Model: BoxModelBuilder Class



```
OptimizationModel configureAnalysisModel(OptimizationModel om)
    throws ContextException, EvaluationException {

    Evaluator evalF = evaluator("evalF", "2.0*(w*h + d*h + 2.0*w*d)");
    om.setResponseEvaluator("f", evalF);
    evalF.addArgs(om.getDesignVars("d", "h", "w"));

    Evaluator evalG = evaluator("evalG", "2.0 - h*w*d");
    om.setResponseEvaluator("g", evalG);
    evalG.addArgs(om.getDesignVars("d", "h", "w"));

    return om;
}
```



…/dot/requestor/src/engineering/optimization/
dot/requestor/BoxModelBuilder.java

# Creating a SORCER Model: BoxModelBuilder Class

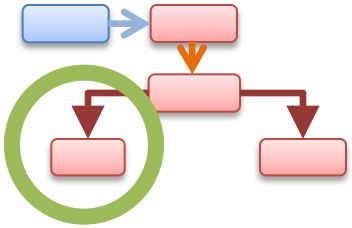


```
OptimizationModel configureAnalysisModel(O
    throws ContextException, EvaluationExce

    Evaluator evalF = evaluator("evalF", "2.0*(w*h + d*h + 2.0*w*d)");
    om.setResponseEvaluator("f", evalF);
    evalF.addArgs(om.getDesignVars("d", "h", "w"));

    Evaluator evalG = evaluator("evalG", "2.0 - h*w*d");
    om.setResponseEvaluator("g", evalG);
    evalG.addArgs(om.getDesignVars("d", "h", "w"));

    return om;
}
```

Create an evaluator named evalF with an expression

Associate evaluator with response variable

evalF is a function of d, h, and w



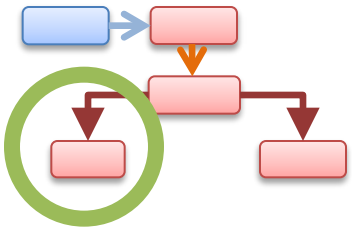
…/dot/requestor/src/engineering/optimization/
dot/requestor/BoxModelBuilder.java

# Creating a SORCER Model: BoxModelBuilder Class



```
OptimizationModel createModel() throws ContextException, RemoteException {
    OptimizationModel om = optimizationModel(
          "Box Model",
          designVars(var("d", 1.0, 0.0, 5.0),
                  var("h", 2.0, 0.0, 5.0),
                  var("w", 1.0, 0.0, 5.0)),
          responseVars("f"),
          responseVars("g"),
          objectiveVars(var("area", "f", Target.min)),
          constraintVars(var("vol", "g", Relation.lte, 0.0)));

    om = configureAnalysisModel(om);
    om = configureSensitivityModel(om);
    om.initialize();
    return om;
}
```

Add sensitivity evaluators to response variables

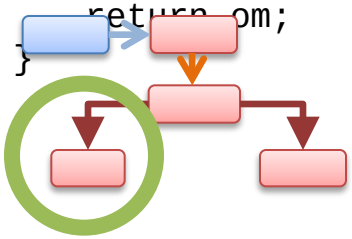


…/dot/requestor/src/engineering/optimization/
dot/requestor/BoxModelBuilder.java

# Creating a SORCER Model: BoxModelBuilder Class



```
OptimizationModel configureSensitivityModel(OptimizationModel om)
    throws ContextException, EvaluationException {

    Evaluator evalDFDH = evaluator("evalDFDH", "2.0*(w+d)",
          args(om.getDesignVars("w“, "d")));
    Evaluator evalDFDW = evaluator("evalDFDW", "2.0*(h+2.0*d)",
          args(om.getDesignVars("h", "d")));
    Evaluator evalDFDD = evaluator("evalDFDD", "2.0*(h+2.0*w)",
          args(om.getDesignVars("w", "h")));
    List<Evaluator> evalGradF = list(evalDFDD, evalDFDH, evalDFDW);
    om.setGradientEvaluators("f", "evalF", "gradF", evalGradF);

    Evaluator evalDGDH = evaluator("evalDGDH", "-d*w",
          args(om.getDesignVars("w", "d")));
    Evaluator evalDGDW = evaluator("evalDGDW", "-h*d“,
          args(om.getDesignVars("h", "d")));
    Evaluator evalDGDD = evaluator("evalDGDD", "-h*w“,
          args(om.getDesignVars("w", "h")));
    List<Evaluator> evalGradG = list(evalDGDD, evalDGDH, evalDGDW);
    om.setGradientEvaluators("g", "evalG", "gradG", evalGradG);

    return om;
}
```



…/dot/requestor/src/engineering/optimization/
dot/requestor/BoxModelBuilder.java

# Creating a SORCER Model: BoxModelBuilder Class

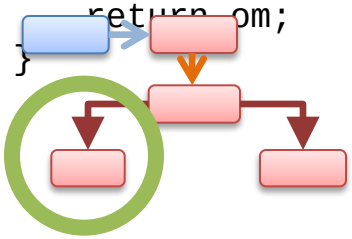


```
OptimizationModel configureSensitivityModel(OptimizationModel om)
    throws ContextException, EvaluationException {

    Evaluator evalDFDH = evaluator("evalDFDH", "2.0*(w+d)",
          args(om.getDesignVars("w", "d")));
    Evaluator evalDFDW = evaluator("evalDFDW", "2.0*(h+2.0*d)",
          args(om.getDesignVars("h", "d")));
    Evaluator evalDFDD = evaluator("evalDFDD", "2.0*(h+2.0*w)",
          args(om.getDesignVars("w", "h")));
    List<Evaluator> evalGradF = list(evalDFDD, evalDFDH, evalDFDW);
    om.setGradientEvaluators("f", "evalF", "gradF", evalGradF);

    Evaluator evalDGDH = evaluator("evalDGDH", "-d*w",
          args(om.getDesignVars("w", "d")));
    Evaluator evalDGDW = evaluator("evalDGDW", "-h*d",
          args(om.getDesignVars("h", "d")));
    Evaluator evalDGDD = evaluator("evalDGDD", "-h*w",
          args(om.getDesignVars("w", "h")));
    List<Evaluator> evalGradG = list(evalDGDD, evalDGDH, evalDGDW);
    om.setGradientEvaluators("g", "evalG", "gradG", evalGradG);

    return om;
}
```

Create an evaluator with an expression and arguments for df/dh

Make a list of evaluations for a "response variable-response variable evaluator" pair

Assign evaluators named gradF to pair "f-evalF"



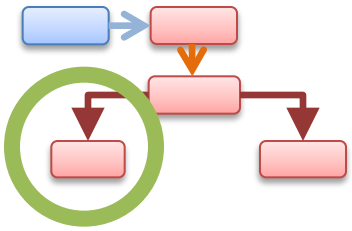
…/dot/requestor/src/engineering/optimization/dot/requestor/BoxModelBuilder.java

# Creating a SORCER Model: BoxModelBuilder Class



```
OptimizationModel createModel() throws ContextException, RemoteException {
    OptimizationModel om = optimizationModel(
          "Box Model",
          designVars(var("d", 1.0, 0.0, 5.0),
                  var("h", 2.0, 0.0, 5.0),
                  var("w", 1.0, 0.0, 5.0)),
          responseVars("f"),
          responseVars("g"),
          objectiveVars(var("area", "f", Target.min)),
          constraintVars(var("vol", "g", Relation.lte, 0.0)));

    om = configureAnalysisModel(om);
    om = configureSensitivityModel(om);
    om.initialize();
    return om;
}
```

Inherited initialization method

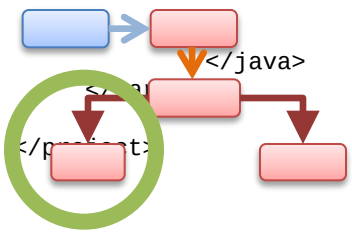


…/dot/requestor/src/engineering/optimization/
dot/requestor/BoxModelBuilder.java

# Publishing a SORCER Model



```
<project name="Box Model Provider" default="run.provider" basedir=".">
    …
    <!-- set property values here -->
    <property name="provider.bean" value="dotoptimization-req" />
    <property name="provider.class" value="sorcer.core.provider.ServiceProvider" />
    <property name="webster" value="${provider.webster.url}" />

    <!-- provider classpath -->
    <path id="project.classpath">
            <pathelement location="${eng.lib}/${provider.bean}.jar" />
            <pathelement location="${sorcer.lib}/sorcer-prv.jar" />
            <pathelement location="${sorcer.lib}/sorcer-lib.jar" />
            <pathelement location="${eng.lib}/dotoptimization.jar" />
            <pathelement location="${sorcer.lib}/sorcer-modeling-lib.jar" />
            …
    </path>

    <!-- provider codeabse jars -->
    <property name="j1" value="${webster}/${provider.bean}.jar" />
    <property name="j2" value="${webster}/sorcer-prv-dl.jar" />
    <property name="j3" value="${webster}/jsk-dl.jar" />
    <property name="j4" value="${webster}/provider-ui.jar" />

    <!-- start provider -->
    <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/${provider.bean}.policy" />
                    <sysproperty key="sorcer.provider.codebase" value="${j1} ${j2} ${j3} ${j4}" />
                    <sysproperty key="sorcer.provider.classpath" value="${toString:project.classpath}" />
                    <sysproperty key="sorcer.provider.impl" value="${provider.class}" />
                    <sysproperty key="sorcer.provider.config" value="../configs/bean-dot-box-model.config" />
                    <sysproperty key="sorcer.env.file" value="${iGrid.home}/configs/sorcer.env" />
                    <arg value="${iGrid.home}/configs/startup-prv.config" />
```



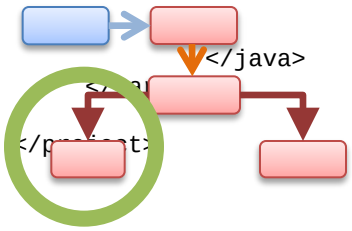

…/dot/requestor/bin/bean-dot-box-model-publish.xml

# Publishing a SORCER Model



```
<project name="Box Model Provider" default="run.provider" basedir=".">
    …
    <!-- set property values here -->
    <property name="provider.bean" value="dotoptimization-req" />
    <property name="provider.class" value="sorcer.core.provider.ServiceProvider" />
    <property name="webster" value="${provider.webster.url}" />

    <!-- provider classpath -->
    <path id="project.classpath">
            <pathelement location="${eng.lib}/${provider.bean}.jar" />
            <pathelement location="${sorcer.lib}/sorcer-prv.jar" />
            <pathelement location="${sorcer.lib}/sorcer-lib.jar" />
            <pathelement location="${eng.lib}/dotoptimization.jar" />
            <pathelement location="${sorcer.lib}/sorcer-modeling-lib.jar" />
            …
    </path>

    <!-- provider codeabse jars -->
    <property name="j1" value="${webster}/${provider.bean}.jar" />
    <property name="j2" value="${webster}/sorcer-prv-dl.jar" />
    <property name="j3" value="${webster}/jsk-dl.jar" />
    <property name="j4" value="${webster}/provider-ui.jar" />

    <!-- start provider -->
    <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/${provider.bean}.policy" />
                    <sysproperty key="sorcer.provider.codebase" value="${j1} ${j2} ${j3} ${j4}" />
                    <sysproperty key="sorcer.provider.classpath" value="${toString:project.classpath}" />
                    <sysproperty key="sorcer.provider.impl" value="${provider.class}" />
                    <sysproperty key="sorcer.provider.config" value="../configs/bean-dot-box-model.config" />
                    <sysproperty key="sorcer.env.file" value="${iGrid.home}/configs/sorcer.env" />
                    <arg value="${iGrid.home}/configs/startup-prv.config" />
```

Class to be published is declared in config

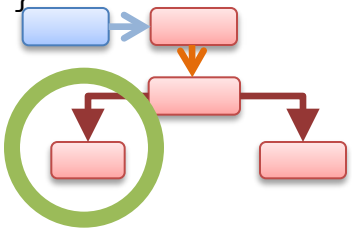



…/dot/requestor/bin/bean-dot-box-model-publish.xml

# Publishing a SORCER Model



```
/*
* Provider dependency injections
* It uses component entry names defined in the SORCER provider
* as defined by sorcer.core.provider.SorcerProvider.
*/
import net.jini.jrmp.*;
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.core.provider.*;
import sorcer.vfe.*;
import sorcer.model.rs.*;

sorcer.core.provider.ServiceProvider {
        /* service provider genetic properties */
           name = "Box Model";
           description = "Box Model Provider";
           location = "AFRL/RBSD";
           publishedInterfaces = new Class[] { sorcer.vfe.OptimizationModeling.class };

           // service beans
           beans = new Object[] { engineering.optimization.dot.requestor.BoxModelBuilder.createModel()
    };

           iconName="sorcer.jpg";
}
```



…/dot/requestor/configs/bean-dot-box-model.config

# Publishing a SORCER Model

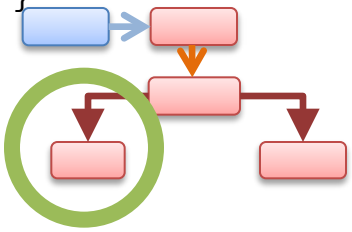


```
/*
* Provider dependency injections
* It uses component entry names defined in the SORCER provider
* as defined by sorcer.core.provider.SorcerProvider.
*/
import net.jini.jrmp.*;
import net.jini.jeri.*;
import net.jini.jeri.tcp.*;
import sorcer.core.*;
import net.jini.core.entry.E
import net.jini.lookup.entry
import sorcer.core.provider.
import sorcer.vfe.*;
import sorcer.models.*;

sorcer.core.provider.ServiceProvider {
        /* service provider genetic properties */
           name = "Box Model";
           description = "Box Model Provider";
           location = "AFRL/RBSD";
           publishedInterfaces = new Class[] { sorcer.vfe.OptimizationModeling.class };

           // service beans
           beans = new Object[] { engineering.optimization.dot.requestor.BoxModelBuilder.createModel()
    };

           iconName="sorcer.jpg";
}
```

"beans" is an iGrid-10 keyword used when calling a method within a class

Published interface

createModel method of published class

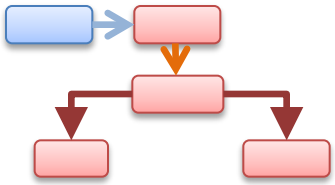


.../dot/requestor/configs/bean-dot-box-model.config

# Running an Optimization



- For event-driven optimization, three more pieces: Explorer, Dispatcher, Optimizer

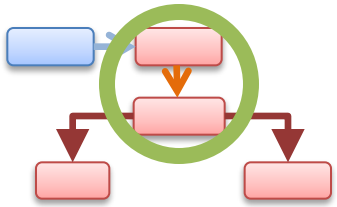


- *For standard optimization* (i.e., DOT is driving without user intervention) these are canned services—***the user does not write these***
  - One point approximation, multi-level, collaborative, and other problems will need custom classes
- All that is left for the user is writing the request

# Publishing an Explorer/Dispatcher



- For standard optimization, the provided explorer and dispatcher may be used
- The explorer is published
  - Class: sorcer.core.context.model.explore.Explorer.class
  - Interface: sorcer.core.context.model.explore.Exploration.class
- The explorer spawns the dispatcher based on the signature in the request
  - Class: engineering.optimization.dot.requestor.DotDispatcher



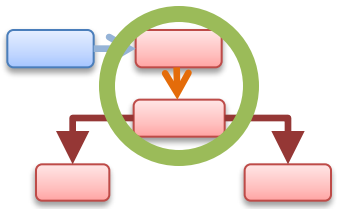
# Publishing an Explorer



- Explorer also published as a service bean
- Note: One explorer per optimization; Explorer is not multi-threaded

```
sorcer.core.provider.ServiceProvider {
        /* service provider genetic properties */
         name = "Box Explorer";
         description = "Box Model Explorer Provider";
         location = "AFRL/RBSD";
         publishedInterfaces = new Class[] {
   sorcer.core.context.model.explore.Exploration.class };

         // service beans
         beanClasses = new Class[] { sorcer.core.context.model.explore.Explorer.class };

         iconName="sorcer.jpg";
}
```

"beanClasses" is an iGrid-10 keyword used when publishing a class



…/dot/requestor/configs/bean-box-explorer.config

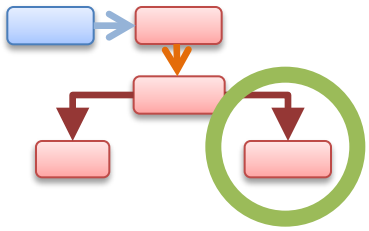
# Publishing DOT



- DOT also published as a service bean
  - Launch using
    …/dot/provider/bin/bean-jna-dot-prv-run.xml

```
sorcer.core.provider.ServiceProvider {
        /* service provider genetic properties */
         name = "DOT Optimizer";
         location = "AFRL/RBSD";
         publishedInterfaces = new Class[] { sorcer.core.context.model.Optimization.class
   };

         // service beans
         beanClasses = new Class[] {
   engineering.optimization.dot.provider.DotOptimizerJna.class };

         iconName="opti.jpg";
}
```

Use cool, custom icons!

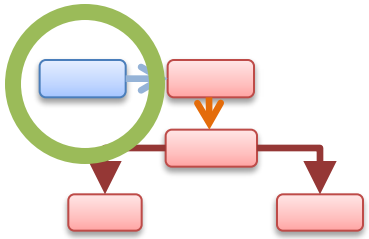


…/dot/provider/configs/bean-jna-dot-prv.config

# Setting Up an Optimization Requestor (Part 1 of 3)



```
public class BoxExplorerRequestor {

   private static Logger logger = Log.getTestLog();
   private static String strategy = new String();
   private static boolean isIntra = true;

   public static void main(String[] args) throws Exception {
         strategy = args[0];
         isIntra = Boolean.parseBoolean(args[1]);
         System.setSecurityManager(new RMISecurityManager());
         BoxExplorerRequestor requestor = new BoxExplorerRequestor();
         requestor.explore();
   }
```



…/dot/requestor/src/engineering/optimization/
dot/requestor/BoxExplorerRequestor.java

# Setting Up an Optimization Requestor (Part 1 of 3)

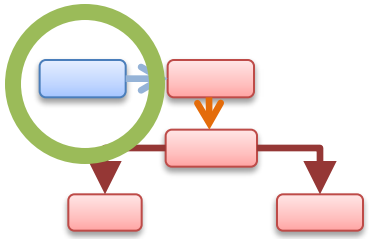


```
public class BoxExplorerRequestor {

   private static Logger logger = Log.getTestLog();
   private static String strategy = new String();
   private static boolean isIntra = true;

   public static void main(String[] args) throws Exception {
        strategy = args[0];
        isIntra = Boolean.parseBoolean(args[1]);
        System.setSecurityManager(new RMISecurityManager());
        BoxExplorerRequestor requestor = new BoxExplorerRequestor();
        requestor.explore();
   }
```

Requestor enters in main

Optimization strategy file location and intraprocess flag passed as arguments

Requestor instantiates itself and calls internal explore method

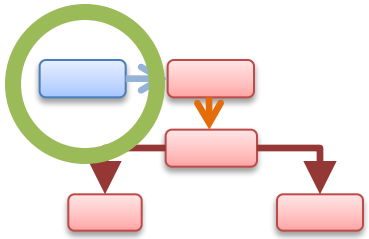


…/dot/requestor/src/engineering/optimization/dot/requestor/BoxExplorerRequestor.java

# Setting Up an Optimization Requestor (Part 2 of 3)



```
private void explore() throws Exception {
   ExploreContext exploreContext = new ExploreContext("Box");
   VarInfoList designInfo = varsInfo(varInfo("d", 1.0),
        varInfo("h", 1.0), varInfo("w", 1.0));
   exploreContext.setDesignVarsInfo(designInfo);
   DotStrategy dotStrat = new DotStrategy(strategy);
   logger.info("DOT Strategy:\n" + dotStrat.echo());
   exploreContext.setObjectivesInfo(null);
   exploreContext.setConstraintsInfo(null);
   exploreContext.setObjectivesGradientInfo(null);
   exploreContext.setConstraintsGradientInfo(null);
   exploreContext.setOptimizerStrategy(dotStrat);
   exploreContext.setDispatcherSignature(sig(null, DotDispatcher.class,
        Process.INTRA));
```



…/dot/requestor/src/engineering/optimization/
dot/requestor/BoxExplorerRequestor.java

# Setting Up an Optimization Requestor (Part 2 of 3)

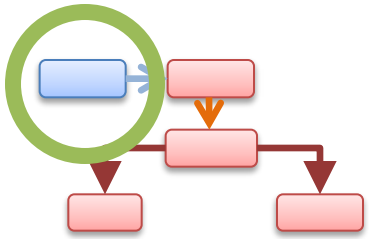


```
private void explore() throws Exception {
    ExploreContext exploreContext = new ExploreContext("Box");
    VarInfoList designInfo = varsInfo(varInfo("d", 1.0),
         varInfo("h", 1.0), varInfo("w", 1.0));
    exploreContext.setDesignVarsInfo(designInfo);
    DotStrategy dotStrat = new DotStrategy(strategy);
    logger.info("DOT Strategy:\n" + dotStrat.echo());
    exploreContext.setObjectivesInfo(null);
    exploreContext.setConstraintsInfo(null);
    exploreContext.setObjectivesGradientInfo(null);
    exploreContext.setConstraintsGradientInfo(null);
    exploreContext.setOptimizerStrategy(dotStrat);
    exploreContext.setDispatcherSignature(sig(null, DotDispatcher.class,
         Process.INTRA));
```

Requestor enters in main

Specify initial design

Instantiate new DotStrategy object using strategy file



.../dot/requestor/src/engineering/optimization/dot/requestor/BoxExplorerRequestor.java

# Setting Up an Optimization Requestor (Part 2 of 3)

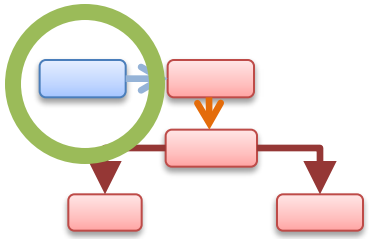


```
private void explore() throws Exception {
   ExploreContext exploreContext = new ExploreContext("Box");
   VarInfoList designInfo = varsInfo(varInfo("d", 1.0),
        varInfo("h", 1.0), varInfo("w", 1.0));
   exploreContext.setDesignVarsInfo(designInfo);
   DotStrategy dotStrat = new DotStrategy(strategy);
   logger.info("DOT Strategy:\n" + dotStrat.echo());
   exploreContext.setObjectivesInfo(null);
   exploreContext.setConstraintsInfo(null);
   exploreContext.setObjectivesGradientInfo(null);
   exploreContext.setConstraintsGradientInfo(null);
   exploreContext.setOptimizerStrategy(dotStrat);
   exploreContext.setDispatcherSignature(sig(null, DotDispatcher.class,
        Process.INTRA));
```

Initialize context with nulls

Put strategy into context

Set dispatcher signature for provided DotDispatcher



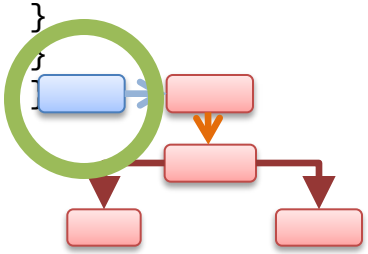
…/dot/requestor/src/engineering/optimization/dot/requestor/BoxExplorerRequestor.java

# Setting Up an Optimization Requestor (Part 3 of 3)



explore() method continued

```
if (isIntra) {
    logger.info("exploreContext: " + exploreContext);
    // optimizer and model are initialized by by Explorer using corresponding signatures
    Explorer explorer = new Explorer();
    exploreContext.setModelSignature(sig("createModel", BoxModelBuilder.class,
          Process.INTRA));
    exploreContext.setOptimizerSignature(sig(null, DotOptimizerJna.class, Process.INTRA));
    ExploreContext outContext = (ExploreContext)explorer.explore(exploreContext);
    logger.info(">>>>>>>>>>>>> query results: " + outContext);
}
else {
    // service providers specified by signatures
    exploreContext.setModelSignature(sig("register", OptimizationModeling.class,
                                         "Box Model"));
    exploreContext.setOptimizerSignature(sig("register", Optimization.class,
                                         "DOT Optimizer"));
    Task responses = task("opti", sig("explore", Exploration.class, "Model Explorer"),
                                    exploreContext);
    Exertion out = exert(responses);
    logger.info(">>>>>>>>>>>>> exceptions: " + out.getExceptions());
    logger.info(">>>>>>>>>>>>> search results: " + out.getContext());
}
}
```



…/dot/requestor/src/engineering/optimization/
dot/requestor/BoxExplorerRequestor.java

# Setting Up an Optimization Requestor (Part 3 of 3)



explore() method continued

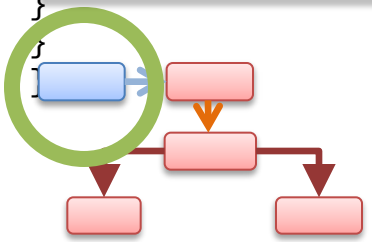
Instantiate Explorer

Specify model builder class and createModel method

Specify optimizer class

Invoke explorer explore method

```
if (isIntra) {
    logger.info("exploreContext: " + exploreContext);
    // optimizer and model are initialized by by Explorer using corresponding signatures
    Explorer explorer = new Explorer();
    exploreContext.setModelSignature(sig("createModel", BoxModelBuilder.class,
          Process.INTRA));
    exploreContext.setOptimizerSignature(sig(null, DotOptimizerJna.class, Process.INTRA));
    ExploreContext outContext = (ExploreContext)explorer.explore(exploreContext);
    logger.info(">>>>>>>>>>>>>> query results: " + outContext);
}
else {
    // service pr                                    ignatures
    exploreContex                                   g("register", OptimizationModeling.class,
                                                      "Box Model"));
    exploreContext.setOptimizerSignature(sig("register", Optimization.c
                                                      "DOT Optimizer"));
    Task responses = task("opti", sig("explore", Exploration.class, "Mo
                                      exploreContext);
    Exertion out = exert(responses);
    logger.info(">>>>>>>>>>>>>> exceptions: " + out.getExceptions());
    logger.info(">>>>>>>>>>>>>> search results: " + out.getContext());
}
}
```



…/dot/requestor/src/engineering/optimization/
dot/requestor/BoxExplorerRequestor.java

# Setting Up an Optimization Requestor (Part 3 of 3)



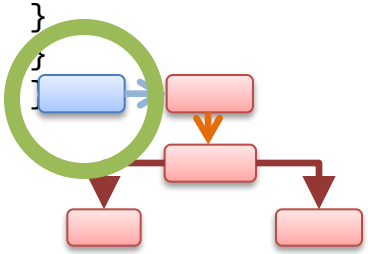
explore() method continued

```
if (isIntra) {
    logger.info("explor                                    ext);
    // optimizer and mo                                    Explorer using corr
    Explorer explorer =
    exploreContext.setM                                    Model", BoxModelBuil
          Process.INTRA
    exploreContext.setO                                    l, DotOptimizerJna.c
    ExploreContext outC                                    explorer.explore(exp
    logger.info(">>>>>>                                    + outContext);
}
else {
    // service providers specified by signatures
    exploreContext.setModelSignature(sig("register", OptimizationModeling.class,
                                         "Box Model"));
    exploreContext.setOptimizerSignature(sig("register", Optimization.class,
                                         "DOT Optimizer"));
    Task responses = task("opti", sig("explore", Exploration.class, "Model Explorer"),
                                  exploreContext);
    Exertion out = exert(responses);
    logger.info(">>>>>>>>>>>>>> exceptions: " + out.getExceptions());
    logger.info(">>>>>>>>>>>>>> search results: " + out.getContext());
}
}
```

Specify model implementing OptimizationModeling with optional name and use method register

Specify optimizer implementing Optimization with optional name and use method register

Exert a task named opti using the explore method of a service implementing Exploration



…/dot/requestor/src/engineering/optimization/dot/requestor/BoxExplorerRequestor.java

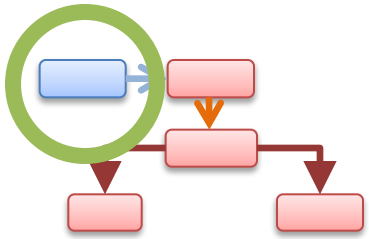
# Running the Requestor



- Launch requestor using …/dot/requestor/bin/dot-box-explorer-req-run.xml

```
…
<target name="run.requestor">
    <java classname="${requestor.class}" fork="yes">
          <classpath refid="project.classpath" />
          <sysproperty key="java.util.logging.config.file“
                    value="${iGrid.home}/configs/sorcer.logging" />
          <sysproperty key="java.security.policy" value="../policy/${requestor.name}.policy" />
          <sysproperty key="java.rmi.server.codebase" value="${j1} ${j2} ${j3} ${j4} ${j5}" />
          <sysproperty key="sorcer.env.file" value="${iGrid.home}/configs/sorcer.env" />
          <sysproperty key="jna.library.path" value="${local.lib}" />
          <arg value="../configs/box-dotoptimization-strategy.dat" />
          <arg value="true" />
    </java>
</target>
…
```

Arguments for requestor

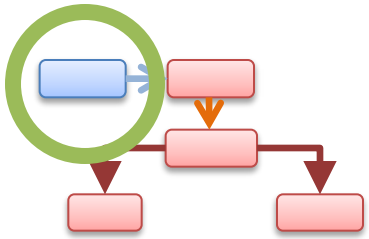


# DOT Strategy File



- DOT execution is controlled using the optimization strategy file
- Set of keyword-value pairs
  - Allowable keywords enumerated in DotConstants
  - See DOT manual for descriptions
  - Also “NAME” keyword accepting string data
- Any values may be omitted
  - Default to values in DotStrategy class

*NAME=Box Surface Area Minimization Test Problem*
*IPRINT=4*
*MINMAX=0*
*IGRAD=1*
*IWRITE=8*



…/dot/requestor/configs/box-dotoptimization-strategy.dat

# Outline



- Status update
- Simplistic but realistic example problem
  - Problem definition
  - Setting up and publishing a SORCER model
  - Publishing an explorer, dispatcher, and optimizer
  - Requesting a single optimization
- Behind the scenes of an optimization request
- Requesting multiple optimizations
- Real engineering problems

# Initial State



Explorer

BoxModel

Dot library is "synchronized" to allow concurrent optimizations

DotOptimizerJna


Initially, three services exist (published, or instantiated for intraprocess):

- Explorer (which will launch the optimization on request and wait for the result)
- BoxModel (which will call response evaluators)
- DotOptimizerJna (which exposes the DOT library)

# Launching an Optimization



Explore Context → Explorer

BoxModel

DotOptimizerJna


The optimization process is launched by providing an ExploreContext and invoking the explore method on the Explorer

# Create Dispatcher & Register



Explore Context

Explorer

DotDispatcher

BoxModel

DotOptimizerJna


The Explorer spawns a dispatcher as prescribed by the dispatcher signature in the ExploreContext, registers with the model and optimizer prescribed in the ExploreContext, and initiates a call to the optimizer

# Service Interaction



Explore Context

Explorer

DotDispatcher

BoxModel

DotOptimizerJna

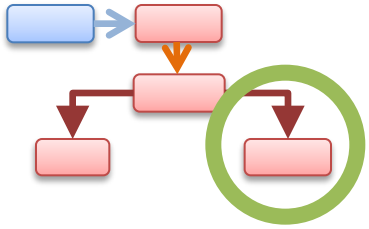

Dispatcher coordinates requests for new designs, model responses, and gradients between model and optimizer

When exploration completes (optimization terminates), results are reported back to explorer

# DOT Provider Classes



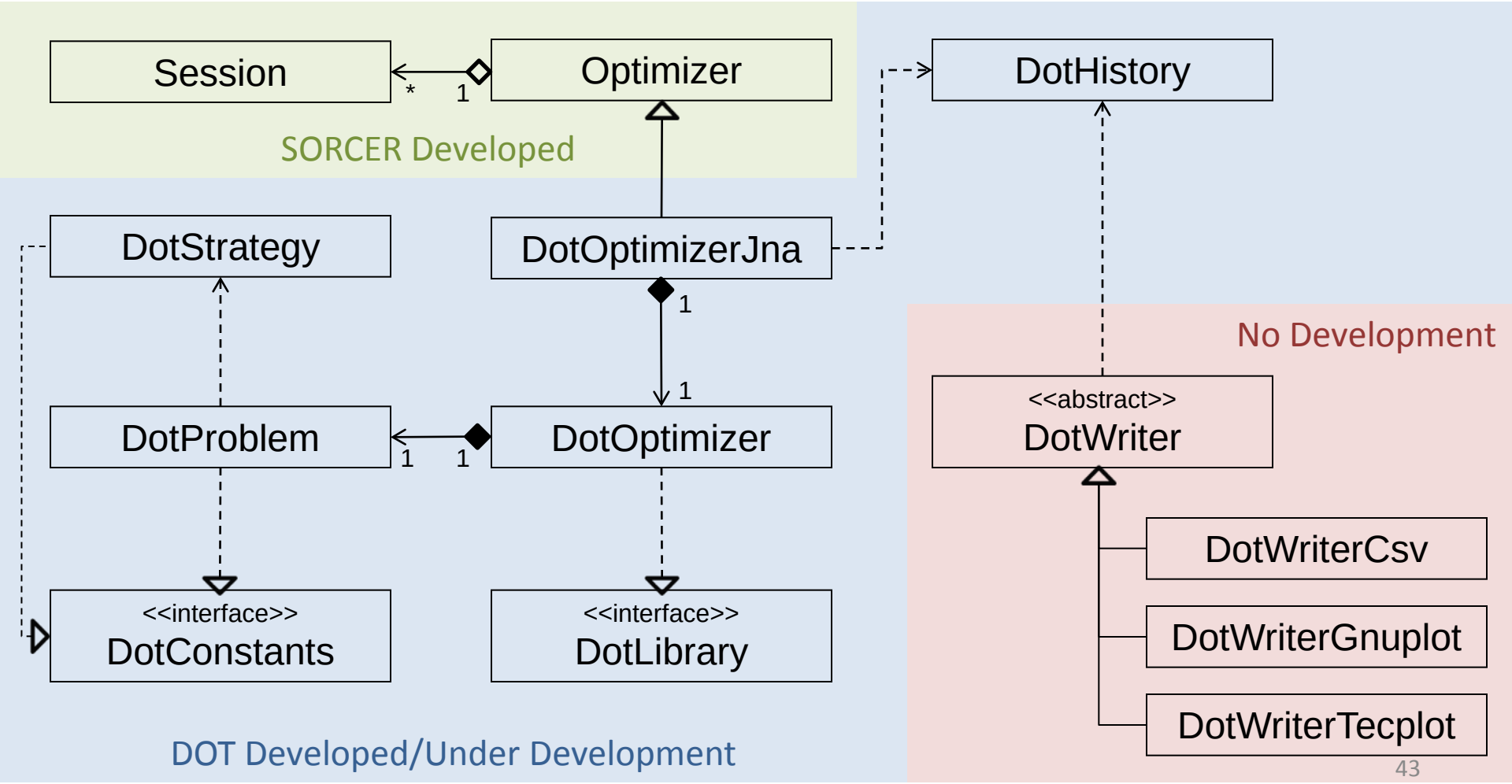
- DotOptimizerJna: Executive class
- DotOptimizer: Loads and calls DOT library, instantiated in and invoked by DotOptimizerJna
- DotProblem: Contains all data used by DOT library, maintains state, subclasses Box, RS (Rosen-Suzuki), and Square for testing
- DotStrategy: Specifies DOT controlling options, configurable by user
- DotConstants: Keyword interface implemented by DotStrategy
- DotHistory: Reserved for future convergence history logging
- DotOptimization: Currently unused



…/dot/provider/src/engineering/optimization/
dot/provider/

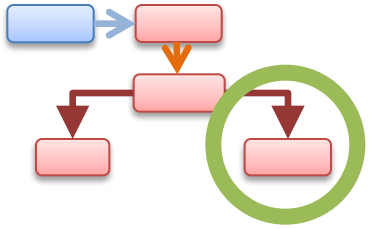
# Class Diagram



# Optimization Event Invocation

- `DotOptimizerJna#processSearchContext` called by `Optimizer#search` invoked by search event
  - `DotOptimizerJna#processSearchContext` overrides `Optimizer#processSearchContext`
- `processSearchContext` calls `optimize`, the executive method



…/dot/provider/src/engineering/optimization/dot/provider/DotOptimizerJna.java

# Unpacking the Problem

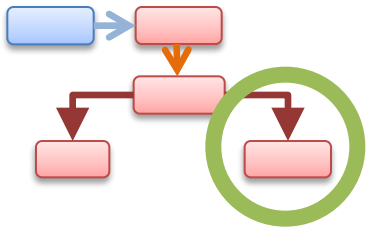


- Retrieve var info from context
  - Obj/con info's contain updated responses
  - Dvar info's will be updated with new design after DOT call

```
objInfo = ((VarInfoList)searchContext.getObjectivesInfo()).toArray();
conInfo = ((VarInfoList)searchContext.getConstraintsInfo()).toArray();
dvarInfo = ((VarInfoList)searchContext.getDesignVarsInfo()).toArray();
objGrads =(TableList)searchContext.getObjectivesGradientValues();
conGrads = (TableList)searchContext.getConstraintsGradientValues();
```

- If initial call, construct dot problem using strategy
- If subsequent, retrieve dot problem from session

```
DotProblem dotProb = null;
if (searchContext.getStatus() == ExecState.INITIAL) {
    DotStrategy dotStrat = (DotStrategy) searchContext.getOptimizerStrategy();
    dotProb = new DotProblem(dotStrat, objInfo[0], conInfo, dvarInfo);
}
else {
    dotProb = (DotProblem)
    getSessions().get(searchContext.getEventInfo().getID()).getState();
}
```



…/dot/provider/src/engineering/optimization/
dot/provider/DotOptimizerJna.java

# Updating and Calling DOT

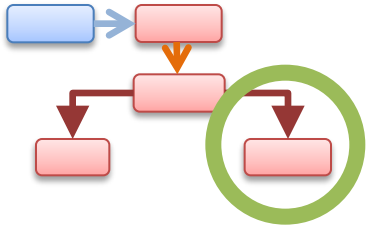


- Update the problem with the latest responses

```
switch (dotProb.getInfo()) {
case 2: dotProb.updateVarGrads(objGrads, objInfo[0], conGrads, conInfo, dvarInfo);
default: dotProb.updateVars(objInfo[0], conInfo, dvarInfo);
}
```

- DotProblem object is passed to DotOptimizer object

```
dotProb = optimizer.optimize(dotProb);
```



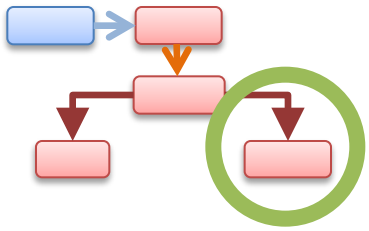
…/dot/provider/src/engineering/optimization/
dot/provider/DotOptimizerJna.java

# Call the DOT Library



- DotOptimizer (implements Runnable for threading) object calls DOT library

```
DotLibrary lib;
DotProblem prob;
public synchronized DotProblem optimize(DotProblem inProb) {
    prob = inProb;
    t = new Thread(this, "DotOptimizer");
    t.start();
    t.join();
    return prob;
}
public void run() {
    info = new IntByReference(prob.info);
    …
    obj = new DoubleByReference(prob.obj);

    lib.dot_(info, method, iprint, ndv, ncon, prob.x, prob.xl, prob.xu, obj,
          minmax, prob.g, prob.rprm, prob.iprm, prob.wk, nrwk, prob.iwk, nriwk);

    prob.info = info.getValue();
    …
    prob.obj = obj.getValue();
}
```



…/dot/provider/src/engineering/optimization/
dot/provider/DotOptimizer.java

# Repackage the Problem



- Set optimizer status

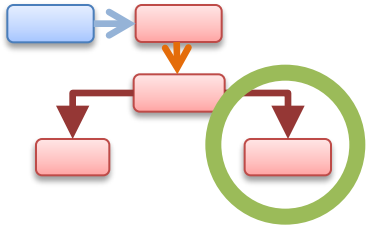
```
searchContext.setOptimizerState(new Integer(dotProb.getInfo()));
```

- Update design variables info and mark responses and gradients requested

```
dvarInfo = dotProb.updateDvarInfoArray
switch (dotProb.getInfo()) {
   case 2: {
        int[] conGradsReq = dotProb.getConGradsReq();
        addGradForEval(objInfo[0], objGradsToEval, conInfo,
                conGradsToEval, conGradsReq);
        break;
   }
   case 1: {
        addRespForEval(objInfo[0], objToEval, conInfo, conToEval);
        break;
   }
```

- Update the context and store optimizer state

```
searchContext = updateContext(searchContext, objInfo, conInfo, dvarInfo,
        objGradsToEval, conGradsToEval, dotProb);
getSessions().get(searchContext.getEventInfo().getID()).setState(dotProb);
```



…/dot/provider/src/engineering/optimization/
dot/provider/DotOptimizerJna.java

# Outline



- Status update
- Simplistic but realistic example problem
  - Problem definition
  - Setting up and publishing a SORCER model
  - Publishing an explorer, dispatcher, and optimizer
  - Requesting a single optimization
- Behind the scenes of an optimization request
- Requesting multiple optimizations
- Real engineering problems

# Concurrent Optimization

- An example of three concurrent optimizations using multithreading is given in MultiExplorerRequestor
  - One of many ways, can launch several singleton cases
- Logic is same as singleton case, except name explorer and model must be given because interfaces same for each problem
- Explorer can only handle one optimization at a time
  - Use one explorer per optimization
- Apparent change in convergence path must be investigated
  - Test problems converge to correct solution, but take more/fewer iterations

# Outline

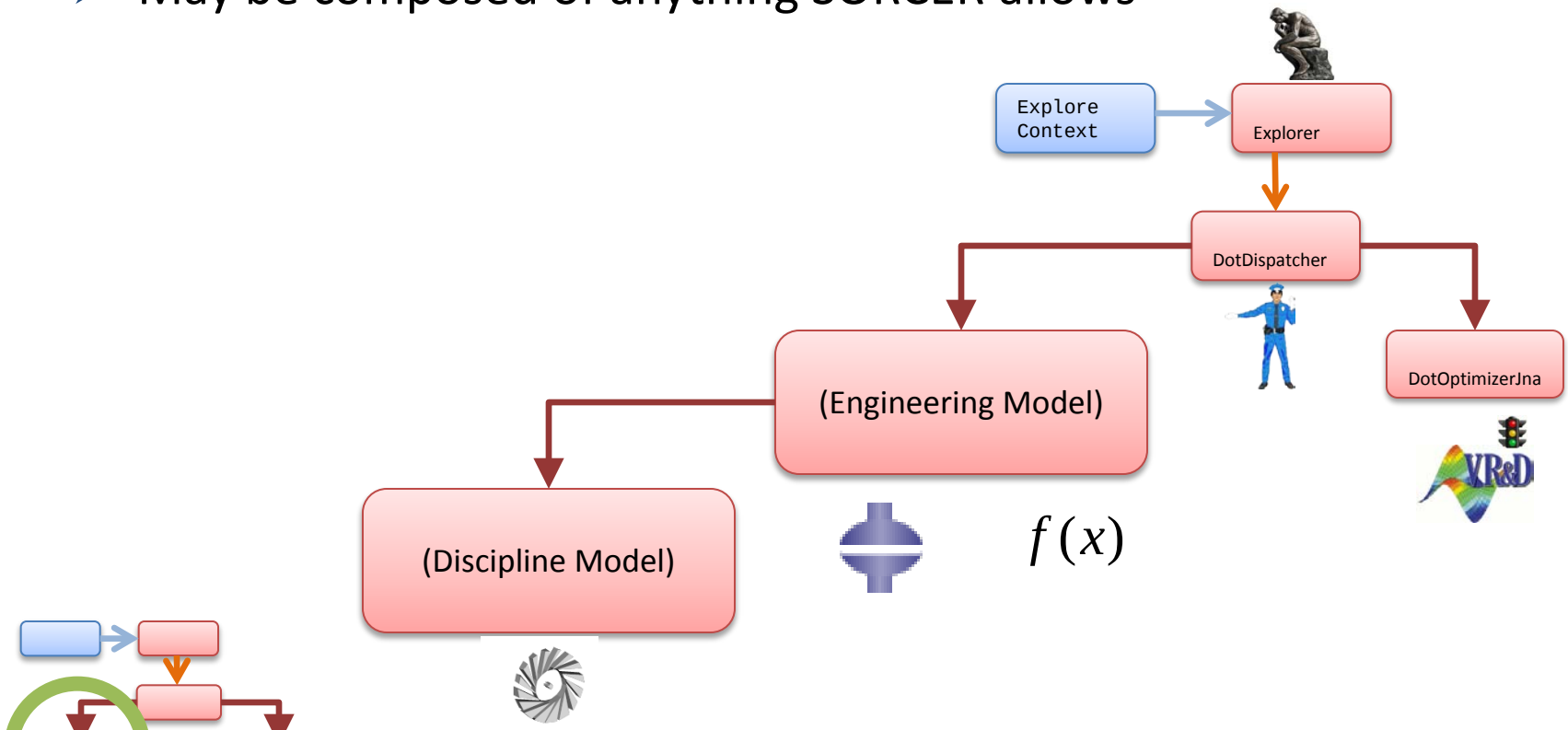


- Status update
- Simplistic but realistic example problem
  - Problem definition
  - Setting up and publishing a SORCER model
  - Publishing an explorer, dispatcher, and optimizer
  - Requesting a single optimization
- Behind the scenes of an optimization request
- Requesting multiple optimizations
- Real engineering problems

# Standard Optimizations



- For a standard optimization (i.e., normal DOT procedure), work is for analyst to create SORCER model
- Design variables, objectives, constraints, their sensitivities (if supplied), and their evaluators are exposed by the model
- May be composed of anything SORCER allows



# Advanced Optimizations



- Advanced optimizations will require development of custom dispatchers
  - One point approx, collaborative optimization, multi-level, etc.