Appendix B: Source Code

1.1 Appendix B.1: NAOMI Core Sources

1.2 Appendix B.1.1: NAOMI Core Interfaces

NaomiActivator.groovy

```groovy
// @file     NaomiActivator
// @author   Mac Radigan

package org.radigan.naomi.service

import org.radigan.naomi.utilities.ServiceFactory
import org.radigan.naomi.service.Factory

import java.util.Properties
import org.osgi.framework.BundleActivator
import org.osgi.framework.BundleContext
import org.osgi.framework.ServiceListener
import org.osgi.framework.ServiceEvent
import org.osgi.framework.ServiceRegistration
import org.osgi.framework.ServiceReference
import org.osgi.service.cm.ConfigurationAdmin
import org.osgi.service.cm.Configuration
import org.osgi.util.tracker.ServiceTracker

public class NaomiActivator implements BundleActivator {

  protected ServiceRegistration registration = null
  protected Factory factory = null

  public void start(BundleContext context) {
    try {
      def propertiesId = 'naomi'
      def propertyHome = 'home'
      def tracker = new ServiceTracker(context, ConfigurationAdmin.class.getName(), null)
      tracker.open()
      def configAdmin = (ConfigurationAdmin)tracker.getService()
      def configuration = configAdmin.getConfiguration(propertiesId, null)
      def properties = configuration.getProperties()
      if(!properties) throw new Exception("""No such properties ID "${propertiesId}".""")
      factory = ServiceFactory.getInstance(new File(properties.get(propertyHome)))
      if(factory) {
        factory.getNaomi().start()
      } else {
        throw new Exception("ERROR:  unable to register ${Factory.class.getName()}")
      }
      def props = new Properties()
      props.put(propertyHome, properties.get(propertyHome))
      registration = context.registerService(Factory.class.getName(), factory, props)
    } catch(Exception e) {
      e.printStackTrace()
    }
  }
```

```
  public void stop(BundleContext context) {
    try {
      if(factory) {
        factory.getNaomi().stop()
        factory = null
      }
      if(registration) {
        registration.unregister()
        registration = null
      }
    } catch(Exception e) {
      e.printStackTrace()
    }
  }

}

// *EOF*
```

Factory.java

```
// @file     Factory
// @author   Mac Radigan

package org.radigan.naomi.service;

import org.radigan.naomi.service.Naomi;
import org.radigan.naomi.roar.service.Roar;
import org.radigan.naomi.wumpus.service.Wumpus;
import org.radigan.naomi.service.Module;
import java.util.List;

public interface Factory {
  public Naomi getNaomi();
  public Roar getRoar();
  public Wumpus getWumpus();
  public void loadModules();
  public List<Module> getModules();
  public List<Module> getModules(Class clazz);
  public void clearModules();
  public void addModules(List<Class> classes);
}

// *EOF*
```

Module.java

```
// @file     Module.java
// @author   Mac Radigan

package org.radigan.naomi.service;

public interface Module {
  public void initialize();
}

// *EOF*
```

Naomi.java

```java
// @file     Naomi.java
// @author   Mac Radigan

package org.radigan.naomi.service;

public interface Naomi {
  public void start();
  public void stop();
}

// *EOF*
```

TypesDatabase.java

```java
// @file     TypesDatabase.java
// @author   Mac Radigan

package org.radigan.naomi.service;

import java.util.List;

public interface TypesDatabase {
}

// *EOF*
```

1.3 Appendix B.1.2: NAOMI Core Implementations

NaomiImpl.groovy

```groovy
// @file     NaomiImpl
// @author   Mac Radigan

package org.radigan.naomi.impl

import org.radigan.naomi.service.Naomi
import org.radigan.naomi.roar.impl.RoarImpl
import org.radigan.naomi.roar.service.Roar
import org.radigan.naomi.utilities.ServiceFactory
import org.apache.log4j.Logger

public class NaomiImpl implements Naomi {
  protected Logger log = null
  protected ServiceFactory serviceFactory = null
  protected Roar roar = null
  public NaomiImpl() {
    serviceFactory = ServiceFactory.getInstance()
    log = serviceFactory.getLogger(this)
    roar = serviceFactory.getRoar()
    initializeServer()
  }
  protected void initializeServer() {
    def modules = []
    //modules << org.radigan.naomi.wumpus.impl.SimulationFunctor
    //modules << org.radigan.naomi.roar.impl.InterfaceResource
    //modules << org.radigan.naomi.nyancat.impl.GraphvizReport
```

```
    serviceFactory.getResourceManager().getText('/META-INF/services/org.radigan.naomi.service.Module').
      if(!line.startsWith('#')) modules << Class.forName(line)
    }
    serviceFactory.addModules(modules)
  }
  public void start() {
    roar.start()
  }
  public void stop() {
    roar.stop()
  }
}

// *EOF*
```

TypesDatabaseImpl.groovy

```
// @file     TypesDatabaseImpl.groovy
// @author   Mac Radigan

package org.radigan.naomi.impl

import org.radigan.naomi.service.TypesDatabase
import javax.swing.tree.TreeNode
import javax.swing.tree.DefaultMutableTreeNode as Node
import groovy.util.slurpersupport.GPathResult
import org.apache.log4j.Logger

public class TypesDatabaseImpl implements TypesDatabase {

  public static final String BASE_OBJECT = "Object"
  private Node root = new Node(BASE_OBJECT)

  private TypesDatabaseImpl() {
    initialize()
  }
  private static TypesDatabaseImpl ref = null
  public static TypesDatabase getInstance() {
    if(!ref) ref=new TypesDatabaseImpl()
    return ref
  }

  private void initialize() {
  }

  public TreeNode find(String type) {
    return find(root, type)
  }
  public TreeNode find(TreeNode node, String type) {
    if(node.getUserObject()==type) {
      return node
    } else {
      def rval = null
      node.children().each { n ->
        rval = find(n, type)
      }
```

```
      return rval
    }
  }

  public String qname(String type) {
    def a = []
    def n = find(root,type)
    a << n.getUserObject()
    while(n=n.getParent()) { a << n.getUserObject() }
    return a.reverse().join(':')
  }

  public boolean typeOf(String t, String b) {
    //def tn = find(t)
    //if(!tn) throw new IllegalArgumentException("Unknown type: $t")
    //return find(tn,t)&&1
    return find(find(t),t)&&1
  }

  public void parse(String text) {
    def xml = new XmlSlurper().parseText(text).declareNamespace(
      t:'http://org.radigan.naomi/datatypes'
      )
    xml.children().each { n ->
      parse(n)
    }
  }
  public void parse(GPathResult node) {
    def name = node.'@name' as String
    def type = node.'@type' as String
    def p = find(type)
    if(p) p.add(new Node(name))
    node.children().each { parse(it) }
  }

  public String toString() {
    def sb = new StringBuilder()
    sb << toString(root, 0)
    return sb.toString()
  }
  public String toString(TreeNode node, int ind) {
    def sb = new StringBuilder()
    sb << " ".multiply(ind) << node.getUserObject() << "\n"
    node.children().each { n ->
      sb << toString(n, ind+2)
    }
    return sb.toString()
  }

}

// *EOF*
```

DatabaseParser.groovy

```
// @file      DatabaseParser.groovy
```

```groovy
// @author   Mac Radigan

package org.radigan.naomi.utilities

import org.radigan.system.utilities.Recordset
import org.radigan.system.utilities.Record
import groovy.util.slurpersupport.GPathResult

public class DatabaseParser {

  public DatabaseParser() {
  }

  public static Recordset parse(File file) {
    def xml = new XmlSlurper().parseText(file.text)
    return parse(xml)
  }

  public static Recordset parse(GPathResult xml) {
    Recordset recordset = new Recordset()
    xml.children().each { row ->
      def fields = [:]
      row.attributes().each { key, value ->
        fields[key] = null
        if(!fields[key]) try {
          fields[key] = Integer.parseInt(value)
        } catch(e) {
        }
        if(!fields[key]) try {
          fields[key] = Boolean.parseBoolean(value)
        } catch(e) {
        }
        if(!fields[key]) try {
          fields[key] = value
        } catch(e) {
        }
      }
      recordset << new Record(fields)
    }
    return recordset
  }

}

// *EOF*
```

ServiceFactory.groovy

```groovy
// @file     ServiceFactory
// @author   Mac Radigan

package org.radigan.naomi.utilities

import java.util.Map
import org.apache.log4j.Logger
```

```
import org.radigan.naomi.service.Factory
import org.radigan.naomi.service.Module
import org.radigan.naomi.wumpus.service.FunctorList
import org.radigan.naomi.wumpus.impl.SimulationFunctor
import org.radigan.system.configuration.Configuration
import org.radigan.system.utilities.ResourceManager
import org.radigan.naomi.service.Naomi
import org.radigan.naomi.impl.NaomiImpl
import org.radigan.naomi.wumpus.service.Wumpus
import org.radigan.naomi.wumpus.impl.WumpusImpl
import org.radigan.naomi.roar.service.Roar
import org.radigan.naomi.roar.impl.RoarImpl
import com.thoughtworks.xstream.*
import com.thoughtworks.xstream.io.xml.DomDriver
import java.net.InetAddress
import java.util.ServiceLoader

public class ServiceFactory implements Factory {

  protected Configuration configuration = null
  protected ConfigObject naomiConfig = null
  protected Naomi naomi = null
  protected Wumpus wumpus = null
  protected Roar roar = null
  protected String environment = null
  protected GroovyClassLoader classloader = null
  protected Logger log = getLogger(this)
  protected List<Module> modules = []
  protected boolean initialized = false

  private static ServiceFactory ref = null
  private ServiceFactory(File systemHome=null) {
    initialize(systemHome)
  }
  public static ServiceFactory getInstance(File systemHome=null) {
    if(!ref) ref = new ServiceFactory(systemHome)
    return ref
  }

  private void initialize(File systemHome=null) {
    classloader = new GroovyClassLoader(getClass().getClassLoader())
    configuration = Configuration.getInstance(systemHome)
    environment = InetAddress.getLocalHost().getHostName()
    naomiConfig = configuration.getConfiguration(environment, new File("naomi.conf"))
  }

  public void loadModules() {
    if(initialized) return
    initialized = true
    def path = new File("${getRootDirectory()}/deploy")
    path.eachFileRecurse() { file ->
      if(file.getName().contains(".naomi")) {
        try {
          log.debug "compiling ${file}"
```

```
          classloader.parseClass(file)
        } catch(e) {
          log.error "Compilation failed.  Reason: ${e.getMessage()}"
          throw e
        }
      } else if(file.getName().endsWith(".jar") || file.getName().endsWith(".nar")) {
        try {
          log.debug "loading ${file}"
          classloader.addURL(file.toURI().toURL())
        } catch(e) {
          log.error "Failed to load class.  Reason: ${e.getMessage()}"
          throw e
        }
      }
    }
    ServiceLoader.load(Module.class, classloader).each() { m ->
      def clazz = m.getClass()
      if(clazz in Module && !modules.contains(clazz)) {
        log.debug "adding class ${clazz}"
        modules << clazz
      }
    }
    classloader.getLoadedClasses().each { clazz ->
      if(clazz in Module && !modules.contains(clazz)) {
        log.debug "adding class ${clazz}"
        modules << clazz
      }
    }
  }

  public List<Module> getModules() { return modules }
  public List<Module> getModules(Class clazz) {
     def rval = []
     modules.each { if(it in clazz) rval<<it }
     return rval
  }
  public void clearModules() { classloader.clearCache() }
  public void addModules(List<Class> classes) { modules.addAll(classes) }

  public ClassLoader getClassLoader() {
    return classloader
  }

  public String getEnvironment() {
    return environment
  }

  public ResourceManager getResourceManager() {
    return configuration.getResourceManager()
  }

  public ConfigObject getConfiguration() {
    return naomiConfig
  }
```

```
  public File getRootDirectory() {
    return configuration.getRootDirectory()
  }

  public Logger getLogger(object) {
    return Logger.getLogger(object.class.getName())
  }

  public Naomi getNaomi() {
    if( ! this.naomi) this.naomi = new NaomiImpl()
    return this.naomi
  }

  public Wumpus getWumpus() {
    if( ! this.wumpus) this.wumpus = new WumpusImpl()
    return this.wumpus
  }

  public Roar getRoar() {
    if( ! this.roar) this.roar = new RoarImpl()
    return this.roar
  }

  public void save(File file, Object object) {
    def xstream = new XStream()
    file.withOutputStream { ostream ->
      xstream.toXML(object, ostream)
    }
  }

  public Object load(File file) {
    def className = new XmlSlurper().parseText(file.text).name()
    def object = Class.forName(className)
    def xstream = new XStream()
    file.withInputStream { istream ->
      object = xstream.fromXML(istream)
    }
    return object
  }

}

/* *EOF* */
```

TypesCategory.groovy

```
// @file     TypeCategory.groovy
// @author   Mac Radigan

package org.radigan.naomi.utilities

import org.radigan.naomi.service.TypesDatabase
import org.apache.log4j.Logger

public class TypeCategory {
```

```
  public String isCase(String t) {
    println "invoked isCase"
    return TypesDatabase.getInstance().typeOf(t.toString(),this.toString())
  }

}

// *EOF*
```

1.4 Appendix B.2: ROAR Sources

1.5 Appendix B.2.1: ROAR Interfaces

Resource.java

```
// @file     Resource.java
// @author   Mac Radigan

package org.radigan.naomi.roar.service;

import java.util.List;
import java.util.Map;
import java.net.URI;
import java.io.OutputStream;

public interface Resource {
  public void setUtilities(Map utilities);
  public boolean canDecode(URI uri);
}

// *EOF*
```

AbstractResource.java

```
// @file     AbstractResource.java
// @author   Mac Radigan

package org.radigan.naomi.roar.service

import org.radigan.naomi.nyancat.service.Report
import org.radigan.naomi.service.Module
import javax.servlet.http.HttpServlet

public abstract class AbstractResource extends HttpServlet implements Resource, Module {
  protected Map util = [:]
  public AbstractResource() { }
  public void setUtilities(Map util) {
    this.util = util
  }
  public abstract void initialize()
  public abstract boolean canDecode(URI uri)
}

// *EOF*
```

Roar.java

```
// @file     Roar.java
```

```java
// @author   Mac Radigan

package org.radigan.naomi.roar.service;

import java.util.List;
import org.radigan.naomi.wumpus.service.Functor;

public interface Roar {
  public boolean isRunning();
  public void start();
  public void stop();
  public void registerFunctor(List<Functor> functors);
  public void unregisterFunctor(List<Functor> functors);
  public void registerInterfaces(List<Functor> functors);
  public void unregisterInterfaces(List<Functor> functors);
}

// *EOF*
```

1.6 Appendix B.2.2: ROAR Implementations

InterfaceResource.groovy

```groovy
// @file     InterfaceResource.java
// @author   Mac Radigan

package org.radigan.naomi.roar.impl

import org.radigan.naomi.nyancat.service.Report
import org.radigan.naomi.nyancat.impl.GraphvizReport
import org.radigan.naomi.nyancat.impl.WumpusReport
import org.radigan.naomi.nyancat.impl.OctaveReport
import org.radigan.naomi.roar.service.Resource
import org.radigan.naomi.roar.service.AbstractResource
import org.radigan.naomi.wumpus.utilities.WumpusUtil
import org.apache.commons.io.FileUtils
import org.apache.commons.io.IOUtils
import java.io.OutputStream
import javax.servlet.*
import javax.servlet.http.*
import static javax.servlet.http.HttpServletResponse.*
import javax.servlet.ServletException

public class InterfaceResource extends AbstractResource {
  protected WumpusUtil wu = null
  protected File cacheDir = null
  public InterfaceResource() {
    super()
  }
  public void setUtilities(Map util) {
    this.util = util
    wu = util['wu']
    cacheDir = util['cache']
  }
  public void initialize() { }
```

```
protected Map<String,String> parse(URI uri) {
  def map = [:]
  def path = uri.toString()
  if(path.startsWith("/roar/")) {
    def matcher = path=~/\/roar\/([A-z][A-z0-9]*)\/([A-z][A-z0-9]*)\/([A-z]+)\/([A-z][A-z0-9]*)\.(gif
    if(matcher.matches()) {
      map['namespace'] = matcher[0][1]
      map['nameDir'] = matcher[0][2]
      map['cat'] = matcher[0][3]
      map['name'] = matcher[0][4]
      map['ext'] = matcher[0][5]
      if(map['name']!=map['nameDir']) return []
      if(!(map['cat'] in ['interface', 'style'])) return []
      return map
    } else {
      return map
    }
  }
  return []
}
public boolean canDecode(URI uri) {
  def map = parse(uri)
  if(map.size()>0) {
    def index = wu.getFunctorIndex(map['name'], map['namespace'])
    return (null!=index)
  }
}
public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOExcepti
  def uri = new URI(req.getRequestURI())
  def map = parse(uri)
  switch(map['cat']) {
    case 'interface':
      def resDir = new File("${cacheDir}/${map['namespace']}/${map['name']}/${wu.getId()}")
      def fn = wu.getFunctorIndex(map['name'], map['namespace'])
      def wf = wu.filter(fn as int)
      switch(map['ext']) {
        case ['html','xhtml']:
          res.getWriter().println(new WumpusReport(wf).toString())
          break
        case 'm':
          res.getWriter().println(new OctaveReport(wf).toString())
          break
        case 'dot':
          res.getWriter().println(new GraphvizReport(wf).toString())
          break
        case ['gif','png','tif','jpg']:
          OutputStream output = res.getOutputStream()
          def filename = new File(uri.getPath()).getName()
          res.setContentType("image/gif")
          res.setHeader('Content-Transfer-Encoding', 'binary')
          /*
          res.setHeader("""Content-Disposition','attachment; filename=\"${filename}\"""" as String)
          */
          def rptFile = new File("${resDir}/${map['name']}.${map['ext']}")
```

```
            if( !rptFile.exists()) {
              FileUtils.forceMkdir(resDir)
              def report = new GraphvizReport(wf)
              report.save(rptFile, map['ext'])
            }
            def input = new FileInputStream(rptFile)
            IOUtils.copy(input, output)
            IOUtils.closeQuietly(output)
            break
          default:
            throw new IllegalStateException("Invalid extension: ${map['ext']}")
        }
        break
      case 'style':
        break
      default:
        throw new IllegalStateException("Invalid category: ${map['cat']}")
    }
  }
}

// *EOF*
```

RoarImpl.groovy

```
// @file     RoarImpl.groovy
// @author   Mac Radigan

package org.radigan.naomi.roar.impl

import org.radigan.naomi.roar.service.Roar
import org.radigan.naomi.roar.service.Resource
import org.radigan.naomi.utilities.ServiceFactory
import org.radigan.naomi.wumpus.service.Functor
import org.radigan.naomi.wumpus.service.Wumpus
import org.mortbay.jetty.*
import org.mortbay.jetty.nio.*
import org.mortbay.jetty.servlet.*
import org.mortbay.jetty.deployer.*
import groovy.servlet.*
import org.mortbay.jetty.bio.SocketConnector
import groovy.util.ConfigObject
import java.io.OutputStream
import javax.servlet.http.*
import javax.servlet.ServletException

public class RoarImpl implements Roar {
  protected ServiceFactory serviceFactory = null
  protected File webappDir = null
  protected File cache = null
  protected Server server = new org.mortbay.jetty.Server()
  protected ConfigObject config = null
  protected int port = 80
  protected Wumpus wumpus = null
  protected List<Resource> resources = []
  public RoarImpl() {
```

```
    serviceFactory = ServiceFactory.getInstance()
    config = serviceFactory.getConfiguration()
    wumpus = serviceFactory.getWumpus()
    webappDir = new File("${serviceFactory.getRootDirectory()}/webapps")
    port = config.roar.port
    cache = new File(config.roar.cache)
    initializeResources()
    initializeServer()
  }
  protected void initializeServer() {
    def connector = new SocketConnector()
    connector.setPort(port as int)
    server.setConnectors([connector] as Connector[])
    def deployer = new WebAppDeployer()
    deployer.setContexts(server)
    deployer.setWebAppDir("${webappDir}")
    deployer.setExtract(true)
    deployer.setParentLoaderPriority(false)
    deployer.start()
    def rootContext = new Context(server, "/", Context.SESSIONS)
    rootContext.setResourceBase(".")
    rootContext.addServlet(new ServletHolder(new TemplateServlet()), "*.html")
    rootContext.addServlet(new ServletHolder(new GroovyServlet()), "*.glet")
    rootContext.addServlet(new ServletHolder(new RoarServlet(this)), "/roar/*")
    server.setStopAtShutdown(true)
    server.start()
  }
  protected initializeResources() {
    serviceFactory.loadModules()
    serviceFactory.getModules(Resource).each { m ->
      def r = m.newInstance()
      r.initialize()
      r.setUtilities([
        'wu':wumpus.getWumpusUtil(),
        'cache':cache
        ])
      resources << r
    }
  }
  public boolean isRunning() {
    return server.isRunning()
  }
  public void start() {
  }
  public void stop() {
    server.start()
  }
  protected Resource findResource(URI uri) {
    def rval = null
    resources.each { r -> if(r.canDecode(uri)) { rval=r } }
    return rval
  }
  protected Resource findResource(HttpServletRequest req) {
    def uri = new URI(req.getRequestURI())
```

```
    return findResource(uri)
  }
  public boolean canDecode(URI uri) {
    return null!=findResource(uri)
  }
  protected boolean canDecode(HttpServletRequest req) {
    def uri = new URI(req.getRequestURI())
    return canDecode(uri)
  }
  public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOExcepti
    findResource(req)?.doGet(req, res)
  }
  public void registerFunctor(List<Functor> functors) {
  }
  public void unregisterFunctor(List<Functor> functors) {
  }
  public void registerInterfaces(List<Functor> functors) {
  }
  public void unregisterInterfaces(List<Functor> functors) {
  }
}

// *EOF*
```

RoarServlet.groovy

```
// @file     RoarServlet.groovy
// @author   Mac Radigan

package org.radigan.naomi.roar.impl

import java.io.*
import javax.servlet.http.*
import static javax.servlet.http.HttpServletResponse.*
import javax.servlet.*
import org.apache.commons.io.IOUtils

public class RoarServlet extends HttpServlet {

  protected RoarImpl roar = null

  public RoarServlet(RoarImpl roar) {
    this.roar = roar
  }

  public void doHead(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOExcept
  }

  public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOExcepti
    try {
      if(roar.canDecode(req)) {
        roar.doGet(req, res)
      } else {
        res.sendError(SC_NOT_FOUND)
      }
    } catch(e) {
```

```
      e.printStackTrace()
      res.sendError(SC_INTERNAL_SERVER_ERROR)
    } finally {
      //IOUtils.closeQuietly(out)
    }
  }


}

// *EOF*
```

1.7 Appendix B.3: WUMPUS Sources

1.8 Appendix B.3.1: WUMPUS Interfaces

Wumpus.java

```
// @file     Wumpus.java
// @author   Mac Radigan

package org.radigan.naomi.wumpus.service;

import java.util.List;
import java.util.Map;
import org.radigan.naomi.wumpus.utilities.WumpusUtil;

public interface Wumpus {
  public String toString();
  public WumpusUtil getWumpusUtil();
}

// *EOF*
```

Functor.java

```
// @file     Functor.java
// @author   Mac Radigan

package org.radigan.naomi.wumpus.service;

import java.util.List;
import java.util.Map;

public interface Functor {
  public String getNamespace();
  public String getName();
  public List<String> getProducts();
  public List<String> getParameters();
  public List<String> getDependencies();
  public void call(Map<String,String> parameters) throws RuntimeException;
}

// *EOF*
```

AbstractFunctor.groovy

```
// @file     AbstractFunctor.groovy
// @author   Mac Radigan
```

```
package org.radigan.naomi.wumpus.service
import org.radigan.naomi.service.Module

public abstract class AbstractFunctor implements Functor, Module {
  public static final String DEFAULT_NAMESPACE = "ns1"
  public abstract void initialize()
  public abstract String getNamespace()
  public abstract String getName()
  public abstract List<String> getProducts();
  public abstract List<String> getParameters();
  public abstract List<String> getDependencies();
  public abstract void call(Map<String,String> parameters) throws RuntimeException;
  public String toString() {
    def sb = new StringBuilder()
    sb << "[${getProducts().join(',')}]"
    if(getNamespace()==DEFAULT_NAMESPACE) {
      sb << "=${getNamespace()}:${getName()}"
    } else {
      sb << "=${getName()}"
    }
    sb << "([${getDependencies().join(',')}])"
    return sb.toString()
  }
}

// *EOF*
```

FunctorList.groovy

```
// @file     FunctorList.groovy
// @author   Mac Radigan

package org.radigan.naomi.wumpus.service

import org.radigan.naomi.wumpus.service.Functor
import org.radigan.naomi.service.Module
import java.util.ArrayList

public class FunctorList extends ArrayList<Functor> implements Module {
  public void initialize() {}
}

// *EOF*
```

1.9 Appendix B.3.2: WUMPUS Implementations

SimulationFunctor.groovy

```
// @file     SimulationFunctor.groovy
// @author   Mac Radigan

package org.radigan.naomi.wumpus.impl

import org.apache.log4j.Logger
import org.radigan.naomi.wumpus.service.AbstractFunctor
```

```
public class SimulationFunctor extends AbstractFunctor {
  def log = Logger.getLogger(SimulationFunctor.class)
  def namespace = DEFAULT_NAMESPACE
  def name = ""
  def dependencies = []
  def products = []
  def parameters = []
  public SimulationFunctor() {} // SPI support
  public SimulationFunctor(
    String name,
    List<String> products,
    List<String> dependencies,
    List<String> parameters) {
    this.name = name
    this.dependencies = dependencies
    this.products = products
    this.parameters = parameters
  }
  public SimulationFunctor(
    String namespace,
    String name,
    List<String> products,
    List<String> dependencies,
    List<String> parameters) {
    this.namespace = namespace
    this.name = name
    this.dependencies = dependencies
    this.products = products
    this.parameters = parameters
  }
  public void initialize() {
  }
  public String getNamespace() {
    return namespace
  }
  public String getName() {
    return name
  }
  public String getQname() {
    return "${namespace}:${name}"
  }
  public List<String> getProducts() {
    return products
  }
  public List<String> getParameters() {
    return parameters
  }
  public List<String> getDependencies() {
    return dependencies
  }
  public void call(Map<String,String> parameters) throws RuntimeException {
    log.info("calling ${getName()} with parameters: ${parameters}")
  }
}
```

```
// *EOF*
```

WumpusImpl.groovy

```groovy
// @file     WumpusImpl.groovy
// @author   Mac Radigan

package org.radigan.naomi.wumpus.impl

import org.radigan.naomi.wumpus.service.Wumpus
import org.radigan.naomi.wumpus.service.Functor
import org.radigan.naomi.wumpus.service.FunctorList
import org.radigan.naomi.wumpus.utilities.WumpusUtil
import org.radigan.naomi.utilities.ServiceFactory
import org.apache.log4j.Logger

public class WumpusImpl implements Wumpus {

  protected ServiceFactory serviceFactory = null
  protected WumpusUtil wumpusUtil = null
  protected Logger log = null
  protected List<Functor> functors = []

  public WumpusImpl() {
    serviceFactory = ServiceFactory.getInstance()
    log = serviceFactory.getLogger(this)
    initializeFunctors()
  }

  public initializeFunctors() {
    serviceFactory.loadModules()
    serviceFactory.getModules(FunctorList).each { m ->
      def f = m.newInstance()
      f.initialize()
      functors.addAll(f)
    }
    serviceFactory.getModules(Functor).each { m ->
      def f = m.newInstance()
      f.initialize()
      functors << f
    }
    wumpusUtil = new WumpusUtil(functors)
  }

  public WumpusUtil getWumpusUtil() {
    return wumpusUtil
  }

  public String toString() {
    def sb = new StringBuilder()
    sb << getWumpusUtil()
    return sb.toString()
  }

}
```

```
// *EOF*
```

WumpusUtil.groovy

```
// @file      WumpusUtil.groovy
// @author    Mac Radigan

package org.radigan.naomi.wumpus.utilities

import org.radigan.system.utilities.MatrixImpl
import org.radigan.system.utilities.Matrix
import org.radigan.naomi.wumpus.service.Functor
import org.radigan.naomi.wumpus.impl.SimulationFunctor
import org.radigan.system.utilities.Md5
import java.util.List

public class WumpusUtil {

  Map<Integer,List<Integer> > r = [:] // representatives
  List<Functor> f = []                // functors
  List<String> e = []                 // edges
  MatrixImpl w = null                 // oriented incident matrix
  MatrixImpl wu = null                // unit oriented incident matrix
  MatrixImpl l = null                 // Laplacian matrix
  MatrixImpl a = null                 // adjacency matrix
  MatrixImpl ap = null                // positive flux adjacency matrix
  MatrixImpl an = null                // negative flux adjacency matrix
  MatrixImpl d = null                 // degree matrix
  int n = 0                           // determine number of nodes
  MatrixImpl c = null                 // transitive closure
  double ac = 0.0                     // algebraic connectivity
  boolean con = false                 // connectivity
  List<Integer> ss = []               // source nodes
  List<Integer> s = []                // starting nodes
  List<Integer> t = []                // sink nodes

  public WumpusUtil(Matrix w, List<Functor> f, List<String> e) {
    this.f = f
    this.e = e
    eval(w)
  }
  public WumpusUtil(List<Functor> f) { eval(f) }

  protected void eval(Matrix w) {
    this.w = w                  // oriented incident matrix, W
    wu = w.signum()             // unit oriented incident matrix, Wu = sign(W)
    l = wu.t()*wu               // Laplacian matrix, L = Wu'*Wu
    a = (l**(l<<0)).abs()       // adjacency matrix, A = abs(L.*(L<0))
    d = l+a                     // degree matrix, D
    n = d.lastNonzeroRow()      // determine number of nodes, N
    def sm = MatrixImpl.zeros(a).fill(MatrixImpl.eye(a))
    c = (a+sm)^n                // transitive closure, C=(A+s)^N
    ac = l.eigW(1)              // algebraic connectivity, ac = eigW(L, 2)
    con = ac>0                  // check connectivity, con = ac>0
    ss = ((wu.min()<<0)&(wu.max()<<1)).find() // source nodes, SS
```

```
  s = c.indToRows(c.find(ss,[n]))           // starting nodes, S
  ap = MatrixImpl.zeros(a)   // positive flux adjacency matrix
  an = MatrixImpl.zeros(a)   // negative flux adjacency matrix
  def edges = getEdges()
  [edges['in'],edges['out']].transpose().each { p ->
    ap.setEntry(p[0],p[1], ap.getEntry(p[0],p[1])+1)
    an.setEntry(p[1],p[0], an.getEntry(p[1],p[0])+1)
  }
  //ap = a**(wu>>0)               // positive flux adjacency matrix, Ap = A.*(Wu>0)
  //an = a**(wu.t()>>0)           // negative flux adjacency matrix, An = A.*(Wu<0)
  t = [traverse(s[0], +1).last()] // sink nodes, t
}

protected void eval(List<Functor> f) {
  this.f = f
  def edg = []
  for(fd in 0 .. f.size()-1) {
    for(fp in 0 .. f.size()-1) {
      //f[fd].getDependencies().unique().each { dn ->
      f[fd].getDependencies().each { dn ->
        def deg = f[fp].getProducts().count{it==dn}
        if(deg) edg << [fd, fp, deg, dn]
      }
    }
  }
  this.w = MatrixImpl.zeros(edg.size(),f.size())
  this.e = []
  edg.eachWithIndex { en, ind ->
    e << en[3]
    w.setEntry(ind,en[1],-1*en[2])
    w.setEntry(ind,en[0],+1*en[2])
  }
  eval(w)
}

public WumpusUtil filter(int n) {
  def retF = []
  traverse(n,-1).each { ind -> retF << f[ind] }
  def tempF = f.clone()
  tempF.retainAll(retF)
  return new WumpusUtil(tempF)
  //def tempW = w.sub(traverse(n,-1).reverse(), w.getColRange())
  //return new WumpusUtil(tempW, tempF, e)
}

public List<Integer> traverse(int n, int flux) {
  def ax = (flux>0) ? ap : an
  return [n] + traverse(n, ax)
}
private List<Integer> traverse(int n, Matrix ax) {
  def rval = []
  def l = ax.find([n], ax.getColRange())
  if(l) ax.indToCols(l).each { ln -> rval += [ln] + traverse(ln, ax) }
  return rval
```

```
}

public List<Integer> step(int n, int flux) {
  def rval = []
  def ax = (flux>0) ? ap : an
  def l = ax.find([n], ax.getColRange())
  if(l) ax.indToCols(l).each { ln -> rval += [ln] }
  return rval
}

public List<Integer> getNodes() {
  def rval = []
  s.each { n -> rval += traverse(n, +1) }
  return rval
}

public void execute(Map<String,Integer> sv) {
  def svc = MatrixImpl.zeros(e.size(),1)
  e.eachWithIndex { en, ind ->
    if(sv[en]) {
      def val = svc.getEntry(ind,0)
      val+=sv[en]
      svc.setEntry(ind,0,val)
    }
  }
  def rho = w*svc
  println "w:\n${w}"
  println "svc:\n${svc}"
  println "rho:\n${rho}"
}
public void execute(int fn) {
  println f[fn].toString()
  ap.getRow(fn).eachWithIndex { v, n ->
    if(v>0) execute(n)
  }
}
public void execute() {
  s.each { n -> execute(n) }
}

public void connectedComponents() {
  f.eachWithIndex { fn, v -> makeSet(v) }
  e.eachWithIndex { en, n ->
    def wp = w>>0
    def u = (wp.find([n], wp.getColRange()))[0]
    def wn = w<<0
    def v = (wn.find([n], wn.getColRange()))[0]
    if(findSet(u)==findSet(v)) union(u,v)
  }
}
public boolean sameComponent(int u, int v) {
  return findSet(u)==findSet(v)
}
public void makeSet(int x) {
```

```
  r[x] = [x]
}
public void clearSet() {
  r.clear()
}
public void union(int x, int y) {
  def sx = null
  def sy = null
  r.each { k, v ->
    if(v.contains(x)) sx = k
    if(v.contains(y)) sy = k
  }
  if(!sx) throw new IllegalStateException("Representative not found for ${x}")
  if(!sy) throw new IllegalStateException("Representative not found for ${y}")
  r[sx] = r[sx]+r[sy]
  r.remove(sy)
}
public int findSet(int x) {
  r.each { k, v -> if(v.contains(x)) return k }
  throw new IllegalStateException("No such representative: ${x}")
}

public Map getEdges() {
  def fi = []
  def fo = []
  w.getRowRange().each { row ->
    w.getColRange().each { col ->
      if(w.getEntry(row,col)<0) fi << col
      if(w.getEntry(row,col)>0) fo << col
    }
  }
  return ['name':e, 'in':fi, 'out':fo]
}

public Matrix getWumpus() { return w }
public Matrix getUnitWumpus() { return wu }
public Matrix getLaplacian() { return l }
public Matrix getAdjacency() { return a }
public Matrix getPositiveFlux() { return ap }
public Matrix getNegativeFlux() { return an }
public Matrix getDegree() { return d }
public Matrix getTransitiveClosure() { return c }
public double getAlgebraicConnectivity() { return ac }
public boolean isConnected() { return con }
public List<Integer> getSources() { return ss }
public List<Integer> getSinks() { return t }
public List<Integer> getStart() { return s }
public String getEdge(int n) { return e[n] }
public List<Functor> getFunctors(String namespace=SimulationFunctor.DEFAULT_NAMESPACE) {
  def rval = []
  f.each { fn -> if(fn.getNamespace()==namespace) rval<<fn }
  return rval
}
public List<Functor> getFunctor(String name, String namespace=SimulationFunctor.DEFAULT_NAMESPACE) {
```

```
    def rval = []
    f.each { fn -> if(fn.getName()==name && fn.getNamespace()==namespace) rval<<fn }
    return rval
  }
  public Integer getFunctorIndex(String name, String namespace=SimulationFunctor.DEFAULT_NAMESPACE) {
    Integer index = null
    f.eachWithIndex { fn, ind ->
      if(fn.getName()==name && fn.getNamespace()==namespace) index = ind
    }
    return index
  }
  Map<Integer,List<Integer> > getRepresentatives() { return r }

  public String toString() {
    def sb = new StringBuilder()
    sb << "f:\n"
    f.each { sb << "${it}\n" }
    sb << "e: ${e.join(',')}\n"
    sb << "W:\n${w}"
    sb << "Wu:\n${wu}"
    sb << "L:\n${l}"
    sb << "A:\n${a}"
    sb << "An:\n${an}"
    sb << "Ap:\n${ap}"
    sb << "D:\n${d}"
    sb << "C:\n${c}"
    sb << "N: ${n}\n"
    sb << "ac: ${ac}\n"
    sb << "con: ${con}\n"
    sb << "SS: ${ss}\n"
    sb << "S: ${s}\n"
    return sb.toString()
  }

  public String getId() {
    def sb = new StringBuffer()
    sb << 'f:=' << f*.getQname().join(',')
    sb << '&e:=' << e.join(',')
    sb << '&w:=' << '['
    w.getRowRange().eachWithIndex { row, rind ->
      w.getColRange().eachWithIndex { col, cind ->
        sb << "${w.getEntry(rind,cind)}"
        if(cind!=w.getCols()-1) sb << ','
      }
      if(rind!=w.getRows()-1) sb << ';'
    }
    sb << ']'
    return Md5.encode(sb.toString())
  }

}

// *EOF*
```

1.10 Appendix B.4: NYANCAT Sources

1.11 Appendix B.4.1: NYANCAT Interfaces

Report.java

```
// @file     Report.java
// @author   Mac Radigan

package org.radigan.naomi.nyancat.service;

import java.util.List;
import java.io.File;

public interface Report {
  public String toString();
  public void save(File filename);
  public void save(String filename);
}

// *EOF*
```

AbstractReport.groovy

```
// @file     AbstractReport.java
// @author   Mac Radigan

package org.radigan.naomi.nyancat.service

import org.radigan.naomi.service.Module
import org.radigan.naomi.nyancat.service.Report
import org.radigan.system.configuration.Configuration
import org.radigan.system.utilities.ResourceManager
import org.radigan.naomi.utilities.ServiceFactory
import org.radigan.system.utilities.Xsp
import org.radigan.system.utilities.Shell
import java.io.File

public abstract class AbstractReport implements Report, Module {
  protected ServiceFactory serviceFactory = null
  protected Configuration configuration = null
  protected ResourceManager resourceManager = null
  protected Xsp xsp = new Xsp()
  protected Shell shell = new Shell()
  public AbstractReport() {
    serviceFactory = ServiceFactory.getInstance()
    configuration = Configuration.getInstance()
    resourceManager = configuration.getResourceManager()
  }
  public abstract void initialize()
  public abstract String toString()
  public abstract void save(File filename);
  public void save(String filename) {
    save(new File(filename));
  }
}

// *EOF*
```

1.12 Appendix B.4.1: NYANCAT Implementations

GraphvizReport.groovy

```
// @file     GraphvizReport.java
// @author   Mac Radigan

package org.radigan.naomi.nyancat.impl

import org.radigan.naomi.nyancat.service.AbstractReport
import org.radigan.naomi.wumpus.utilities.WumpusUtil

public class GraphvizReport extends AbstractReport {
  protected final String TEMPLATE_PATH = "/org/radigan/naomi/data/reports/graphviz.dot"
  protected String template = ""
  protected WumpusUtil util = null
  protected final static String DEFAULT_FORMAT = "gif"

  public GraphvizReport() { // SPI support
    super()
  }
  public GraphvizReport(WumpusUtil util) {
    super()
    template = resourceManager.getText(TEMPLATE_PATH)
    this.util = util
  }

  public void initialize() { }

  public String toString() {
    return xsp.process(template, ['util':util])
  }

  public void save(File filename, String format=DEFAULT_FORMAT) {
     def tempFile = File.createTempFile("temp",".dot")
     //def tempFile = new File("/var/tmp/test.dot")
     tempFile.write(toString())
     def exe = configuration.searchBin(new File("dot"))
     def timeout = 5*60*1000
     def cmd = "${exe} -T${format}:cairo -o ${filename} ${tempFile}"
     shell.execute(cmd)
     tempFile.delete()
  }
}

// *EOF*
```

OctaveReport.groovy

```
// @file     OctaveReport.java
// @author   Mac Radigan

package org.radigan.naomi.nyancat.impl

import org.radigan.naomi.nyancat.service.AbstractReport
import org.radigan.naomi.wumpus.utilities.WumpusUtil

public class OctaveReport extends AbstractReport {
```

```
  protected final String TEMPLATE_PATH = "/org/radigan/naomi/data/reports/octave.m"
  protected String template = ""
  protected WumpusUtil util = null

  public OctaveReport() { // SPI support
    super()
  }
  public OctaveReport(WumpusUtil util) {
    super()
    template = resourceManager.getText(TEMPLATE_PATH)
    this.util = util
  }

  public void initialize() { }

  public String toString() {
    return xsp.process(template, ['util':util])
  }

  public void save(File filename) {
     filename << toString()
  }

}

// *EOF*
```

WumpusReport.groovy

```
// @file     WumpusReport.java
// @author   Mac Radigan

package org.radigan.naomi.nyancat.impl

import org.radigan.naomi.nyancat.service.AbstractReport
import org.radigan.naomi.wumpus.utilities.WumpusUtil

public class WumpusReport extends AbstractReport {
  protected final String TEMPLATE_PATH = "/org/radigan/naomi/data/reports/wumpus.html"
  protected String template = ""
  protected WumpusUtil util = null

  public WumpusReport() { // SPI support
    super()
  }
  public WumpusReport(WumpusUtil util) {
    super()
    template = resourceManager.getText(TEMPLATE_PATH)
    this.util = util
  }

  public void initialize() { }

  public String toString() {
    return xsp.process(template, ['util':util])
  }
```

```
  public void save(File filename) {
     filename << toString()
  }

}

// *EOF*
```

1.13 Appendix B.4.3: NYANCAT Resources

graphviz.dot

```
<%
  sources = util.getStart()
  nodes = util.getNodes()
  sinks = util.getSinks()
  functors = nodes-sources-sinks
  Ap = util.getPositiveFlux()
  edges = util.getEdges()
  ename = edges['name']
  ei = edges['in']
  eo = edges['out']
  f = util.getFunctors()
  fname = f[sinks[0]].getName()
  namespace = f[sinks[0]].getNamespace()
  qname = f[sinks[0]].getQname()
  wfid = util.getId()
%>
// ${fname}.dot
// Mac Radigan
// workflow: ${fname} (${namespace}) [${wfid}]
digraph G {
 labelloc="t";
 label="Workflow ${fname} (${namespace})";
 compound=true;
 nodesep=0.5;
 rankdir=LR;

// sinks
 subgraph cluster_sinks {
  label = "sinks";
<% sinks.each { n -> %> f${n} [label="${f[n].getName()}", fillcolor=green, style="rounded,filled", shap
<% } %>
 }

// functors
 subgraph cluster_functors {
  label = "functors";
<% functors.each { n -> %> f${n} [label="${f[n].getName()}", fillcolor=darkslategray1, style="filled",
<% } %>
 }

// sources
 subgraph cluster_sources {
```

```
<% sources.each { n -> %> label = "sources";
  f${n} [label="${f[n].getName()}", fillcolor=yellow, style="rounded,filled", shape=doublecircle];
<% } %>
 }

// edges
<% ename.eachWithIndex { e, n -> %> f${ei[n]} -> f${eo[n]} [label="${e}"];
<% } %>

}
// *EOF*
```

octave.m

```
<%
  sources = util.getStart()
  nodes = util.getNodes()
  sinks = util.getSinks()
  functors = nodes-sources-sinks
  f = util.getFunctors()
  edges = util.getEdges()
  ename = edges['name']
  ei = edges['in']
  eo = edges['out']
  fname = f[sinks[0]].getName()
  namespace = f[sinks[0]].getNamespace()
  qname = f[sinks[0]].getQname()
  fn = f[sinks[0]]
  wfid = util.getId()
  getMatrix = { m ->
    def sb = new StringBuffer()
    sb << "["
    m.getRowRange().eachWithIndex { row, rind ->
      m.getColRange().eachWithIndex { col, cind ->
        sb << "${(int)m.getEntry(rind,cind)}"
        if(cind!=m.getCols()-1) sb << ","
      }
      if(rind!=m.getRows()-1) sb << ";"
    }
    sb << "]"
    return sb.toString()
  }
%>
% £{fname}.m
% Mac Radigan
% workflow: £{fname} (£{namespace}) [£{wfid}]

% storage
w = {};
% workflow id
w{end+1}.wfid = '${wfid}';

% products
w{end}.products = {'${fn.getProducts().join("','")}'};

% dependencies
```

```
w{end}.dependencies = {'${fn.getDependencies().join("','")}'};

% parameters
w{end}.parameters = {'${fn.getParameters().join("','")}'};

% algebraic connectivity
w{end}.ac = ${util.getAlgebraicConnectivity()};

% Oriented Incident Matrix
w{end}.W = ${getMatrix(util.getWumpus())};

% Unit Oriented Incident Matrix
w{end}.Wu = ${getMatrix(util.getWumpus())};

% Laplacian Matrix
w{end}.L = ${getMatrix(util.getLaplacian())};

% Degree Matrix
w{end}.D = ${getMatrix(util.getDegree())};

% Adjacency Matrix
w{end}.A = ${getMatrix(util.getAdjacency())};

% Positive Flux Adjacency Matrix
w{end}.Ap = ${getMatrix(util.getPositiveFlux())};

% Negative Flux Adjacency Matrix
w{end}.An = ${getMatrix(util.getNegativeFlux())};

% Transitive Closure Matrix
w{end}.C = ${getMatrix(util.getTransitiveClosure())};

% *EOF*
```

wumpus.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
<%
  sources = util.getStart()
  nodes = util.getNodes()
  sinks = util.getSinks()
  functors = nodes-sources-sinks
  f = util.getFunctors()
  edges = util.getEdges()
  ename = edges['name']
  ei = edges['in']
  eo = edges['out']
  fname = f[sinks[0]].getName()
  namespace = f[sinks[0]].getNamespace()
  qname = f[sinks[0]].getQname()
  fn = f[sinks[0]]
  wfid = util.getId()
  getFunction = { f ->
    def sb = new StringBuffer()
    sb << "\\left(${f.getProducts().join(', ')}\\right)"
    sb << " & \\leftarrow \\left( ${f.getQname()}, "
```

```
    sb << "\\{ ${f.getDependencies().join(', ')} \\}, "
    sb << "\\{ ${f.getParameters().join(', ')} \\}"
    sb << "\\right)"
    return sb.toString()
  }
  getMatrix = { m ->
    def sb = new StringBuffer()
    sb << "\\left["
    sb << "\\begin{array}{c}"
    m.getRowRange().eachWithIndex { row, rind ->
      m.getColRange().eachWithIndex { col, cind ->
        sb << "${(int)m.getEntry(rind,cind)}"
        if(cind!=m.getCols()-1) sb << " & "
      }
      if(rind!=m.getRows()-1) sb << "\\\\"
    }
    sb << "\\end{array}"
    sb << "\\right]"
    return sb.toString()
  }
%>
<!-- £{fname}.html -->
<!-- Mac Radigan -->
<!-- workflow: £{fname} (£{namespace}) [£{wfid}] -->
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Workflow ${fname} (${namespace}) [${wfid}]</title>
    <meta name="author" content="Mac Radigan">
    <meta name="description" content="WUMPUS html interfae for ${fname} (${namespace}) [${wfid}]">
    <meta name="keywords" content="WUMPUS ${namespace} ${fname} ${wfid}">
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <script type="text/x-mathjax-config">
      MathJax.Hub.Config({ TeX: { equationNumbers: {autoNumber: "all"} } });
      </script>
      <script type="text/javascript" src="http://cdn.mathjax.org/mathjax/latest/MathJax.js?config=TeX-A
      <style>
      h1 {
         background: #CCCCCC;
           padding: .2em 1em;
             border-top: 3px solid #666666;
               border-bottom: 3px solid #999999;
      }
      </style>
  </head>
  <body>
    <center><h1>Workflow ${fname} (${namespace})</h1></center>
    <div style="padding:0 2em">

      <hr>
        <center><img src="http://localhost:8080/roar/${namespace}/${fname}/interface/${fname}.gif" alt=
      <p>
```

```
      <hr>
        Workflow:
        \\begin{equation}
          \\begin{split}
<% nodes.each { n -> %> ${getFunction(f[n])} \\\\
<% } %>
          \\end{split}
          \\label{eq:productions}
        \\end{equation}
      <p>

      <hr>
        Name: ${fname}
        <br>
        Namespace: ${namespace}
        <br>
        Unique Id: ${wfid}
        <br>
        Products: ${fn.getProducts().join(", ")}
        <br>
        Dependencies: ${fn.getDependencies().join(", ")}
        <br>
        Parameters: ${fn.getParameters().join(", ")}
        <br>
        Algebraic Connectivity: ${util.getAlgebraicConnectivity()}
        <br>
      <p>

    <center><h1>Workflow Unified as a single Martrix for Processing Unlimited Services (WUMPUS)</h1></c

      <hr>
        Oriented Incident Matrix
        \\begin{equation}
          \\mathbb{W} = ${getMatrix(util.getWumpus())}
          \\label{eq:W}
        \\end{equation}
      <p>

      <hr>
        Unit Oriented Incident Matrix
        \\begin{equation}
          \\mathbb{W}_{u} = ${getMatrix(util.getUnitWumpus())}
          \\label{eq:Wu}
        \\end{equation}
      <p>

      <hr>
        Laplacian
        \\begin{equation}
          \\mathbb{L} = ${getMatrix(util.getLaplacian())}
          \\label{eq:L}
        \\end{equation}
      <p>
```

```
      <hr>
        Degree
        \\begin{equation}
          \\mathbb{D} = ${getMatrix(util.getDegree())}
          \\label{eq:D}
        \\end{equation}
      <p>

      <hr>
        Adjacency
        \\begin{equation}
          \\mathbb{A} = ${getMatrix(util.getAdjacency())}
          \\label{eq:A}
        \\end{equation}
      <p>

      <hr>
        Positive Flux Adjacency
        \\begin{equation}
          \\mathbb{A}^{+} = ${getMatrix(util.getPositiveFlux())}
          \\label{eq:Ap}
        \\end{equation}
      <p>

      <hr>
        Negative Flux Adjacency
        \\begin{equation}
          \\mathbb{A}^{-} = ${getMatrix(util.getNegativeFlux())}
          \\label{eq:An}
        \\end{equation}
      <p>

      <hr>
        Transitive Closure
        \\begin{equation}
          \\mathbb{C}^{*} = ${getMatrix(util.getTransitiveClosure())}
          \\label{eq:C}
        \\end{equation}
      <p>


    </div>
  </body>
</html>
```

1.14 Appendix B.5: RUTH Sources

1.15 Appendix B.5.1: RUTH Interfaces

1.16 Appendix B.5.2: RUTH Implementations

1.17 Appendix B.6: Base Sources

1.18 Appendix B.6.1: Base Interfaces

Tool.java

```
// @file     Tool.java
// @author   Mac Radigan

package org.radigan.system.tools;

import groovy.util.OptionAccessor;

public interface Tool {
  public OptionAccessor parse(String[] args);
  public int process(String[] args);
  public void initialize();
  public String getName();
  public String getDescription();
  public int run();
}
// *EOF*
```

Matrix.java

```
// @file     Matrix.java
// @author   Mac Radigan

package org.radigan.system.utilities;

import java.util.List;

public interface Matrix {
  public Matrix pow(int n);
  public double eigW(int index);
  public Matrix transpose();
  public String toString();
  public Matrix mult(Matrix m);
  public Matrix multElementwise(Matrix m);
  public Matrix add(Matrix m);
  public Matrix signum();
  public Matrix gt(double th);
  public Matrix lt(double th);
  public Matrix abs();
  public int lastNonzeroRow();
  public Matrix fill(Matrix m);
  public Matrix min();
  public Matrix max();
  public Matrix sum();
  public Matrix land(Matrix a);
  public List<Integer> indToRows(List<Integer> ind);
  public List<Integer> indToCols(List<Integer> ind);
  public List<Integer> find();
```

```
  public List<Integer> find(List<Integer> rows, List<Integer> cols);
  public int getRows();
  public List<Integer> getRowRange();
  public int getCols();
  public List<Integer> getColRange();
  public double getEntry(int row, int col);
  public void setEntry(int row, int col, double val);
  public List<Matrix> qr(Matrix a);
  public Matrix solve(Matrix b);
  public Matrix inverse();
  public Matrix sub(List<Integer> rows, List<Integer> cols);
  public List<Integer> size();
}

// *EOF*
```

1.19 Appendix B.6.2: Base Implementations

MatrixImpl.groovy

```
// @file     Matrix.groovy
// @author   Mac Radigan

package org.radigan.system.utilities

import org.apache.commons.math3.linear.RealMatrix
import org.apache.commons.math3.linear.Array2DRowRealMatrix
import org.apache.commons.math3.linear.EigenDecomposition
import org.apache.commons.math3.linear.MatrixUtils
import org.apache.commons.math3.linear.DecompositionSolver
import org.apache.commons.math3.linear.QRDecomposition
import org.apache.log4j.Logger

public class MatrixImpl implements Matrix {

  RealMatrix m = null

  public MatrixImpl(double[][] data) {
    m = new Array2DRowRealMatrix(data, true)
  }
  public MatrixImpl(Matrix m) {
    this.m = new Array2DRowRealMatrix(m.getData(), true)
  }
  public MatrixImpl(RealMatrix m) {
    this.m = m
  }

  public int getRows() {
    m.getRowDimension()
  }
  public List<Integer> getRowRange() {
    return 0 .. (getRows()-1)
  }
  public int getCols() {
    m.getColumnDimension()
```

```
}
public List<Integer> getColRange() {
  return 0 .. (getCols()-1)
}
public void setEntry(int row, int col, double val) {
  m.setEntry(row, col, val)
}
public double getEntry(int row, int col) {
  return m.getEntry(row, col)
}

public RealMatrix getUnderlying() {
  return m
}

public Matrix inverse() {
  // https://issues.apache.org/jira/browse/MATH-858
  return new MatrixImpl(new QRDecomposition(m).getSolver().getInverse())
}

public Matrix pow(int n) {
  if(n==1) {
    return this
  } else if(n>=0) {
    return new MatrixImpl(m.power(n))
  } else {
    return inverse().pow(-1*n)
  }
}
public Matrix xor(int n) {
  return pow(n)
}

public double eigW(int index) {
  final eigd = new EigenDecomposition(m, 0.001)
  return eigd.getRealEigenvalue(index)
}

public Matrix transpose() {
  return new MatrixImpl(m.transpose())
}
public Matrix t() {
  return transpose()
}

public Matrix mult(Matrix m) {
  return new MatrixImpl(this.m.multiply(m.getUnderlying()))
}
public Matrix multiply(Matrix m) {
  return mult(m)
}

public List<Integer> size() {
  return [getRows(), getCols()]
```

```
}

public Matrix multElementwise(Matrix m) {
  def rval = MatrixUtils.createRealMatrix(getRows(), getCols())
  for(int row=0; row<getRows(); row++) {
    for(int col=0; col<getCols(); col++) {
      rval.setEntry(row,col,this.m.getEntry(row,col)*m.getEntry(row,col))
    }
  }
  return new MatrixImpl(rval)
}
public Matrix power(Matrix m) {
  return multElementwise(m)
}

public Matrix add(Matrix m) {
  return new MatrixImpl(this.m.add(m.getUnderlying()))
}
public Matrix plus(Matrix m) {
  return add(m)
}

public static Matrix vertcat(List<Matrix> m) {
  return null
}

public static Matrix horzcat(List<Matrix> m) {
  return null
}

public Matrix gt(double th) {
  def rval = MatrixUtils.createRealMatrix(m.getRowDimension(), m.getColumnDimension())
  for(int row=0; row<m.getRowDimension(); row++) {
    for(int col=0; col<m.getColumnDimension(); col++) {
      if(m.getEntry(row,col)>th) rval.setEntry(row,col,1)
    }
  }
  return new MatrixImpl(rval)
}
public Matrix rightShift(double th) {
  return gt(th)
}

public Matrix lt(double th) {
  def rval = MatrixUtils.createRealMatrix(m.getRowDimension(), m.getColumnDimension())
  for(int row=0; row<m.getRowDimension(); row++) {
    for(int col=0; col<m.getColumnDimension(); col++) {
      if(m.getEntry(row,col)<th) rval.setEntry(row,col,1)
    }
  }
  return new MatrixImpl(rval)
}
public Matrix leftShift(double th) {
  return lt(th)
```

```
}

public Matrix signum() {
  def rval = MatrixUtils.createRealMatrix(m.getRowDimension(), m.getColumnDimension())
  for(int row=0; row<m.getRowDimension(); row++) {
    for(int col=0; col<m.getColumnDimension(); col++) {
      if(m.getEntry(row,col)>0) rval.setEntry(row,col,+1)
      if(m.getEntry(row,col)<0) rval.setEntry(row,col,-1)
      if(m.getEntry(row,col)==0) rval.setEntry(row,col,0)
    }
  }
  return new MatrixImpl(rval)
}

public Matrix abs() {
  def rval = MatrixUtils.createRealMatrix(m.getRowDimension(), m.getColumnDimension())
  for(int row=0; row<m.getRowDimension(); row++) {
    for(int col=0; col<m.getColumnDimension(); col++) {
      rval.setEntry(row,col,m.getEntry(row,col).abs())
    }
  }
  return new MatrixImpl(rval)
}

public Matrix fill(Matrix m) {
  def rval = new MatrixImpl(getUnderlying())
  for(int row=0; row<m.getRows(); row++) {
    for(int col=0; col<m.getCols(); col++) {
      rval.setEntry(row,col,m.getEntry(row,col))
    }
  }
  return rval
}

public Matrix min() {
  def rval = MatrixUtils.createRealMatrix(1, getCols())
  for(int col=0; col<getCols(); col++) {
    def val = Double.MAX_VALUE
    for(int row=0; row<getRows(); row++) {
      def cur = m.getEntry(row,col)
      if(cur<val) val = cur
    }
    rval.setEntry(0,col,val)
  }
  return new MatrixImpl(rval)
}
public Matrix max() {
  def rval = MatrixUtils.createRealMatrix(1, getCols())
  for(int col=0; col<getCols(); col++) {
    def val = Double.MIN_VALUE
    for(int row=0; row<getRows(); row++) {
      def cur = m.getEntry(row,col)
      if(cur>val) val = cur
    }
```

```
    rval.setEntry(0,col,val)
  }
  return new MatrixImpl(rval)
}

public Matrix sum() {
  def rval = MatrixUtils.createRealMatrix(1, getCols())
  for(int col=0; col<getCols(); col++) {
    def val = 0
    for(int row=0; row<getRows(); row++) {
      val += m.getEntry(row,col)
    }
    rval.setEntry(0,col,val)
  }
  return new MatrixImpl(rval)
}

public Matrix land(Matrix a) {
  def rval = MatrixUtils.createRealMatrix(m.getRowDimension(), m.getColumnDimension())
  for(int row=0; row<m.getRowDimension(); row++) {
    for(int col=0; col<m.getColumnDimension(); col++) {
      def mval = m.getEntry(row,col)
      def aval = a.getEntry(row,col)
      mval&&aval ? rval.setEntry(row,col,1) : rval.setEntry(row,col,0)
    }
  }
  return new MatrixImpl(rval)
}
public Matrix and(Matrix a) {
  return land(a)
}

public List<Integer> find(List<Integer> rows, List<Integer> cols) {
  def rval = new ArrayList<Integer>()
  rows.each { row ->
    cols.each { col ->
      //def ind = row*(getCols()-1)+col
      def ind = row*getCols()+col
      if(m.getEntry(row,col)) rval << ind
    }
  }
  return rval
}
public List<Integer> find() {
  def rval = new ArrayList<Integer>()
  def ind = 0
  for(int row=0; row<getRows(); row++) {
    for(int col=0; col<getCols(); col++) {
      if(m.getEntry(row,col)) rval << ind
      ind++
    }
  }
  return rval
}
```

```
public List<Integer> indToRows(List<Integer> ind) {
  return ind.collect{(int)Math.floor(it/getRows())}
}
public List<Integer> indToCols(List<Integer> ind) {
  return ind.collect{(int)Math.floor(it%getCols())}
}

public String toString() {
  def sb = new StringBuffer()
  for(int row=0; row<m.getRowDimension(); row++) {
    for(int col=0; col<m.getColumnDimension(); col++) {
      sb << String.format('%1$04.2f ', m.getEntry(row,col)).padLeft(6)
    }
    sb << String.format('%n')
  }
  return sb.toString()
}

public int lastNonzeroRow() {
  def last = 0
  for(int row=0; row<m.getRowDimension(); row++) {
    def sum = 0
    for(int col=0; col<m.getColumnDimension(); col++) {
      sum += m.getEntry(row,col).abs()
    }
    if(sum) last = row
  }
  return last
}

public static Matrix eye(int n) {
  return new MatrixImpl(MatrixUtils.createRealIdentityMatrix(n))
}
public static Matrix eye(Matrix a) {
  def rval = new MatrixImpl(MatrixUtils.createRealMatrix(a.getRows(), a.getCols()))
  for(int d=0; d<rval.getRows(); d++) {
    rval.setEntry(d,d,1)
  }
  return rval
}

public static Matrix zeros(int m, int n) {
  return new MatrixImpl(MatrixUtils.createRealMatrix(m,n))
}
public static Matrix zeros(Matrix a) {
  return new MatrixImpl(MatrixUtils.createRealMatrix(a.getRows(), a.getCols()))
}

public List getRow(int n) {
  return m.getRowVector(n).toArray()
}
public List getCol(int n) {
  return m.getColumnVector(n).toArray()
}
```

```
  public Matrix solve(Matrix b) {
    return new MatrixImpl(new QRDecomposition(m).getSolver().solve(b))
  }
  public Matrix divide(Matrix b) {
    return solve(b)
  }

  public List<Matrix> qr(Matrix a) {
    def qrd = new QRDecomposition(m)
    def rval = []
    return rval << qrd.getQ() << qrd.getR()
  }

  public Matrix sub(List<Integer> rows, List<Integer> cols) {
    return new MatrixImpl(m.getSubMatrix(rows.toArray() as int[],cols.toArray() as int[]))
  }
  public Matrix filter(List<Integer> rows, List<Integer> cols) {
    def nrows = getRowRange().removeAll(rows)
    def ncols = getColRange().removeAll(cols)
    def rval = MatrixUtils.createRealMatrix(nrows.size(), ncols.size())
    nrows.eachWithIndex { row, rind ->
      ncols.eachWithIndex { col, cind ->
        def val = m.getEntry(row,col)
        rval.setEntry(rind,cind,val)
      }
    }
    return new MatrixImpl(rval)
  }

}

// *EOF*
```

ClojureTool.groovy

```
// @file     ClojureTool.groovy
// @author   Mac Radigan

package org.radigan.system.interpreter

import org.radigan.system.tools.AbstractTool
import org.radigan.system.utilities.Debug
import clojure.lang.RT
import clojure.lang.Var
import clojure.lang.Compiler
import javax.script.ScriptEngine
import javax.script.ScriptEngineManager

public class ClojureTool extends AbstractTool {
  protected List<String> args = null

  public String getName() {
    return "clojure"
  }
```

```
  public String getDescription() {
    return "Runs the clojure interpreter."
  }

  public void initialize() {
    cli.usage = "${getName()} -f <filename> [-h]"
    cli.with {
        h longOpt: 'help',   'show usage information'
        g longOpt: 'debug',  'turn debugging on'
        f longOpt: 'file',   'file',  args:1, argName:'file', required:true
    }
  }

  @Override public int process(String[] args) {
    this.args = args.collect{it}[2..-1]
    opt = parse(args)
    if(!opt) return 1
    if(opt.help) { cli.usage(); return 1 }
    return run()
  }

  public int run() {
    def manager = new ScriptEngineManager()
    def engine = manager.getEngineByName("Clojure")
    def cljArgs = new String[this.args.size()]
    def index = 0
    args.each { cljArgs[index++] = it }
    engine.put("*command-line-args*", cljArgs)
    engine.put("args", cljArgs)
    engine.eval(new File(opt.f).text)
    return 0
  }

}

/* *EOF* */
```

ScriptTool.groovy

```
// @file     ScriptTool.groovy
// @author   Mac Radigan

package org.radigan.system.interpreter

import org.radigan.system.tools.AbstractTool
import org.apache.bsf.BSFManager
import org.apache.bsf.util.IOUtils
import org.radigan.system.utilities.Debug

public class ScriptTool extends AbstractTool {

  public String getName() {
    return "script"
  }

  public String getDescription() {
```

```
    return "Runs the script interpreter."
  }

  public void initialize() {
    cli.usage = "${getName()} -f <filename> [-h]"
    cli.with {
        h longOpt: 'help',   'show usage information'
        g longOpt: 'debug',  'turn debugging on'
        f longOpt: 'file',   'file',  args:1, argName:'file', required:true
    }
  }

  public int run() {
    BSFManager.registerScriptingEngine("groovy", "org.codehaus.groovy.bsf.GroovyEngine", ["groovy", "gy
    def manager = new BSFManager()
    def debug = Debug.getInstance()
    debug.setEngine(manager)
    def language = manager.getLangFromFilename(opt.f)
    def script = IOUtils.getStringFromReader(new FileReader(opt.f))
    manager.exec(language, opt.f, 0, 0, script)
    return 0
  }

}

/* *EOF* */
```

Configuration.groovy

```
// @file     Configuration.groovy
// @author   Mac Radigan
// @version  $Id: Configuration.groovy 79 2012-04-04 07:46:41Z mac.radigan $

package org.radigan.system.configuration

import org.radigan.system.utilities.ResourceManager
import org.apache.log4j.xml.DOMConfigurator
import org.apache.log4j.PropertyConfigurator
import groovy.util.ConfigSlurper
import java.net.InetAddress
//import javax.naming.ConfigurationException

public class Configuration {

  public ConfigObject config
  protected static String SYSTEM_NAME = "naomi"
  protected static Configuration ref = null
  protected ResourceManager resourceManager = new ResourceManager()
  protected String environment = null
  protected List libPath
  protected List configPath
  protected List resPath
  protected List binPath
  protected List homePath
  protected String root
```

```
private Configuration(File systemHome) {
  initialize(systemHome)
}

public static Configuration getInstance(File systemHome=null) {
   if(null==ref) { ref = new Configuration(systemHome) }
   return ref
}

public ResourceManager getResourceManager() {
   return resourceManager
}

public File getRootDirectory() {
  return new File(root)
}

private initialize(File systemHome=null) {
  //root = resourceManager.getRootDirectory()
  def skelPath = []
  def skelPathVar = System.getenv("PATH")
  if(skelPathVar) skelPath = skelPathVar.tokenize(':')
  homePath = ["./target", "./lib"]
  configPath = [".", "./config"]
  libPath = ["./lib"]
  binPath = ["./bin"] + skelPath
  resPath = ["res/system"]
  if(!systemHome) systemHome = new File(System.getenv("${SYSTEM_NAME.toUpperCase()}_HOME")) // adde
  root = "${systemHome}"
  if(!systemHome) {
    install()
    configureExecutables()
  }
  configureClasspath()
  configureSettings()
  configureLogging()
}

private install() {
  resPath.each { path ->
    resourceManager.extractResource("${path}", new File("${root}"))
  }
}

private File searchBin(File file) {
  def result = null
  binPath.each { path ->
    def testFile = new File("${root}/${path}/${file}")
    if(testFile.exists()) {
      result = testFile
      return
    }
  }
  binPath.each { path ->
```

```
    def testFile = new File("${path}/${file}")
    if(testFile.exists()) {
      result = testFile
      return
    }
  }
  return result
}

private File searchConfig(File file) {
  def result = null
  configPath.each { path ->
    def testFile = new File("${root}/${path}/${file}")
    if(testFile.exists()) {
      result = testFile
      return
    }
  }
  return result
}

private configureLogging() {
  //def configFile = searchConfig(new File("logging.xml"))
  def configFile = searchConfig(new File("logging.properties"))
  try {
    //if(configFile) DOMConfigurator.configure(configFile.toURL())
    if(configFile) PropertyConfigurator.configure(configFile.toURL())
  } catch(Exception e) {
    e.printStackTrace()
  }
}

public ConfigObject getConfiguration(String environment, File file) {
  def configFile = searchConfig(file)
  //if( ! configFile) throw new ConfigurationException("""Configuration file not found: "${file}"."""
  if( ! configFile) throw new Exception("""Configuration file not found: "${file}".""")
  def configSlurper = null
  if(environment) {
    configSlurper = new ConfigSlurper(environment)
  } else {
    configSlurper = new ConfigSlurper()
  }
  configSlurper.classLoader = resourceManager.getClassLoader()
  return configSlurper.parse(configFile.toURL())
}

public static ConfigObject getConfiguration() {
  return getInstance().config
}

private configureSettings() {
  environment = InetAddress.getLocalHost().getHostName()
  config = getConfiguration(environment, new File("${SYSTEM_NAME.toLowerCase()}.conf"))
}
```

```
  private configureClasspath() {
    libPath.each { path ->
      def dir = new File("${root}/${path}")
      if(dir.exists()) resourceManager.addClasspath(dir)
    }
  }

  private configureExecutables() {
    binPath.each { path ->
      new File("${root}/${path}").eachFileRecurse() { file ->
        if(file.canWrite()) file.setExecutable(true, false)
      }
    }
  }


}

// *EOF*
```

AbstractTool.groovy

```
// @file     AbstractTool.groovy
// @author   Mac Radigan

package org.radigan.system.tools

import groovy.util.CliBuilder
import groovy.util.OptionAccessor

public abstract class AbstractTool implements Tool {
  public CliBuilder cli = new CliBuilder()
  public OptionAccessor opt = null
  public AbstractTool() {
    initialize()
  }
  public OptionAccessor parse(String[] args) {
    opt = cli.parse(args as String[])
    return opt
  }
  public int process(String[] args) {
    opt = parse(args)
    if(!opt) return 1
    if(opt.help) { cli.usage(); return 1 }
    return run()
  }
  public abstract void initialize()
  public abstract String getName()
  public abstract String getDescription()
  public abstract int run()
}
// *EOF*
```

Command.groovy

```
// @file     Command.groovy
// @author   Mac Radigan
```

```groovy

package org.radigan.system.tools

import java.util.ServiceLoader
import org.radigan.system.tools.AbstractTool
import org.radigan.system.configuration.Configuration
import org.apache.log4j.Logger

public class Command {

  public static void main(String[] args) {
    def log = Logger.getLogger(Command.class)
    Configuration.getInstance()
    def providers = ServiceLoader.load(AbstractTool.class)
    if(args.size()) {
      try {
        providers.each() { tool ->
          if(tool.getName()==args[0]) {
            def newargs = new String[args.size()-1]
            for(int argIndex=0; argIndex<args.size()-1; argIndex++) {
              newargs[argIndex] = args[argIndex+1]
            }
            int returnCode = tool.process(newargs)
            System.exit(returnCode)
          }
        }
        usage(providers)
        println "no such command:  ${args[0]}"
        System.exit(1)
      } catch(e) {
        e.printStackTrace()
      }
    } else {
      usage(providers)
      System.exit(0)
    }
  }

  public static void usage(providers) {
    try {
      println "tools:"
      def index = 0
      providers.each() { tool ->
        println "    ${++index}) ${tool.getName()} - ${tool.getDescription()}"
      }
    } catch(e) {
      e.printStackTrace()
    }
  }
}
// *EOF*
```

Command2.groovy

```groovy
// @file     Command2.groovy
// @author   Mac Radigan
```

```
package org.radigan.system.tools

import org.apache.commons.discovery.tools.DiscoverSingleton
import org.apache.commons.discovery.resource.ClassLoaders
import org.apache.commons.discovery.resource.classes.DiscoverClasses
import org.apache.commons.discovery.tools.DiscoverClass
import org.apache.commons.discovery.ResourceClassIterator
import org.apache.commons.discovery.ResourceClass
import org.apache.commons.discovery.tools.SPInterface
import org.apache.commons.discovery.ResourceClass
//import static org.apache.commons.discovery.tools.SPInterface.newSPInterface
//import static org.apache.commons.discovery.tools.Service.providers
import org.radigan.system.test.TestTool
import org.radigan.system.utilities.ResourceManager
import org.radigan.system.tools.AbstractTool
import org.radigan.system.tools.Tool
import org.radigan.system.configuration.Configuration
import org.apache.log4j.Logger

public class Command2 {

  public static void main(String[] args) {
    def log = Logger.getLogger(Command.class)
    def configuration = Configuration.getInstance()
    def resourceManager = configuration.getResourceManager()
    def loaders = ClassLoaders.getAppLoaders(Tool.class, getClass(), false)
    def providerClasses = []
    def buffer = resourceManager.getStream("/META-INF/services/org.radigan.system.tools.AbstractTool")
    buffer.eachLine { line -> if( ! line.startsWith("#")) providerClasses << line }
    def discoverClasses = new DiscoverClasses<Tool>(loaders)
    //def classList = []
    def providers = []
    providerClasses.each { className ->
      def classIterator = discoverClasses.findResourceClasses(className)
      while(classIterator.hasNext()) {
        def resource = classIterator.nextResourceClass()
        def clazz = resource.loadClass()
        providers << clazz.newInstance()
      }
    }
    if(args.size()) {
      try {
        providers.each() { tool ->
          if(tool.getName()==args[0]) {
            def newargs = new String[args.size()-1]
            for(int argIndex=0; argIndex<args.size()-1; argIndex++) {
              newargs[argIndex] = args[argIndex+1]
            }
            int returnCode = tool.process(newargs)
            System.exit(returnCode)
          }
        }
        usage(providers)
```

```
        println "no such command:  ${args[0]}"
        System.exit(1)
      } catch(e) {
        e.printStackTrace()
      }
    } else {
      usage(providers)
      System.exit(0)
    }
  }

  public static void usage(providers) {
    try {
      println "tools:"
      def index = 0
      providers.each() { tool ->
        println "    ${++index}) ${tool.getName()} - ${tool.getDescription()}"
      }
    } catch(e) {
      e.printStackTrace()
    }
  }

}
// *EOF*
```

Encoder.groovy

```
// @file     Encoder.groovy
// @author   Mac Radigan

package org.radigan.system.utilities

public class Encoder {

  private String encoding = null

  public Encoder(String encoding) {
    this.encoding = encoding
  }

  public String encode(String message) {
    switch(encoding) {
      case "rot13":
        return Rot13.encode(message)
      case "md5":
        return Md5.encode(message)
    }
  }

  public String decode(String message) {
    switch(encoding) {
      case "rot13":
        return Rot13.decode(message)
    }
  }
```

```
}

// *EOF*
```

Entropy.groovy

```
// @file     Entropy.groovy
// @author   Mac Radigan

package org.radigan.system.utilities

import java.util.LinkedHashMap
import java.util.List

public class Entropy {
  protected Map histogram = [:]
  protected double nats
  protected double NATS_TO_BITS = 1/Math.log(2.0D)

  public Entropy() {
  }

  public double getNats() {
    return nats
  }

  public double getBits() {
    return nats * NATS_TO_BITS
  }

  private void update() {
    nats = 0
    def sum = histogram.values().sum()
    histogram.each { symbol, cnt ->
      def probability = cnt / sum
      nats -= probability * Math.log(probability as double)
    }
  }

  public void addAll(List data) {
    data.each { symbol ->
      if( ! histogram[symbol]) histogram[symbol]=1
      histogram[symbol]++
    }
    update()
  }

}

// *EOF*
```

FileCollection.groovy

```
// @file     FileCollection.groovy
// @author   Mac Radigan

package org.radigan.system.utilities
```

```
import org.apache.log4j.Logger
import java.util.HashMap
import org.radigan.system.utilities.Md5
import java.rmi.server.UID
import javax.swing.text.DateFormatter
import java.util.Date
import java.text.SimpleDateFormat
import java.util.TimeZone
//import groovy.util.AntBuilder
import org.apache.commons.io.FileUtils

public class FileCollection extends HashMap<File,String> {
  protected static log = Logger.getLogger(FileCollection.class.getName())
  protected String collectionId = null
  protected boolean deleteInput = false
  protected String uniqueId = null
  protected File directory = null
  //protected AntBuilder ant = new AntBuilder()

  public FileCollection(File directory) {
    this.uniqueId = new UID().toString().replaceAll("-","").replaceAll(":","")
    this.collectionId = createCollectionId(uniqueId)
    this.directory = new File("${directory}/${collectionId}")
    initialize()
  }

  public String setPrimaryId(File file, boolean deleteInput) {
    setDeleteInput(deleteInput)
    this.uniqueId = Md5.encode(file.text)
    this.collectionId = createCollectionId(uniqueId)
    this.directory = new File("${this.directory}/${collectionId}")
    def newFile = add(file)
    return newFile
  }

  private initialize() {
    if( ! directory.exists()) {
      log.info """        creating directory ${directory}"""
      directory.mkdirs()
    }
  }

  protected String getCollectionId() {
    return collectionId
  }

  protected String getUniqueId() {
    return uniqueId
  }

  protected File getDirectory() {
    return directory
  }
```

```
protected String createCollectionId(String uniqueId) {
  def dateFormatter = new SimpleDateFormat("yyyy-MM-dd_kk-mm-ss")
  dateFormatter.setTimeZone(TimeZone.getTimeZone("GMT"))
  def date = dateFormatter.format(new Date())
  return "${date}_${uniqueId}"
}

public File updateFile(String filename) {
  def file = new File("${directory}/${filename}")
  if(containsKey(filename)) {
    remove(filename)
    put(Md5.encode(file.text), file)
  } else {
    put(filename, file)
  }
  return file
}

public void setDeleteInput(boolean deleteInput) {
  this.deleteInput = deleteInput
}

public void leftShift(File file) {
  add(file)
}

public void cleanup() {
  def sb = new StringBuffer()
  def endl = java.lang.System.getProperty('line.separator')
  sb << """        deleting file collection""" << endl
  sb << toString()
  log.debug sb.toString()
  //ant.delete(dir:"${directory}")
  FileUtils.deleteDirectory(directory)
}

public File getFileByExtension(String extensionList) {
  def file = null
  this.each { key, value ->
    extensionList.tokenize(",").each { extension ->
      def ext = value.name.substring(value.name.indexOf(".")+1)
      if(extension.equalsIgnoreCase(ext)) file = value
    }
  }
  return file
}

public String add(File file) {
  def newFile = new File("${directory}/${file.getName()}")
  log.info """        copying file ${file} to ${newFile}"""
  //ant.copy(file:"${file}", toFile:"${newFile}")
  FileUtils.copyFile(file, newFile)
  if(deleteInput) {
```

```
      log.info """        deleting file ${file}"""
      file.delete()
    }
    def md5 = Md5.encode(newFile.text)
    this.put(md5, newFile)
    return md5
  }

  @Override public String toString() {
    def sb = new StringBuffer()
    def endl = java.lang.System.getProperty('line.separator')
    sb << "collection<" << collectionId << ">" << endl
    this.each { key, value ->
      sb << "  [" << key.padLeft(32) << "] " << value << endl
    }
    return sb.toString()
  }

}

// *EOF*
```

Md5.groovy

```
// @file     Md5.groovy
// @author   Mac Radigan

package org.radigan.system.utilities

import java.security.MessageDigest

public class Md5 {

  public static String encode(String message) {
    def digest = MessageDigest.getInstance("MD5")
    digest.update(message.bytes)
    def big = new BigInteger(1,digest.digest())
    return big.toString(16).padLeft(32,"0")
  }

}

// *EOF*
```

Record.groovy

```
// @file     Record.groovy
// @author   Mac Radigan

package org.radigan.system.utilities

import java.util.LinkedHashMap

public class Record extends LinkedHashMap {

  public Record() {
  }
```

```
}

// *EOF*
```

Recordset.groovy

```
// @file     Recordset.groovy
// @author   Mac Radigan

package org.radigan.system.utilities

import java.util.ArrayList
import java.util.ServiceLoader
import org.apache.log4j.Logger

public class Recordset extends ArrayList<Record> {
  private static log = Logger.getLogger(Recordset.class)

  public Recordset() {
  }

  public Recordset filter(closure) {
    def resultRecordset = new Recordset()
    this.each { record ->
      if(closure.call(record)) {
      } else {
        resultRecordset << record
      }
    }
    return resultRecordset
  }

  public List<String> getValues() {
    def list = []
    this.each { record ->
      record.each { key, value ->
        list << value
      }
    }
    return list
  }

  public Recordset groupBy(String expression, int nameIndex, int idIndex) {
    def groups = [:]
    def misc = [:]
    def resultRecordset = new Recordset()
    this.each { record ->
      record.each { key, value ->
        def matcher = (key=~expression)
        def resultRecord = null
        if(matcher.matches()) {
          if( ! groups[matcher[0][idIndex]]) groups[matcher[0][idIndex]] = new Record()
          resultRecord = groups[matcher[0][idIndex]]
          resultRecord[matcher[0][nameIndex]] = value
        }
      }
```

```
    record.each { key, value ->
      def matcher = (key=~expression)
      def resultRecord = null
      if(!matcher.matches()) {
        groups.each { gkey, gvalue ->
          groups[gkey][key] = value
        }
      }
    }
    groups.each { gkey, gvalue ->
      resultRecordset << gvalue
    }
    misc.each { mkey, mvalue ->
      resultRecordset << mvalue
    }
  }
  return resultRecordset
}

public String toShortString() {
  def sb = new StringBuffer()
  def endl = java.lang.System.getProperty('line.separator')
  def border = 1
  def indent = 2
  def headers = [:]
  def max = 0
  each { record ->
    record.each { key, value ->
      headers[key] = Math.max(headers[key]?headers[key]:key.size(),"${value}".size())
      max = Math.max(headers[key],max)
    }
  }
  def index = 0
  each { record ->
    sb << "Record " << ++index << ">" << endl
    record.each { key, value ->
      sb << " ".multiply(indent) << key.padLeft(max+indent) << ": " << value << endl
    }
    sb << endl
  }
  return sb.toString()
}

@Override public String toString() {
  def sb = new StringBuffer()
  def endl = java.lang.System.getProperty('line.separator')
  def border = 1
  def headers = [:]
  each { record ->
    record.each { key, value ->
      headers[key] = Math.max(headers[key]?headers[key]:key.size(),"${value}".size())
    }
  }
  if(headers.size()) {
```

```
      headers.each { key, length -> sb << key.padRight(length+border) }; sb << endl
      headers.each { key, length -> sb << "=".multiply(length)+" ".multiply(border) }; sb << endl
    }
    each { record ->
      record.each { key, value ->
        if(!headers[key]) {
          sb << " ".padLeft(headers[key]) << " ".multiply(border)
        } else {
          sb << "${value}".padLeft(headers[key]) << " ".multiply(border)
        }
      }
      sb << endl
    }
    return sb.toString()
  }

}

// *EOF*
```

ResourceManager.groovy

```
// @file:    ResourceManager.groovy
// @author:  Mac Radigan

package org.radigan.system.utilities

import java.io.BufferedOutputStream
import java.io.File
import java.io.FileInputStream
import java.io.IOException
import java.io.InputStream
import java.util.jar.JarEntry
import java.util.jar.JarInputStream
import java.util.jar.JarOutputStream
import groovy.lang.GroovyClassLoader
import java.lang.ClassLoader
//import java.net.URLClassLoader

public class ResourceManager {

  protected static final int BUFFER_SIZE = 2156
  //protected GroovyClassLoader classloader = null
  protected URLClassLoader classloader = null
  protected static String SYSTEM_NAME = "naomi"

  public ResourceManager() {
    classloader = new GroovyClassLoader(getClass().getClassLoader())
    //classloader = new URLClassLoader([] as URL[], getClass().getClassLoader())
  }

  public ClassLoader getClassLoader() {
    return classloader
  }

  public void addClasspath(File path) {
```

```
  //def classLoader = new GroovyClassLoader()
  path.eachFileRecurse() { file ->
    if(file.isFile() && file.getAbsolutePath().endsWith(".jar")) {
      println "loading ${file}"
      this.class.getClassLoader().addURL(file.toURI().toURL())
      //classloader.addURL(file.toURI().toURL())
    }
  }
}

public File getJarFile() {
  //def protectionDomain = ResourceManager.class.getProtectionDomain()
  def protectionDomain = getClass().getProtectionDomain()
  def location = protectionDomain.getCodeSource().getLocation()
  return new File(location.toURI())
}

public File getRootDirectory() {
  def systemHome = System.getenv("${SYSTEM_NAME.toUpperCase()}_HOME") // added for OSGi support
  println "systemHome: ${systemHome}"
  if(systemHome) {
    def file = new File(systemHome)
    if(!file.exists()) throw new Exception("""Environment variable ${SYSTEM_NAME.toUpperCase()}_HOM
    return file
  }
  def protectionDomain = getClass().getProtectionDomain()
  def location = protectionDomain.getCodeSource().getLocation()
  return new File(location.toURI()).getParentFile().getParentFile()
}

protected String reducePath(String resourcePath, JarEntry jarEntry) {
  def resource = jarEntry.getName()
  return new File(resource.substring(resourcePath.length(), resource.length()))
}

public InputStream getStream(String resourcePath) {
  //return ResourceManager.class.getResourceAsStream(resourcePath)
  return getClass().getResourceAsStream(resourcePath)
}

public String getText(String resourcePath) {
  //def inputStream = ResourceManager.class.getResourceAsStream(resourcePath)
  def inputStream = getClass().getResourceAsStream(resourcePath)
  if(null==inputStream) throw new Exception("Resource not found ${resourcePath}")
  def bufferedReader = new BufferedReader(new InputStreamReader(inputStream))
  def lineSep = System.getProperty("line.separator")
  def sb = new StringBuffer()
  def line = null
  while((line=bufferedReader.readLine())!=null) {
    sb.append(line)
    sb.append(lineSep)
  }
  return sb.toString()
}
```

```
  public void extractResource(String resourcePath, File destinationDirectory) {
    def destination = null
    def jarInputStream = new JarInputStream(new FileInputStream(getJarFile()))
    def jarEntry = null
    while((jarEntry=jarInputStream.getNextJarEntry())!=null) {
      if(jarEntry.getName().startsWith(resourcePath)) {
        def reduced = reducePath(resourcePath, jarEntry)
        if(null!=reduced) {
          if(jarEntry.isDirectory()) {
            def dir = new File(destinationDirectory, reduced)
            if(!dir.exists()) {
              println "extracting ${dir.getAbsolutePath()}"
              dir.mkdir()
              if(jarEntry.getTime()!=-1) {
                dir.setLastModified(jarEntry.getTime())
              }
            }
          } else { // entry is a file
            def byteCount = 0
            def data = new byte[BUFFER_SIZE]
            def file = new File(destinationDirectory, reduced)
            if(!file.exists()) {
              destination = new BufferedOutputStream(new FileOutputStream(file), BUFFER_SIZE)
              println "extracting ${file.getAbsolutePath()}"
              while((byteCount=jarInputStream.read(data, 0, BUFFER_SIZE))!=-1) {
                destination.write(data, 0, byteCount)
              }
              destination.flush()
              destination.close()
              if(jarEntry.getTime()!=-1) {
                file.setLastModified(jarEntry.getTime())
              }
            }
          }
        }
      }
    }
    jarInputStream.close()
  }

}

// *EOF*
```

Rot13.groovy

```
// @file     Rot13.groovy
// @author   Mac Radigan

package org.radigan.system.utilities

public class Rot13 {

  public static String encode(String message) {
    return rot13(message)
```

```
  }

  public static String decode(String message) {
    return rot13(message)
  }

  public static String rot13(String message) {
    def sb = new StringBuffer()
    for (int index = 0; index < message.length(); index++) {
      char c = message.charAt(index)
      if       (c >= 'a' && c <= 'm') c += 13;
      else if  (c >= 'n' && c <= 'z') c -= 13;
      else if  (c >= 'A' && c <= 'M') c += 13;
      else if  (c >= 'A' && c <= 'Z') c -= 13;
      sb.append(c)
    }
    return sb.toString()
  }

}

// *EOF*
```

Shell.groovy

```
// @file     Shell.groovy
// @author   Mac Radigan

package org.radigan.system.utilities

import org.apache.log4j.Logger;
import java.util.concurrent.ExecutionException
import org.radigan.system.configuration.Configuration
import org.apache.commons.io.FileUtils
import org.apache.commons.io.IOUtils

public class Shell {
  protected static log = Logger.getLogger(Shell.class.getName())
  private long timeout = 0
  private long defaultTimeout = 4*60*1000

  public Shell() {
  }

  public Shell(Logger log) {
    this.log = log
  }

  public setLogger(Logger log) {
    this.log = log
  }

  public int execute(String command, long timeout=0) {
    try {
      def exitValue = 0
      def initialSize = 4096
```

```
      def outStream = new ByteArrayOutputStream(initialSize)
      def errStream = new ByteArrayOutputStream(initialSize)
      log.debug "[executing] ${command}"
      def env = null
      def environment = []
      def pathVar = null
      pathVar = System.getenv("DYLD_LIBRARY_PATH")  // Mac OS X
      if(pathVar) environment += pathVar.tokenize()
      pathVar = System.getenv("LD_LIBRARY_PATH")    // Linux
      if(pathVar) environment += pathVar.tokenize()
      def proc = command.execute(environment as String[], new File('.'))
      proc.consumeProcessOutput(outStream, errStream)
      if(timeout) {
        proc.waitFor()
      } else {
        proc.waitForOrKill(defaultTimeout)
      }
      exitValue = proc.exitValue()
      log.debug "[stdout] ${outStream.toString()}"
      log.debug "[stderr] ${errStream.toString()}"
      if(exitValue) throw new ExecutionException("Command failed with exit code ${proc.exitValue()}.\n$
      return exitValue
    } catch(e) {
      log.debug """Error executing command ${command}.""", e
      throw new Exception("""Error running "${command}".""", e)
    }
  }

}

// *EOF*
```

XmlUtilities.groovy

```
// @file     XmlUtilities.groovy
// @author   Mac Radigan

package org.radigan.system.utilities

// DOM
import javax.xml.parsers.DocumentBuilder
import javax.xml.parsers.DocumentBuilderFactory
import org.w3c.dom.Document
import org.w3c.dom.Element
import javax.xml.transform.dom.DOMSource
import javax.xml.transform.dom.DOMResult
import javax.xml.transform.stream.StreamSource
import javax.xml.transform.stream.StreamResult
// XPATH
import org.apache.xpath.domapi.XPathEvaluatorImpl
import org.w3c.dom.xpath.XPathEvaluator
import org.w3c.dom.xpath.XPathNSResolver
import org.w3c.dom.xpath.XPathResult
//import org.xml.sax.SAXException
// XSL
import javax.xml.transform.TransformerFactory
```

```
import net.sf.saxon.TransformerFactoryImpl
import javax.xml.transform.Templates
// list of XSL transformers:
// GOOD // import net.sf.saxon.TransformerFactoryImpl
// OK   // import org.apache.xalan.processor.TransformerFactoryImpl
// BAD  // import com.sun.org.apache.xalan.internal.processor.TransformerFactoryImpl
// BAD  // import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl
//import org.ccil.cowan.tagsoup.Parser

public class XmlUtilities {

  public static newDocument(xmldata) {
    DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance()
    docBuilderFactory.setNamespaceAware(true)
    DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder()
    ByteArrayInputStream bais = new ByteArrayInputStream(xmldata.getBytes())
    Document xml = docBuilder.parse(bais)
    return xml
  }

  public static docToStreamSource(doc) {
    def factory = net.sf.saxon.TransformerFactoryImpl.newInstance()
    def streamResult = new StreamResult( new java.io.ByteArrayOutputStream() )
    def transformer = factory.newTransformer()
    transformer.transform( new DOMSource(doc), streamResult )
    def os = streamResult.getOutputStream()
    def baos = (ByteArrayOutputStream)os
    def is = new ByteArrayInputStream( baos.toByteArray() )
    def streamSource = new StreamSource( is )
    return streamSource
  }

  public static xslt(xml,xsl) {
    def factory = net.sf.saxon.TransformerFactoryImpl.newInstance()
    def domResult = new DOMResult()
    def domSource = new DOMSource(xml);
    def xslStreamSource = docToStreamSource(xsl)
    def templates = factory.newTemplates( xslStreamSource )
    def transformer = templates.newTransformer()
    transformer.transform(domSource, domResult)
    return domResult.getNode()
  }

  public static parseToString(node) {
    def factory = net.sf.saxon.TransformerFactoryImpl.newInstance()
    def transformer = factory.newTransformer()
    def stringWriter = new StringWriter(128);
    transformer.transform(new DOMSource(node), new StreamResult(stringWriter))
    StringBuffer buffer =  stringWriter.getBuffer()
    return buffer.toString()
  }

  public static findNodes(node, xpath) {
    def xpathEvaluator = new XPathEvaluatorImpl( node )
```

```
    def xpathNSResolver = xpathEvaluator.createNSResolver( node )
    def xpathResult = (XPathResult)xpathEvaluator.evaluate( xpath, node, xpathNSResolver, XPathResult.
    return xpathResult
  }

  public static findSingleNode(node, xpath) {
    return findNodes(node, xpath).snapshotItem(0)
  }

/*
  public static download(url) {
    def location = url.toURL()
    def sb = new StringBuffer()
    def input = new BufferedReader(new InputStreamReader(location.openStream()))
    def inputLine
    while ((inputLine = input.readLine()) != null) sb.append(inputLine)
    input.close()
    def tagSoupParser = new org.ccil.cowan.tagsoup.Parser()
    def htmlParser = new XmlSlurper(tagSoupParser)
    def xhtml = htmlParser.parseText(sb.toString())
    def cleanHtmlWriter = new StringWriter()
    cleanHtmlWriter << new groovy.xml.StreamingMarkupBuilder().bind( {
      mkp.declareNamespace( 'html': 'http://www.w3.org/1999/xhtml' )
      mkp.yield( xhtml ) }
    )
    return newDocument(cleanHtmlWriter.toString().trim())
  }
*/

}

// *EOF*
```

Xsp.groovy

```
// @file     Xsp.groovy
// @author   Mac Radigan

package org.radigan.system.utilities

import groovy.text.GStringTemplateEngine
import groovy.io.PlatformLineWriter

public class Xsp {

  protected GStringTemplateEngine engine = new GStringTemplateEngine()

  public Xsp() {
  }

  public String process(String xsp, Map binding) {
    def template = engine.createTemplate(xsp).make(binding)
    def stringWriter = new StringWriter()
    def platformLineWriter = new PlatformLineWriter(stringWriter)
    template.writeTo(platformLineWriter)
    platformLineWriter.flush()
```

```
    return stringWriter.toString()
  }

}

// *EOF*
```

1.20 Appendix B.7: NAOMI Configuration Sources

datatypes.xml

```
<?xml version="1.0"?>
<t:datatype xmlns:t="http://org.radigan.naomi/datatypes" name="Object" type="Root">
  <t:datatype name="Foo" type="Object">
    <t:datatype name="Bar" type="Foo">
    </t:datatype>
  </t:datatype>
</t:datatype>
```

logging.properties

```
## logging.properties
## Mac Radigan

#log4j.rootLogger=DEBUG, file, stdout
log4j.rootLogger=NONE

#log4j.appender.stdout= org.apache.log4j.ConsoleAppender
#log4j.appender.stdout.Threshold=DEBUG
#log4j.appender.stdout.target=System.out
#log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
#log4j.appender.stdout.layout.ConversionPattern=%d{dd.MM.yyyy HH:mm:ss.SSS} %5p [%t] %c\n%m%n

log4j.appender.syslog=org.apache.log4j.net.SyslogAppender
log4j.appender.syslog.threshold=DEBUG
log4j.appender.syslog.syslogHost=localhost
log4j.appender.syslog.facility=LOCAL1
log4j.appender.syslog.facilityPrinting=false
log4j.appender.syslog.layout=org.apache.log4j.PatternLayout
log4j.appender.syslog.layout.conversionPattern=%d{dd.MM.yyyy HH:mm:ss.SSS} %5p [%t] %c\n%m%n

log4j.appender.stderr= org.apache.log4j.ConsoleAppender
log4j.appender.stderr.Threshold=DEBUG
log4j.appender.stderr.target=System.err
log4j.appender.stderr.layout=org.apache.log4j.PatternLayout
log4j.appender.stderr.layout.ConversionPattern=%d{dd.MM.yyyy HH:mm:ss.SSS} %5p [%t] %c\n%m%n

log4j.appender.stdout= org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Threshold=DEBUG
log4j.appender.stdout.target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
#log4j.appender.stdout.layout.ConversionPattern=%d{dd.MM.yyyy HH:mm:ss.SSS} %5p [%t] %c\n%m%n
log4j.appender.stdout.layout.ConversionPattern=%m%n

#log4j.appender.sock=org.apache.log4j.net.SocketAppender
#log4j.appender.sock.Threshold=DEBUG
#log4j.appender.sock.layout=org.apache.log4j.PatternLayout
```

```
##log4j.appender.sock.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
#log4j.appender.sock.layout.ConversionPattern=%d{dd.MM.yyyy HH:mm:ss.SSS} %-5p %c - %m%n
#log4j.appender.sock.Port=4445
#log4j.appender.sock.RemoteHost=127.0.0.1
#log4j.appender.sock.ReconnectionDelay=60000

log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.Threshold=DEBUG
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
log4j.appender.file.File=naomi.log
log4j.appender.file.MaxFileSize=10MB
log4j.appender.file.MaxBackupIndex=3

log4j.logger.org.radigan=DEBUG, file, syslog, stdout
log4j.logger.org.apache.bsf=ERROR, stdout
log4j.logger.org.mortbay=NONE
log4j.logger.org.apache.commons.httpclient=NONE

## *EOF*
```

naomi.conf

```
// @file:     naomi.config
// @author:   Mac Radigan

environments {

  omni {
    roar {
      port = 8080
      cache = '/Users/mac/temp/cache'
    }
  }

}

// *EOF*
```

1.21 Appendix B.8: Build Sources and Packaging

pom.xml (NAOMI Project)

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  @file     pom.xml
  @author   Mac Radigan
-->
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance

  <modelVersion>4.0.0</modelVersion>

  <groupId>org.radigan</groupId>
  <artifactId>naomi</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>NAOMI</name>
```

```
<dependencies>
<dependency>
    <groupId>org.osgi</groupId>
    <artifactId>org.osgi.compendium</artifactId>
    <version>4.2.0</version>
  </dependency>
  <dependency>
    <groupId>org.osgi</groupId>
    <artifactId>org.osgi.core</artifactId>
    <version>4.2.0</version>
  </dependency>
  <dependency>
    <groupId>org.hdfgroup</groupId>
    <artifactId>hdf-java</artifactId>
    <version>2.6.1</version>
  </dependency>
  <dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-math3</artifactId>
    <version>3.0</version>
  </dependency>
  <dependency>
    <groupId>org.clojure</groupId>
    <artifactId>clojure</artifactId>
    <version>1.5.0-alpha2</version>
  </dependency>
  <dependency>
    <groupId>clojure-jsr223</groupId>
    <artifactId>clojure-jsr223</artifactId>
    <version>1.0</version>
  </dependency>
  <dependency>
    <groupId>org.clojure</groupId>
    <artifactId>tools.cli</artifactId>
    <version>0.2.1</version>
  </dependency>
  <dependency>
    <groupId>commons-discovery</groupId>
    <artifactId>commons-discovery</artifactId>
    <version>20040218.194635</version>
  </dependency>
  <!--dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>com.springsource.org.apache.commons.discovery</artifactId>
    <version>0.4.0</version>
  </dependency-->
  <dependency>
    <groupId>commons-cli</groupId>
    <artifactId>commons-cli</artifactId>
    <version>1.0</version>
  </dependency>
  <dependency>
    <groupId>commons-io</groupId>
```

```
    <artifactId>commons-io</artifactId>
    <version>2.2</version>
  </dependency>
  <dependency>
    <groupId>jline</groupId>
    <artifactId>jline</artifactId>
    <version>1.0</version>
  </dependency>
  <dependency>
    <groupId>org.codehaus.groovy</groupId>
    <artifactId>groovy</artifactId>
    <!--version>1.7.0</version-->
    <version>1.8.0</version>
  </dependency>
  <dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.16</version>
  </dependency>
  <dependency>
    <groupId>xstream</groupId>
    <artifactId>xstream</artifactId>
    <version>1.2.2</version>
  </dependency>
  <dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-email</artifactId>
    <version>1.2</version>
  </dependency>
  <!--dependency>
    <groupId>javax.jmdns</groupId>
    <artifactId>jmdns</artifactId>
    <version>3.2.2</version>
    <version>3.4.1</version>
  </dependency-->
  <!--dependency>
    <groupId>org.objectweb.bonita</groupId>
    <artifactId>chainsaw</artifactId>
    <version>2.1</version>
  </dependency-->
  <dependency>
    <groupId>xerces</groupId>
    <artifactId>xercesImpl</artifactId>
    <version>2.10.0</version>
  </dependency>
  <dependency>
    <groupId>xalan</groupId>
    <artifactId>xalan</artifactId>
    <version>2.7.1</version>
  </dependency>
  <dependency>
    <groupId>bsf</groupId>
    <artifactId>bsf</artifactId>
    <version>2.4.0</version>
```

```
</dependency>
<dependency>
  <groupId>org.apache.ant</groupId>
  <artifactId>ant</artifactId>
  <version>1.8.2</version>
</dependency>
<dependency>
  <groupId>org.apache.ftpserver</groupId>
  <artifactId>ftpserver-core</artifactId>
  <version>1.0.6</version>
</dependency>
<dependency>
  <groupId>org.apache.mina</groupId>
  <artifactId>mina-core</artifactId>
  <version>2.0.4</version>
  <!--packaging>bundle</packaging-->
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.6.4</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.6.4</version>
</dependency>
<dependency>
  <groupId>org.apache.ftpserver</groupId>
  <artifactId>ftplet-api</artifactId>
  <version>1.0.6</version>
  <!--packaging>bundle</packaging-->
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>1.6.4</version>
</dependency>
<dependency>
  <groupId>org.mortbay.jetty</groupId>
  <artifactId>jetty</artifactId>
  <version>7.0.0.pre5</version>
</dependency>
<dependency>
  <groupId>org.mortbay.jetty</groupId>
  <artifactId>jetty-deploy</artifactId>
  <version>7.0.0.pre5</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.5</version>
</dependency>
<dependency>
```

```
    <groupId>commons-httpclient</groupId>
    <artifactId>commons-httpclient</artifactId>
    <version>3.1</version>
  </dependency>
  <dependency>
    <groupId>tagsoup</groupId>
    <artifactId>tagsoup</artifactId>
    <version>0.9.7</version>
  </dependency>
  <dependency>
    <groupId>net.sf.saxon</groupId>
    <artifactId>saxon</artifactId>
    <version>8.7</version>
  </dependency>
  <!--dependency>
    <groupId>xch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>0.9.29</version>
  </dependency-->
  <dependency>
    <groupId>org.apache.karaf.shell</groupId>
    <artifactId>org.apache.karaf.shell.console</artifactId>
    <version>2.2.3</version>
  </dependency>
  <dependency>
    <groupId>org.apache.felix</groupId>
    <artifactId>org.apache.felix.main</artifactId>
    <version>3.2.2</version>
  </dependency>
  <dependency>
    <groupId>org.apache.felix</groupId>
    <artifactId>org.apache.felix.configadmin</artifactId>
    <version>1.2.4</version>
  </dependency>
</dependencies>

<build>
  <plugins>

    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>2.3.2</version>
      <configuration>
        <compilerId>groovy-eclipse-compiler</compilerId>
        <verbose>true</verbose>
        <debug>true</debug>
      </configuration>
      <executions>
        <execution>
          <goals>
            <goal>compile</goal>
          </goals>
        </execution>
      </executions>
```

```
  <dependencies>
    <dependency>
      <groupId>org.codehaus.groovy</groupId>
      <artifactId>groovy-eclipse-compiler</artifactId>
      <version>2.6.0-01</version>
    </dependency>
  </dependencies>
</plugin>
<plugin>
  <groupId>org.codehaus.groovy</groupId>
  <artifactId>groovy-eclipse-compiler</artifactId>
  <version>2.6.0-01</version>
  <extensions>true</extensions>
</plugin>

<!--plugin>
  <groupId>org.codehaus.gmaven</groupId>
  <artifactId>gmaven-plugin</artifactId>
  <version>1.4</version>
  <configuration>
    <debug>true</debug>
    <verbose>true</verbose>
    <stacktrace>true</stacktrace>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>generateStubs</goal>
        <goal>compile</goal>
        <goal>generateTestStubs</goal>
        <goal>testCompile</goal>
      </goals>
    </execution>
  </executions>
</plugin-->

<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-shade-plugin</artifactId>
  <version>1.5</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>shade</goal>
      </goals>
      <configuration>
        <filters>
          <filter>
            <artifact>*:*</artifact>
            <excludes>
              <exclude>META-INF/*.SF</exclude>
              <exclude>META-INF/*.DSA</exclude>
              <exclude>META-INF/*.RSA</exclude>
```

```
            </excludes>
          </filter>
        </filters>
        <artifactSet>
          <excludes>
            <exclude>junit:junit</exclude>
            <exclude>jmock:jmock</exclude>
          </excludes>
        </artifactSet>
        <transformers>
          <!--mainClass>org.radigan.system.tools.Command</mainClass-->
          <transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTr
            <manifestEntries>
              <Main-Class>org.radigan.system.tools.Command</Main-Class>
              <Manifest-Version>1.0</Manifest-Version>
              <Bundle-Activator>org.radigan.naomi.service.NaomiActivator</Bundle-Activator>
              <Bundle-ManifestVersion>2</Bundle-ManifestVersion>
              <Bundle-Name>NAOMI</Bundle-Name>
              <Bundle-SymbolicName>naomi</Bundle-SymbolicName>
              <Bundle-Version>1.0.0.SNAPSHOT</Bundle-Version>
              <Export-Package>org.radigan.naomi.service;version="1.0.0.SNAPSHOT",
                              org.apache.karaf.branding;version="1.0.0.SNAPSHOT"</Export-Package>
              <Import-Package>org.osgi.service.cm;version="1.2",
                              org.osgi.framework,sun.misc;resolution:=optional</Import-Package>
            </manifestEntries>
          </transformer>
          <transformer implementation="org.apache.maven.plugins.shade.resource.AppendingTransform
            <resource>META-INF/services/org.radigan.system.tools.ITool</resource>
          </transformer>
          <transformer implementation="org.apache.maven.plugins.shade.resource.AppendingTransform
            <resource>META-INF/services/org.radigan.system.tools.Tool</resource>
          </transformer>
        </transformers>
      </configuration>
    </execution>
  </executions>
</plugin>

<!--plugin>
  <groupId>org.apache.felix</groupId>
  <artifactId>maven-bundle-plugin</artifactId>
  <extensions>true</extensions>
  <configuration>
    <instructions>
      <Export-Package>org.radigan,org.apache.karaf.branding</Export-Package>
      <Bundle-SymbolicName>${project.artifactId}</Bundle-SymbolicName>
      <Bundle-Activator>org.radigan.naomi.osgi.activator.DaemonActivator</Bundle-Activator>
      <Embed-Dependency>*;scope=runtime|compile;optional=false</Embed-Dependency>
    </instructions>
  </configuration>
</plugin-->

</plugins>
```

```
  </build>

  <distributionManagement>
    <repository>
      <name>NAOMI</name>
      <id>releases</id>
      <url>http://naomi-2:8081/nexus/content/repositories/releases/</url>
    </repository>
    <snapshotRepository>
       <name>NAOMI</name>
       <uniqueVersion>false</uniqueVersion>
      <id>snapshots</id>
      <url>http://naomi-2:8081/nexus/content/repositories/snapshots</url>
    </snapshotRepository>
  </distributionManagement>
</project>
<!-- *EOF* -->
```

pom.xml (NAOMI Bundle)

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  @file     pom.xml
  @author   Mac Radigan
-->
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance

  <modelVersion>4.0.0</modelVersion>

  <groupId>org.radigan</groupId>
  <artifactId>naomi</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>NAOMI</name>

  <dependencies>
  <dependency>
      <groupId>org.osgi</groupId>
      <artifactId>org.osgi.compendium</artifactId>
      <version>4.2.0</version>
    </dependency>
    <dependency>
      <groupId>org.osgi</groupId>
      <artifactId>org.osgi.core</artifactId>
      <version>4.2.0</version>
    </dependency>
    <dependency>
      <groupId>org.hdfgroup</groupId>
      <artifactId>hdf-java</artifactId>
      <version>2.6.1</version>
    </dependency>
    <dependency>
      <groupId>org.apache.commons</groupId>
      <artifactId>commons-math3</artifactId>
      <version>3.0</version>
    </dependency>
```

```
<dependency>
  <groupId>org.clojure</groupId>
  <artifactId>clojure</artifactId>
  <version>1.5.0-alpha2</version>
</dependency>
<dependency>
  <groupId>clojure-jsr223</groupId>
  <artifactId>clojure-jsr223</artifactId>
  <version>1.0</version>
</dependency>
<dependency>
  <groupId>org.clojure</groupId>
  <artifactId>tools.cli</artifactId>
  <version>0.2.1</version>
</dependency>
<dependency>
  <groupId>commons-discovery</groupId>
  <artifactId>commons-discovery</artifactId>
  <version>20040218.194635</version>
</dependency>
<!--dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>com.springsource.org.apache.commons.discovery</artifactId>
  <version>0.4.0</version>
</dependency-->
<dependency>
  <groupId>commons-cli</groupId>
  <artifactId>commons-cli</artifactId>
  <version>1.0</version>
</dependency>
<dependency>
  <groupId>commons-io</groupId>
  <artifactId>commons-io</artifactId>
  <version>2.2</version>
</dependency>
<dependency>
  <groupId>jline</groupId>
  <artifactId>jline</artifactId>
  <version>1.0</version>
</dependency>
<dependency>
  <groupId>org.codehaus.groovy</groupId>
  <artifactId>groovy</artifactId>
  <!--version>1.7.0</version-->
  <version>1.8.0</version>
</dependency>
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.16</version>
</dependency>
<dependency>
  <groupId>xstream</groupId>
  <artifactId>xstream</artifactId>
```

```
    <version>1.2.2</version>
  </dependency>
  <dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-email</artifactId>
    <version>1.2</version>
  </dependency>
  <!--dependency>
    <groupId>javax.jmdns</groupId>
    <artifactId>jmdns</artifactId>
    <version>3.2.2</version>
    <version>3.4.1</version>
  </dependency-->
  <!--dependency>
    <groupId>org.objectweb.bonita</groupId>
    <artifactId>chainsaw</artifactId>
    <version>2.1</version>
  </dependency-->
  <dependency>
    <groupId>xerces</groupId>
    <artifactId>xercesImpl</artifactId>
    <version>2.10.0</version>
  </dependency>
  <dependency>
    <groupId>xalan</groupId>
    <artifactId>xalan</artifactId>
    <version>2.7.1</version>
  </dependency>
  <dependency>
    <groupId>bsf</groupId>
    <artifactId>bsf</artifactId>
    <version>2.4.0</version>
  </dependency>
  <dependency>
    <groupId>org.apache.ant</groupId>
    <artifactId>ant</artifactId>
    <version>1.8.2</version>
  </dependency>
  <dependency>
    <groupId>org.apache.ftpserver</groupId>
    <artifactId>ftpserver-core</artifactId>
    <version>1.0.6</version>
  </dependency>
  <dependency>
    <groupId>org.apache.mina</groupId>
    <artifactId>mina-core</artifactId>
    <version>2.0.4</version>
    <!--packaging>bundle</packaging-->
  </dependency>
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.6.4</version>
  </dependency>
```

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.6.4</version>
</dependency>
<dependency>
  <groupId>org.apache.ftpserver</groupId>
  <artifactId>ftplet-api</artifactId>
  <version>1.0.6</version>
  <!--packaging>bundle</packaging-->
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>1.6.4</version>
</dependency>
<dependency>
  <groupId>org.mortbay.jetty</groupId>
  <artifactId>jetty</artifactId>
  <version>7.0.0.pre5</version>
</dependency>
<dependency>
  <groupId>org.mortbay.jetty</groupId>
  <artifactId>jetty-deploy</artifactId>
  <version>7.0.0.pre5</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.5</version>
</dependency>
<dependency>
  <groupId>commons-httpclient</groupId>
  <artifactId>commons-httpclient</artifactId>
  <version>3.1</version>
</dependency>
<dependency>
  <groupId>tagsoup</groupId>
  <artifactId>tagsoup</artifactId>
  <version>0.9.7</version>
</dependency>
<dependency>
  <groupId>net.sf.saxon</groupId>
  <artifactId>saxon</artifactId>
  <version>8.7</version>
</dependency>
<!--dependency>
  <groupId>xch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>0.9.29</version>
</dependency-->
<dependency>
  <groupId>org.apache.karaf.shell</groupId>
  <artifactId>org.apache.karaf.shell.console</artifactId>
```

```
    <version>2.2.3</version>
  </dependency>
  <dependency>
    <groupId>org.apache.felix</groupId>
    <artifactId>org.apache.felix.main</artifactId>
    <version>3.2.2</version>
  </dependency>
  <dependency>
    <groupId>org.apache.felix</groupId>
    <artifactId>org.apache.felix.configadmin</artifactId>
    <version>1.2.4</version>
  </dependency>
</dependencies>

<build>
  <plugins>

    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>2.3.2</version>
      <configuration>
        <compilerId>groovy-eclipse-compiler</compilerId>
        <verbose>true</verbose>
        <debug>true</debug>
      </configuration>
      <executions>
        <execution>
          <goals>
            <goal>compile</goal>
          </goals>
        </execution>
      </executions>
      <dependencies>
        <dependency>
          <groupId>org.codehaus.groovy</groupId>
          <artifactId>groovy-eclipse-compiler</artifactId>
          <version>2.6.0-01</version>
        </dependency>
      </dependencies>
    </plugin>
    <plugin>
      <groupId>org.codehaus.groovy</groupId>
      <artifactId>groovy-eclipse-compiler</artifactId>
      <version>2.6.0-01</version>
      <extensions>true</extensions>
    </plugin>

    <!--plugin>
      <groupId>org.codehaus.gmaven</groupId>
      <artifactId>gmaven-plugin</artifactId>
      <version>1.4</version>
      <configuration>
        <debug>true</debug>
        <verbose>true</verbose>
```

```
    <stacktrace>true</stacktrace>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>generateStubs</goal>
        <goal>compile</goal>
        <goal>generateTestStubs</goal>
        <goal>testCompile</goal>
      </goals>
    </execution>
  </executions>
</plugin-->

<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-shade-plugin</artifactId>
  <version>1.5</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>shade</goal>
      </goals>
      <configuration>
        <filters>
          <filter>
            <artifact>*:*</artifact>
            <excludes>
              <exclude>META-INF/*.SF</exclude>
              <exclude>META-INF/*.DSA</exclude>
              <exclude>META-INF/*.RSA</exclude>
            </excludes>
          </filter>
        </filters>
        <artifactSet>
          <excludes>
            <exclude>junit:junit</exclude>
            <exclude>jmock:jmock</exclude>
          </excludes>
        </artifactSet>
        <transformers>
          <!--mainClass>org.radigan.system.tools.Command</mainClass-->
          <transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTr
            <manifestEntries>
              <Main-Class>org.radigan.system.tools.Command</Main-Class>
              <Manifest-Version>1.0</Manifest-Version>
              <Bundle-Activator>org.radigan.naomi.service.NaomiActivator</Bundle-Activator>
              <Bundle-ManifestVersion>2</Bundle-ManifestVersion>
              <Bundle-Name>NAOMI</Bundle-Name>
              <Bundle-SymbolicName>naomi</Bundle-SymbolicName>
              <Bundle-Version>1.0.0.SNAPSHOT</Bundle-Version>
              <Export-Package>org.radigan.naomi.service;version="1.0.0.SNAPSHOT",
                              org.apache.karaf.branding;version="1.0.0.SNAPSHOT"</Export-Package>
```

```
                    <Import-Package>org.osgi.service.cm;version="1.2",
                                    org.osgi.framework,sun.misc;resolution:=optional</Import-Package>
                  </manifestEntries>
                </transformer>
                <transformer implementation="org.apache.maven.plugins.shade.resource.AppendingTransform
                  <resource>META-INF/services/org.radigan.system.tools.ITool</resource>
                </transformer>
                <transformer implementation="org.apache.maven.plugins.shade.resource.AppendingTransform
                  <resource>META-INF/services/org.radigan.system.tools.Tool</resource>
                </transformer>
              </transformers>
            </configuration>
          </execution>
        </executions>
      </plugin>

      <!--plugin>
        <groupId>org.apache.felix</groupId>
        <artifactId>maven-bundle-plugin</artifactId>
        <extensions>true</extensions>
        <configuration>
          <instructions>
            <Export-Package>org.radigan,org.apache.karaf.branding</Export-Package>
            <Bundle-SymbolicName>${project.artifactId}</Bundle-SymbolicName>
            <Bundle-Activator>org.radigan.naomi.osgi.activator.DaemonActivator</Bundle-Activator>
            <Embed-Dependency>*;scope=runtime|compile;optional=false</Embed-Dependency>
          </instructions>
        </configuration>
      </plugin-->

    </plugins>

  </build>

  <distributionManagement>
    <repository>
      <name>NAOMI</name>
      <id>releases</id>
      <url>http://naomi-2:8081/nexus/content/repositories/releases/</url>
    </repository>
    <snapshotRepository>
       <name>NAOMI</name>
       <uniqueVersion>false</uniqueVersion>
      <id>snapshots</id>
      <url>http://naomi-2:8081/nexus/content/repositories/snapshots</url>
    </snapshotRepository>
  </distributionManagement>
</project>
<!-- *EOF* -->
```

MANIFEST.MF

```
Manifest-Version: 1.0
Bnd-LastModified: 1356164882035
Build-Jdk: 1.6.0_33
Built-By: mac
```

```
Bundle-Activator: org.radigan.naomi.service.SystemActivator
Bundle-ManifestVersion: 2
Bundle-Name: NAOMI
Bundle-SymbolicName: org.radigan.naomi
Bundle-Version: 1.0.0.SNAPSHOT
Created-By: Apache Maven Bundle Plugin
Export-Package: org.radigan.naomi.service;version="1.0.0.SNAPSHOT",org.apache.karaf.branding;version="1
Import-Bundle:
Import-Package:
Main-Class: org.radigan.system.tools.Command
SPI-Provider: org.radigan.*
Spring-Context: *;public-context:=false
Tool: Bnd-1.50.0
```

org.radigan.naomi.service.Module

```
## Service Provider List for org.radigan.naomi.service.Module
## Mac Radigan
org.radigan.naomi.wumpus.impl.SimulationFunctor
org.radigan.naomi.roar.impl.InterfaceResource
org.radigan.naomi.nyancat.impl.GraphvizReport
## *EOF*
```

org.radigan.system.tools.AbstractTool

```
## Service Provider List for org.radigan.system.tools.AbstractTool
## Mac Radigan
org.radigan.system.interpreter.ScriptTool
#org.radigan.system.interpreter.ClojureTool
org.radigan.system.test.TestTool
## *EOF*
```

1.22 Appendix B.9: Distributable Sources

naomi.vbs

```
' @file     naomi.vbs
' @author   Mac Radigan
Option Explicit
Set WshShell = WScript.CreateObject("WScript.Shell")
Dim commandLine = "java -jar " & WScript.Path & "\..\" & "naomi.jar" & Join(Wscript.Arguments, " ")
WshShell.Run(commandLine)
WScript.Quit
' *EOF*
```

application.xml

```
// @file     application.naomi
// @author   Mac Radigan

package org.radigan.naomi.modules

import org.radigan.naomi.wumpus.service.FunctorList
import org.radigan.naomi.wumpus.impl.SimulationFunctor as F

public class ApplicationModule extends FunctorList {
  public void initialize() {

/*
    this << new F('p1', ['d1'],           [],              ['a1','a2','a3'])
```

```
    this << new F('p2', ['d2','d3','d3'], ['d1'],          ['a1','a2','a3'])
    this << new F('p3', ['d4'],           ['d2','d3','d3'], ['a1','a2','a3'])
    this << new F('p4', ['d5'],           ['d2'],          ['a1','a2','a3'])
*/

    def f = []
    // sources 01-09
    f << new F('p1',  ['d1','d2','d3','d4'], [],                                  ['a1','a2',
    f << new F('p2',  ['d5'],                [],                                  ['a51','a52
    f << new F('p3',  ['d6'],                [],                                  ['a61','a62
    f << new F('p4',  ['d7'],                [],                                  ['a41','a42
    f << new F('p5',  ['d8'],                [],                                  ['a41','a42
    // functors A 11-19
    f << new F('p11', ['d11'],               ['d1','d2','d3','d4'],               ['a1','a2',
    f << new F('p12', ['d12'],               ['d5'],                              ['a12'])
    f << new F('p13', ['d13'],               ['d6'],                              ['a13'])
    f << new F('p14', ['d14'],               ['d7','d7'],                         ['a14'])
    f << new F('p15', ['d15'],               ['d8'],                              ['a14'])
    // functors B 21-29
    f << new F('p21', ['d21'],               ['d11'],                             ['a1','a2',
    f << new F('p22', ['d22'],               ['d12'],                             ['a1','a8']
    f << new F('p23', ['d23'],               ['d13'],                             ['a61','a67
    f << new F('p24', ['d24'],               ['d13'],                             ['a61','a67
    f << new F('p25', ['d25'],               ['d15','d15'],                       ['a71','a77
    // sinks 31+
    f << new F('p31', ['d31'],               ['d21'],                             ['a1','a2',
    f << new F('p32', ['d32','d33'],         ['d22'],                             ['a15'])
    f << new F('p33', ['d32','d33','d39'],   ['d23'],                             ['a30'])
    f << new F('p34', ['d34'],               ['d1','d2','d3','d4','d5'],          ['a31'])
    f << new F('p35', ['d35'],               ['d1','d2','d3','d4','d1','d2','d3','d4'], ['a31'])
    f << new F('p36', ['d36'],               ['d6','d6'],                         ['a36'])
    f << new F('p37', ['d37'],               ['d13','d13'],                       ['a37'])
    f << new F('p38', ['d38','d39','d40'],   ['d1','d2','d3','d4','d5','d13','d13'], ['a37','a38
    f << new F('p39', ['d39','d40','d41'],   ['d25','d14'],                       ['a47','a48

    this.addAll(f)

  }
}

// *EOF*
```

naomi-features.xml

```
<!-- naomi-featues.xml -->
<!-- Mac Radigan -->
<features xmlns="http://karaf.apache.org/xmlns/features/v1.0.0">
  <feature name="naomi" version="1.0.0">
    <bundle>mvn:org.radigan/naomi/1.0.0-SNAPSHOT</bundle>
  </feature>
</features>
<!-- *EOF* -->
```

naomi.conf

```
// @file:     naomi.config
// @author:   Mac Radigan
```

```
environments {

  omni {
    roar {
      port = 8080
      cache = '/Users/mac/temp/cache'
    }
  }

}

// *EOF*
```

naomi.cfg

```
## naomi.cfg
## Mac Radigan
home=/Users/mac/local/naomi/dist
## *EOF*
```

1.23 Appendix B.10: Resource Files and Branding

org.radigan.system.tools.Tool

```
## Commons Discovery List for org.radigan.system.tools.Tool
## Mac Radigan
org.radigan.system.interpreter.ScriptTool
org.radigan.system.test.TestTool
## *EOF*
```

branding.properties

```
welcome = \
\r\n\
\r\n\
\u001B[36m _|      _|    _|_|      _|_|    _|      _|  _|_|_|  \u001B[0m\r\n\
\u001B[36m _|_|    _|  _|    _|  _|    _|  _|_|  _|_|    _|    \u001B[0m\r\n\
\u001B[36m _|  _|  _|  _|_|_|_|  _|    _|  _|  _|  _|    _|    \u001B[0m\r\n\
\u001B[36m _|    _|_|  _|    _|  _|    _|  _|      _|    _|    \u001B[0m\r\n\
\u001B[36m _|      _|  _|    _|    _|_|    _|      _|  _|_|_|  \u001B[0m\r\n\
\r\n\
\r\n\
\u001B[1mNormalized Algorithm Object Messaging Interface (NAOMI)\u001B[0m\r\n\
\r\n\
Hit '\u001B[1m<tab>\u001B[0m' for a list of available commands\r\n\
and '\u001B[1mcmd --help\u001B[0m' for help on a specific command.\r\n\
Hit '\u001B[1m<ctrl-d>\u001B[0m' or '\u001B[1mosgi:shutdown\u001B[0m' to shutdown NAOMI System.\r\n
\r\n\
```

nyancat.ascii

```
+      o     +              o
    +             o     +       +
o          +
    o  +           +        +
+        o     o       +        o
-_-_-_-_-_-_-_,------,      o
_-_-_-_-_-_-_-|   /\_/\
-_-_-_-_-_-_-~|__( ^ .^)  +     +
_-_-_-_-_-_-_""  ""
```

```
+      o          o   +       o
    +         +
o        o          o      o     +
    o           +
+      +     o         o      +
```

wumpus.ascii

```
          ,@@@@@@@@:‘
         ,@@@@@@@@@#,
        ,@@@@@@@@@@@@:
      ‘,@@@@@@@@@@@@@@:
      ,@@@@@@@@@@@@@@@@:
     .@@@@@@@@@@@@@@@@@@,
    ‘@@@@@@. #@@: #@@@@@@,
  ‘,@@@@@@@‘ #@@: +@@@@@@@,
  ,@@@@@@@#.;.@@ @ @@@@@@@@.
  @@@@@@@@@@@@@@@@@@:@@@@@#
  @@@@@ #@@@@@@@@@@@@@,.@@@@@
  @@@@  @@@@@@@@@@@@@, ‘@@@@
  @@@;  #@@#;@’+@;@@@,  @@@@
  @@@;  ##.@ @‘’@ @ @,  @@@@
  @@@;  ’‘           ,  @@@@
  @@@;  @    ‘ ‘‘ ‘ ,,  @@@@
  @@@;  #@.@ @‘+@ @,@,  @@@@
  @@@;  @@@@@@@@@@@@@,  @@@@
  ’@@@  ‘@#@@@@@@@@#@  ‘@@@’
   ’@@@   @@@@@@@@@#   @@#’
    +@@@   ,,,,,,,,   @@@+‘
     @@@@           ‘@@@@
    @@@@@@         ‘@@@@@@
   #@@@+@@@‘      ‘@@@’#@@#
  #+## ‘+###      ###+  +###
```

wumpus-face.ascii

```
                fL.          fG
               tLL,         tLL:
              .DLLL:        LLLL.
             :fLLLLL.     ,LLLLLf:
             LL,  tLf     fG,  tLf
             LG #L Gf     fG Kf Lf
             LG ,: Gf     fG ,; Lf
             LLi  fLf     fLi  jLf
             ,fLLLLf:     ,GLLLLL:
```

```
,     .                                    ;
L    j.                                  i    f
LL  ,L:     .                      .     .L,  Lf
LLLLLLG     f.    G:    fi    :j     G     fLGLLLf
LL. LLff   .DLE   ;Lf    LL.   GL:   LLi   tfLt  Lf
f    jLL, tGLLj :LLLG  LLLf  LLLf. LLLLj iLLL   .j
```

```
j     LLLiLfLLGtLLLLLjLLLLGffLLLLLtLLLGL,fLL.    i
.    .LG.  fLL:  LLL,  GLLL  fLLL  :LLG   LL
      G    .L,    Lj    LG    LL    ,L:    G
      t     t     ;.    .      ,     G     L
```

1.24 Appendix B.7: Unit Tests and Control Scripts

config.ncs

```
// config.ncs
// Mac Radigan

import org.radigan.naomi.utilities.ServiceFactory

def serviceFactory = ServiceFactory.getInstance()
def config = serviceFactory.getConfiguration()
println "config: ${config}"
println "port: ${config.roar.port}"

// *EOF*
```

matrix.ncs

```
// matrix.ncs
// Mac Radigan

import org.radigan.system.utilities.MatrixImpl

double[][] data = [
  [-1,  1,  0,  0],
  [ 0, -1,  1,  0],
  [ 0, -2,  2,  0],
  [ 0,  0,  0,  0]
]

//  in=-1;                                % flux direction, in
//  Wu = sign(W);                         % oriented incident matrix, Wu
//  L = Wu'*Wu;                           % Laplacian matrix, L
//  A = abs(L.*(L<0));                    % adjacency matrix, A
//  D = L+A;                              % degree matrix, D
//  [i,j]=find(D); N=i(end);              % determine number of nodes
//  k=1:N;                                % submatrix indices, k
//  s=zeros(size(A)); s(k,k)=diag(k)>0;   % submatrix diagonal, s
//  C=(A+s)^N;                            % transitive closure, C
//  lambda = flipud(sort(eig(L)));        % eigenvalues, lambda
//  ac = lambda(2);                       % algebraic connectivity, ac
//  con = ac>0;                           % check connectivity, con
//  SS = find((min(Wu)<0)&(max(Wu)<-in)); % source nodes, SS
//  S = find(C(SS,N));                    % starting nodes

def W = new MatrixImpl(data)
println "W:\n${W}"

// oriented incident matrix, Wu = sign(W)
def Wu = W.signum()
println "Wu:\n${Wu}"
```

```
// Laplacian matrix, L = Wu'*Wu
def L = Wu.t()*Wu
println "L:\n${L}"

// adjacency matrix, A = abs(L.*(L<0))
def A = (L**(L<<0)).abs()
println "A:\n${A}"

// degree matrix, D
def D = L+A
println "D:\n${D}"

// determine number of nodes, N
def N = D.lastNonzeroRow()
println "N:\n${N}"
println "\n"

// transitive closure, C=(A+s)^N
def sm = MatrixImpl.zeros(A).fill(MatrixImpl.eye(A))
def C = (A+sm)^N
println "C:\n${C}"

// algebraic connectivity, ac = eigW(L, 2)
def ac = L.eigW(1)
println "ac:\n${ac}"
println "\n"

// check connectivity, con = ac>0
def con = ac>0
println "con:\n${con}"
println "\n"

// source nodes, SS
//def SS = (Wu.min().lt(0) & Wu.max().lt(1)).find()
def SS = ((Wu.min()<<0)&(Wu.max()<<1)).find()
println "SS:\n${SS}"
println "\n"

// starting nodes, S
def S = C.indToRows(C.find(SS,[N]))
println "S:\n${S}"
println "\n"

// positive flux adjacency matrix, Ap = A.*(Wu>0)
def Ap = A**(Wu>>0)
println "Ap:\n${Ap}"

// negative flux adjacency matrix, Ap = A.*(Wu<0)
def An = A**(Wu<<0)
println "An:\n${An}"

// *EOF*
```

module.ncs

```
// module.ncs
```

```
// Mac Radigan

import org.radigan.naomi.utilities.ServiceFactory
import org.radigan.naomi.service.Module

def serviceFactory = ServiceFactory.getInstance()
def modules = serviceFactory.getModules()
println "modules: ${modules}"

// *EOF*
```

naomi.ncs

```
// naomi.ncs
// Mac Radigan

import org.radigan.naomi.impl.NaomiImpl

def naomi = new NaomiImpl()

naomi.start()

System.in.withReader { println "CRL+C to exit"; println it.readLine() }

// *EOF*
```

nyancat.ncs

```
// nyancat.ncs
// Mac Radigan

import org.radigan.naomi.wumpus.impl.SimulationFunctor as F
import org.radigan.naomi.wumpus.utilities.WumpusUtil as Util
import org.radigan.naomi.nyancat.impl.GraphvizReport
import org.apache.commons.io.FileUtils

def f = []
// sources 01-09
f << new F('p1',  ['d1','d2','d3','d4'], [],                                  ['a1','a2','a3
f << new F('p2',  ['d5'],                [],                                  ['a51','a52','a
f << new F('p3',  ['d6'],                [],                                  ['a61','a62'])
f << new F('p4',  ['d7'],                [],                                  ['a41','a42'])
f << new F('p5',  ['d8'],                [],                                  ['a41','a42'])
// functors A 11-19
f << new F('p11', ['d11'],               ['d1','d2','d3','d4'],                ['a1','a2','a3
f << new F('p12', ['d12'],               ['d5'],                               ['a12'])
f << new F('p13', ['d13'],               ['d6'],                               ['a13'])
f << new F('p14', ['d14'],               ['d7','d7'],                          ['a14'])
f << new F('p15', ['d15'],               ['d8'],                               ['a14'])
// functors B 21-29
f << new F('p21', ['d21'],               ['d11'],                              ['a1','a2','a3
f << new F('p22', ['d22'],               ['d12'],                              ['a1','a8'])
f << new F('p23', ['d23'],               ['d13'],                              ['a61','a67'])
f << new F('p24', ['d24'],               ['d13'],                              ['a61','a67'])
f << new F('p25', ['d25'],               ['d15','d15'],                        ['a71','a77'])
// sinks 31+
f << new F('p31', ['d31'],               ['d21'],                              ['a1','a2','a3
f << new F('p32', ['d32','d33'],         ['d22'],                              ['a15'])
```

```
f << new F('p33', ['d32','d33','d39'],   ['d23'],                                  ['a30'])
f << new F('p34', ['d34'],               ['d1','d2','d3','d4','d5'],               ['a31'])
f << new F('p35', ['d35'],               ['d1','d2','d3','d4','d1','d2','d3','d4'], ['a31'])
f << new F('p36', ['d36'],               ['d6','d6'],                              ['a36'])
f << new F('p37', ['d37'],               ['d13','d13'],                            ['a37'])
f << new F('p38', ['d38','d39','d40'],   ['d1','d2','d3','d4','d5','d13','d13'],   ['a37','a38','a
f << new F('p39', ['d39','d40','d41'],   ['d25','d14'],                            ['a47','a48'])

def wu = new Util(f)
def ext = "gif"

process = { name ->
  def n = wu.getFunctorIndex(name, 'ns1')
  def fn = f[n]
  def wf = wu.filter(n as int)
  println wf
  def cacheDir = new File("/Users/mac/temp/cache")
  def resDir = new File("${cacheDir}/${fn.getNamespace()}/${fn.getName()}/${wu.getId()}")
  def rptFile = new File("${resDir}/${fn.getName()}.${ext}")
  def report = new GraphvizReport(wf)
  println report.toString()
  if( ! rptFile.exists()) {
    FileUtils.forceMkdir(resDir)
    report.save(rptFile, ext)
    println report.toString()
    println "file complete: ${rptFile}"
  } else {
    println "file exists: ${rptFile}"
  }
}

['p39'].each { name -> process(name) }

// *EOF*
```

request.ncs

```
// request.ncs
// Mac Radigan

def path = "/roar/abc.xml"
def matcher = path =~ /\/roar\/([A-z][A-z0-9]*)\.(xml|xsd|xsl|xhtml)/
if( ! matcher.matches()) throw new IllegalArgumentException("Bad request")
def algo = matcher[0][1]
println "algo: ${algo}"

// *EOF*
```

roar.ncs

```
// roar.ncs
// Mac Radigan

import org.radigan.naomi.roar.impl.RoarImpl

def roar = new RoarImpl()

roar.start()
```

```
System.in.withReader { println "CRL+C to exit"; println it.readLine() }

// *EOF*
```

types.ncs

```
// types.ncs
// Mac Radigan

import org.radigan.naomi.impl.TypesDatabaseImpl as Db
import org.radigan.naomi.utilities.TypeCategory

def db = Db.getInstance()

def file = new File("dist/config/datatypes.xml")
db.parse(file.text)
println db.toString()

def t1 = "Foo"
def t2 = "Bar"
use(TypeCategory) {
  println t1 in t2
  println t2 in t1
}

// *EOF*
```

wumpus.ncs

```
// wumpus.ncs
// Mac Radigan

import org.radigan.naomi.wumpus.impl.WumpusImpl

def wumpus = new WumpusImpl()
println wumpus

// *EOF*
```

wumpusutil.ncs

```
// wumpusutil.ncs
// Mac Radigan

import org.radigan.naomi.wumpus.impl.SimulationFunctor as F
import org.radigan.naomi.wumpus.utilities.WumpusUtil as Util
import java.util.concurrent.atomic.AtomicIntegerArray
import org.radigan.system.utilities.MatrixImpl

def f = []
// sources 01-09
f << new F('p1',  ['d1','d2','d3','d4'], [],                          ['a1','a2','a3'])
f << new F('p2',  ['d5'],                [],                          ['a1','a2','a3'])
f << new F('p3',  ['d6'],                [],                          ['a1','a2','a3'])
// functors A 11-19
f << new F('p11', ['d11'],               ['d1','d2','d3','d4'],       ['a1','a2','a3'])
f << new F('p12', ['d12'],               ['d5'],                      ['a1','a2','a3'])
f << new F('p13', ['d13'],               ['d6'],                      ['a1','a2','a3'])
```

```
// functors B 21-29
f << new F('p21', ['d21'],              ['d11'],                    ['a1','a2','a3'])
f << new F('p22', ['d22'],              ['d12'],                    ['a1','a2','a3'])
f << new F('p23', ['d23'],              ['d13'],                    ['a1','a2','a3'])
// sinks 31+
f << new F('p31', ['d31'],              ['d21'],                    ['a1','a2','a3'])

def wu = new Util(f)
println wu

println "state vector"
def sv = [
  'd1':1,
  'd2':2,
  'd3':3,
  'd4':0
  ]
//wu.execute(sv)

def tp = []
tp << Thread.start { println "a" }
tp << Thread.start { println "b" }
tp*.join()

def n = wu.getFunctorIndex('p31', 'ns1')
def fn = f[n]
def wf = wu.filter(n as int)
println wf

println "workflow:"
wf.execute()

println "\nsinks: ${wf.getSinks()}"

// *EOF*
```

1.25 Appendix B.11: Octave/Matlab Sources

exercises.m

```
%% exercises.m
%% Mac Radigan

clear all


types = {'d1','d2','d3','d4'};
functors = {'p1','p2','p3','p4'};

syms(types{:});
syms(functors{:});
productions = cell(length(functors),3);  % initialize production list

%------------------------------
%   p   y1...    x1...
```

```
%------------------------------
productions = {                 ...
  [p1], [d1],    []           ; ...  % starting productions (sources)
  [p2], [d2 d3 d3], [d1]      ; ...  % intermediate productions
  [p3], [d4],    [d2 d3 d3]   ; ...  % final production (sink)
};

W = wumpus(productions, types, functors);
[N Wu A D L C lambda ac con S] = wumpusEval(W);
W
Wu
A
D
L
C
lambda
ac
con
S
wumpusValidate(W, true);
st = wumpusRes(W);
st

%% *EOF*
```

wumpus.m

```
%% wumpus.m
%% Mac Radigan

function W = wumpus(productions, types, functors)

  % define index and flux direction constants
  in=-1; DP=1; DY=2; DX=3;

  % checks if a datatype is in a list
  idx = @(s) regexprep(s,'[A-z]', '');
  tok = @(s) regexp(findsym(s),',','split');

  % create the oriented incident matrix, W
  W = zeros(length(types),length(functors));
  for pp=1:size(productions,1), p=str2num(idx(findsym(productions{pp,DP})));
    if length(productions{pp,DX})
      % out-degree
      for xx=cellfun(@(s) str2num(s), idx(tok(productions{pp,DX})), ...
          'UniformOutput', false); x=xx{:};
        for qq=1:size(productions,1)
          for qy=symToIdx(productions{qq,DY});
            if qy==x
              W(x,p) = W(x,p) -in;
            end
          end
        end
      end
    end
    % in-degree
```

```
    for yy=cellfun(@(s) str2num(s), idx(tok(productions{pp,DY})), ...
           'UniformOutput', false); y=yy{:};
      for qq=1:size(productions,1)
        if length(productions{qq,DX})
          for qx=symToIdx(productions{qq,DX});
            if qx==y
              W(y,p) = W(y,p) +in;
            end
          end
        end
      end
    end
  end

function y = symToIdx(s)
  idx = @(s) regexprep(s,'[A-z]', '');
  y = [];
  for n=1:length(s)
    y(n)= str2num(idx(findsym(s(n))));
  end

%% *EOF*
```

wumpusEval.m

```
%% wumpusEval.m
%% Mac Radigan

function [N Wu A D L C lambda ac con S] = wumpusEval(W)

  nassert = @(a,b) arrayfun(@(x,y)assert(x==y),a,b);

  in=-1;                                % flux direction, in
  Wu = sign(W);                         % oriented incident matrix, Wu
  L = Wu'*Wu;                           % Laplacian matrix, L
  A = abs(L.*(L<0));                    % adjacency matrix, A
  D = L+A;                              % degree matrix, D
  [i,j]=find(D); N=i(end);              % determine number of nodes
  k=1:N;                                % submatrix indices, k
  s=zeros(size(A)); s(k,k)=diag(k)>0;   % submatrix diagonal, s
  C=(A+s)^N;                            % transitive closure, C
  lambda = flipud(sort(eig(L)));        % eigenvalues, lambda
  ac = lambda(2);                       % algebraic connectivity, ac
  con = ac>0;                           % check connectivity, con
  SS = find((min(Wu)<0)&(max(Wu)<-in)); % source nodes, SS
  S = find(C(SS,N));                    % starting nodes

%% *EOF*
```

wumpusExec.m

```
%% wumpusExec.m
%% Mac Radigan

function wumpusExec(W)

  nassert = @(a,b) arrayfun(@(x,y)assert(x==y),a,b);
  [N Wu A D L C lambda ac con S] = wumpusEval(W);
```

```
%% *EOF*
```

wumpusRes.m

```
%% wumpusRes.m
%% Mac Radigan

function st = wumpusRes(W)

  nassert = @(a,b) arrayfun(@(x,y)assert(x==y),a,b);
  [N Wu A D L C lambda ac con S] = wumpusEval(W);

  [st cl] = resolve(Wu, N);


function [vo cl] = resolve(Wu, vi)
  in=-1;                                % flux direction, in
  cl = 0;
  vo = [];
% for v=vi
%   [i dc]  = find(Wu(:,v)==-in);
%   [vd dc] = find(Wu(i,:)==in);
%   if(any(vd))
%     vo=horzcat(vo,resolve(Wu,vd));
%   else
%     vo=[];
%   end
% end

  cx = [];
  vx = [];
  for v=vi
    fprintf(1, 'resolving... %d\n', v);
    [dc i]  = find(Wu(:,v)==-in);
    [dc vd] = find(Wu(:,i)==in);
    if(~any(vd))
      cl = 1;
      vo=horzcat(vo,vi)
    else
      [vx cxx] = resolve(Wu,vd);
      cx(end+1) = cxx;
    end
  end
  if(any(cx)&all(cx))
    cl = 1;
  else
    vo=horzcat(vo,vx)
  end
    %keyboard
    %if(~all(vx))
    %  error 'dependency not found'
    %end
  fprintf(1, 'returning...\n');
  %vi
  %vo
```

```
  %cl

%% *EOF*
```

wumpusValidate.m

```
%% wumpusValidate.m
%% Mac Radigan

function valid = wumpusValidate(W, foe)

  nassert = @(a,b) arrayfun(@(x,y)assert(x==y),a,b);
  sel = @(x,n) x(n);
  [N Wu A D L C lambda ac con S] = wumpusEval(W);

  % assert that the dependencies are satisified by the sources
  if(~con)
    msg = sprintf('connection error: ac=%d', ac);
    if(foe), warning(msg); else error(msg); end
  end

  % assert consistency between Floyd-Warshall and spectral graph theory
  for v=S
    M = Wu;
    M(setdiff(v,S),:) = [];             % skip other sources
    l2 = sel(flipud(sort(eig(L))),2);  % second eigenvalue of M
    if( ~(l2>0&C(N,v))|(~(l2>0)&~C(N,v)) )
      disp('C*(A)'); disp(C)
      msg = sprintf('tautology error: C*(A)[%d,%d]>0 <-> l2>0', N, v);
      if(foe), warning(msg); else error(msg); end
    end
  end

%% *EOF*
```