The Java CoG Kit

Gregor von Laszewski

Argonne National Laboratory

University of Chicago

gregor@mcs.anl.gov

http://www.cogkit.org

Updated slides will be available on the CoG Kit web site

Funding sources & Acknowledgement

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

Community

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

Outline

- What is the CoG Kit?
  - Basic definitions
  - History of the CoG Kit
  - CoG Kit in action
  - Relationship to GT versions
- Selected Project Components
  - Design: Abstractions
  - Programming with Abstractions (Task Graphs)
  - Visual components: Portals & Applications
- Conclusion

Introduction

Observation

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

Abstractions

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

Result: CoG Kits

- CoG Kits make Grid programming simple and new technologies are easy to integrate
- We focus on a CoG Kit for Java
  - Others are possible Python, …
  - Availability: Java CoG Kit since 1997
- Our designs are based on experience reaching back to the beginnings of Meta-computing and Grid-computing

Relationship towards GT

- Since GT3 CoG Kit is an essential part of GT
- CoG Kit protects from an evolving standard
- CoG Kit provides simple programming model
- CoG Kit supports portal and GUI developers
- CoG Kit is a bridge between application and Grid middleware developers.
- CoG Kit has known to be working with
  - GT1.0 GT2.4, GT3.0.2, GT3.2, GT3.21, SSH
  - (under dev.) GT3.9.x, GT4, Condor
  - (community) Unicore

Relationship to WS-RF

- Because …
  - (Quote: Steve Tuecke, at a GGF meeting): "WS-RF is still under development. The OASIS standards process has just begun."
- COG Kit
  - Provides investment protection while standards are developed.
  - Provides a more sophisticated programming model than just services
  - Focus on what many high end-users need
  - You can influence the direction of the CoG Kit by partnering with us
  - Will work with future versions of GT, SSH, Condor (planed), …
  - We intend to support and integrate with upcoming new standards

History

History (cont.)

1992	Term Metacomputer is introduced
1994	von Laszewski: **Graphical Meta-computing environment**
1995	I-Way
Nov. 1996	von Laszewski joins Argonne
1997	Globus version 1 / **first release of jglobus based on concepts of protocols and services includes a high throughput fault tolerant workflow prototype**
1998	CoG Team: LDAP browser wins Novel developers award
1998	Term Grid is introduced
1999	**Term Java CoG Kit is introduced to include jglobus and other components in a single toolkit**
2000	CoG Team: The experimental personal gatekeeper of the Java CoG Kit was been able to be installed in less than 30 seconds on a PC including Windows, a similar Globus service installed by an experienced administrator required one to multiple days.
2001	Cog Team: The Java CoG Kit experimental Infogram Service architecture was defined combining execution and information Services as a single Grid service.
2001	CoG Kit was selected by IBM to demonstrate Grid computing in Boards of Directors meeting
2002	Globus Team defines OGSI / **Java CoG Kit for GT2.x and GT3.0/OGSI based, includes visual components such as the CoG Kit Desktop, GridFTP interface, GRAM interface**

History

2003	WSRF is defined
2002 and 2003	**CoG team rewrites the workflow component and introduces GridAnt and a new workflow engine called Karajan that contains flow and structural control (DAGs, conditions, loops). The workflow concept is expandable. Check pointing and minimal features for fault tolerance are available.** **Result caching is possible based on method signatures.**
2004	**CoG team introduces the concept of Grid providers making it possible that the CoG Kit can in principal submit to GT2, GT3, GT4, or SSH. Community demonstrates also UNICORE provider.**
2004	**A class project shows it is possible to define PBS and LSF providers (not distributed with the CoG Kit)**
2004	**CoG Kit receives best research poster award at SC 2004**
2005	**Major new Java CoG Kit release.** ***GT2, GT3, GT4, SSH providers** *** Workflow** *** Graphical components** *** New manual**

the globus alliance
www.globus.org/cog

Use of CoG Kits

Frameworks

OGCE

Portalware

Middleware

Perl CoG Kit

Python
Java
OGCE
Chef
Cactus
GridLab
GridSphere
GADU
Launchpad
GRIP/Unicore
Nimrod/G
GAF4J
XCAT
Globus Toolkit Version 3
OGSA/OGSI
Globus Toolkit Version 2
DOE
CERN
PACI
NPACI

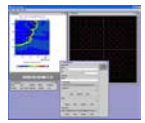
CoG

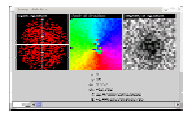
Astronomy
Climate
Nano Materials
HEP
Genome
Chemistry
Chimera
Pegasus
Workflow
Parameter Studies
GridAnt
Task Management
Job Submission
File Transfer
GridFTP & RFT
Security
Information Services

Access Grid

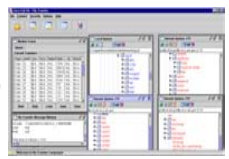
Production

Applications





Grid Services

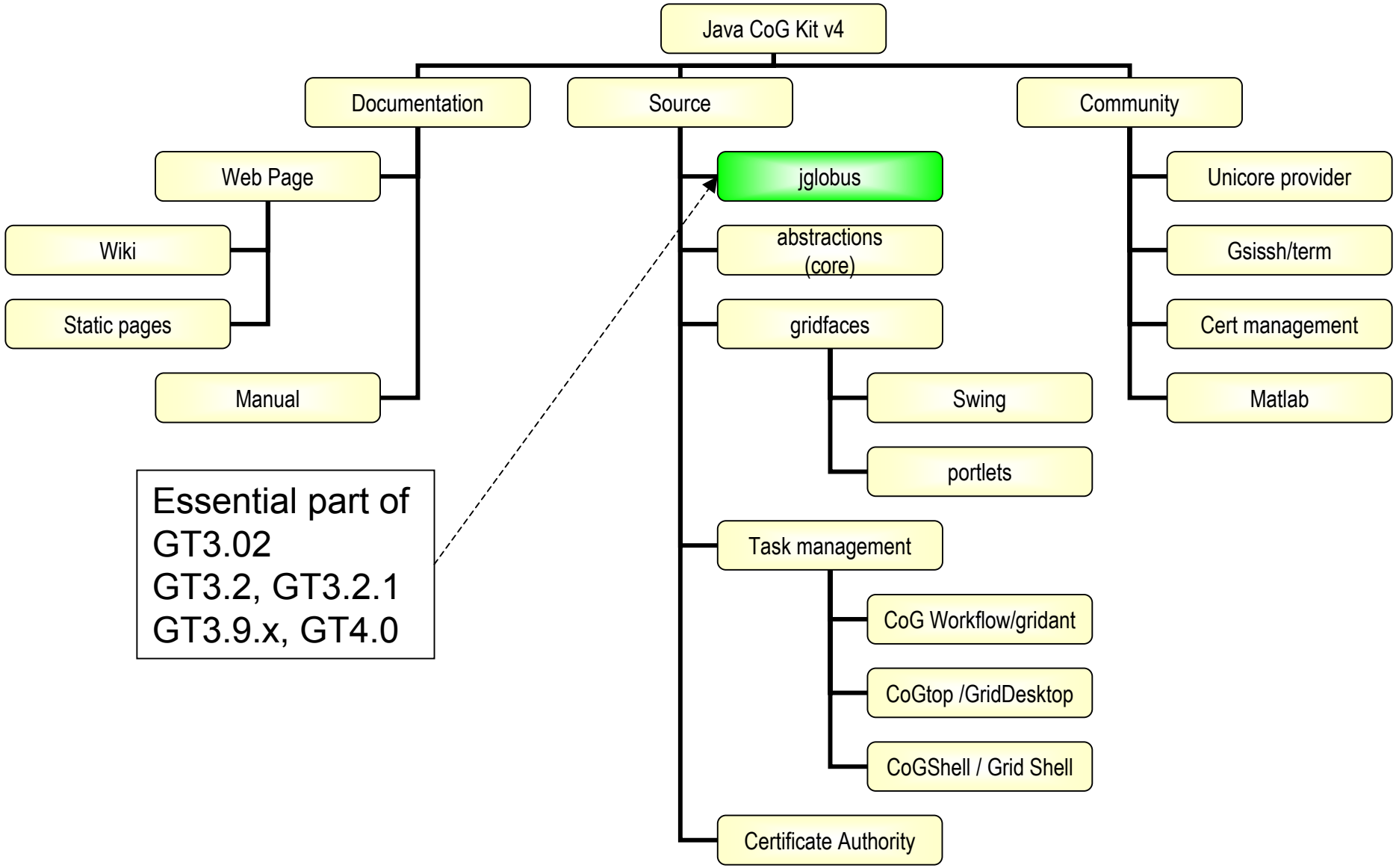


Design

Design

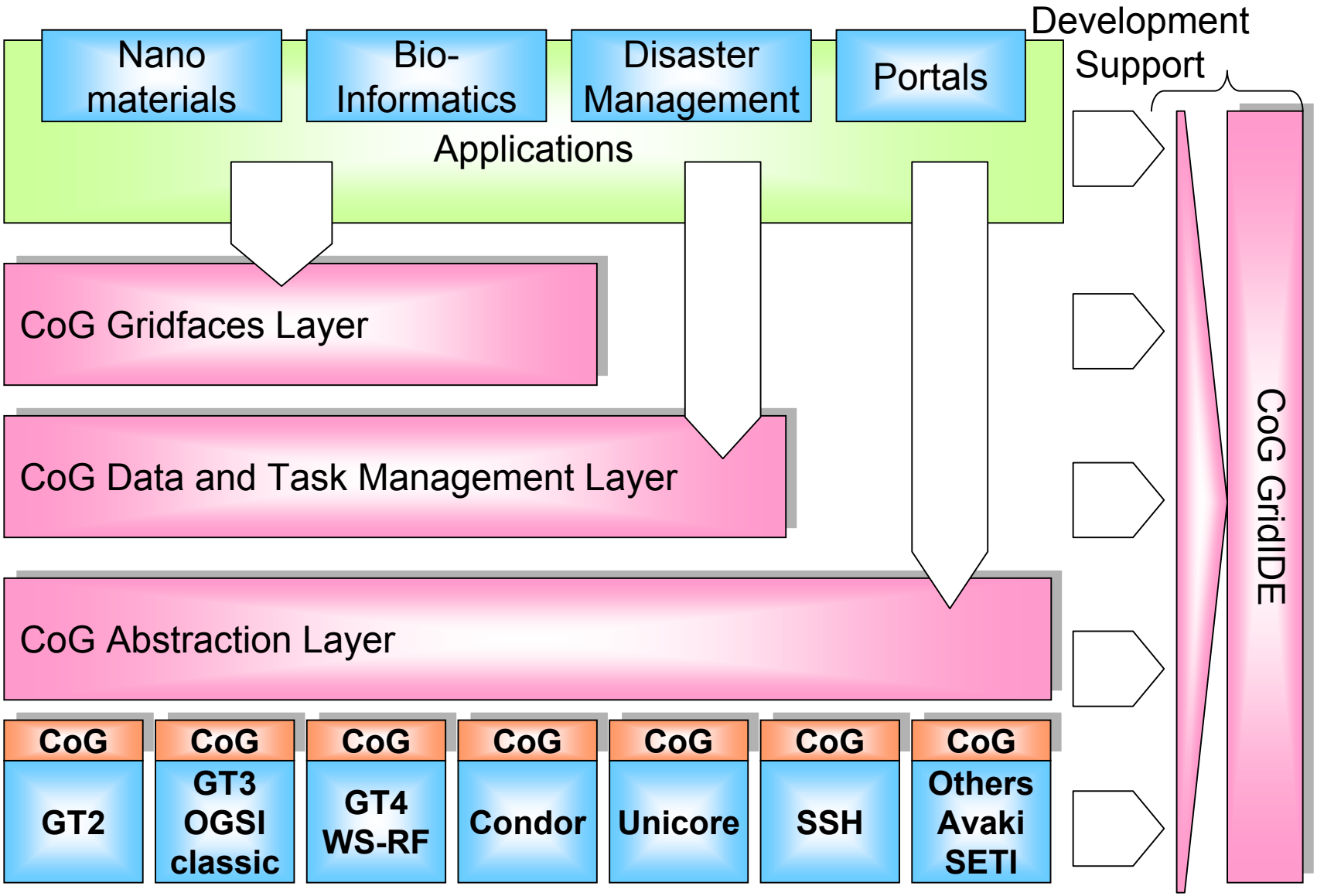
- Based on layered model
- Flexible
- Expandable
- Based on Java interfaces
- Abstracts protocols
- Abstracts services
- Provides workflow

CoG Kit is more than jglobus



the globus alliance
www.globus.org/cog

CoG Abstraction Layers



Selected Project Components

Focus on Reusable APIs & Components

- Abstractions
  - Provide a simple programming model
- Workflow
  - Workflow abstraction
- Portals
  - Supporting APIs, abstractions and implementations for portals.
- jglobus1.2
  - GSI security in Java
  - GRAM protocal & client
  - gridFTP protocol & client
  - Myproxy client

- Not just API's but also their implementation

Focus on Abstractions and Patterns

- Abstraction above Grid Toolkits
  - Task Model
    - Jobs, information query, file transfer, authentication, others
  - Gridfaces model
    - Abstract views of GUIs to the Grid in different implementations (SWING, JSP, Portlets, …)
  - Data Types
    - Queues, Sets, Brokers, Schedulers. Based on Task model

Java CoG Kit abstractions

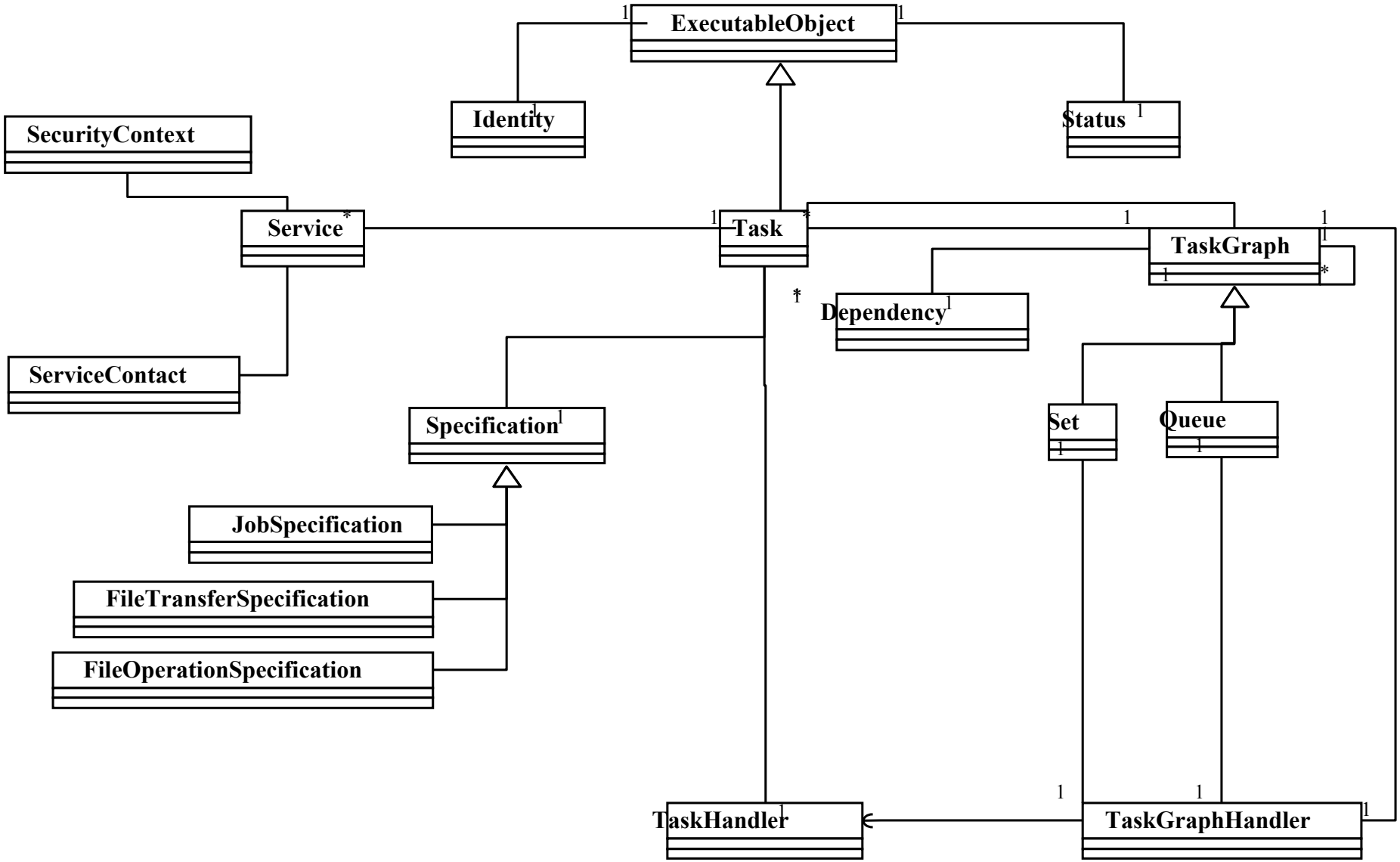
- A programming model based on a **task model** that simplifies elementary Grid patterns such as job execution, file transfer, and file operations.
- A programming model that includes **execution flows** in the form of directed acyclic graphs (DAG).
- The programming model is **decoupeling the definition from the implementation**, thus providing independence from current and future Grid versions.
- Only elementary Grid patterns are considered.
- It makes programming the Grid simple
- It makes developing Grid portals more easy
- Focus is selected functionality

Design

- ExecutableObject
- Task
- TaskGraph
- Handlers
- Events
- Service

the globus alliance
www.globus.org/cog

Design



Programming with Abstractions

A simple Programming Pattern

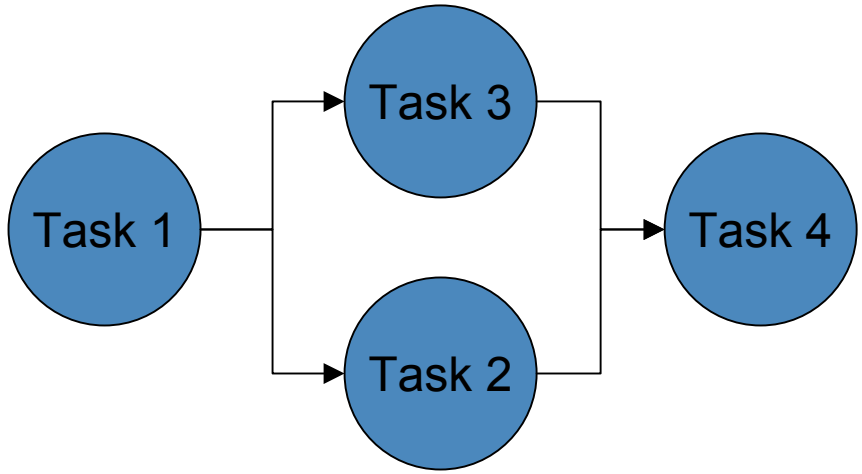
```
public class COG implements StatusListener{
   public void create() { … }
   public void submit () { … }
   public void statusChanged (StatusEvent e)  { … }
   public static void main (String arg[]){
      try {
         COG cog = new COG();
         cog.create();
         cog.submit();
       catch (Exception e) {
         logger.error("Something went wrong:", e);
   }
}
```

Executing a Simple TaskGraph

```
TaskGraph tg = new TaskGraphImpl();

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

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



Create a task

```
Task task1 = new Task();

JobSpecification spec = new JobSpecificationImpl();
spec.setExecutable("/bin/ls");
spec.addArguments("-la");
spec.setStdOutput("output.txt");

task1.setSpecification(spec);

// bind the task (late binding)
```

Status Monitoring

```
public void statusChanged (StatusEvent event) {

   Status status = event.getStatus( );

   logger.debug(``Status changed to '' +
                status.getStatusCode());

   if (status.getStatusCode( ) == Status.COMPLETED) {
      logger.info(``Task Done'');
   elsif (status.getStatusCode( ) == Status.FAILED) {
      logger.info(``Task Failed'');
       System.exit(1);
   }
}
```

Users can design their own Event handeling logic based on status changes

Using the Handler

```
try {
    handler.submit (cog);
  } catch (InvalidSecurityContextException ise) {
    logger.error(``Security Exception'', ise);
    System.exit(1);
  } catch (TaskSubmissionException tse) {
    logger.error(``TaskSubmission Exception'', tse);
    System.exit(1);
  } catch (IllegalSpecException ispe) {
    logger.error(``Specification Exception'', ispe);
    System.exit(1);
  } catch (InvalidServiceContactException isce){
    logger.error(``Service Contact Exception'', isce);
    System.exit(1);
  }
```

Detailed information
Can be retrieved if exceptions are used

Bind a Task to a Service

```
Service service = new ServiceImpl(Service.JOB_SUBMISSION);
service.setProvider(``GT3_2_1'');

// Set Security Context – e.g. certificates and such
SecurityContext securityContext =
    CoreFactory.newSecurityContext(``GT3_2_1'');
securityContext.setCredentials(null); // e.g. set it to default in ./globus
service.setSecurityContext(securityContext);

// Set Contact – e.g. where to go to
ServiceContact serviceContact =
    new ServiceContactImpl(
    "http://127.0.0.1:8080/ogsa/services/base/gram/
    MasterForkManagedJobFactoryService");
service.setServiceContact(serviceContact);

task.setService(Service.JOB_SUBMISSION_SERVICE, service);
```

the globus alliance
www.globus.org/cog

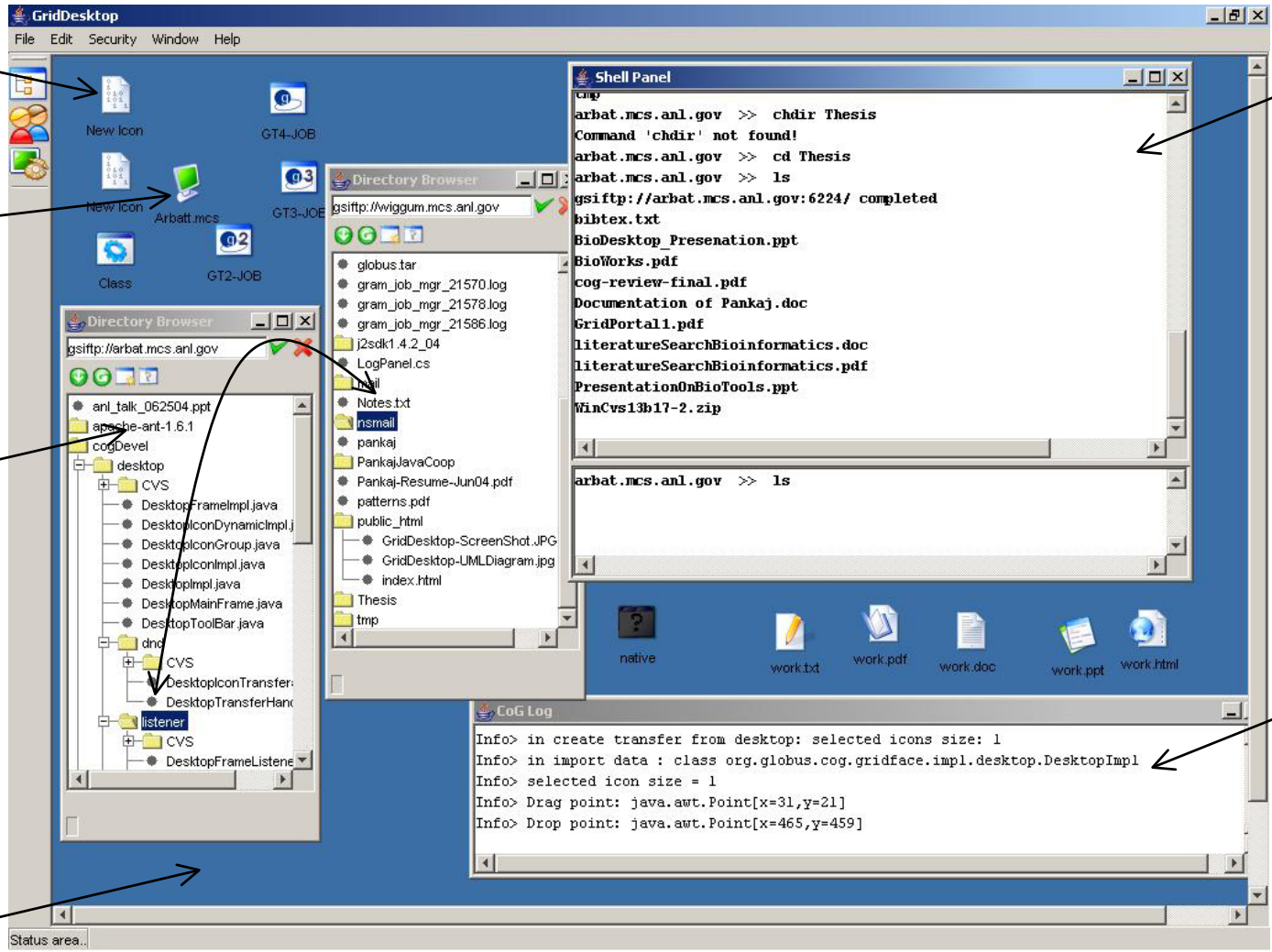
CoG Kit Desktop

Job Icons

Machine Icons

File Transfer GUI

Native Icons

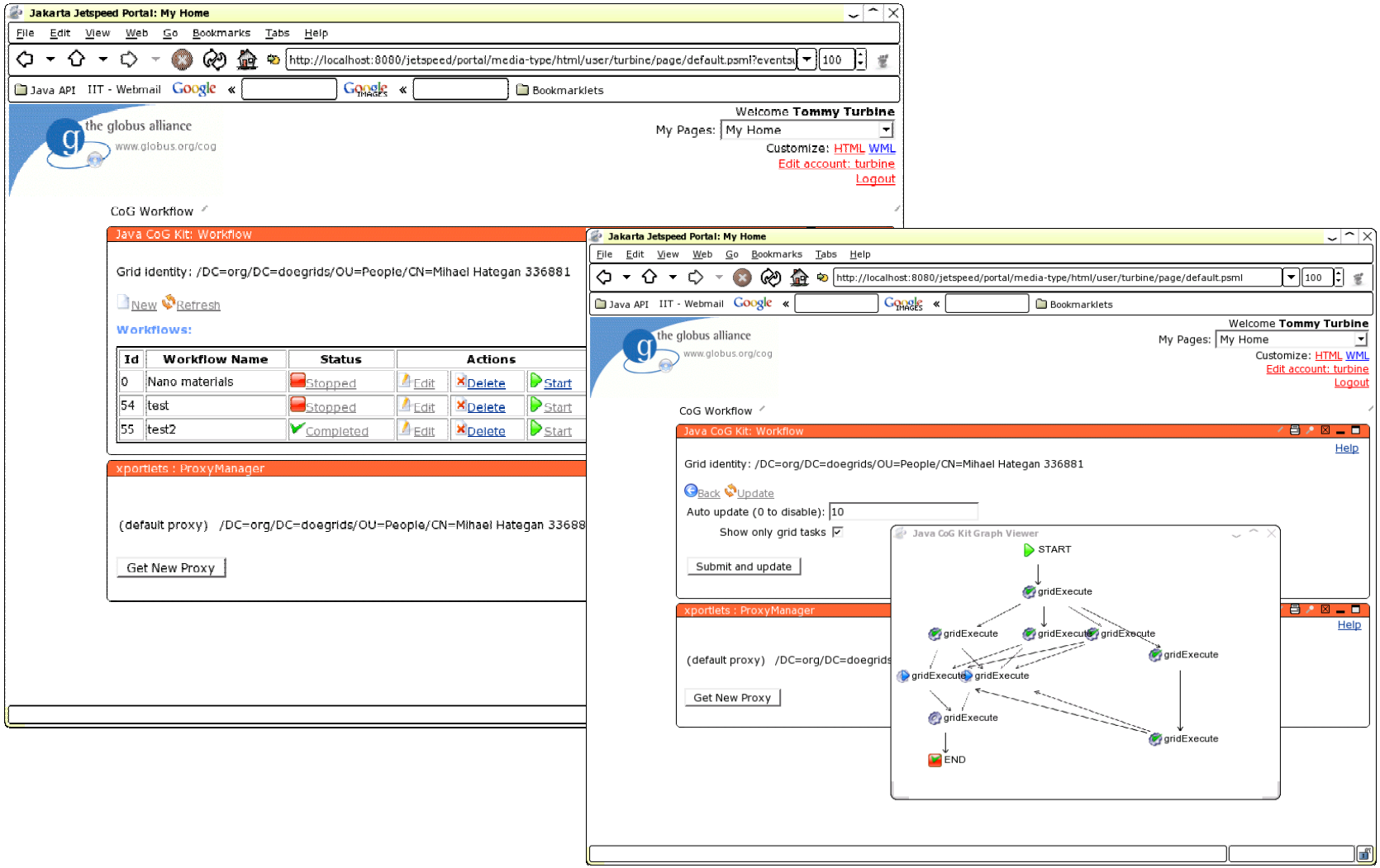


Grid Shell

Grid Log

the globus alliance
www.globus.org/cog

Portlets: OGCE.org



Contributing

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

Conclusion

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

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