XtreemOS

Enabling Linux for the Grid

Grid Application Programming: The SAGA API

Thilo Kielmann

VU University, Amsterdam

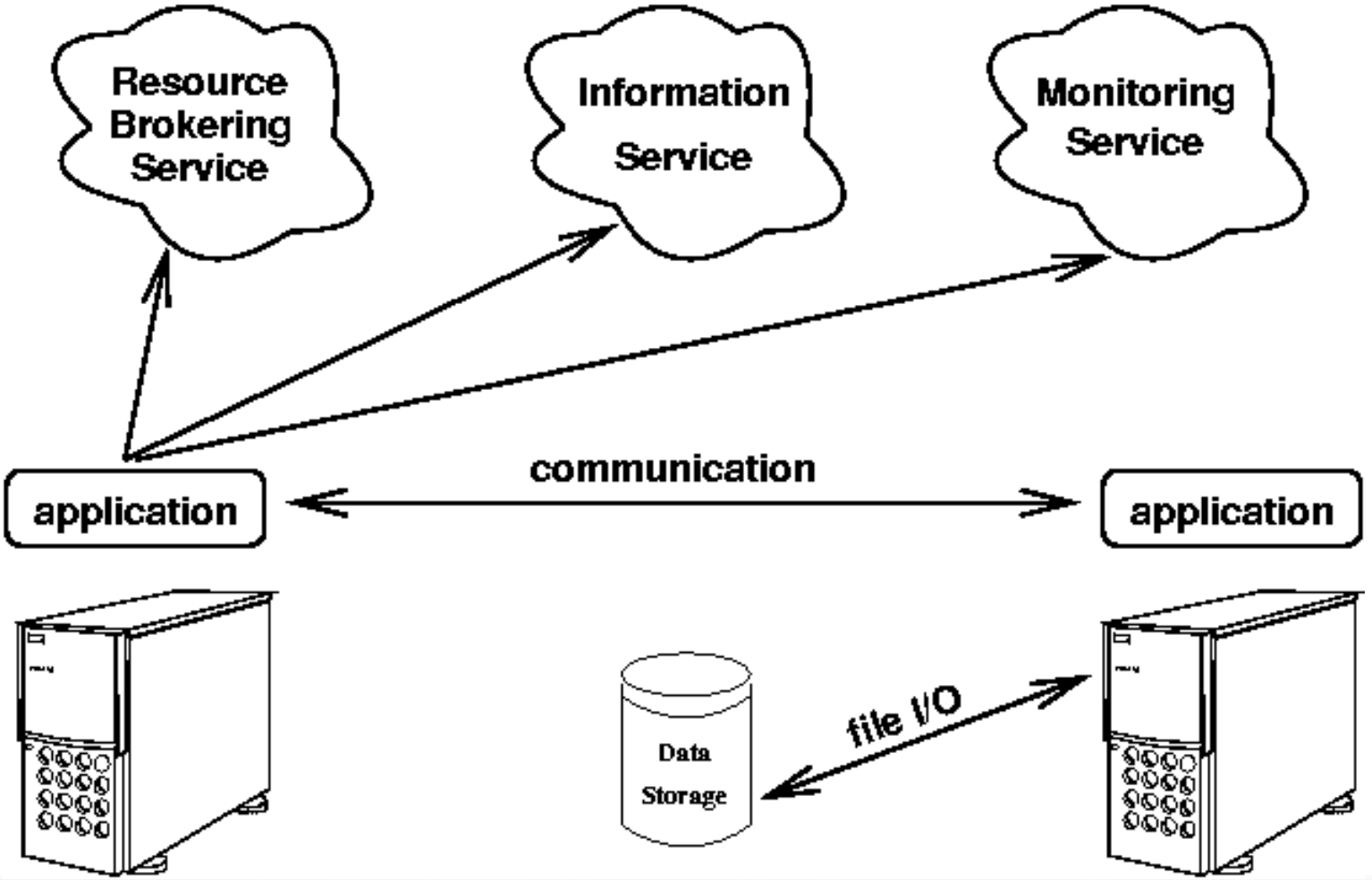
kielmann@cs.vu.nl

XtreemOS IP project
is funded by the European Commission under contract IST-FP6-033576

Information Society Technologies

XtreemOS
Enabling Linux for the Grid

A Grid Application, seen from 10000 Feet



Grid Platforms (1)

- **Writing Grid applications means programming against the interface(s) of the respective middleware:**
  - Globus 2.x (C-based services, proprietary protocols and interfaces), the “de-facto standard”
  - Globus 3.1 (Grid services, OGSA/OGSI, Java-based) outdated before it was widely deployed
  - Globus 4.0.x Web services, WSRF-based, Java services
  - Globus 5.x REST-ful services, back to 2.x look & feel

Grid Platforms (2)

- gLite 3.0
  - DataGrid / EGEE projects provided set of services
  - Proprietary API's / interfaces
- NAREGI (Japanese NAtional REsearch Grid Infrastructure)
  - (OGSA) services to build virtual, integrated supercomputer
  - Provides GridMPI and GridRPC interfaces
- ssh
  - Minimalistic approach (e.g. used in PlanetLab)

Grid Platforms (3)

- Unicore, started as what today would be called a “portal”
  - no real “API”, tries to hide the Grid from the applications
  - Started with proprietary protocols and formats, now Web services
- Avaki, commercial version of the Legion project
  - Now “Sybase Avaki Enterprise Information Integration System”
  - Data access interfaces (data bases via Web services)

Grid Platforms (4)

- **Condor-G**
  - **Condor's high throughput computing, job submission via Globus**
  - **No explicit API and interfaces (Condor hides remoteness of execution)**
- **OMII-UK: Open Middleware Infrastructure Institute UK**
  - **Web services for remote compute/data access**
  - **Aims to provide the SAGA API to its clients**

Amazon Web Services:

- Elastic Compute Cloud (EC2)
  - allows to dynamically create/remove virtual machines with user-defined image (OS + application)
  - payment for CPU per hour
- Simple storage Service (S3)
  - provides persistent object storage, write-once objects
  - payment for storage volume and transfer volume

Highly dynamic service provider for compute and storage capacities

IaaS Cloud Platforms

- **Writing Grid Cloud applications means programming against the interface(s) of the respective middleware:**
  - Amazon EC2 and S3
  - Nimbus
  - Eucalyptus
  - Nebula
  - OpenNebula
  - 3Tera, GoGrid, RightScale, ...
  - OCCI (OGF's Open Cloud Computing Interface)

The Simple API for Grid Applications (SAGA): Towards a Standard

- The need for a standard programming interface
  - Projects keep reinventing the wheel again, yet again, and again
  - MPI as a useful analogy of community standard
  - OGF as the natural choice; established the SAGA-RG
- Community process
  - Design and requirements derived from 23 use cases
  - SAGA Design Team (OGF, Berkeley, VU, LSU, NEC)

OpenGridForum
OPEN FORUM | OPEN STANDARDS

Outline

- **The Simple API for Grid Applications (SAGA)**
  - Motivation & Scope
  - SAGA as an OGF Standard
- **The SAGA Landscape**
  - Interfaces
  - Language Bindings
- **SAGA Implementations**
  - Engine with Adaptors
  - C++, Java, Python

Grid Programming Nightmare: Copy a File with Globus GASS

```
int copy_file (char const* source,   char const* target)
{
globus_url_t                        source_url;
globus_io_handle_t                  dest_io_handle;
globus_ftp_client_operationattr_t   source_ftp_attr;
globus_result_t                     result;
globus_gass_transfer_requestattr_t  source_gass_attr;
globus_gass_copy_attr_t             source_gass_copy_attr;
globus_gass_copy_handle_t           gass_copy_handle;
globus_gass_copy_handleattr_t       gass_copy_handleattr;
globus_ftp_client_handleattr_t      ftp_handleattr;
globus_io_attr_t                    io_attr;
int                                 output_file = -1;

if ( globus_url_parse (source_URL, &source_url) != GLOBUS_SUCCESS ) {
  printf ("can not parse source_URL \"%s\"\n", source_URL);
  return (-1);
}

if ( source_url.scheme_type != GLOBUS_URL_SCHEME_GSIFTP &&
     source_url.scheme_type != GLOBUS_URL_SCHEME_FTP    &&
     source_url.scheme_type != GLOBUS_URL_SCHEME_HTTP   &&
     source_url.scheme_type != GLOBUS_URL_SCHEME_HTTPS  ) {
  printf ("can not copy from %s - wrong prot\n", source_URL);
  return (-1);
}
globus_gass_copy_handleattr_init  (&gass_copy_handleattr);
globus_gass_copy_attr_init        (&source_gass_copy_attr);

globus_ftp_client_handleattr_init (&ftp_handleattr);
globus_io_fileattr_init           (&io_attr);

globus_gass_copy_attr_set_io      (&source_gass_copy_attr, &io_attr);
                                    &io_attr);
globus_gass_copy_handleattr_set_ftp_attr
                                  (&gass_copy_handleattr,
                                    &ftp_handleattr);
globus_gass_copy_handle_init      (&gass_copy_handle,
                                    &gass_copy_handleattr);

if (source_url.scheme_type == GLOBUS_URL_SCHEME_GSIFTP ||
    source_url.scheme_type == GLOBUS_URL_SCHEME_FTP    ) {
  globus_ftp_client_operationattr_init (&source_ftp_attr);
  globus_gass_copy_attr_set_ftp (&source_gass_copy_attr,
                                 &source_ftp_attr);
}
else {
  globus_gass_transfer_requestattr_init (&source_gass_attr,
                                         source_url.scheme);
  globus_gass_copy_attr_set_gass(&source_gass_copy_attr,
             &source_gass_attr);
}

output_file = globus_libc_open ((char*) target,
              O_WRONLY | O_TRUNC | O_CREAT,
              S_IRUSR  | S_IWUSR | S_IRGRP |
              S_IWGRP);
if ( output_file == -1 ) {
  printf ("could not open the file \"%s\"\n", target);
  return (-1);
}
/* convert stdout to be a globus_io_handle */
if ( globus_io_file_posix_convert (output_file, 0,
                                   &dest_io_handle)
     != GLOBUS_SUCCESS) {
  printf ("Error converting the file handle\n");
  return (-1);
}

result = globus_gass_copy_register_url_to_handle (
         &gass_copy_handle, (char*)source_URL,
         &source_gass_copy_attr, &dest_io_handle,
         my_callback, NULL);
if ( result != GLOBUS_SUCCESS ) {
  printf ("error: %s\n", globus_object_printable_to_string
           (globus_error_get (result)));
  return (-1);
}
globus_url_destroy  (&source_url);
return (0);
}
```

Relief: Copy a File with SAGA

```
import org.ogf.saga.error.SagaException;
import org.ogf.saga.file.File;
import org.ogf.saga.file.FileFactory;
import org.ogf.saga.url.URL;

public class CopyFile {

    void copyFile(URL sourceUrl, URL targetUrl) {
        try {
            File f = FileFactory.createFile(sourceUrl);
            f.copy(targetUrl);
        } catch (SagaException e) {
            System.err.println(e);
        }
    }

}
```

- Provides the high level abstraction that application programmers need; will work across different systems
- Shields gory details of lower-level middleware system
- Like MapReduce – leave out details of distribution etc.

SAGA in a nutshell

- **A programming interface for grid applications**
  - provides common grid functionality
  - simple (80/20 rule, limited in scope)
  - integrated (“consistent”)
  - stable: does not change (incompatibly)
  - uniform, across middleware platforms
  - high level, what applications need

What SAGA is and is not

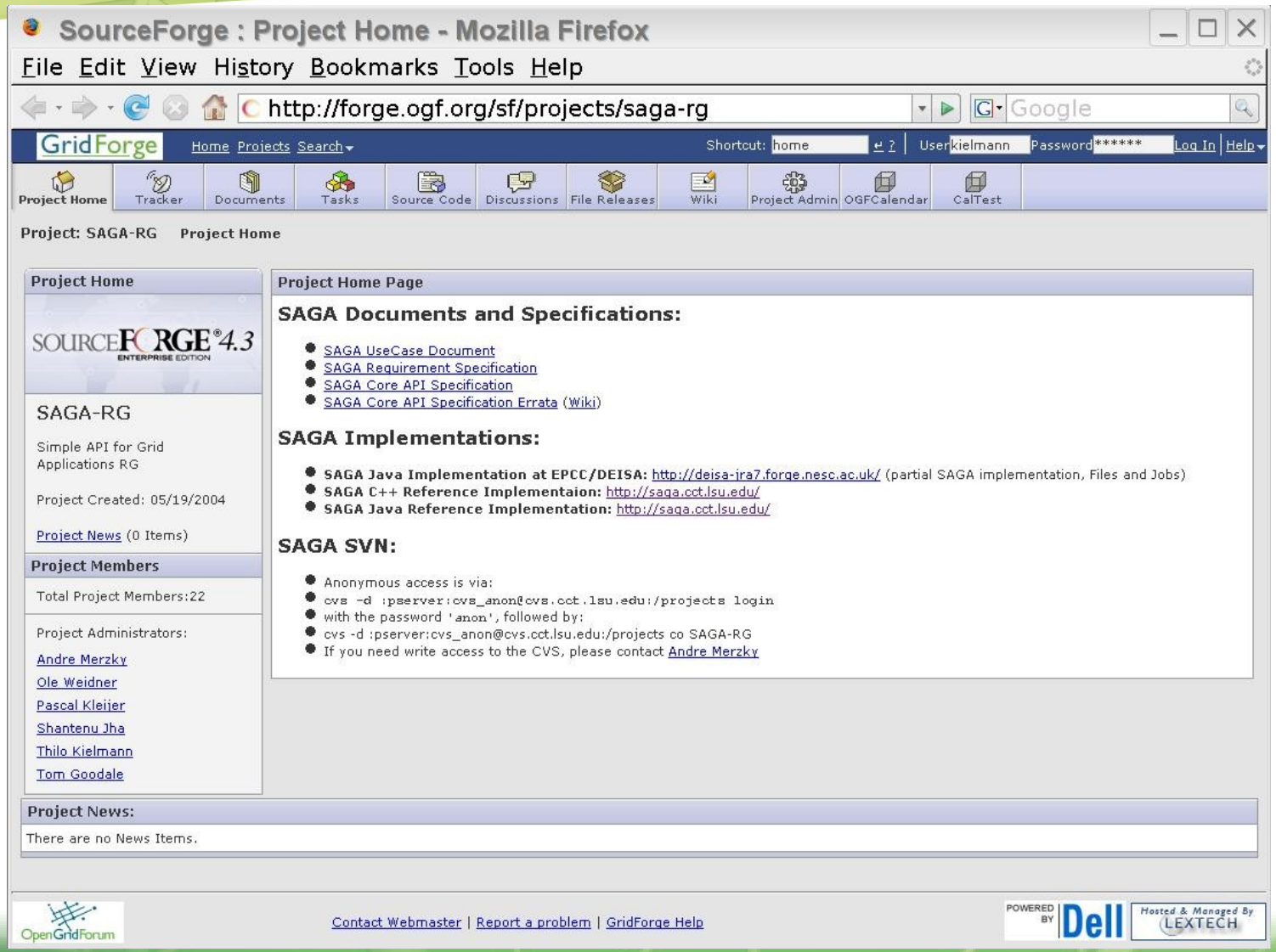
- **Is/Does:**
  - Simple API for Grid-Aware Applications
  - Deals with distributed infrastructure explicitly
  - High-level (= application-level) abstraction
  - A uniform interface to different middleware(s)
  - Client-side software
- **Is/Does NOT:**
  - Middleware
  - A service management interface!
  - Does not hide the resources - remote files, jobs

SAGA API: Towards a Standard

- **Community effort within OGF**
  - MPI as useful analogy of a community standard
- **Scope: (object-oriented) packages**
  - Functional Areas: Job Mgmt, Resource Mgmt, Data Mgmt, Logical Files, Streams, ...
  - Non-functional Areas: Asynchronous, Errors, ...
- **Language independent; specified using Scientific Interface Description Language (SIDL)**
  - Easy to map into specific language
  - Extensible via additional packages

XtreemOS
Enabling Linux for the Grid

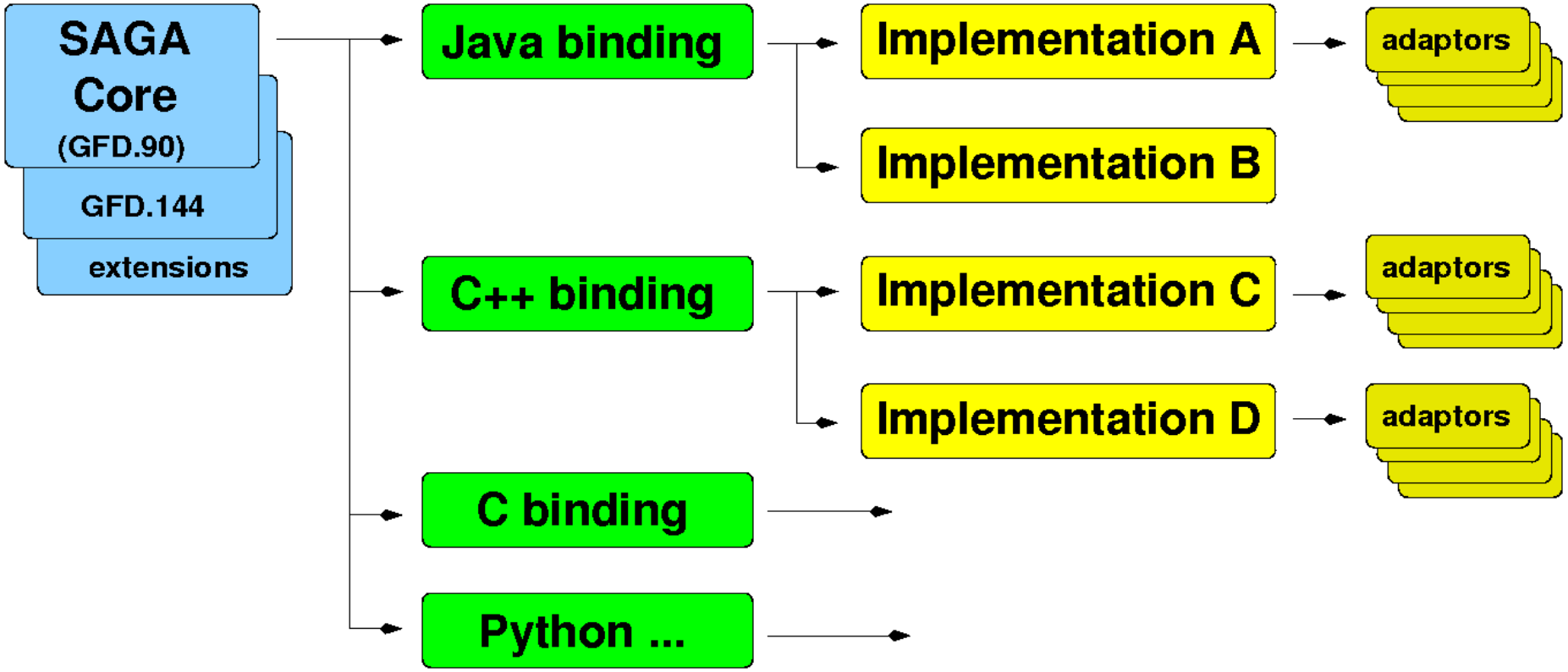
http://forge.ogf.org/sf/projects/saga-rg



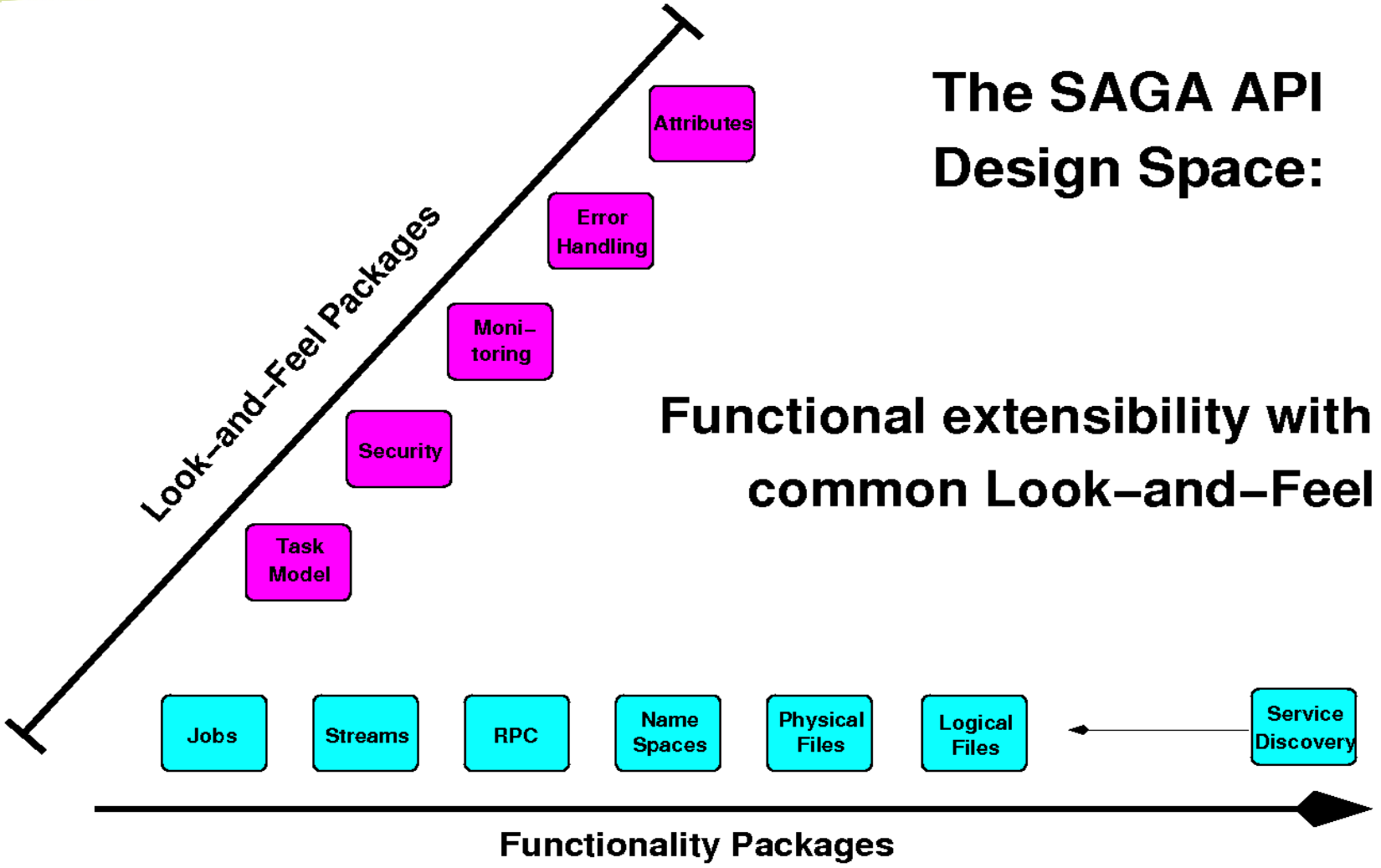
Outline

- **The Simple API for Grid Applications (SAGA)**
  - Motivation & Scope
  - SAGA as an OGF Standard
- **The SAGA Landscape**
  - Interfaces
  - Language Bindings
- **SAGA Implementations**
  - Engine with Adaptors
  - C++, Java, Python

The SAGA Landscape



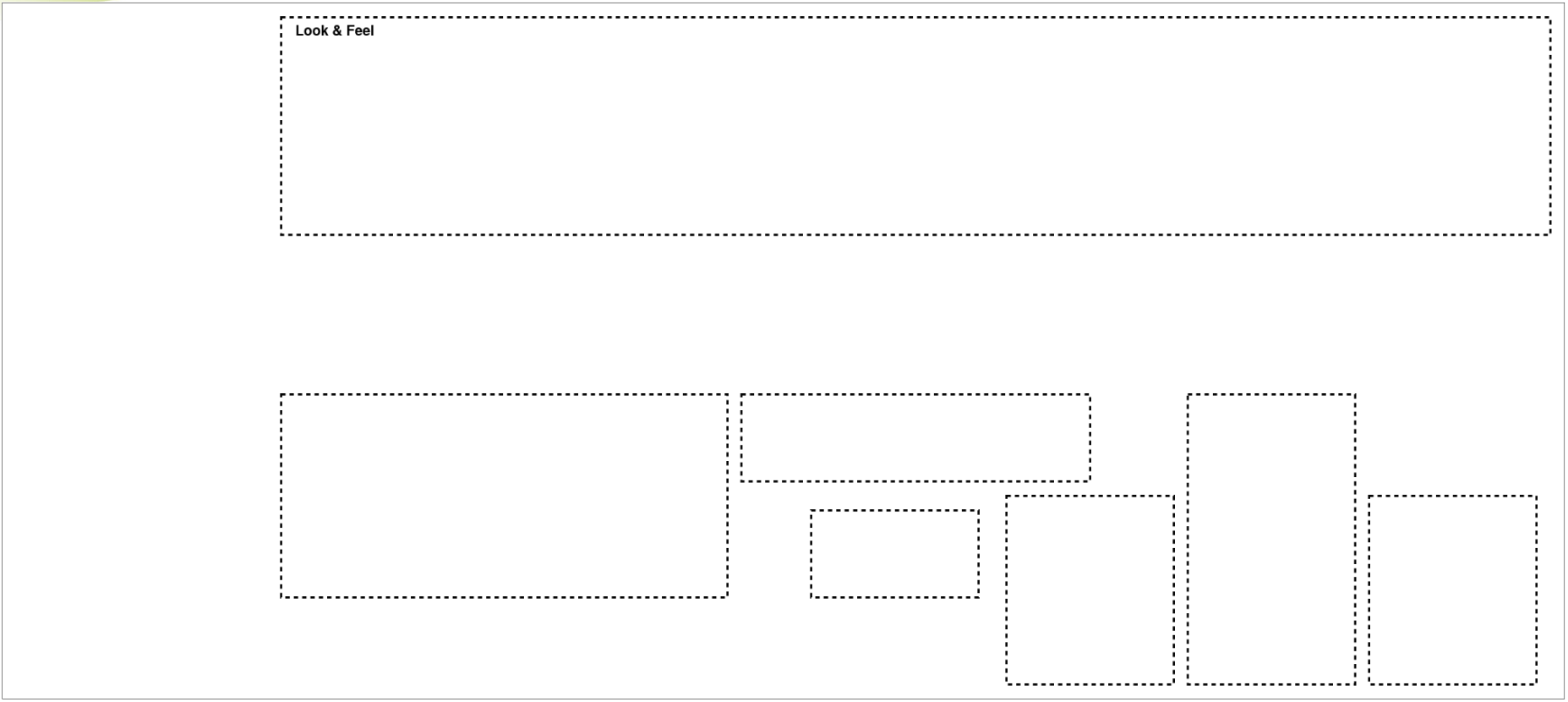
SAGA API Design Overview



The SAGA API Design Space:

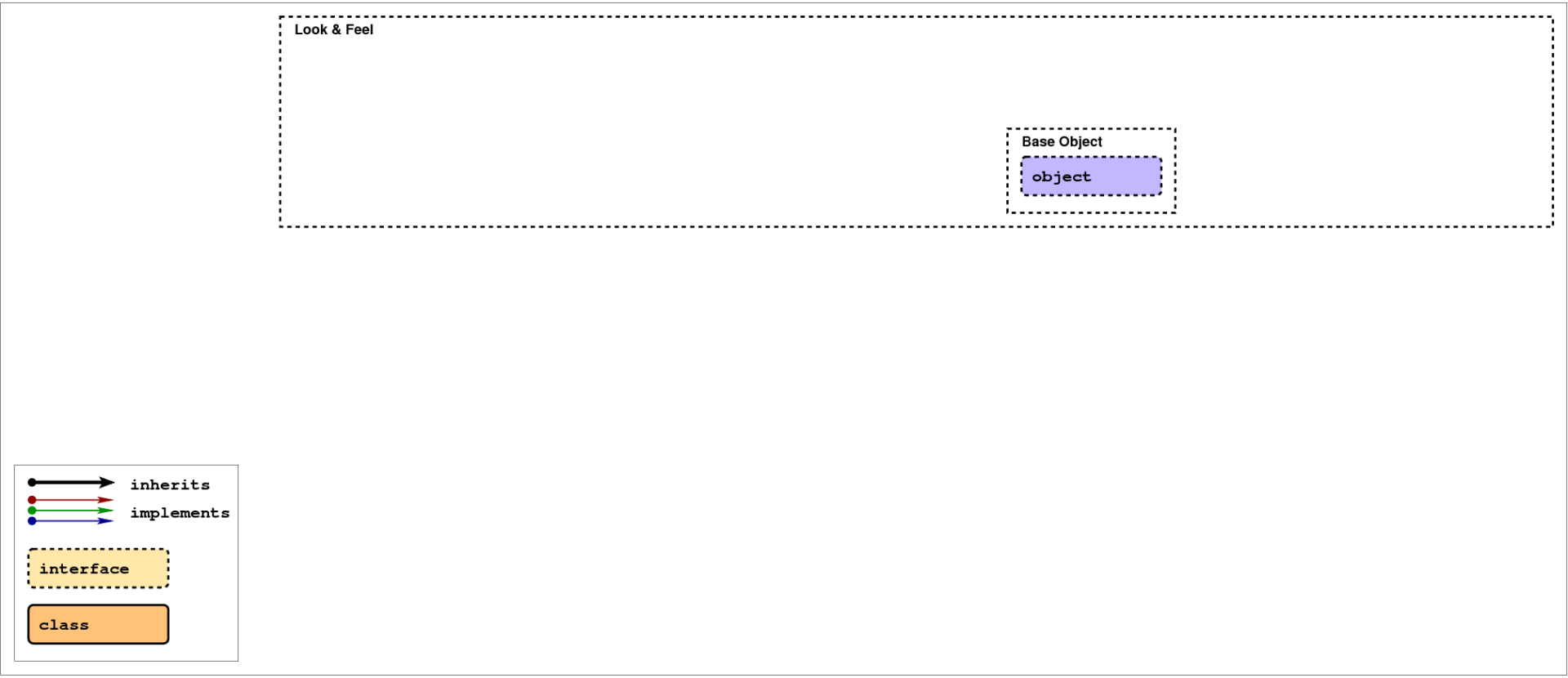
Functional extensibility with common Look–and–Feel

SAGA Interface Hierarchy



Look and feel: Top level Interfaces; Core SAGA objects needed by other API packages that provide specific functionality -- capability providing packages e.g., jobs, files, streams, namespaces etc.

SAGA Interface Tour

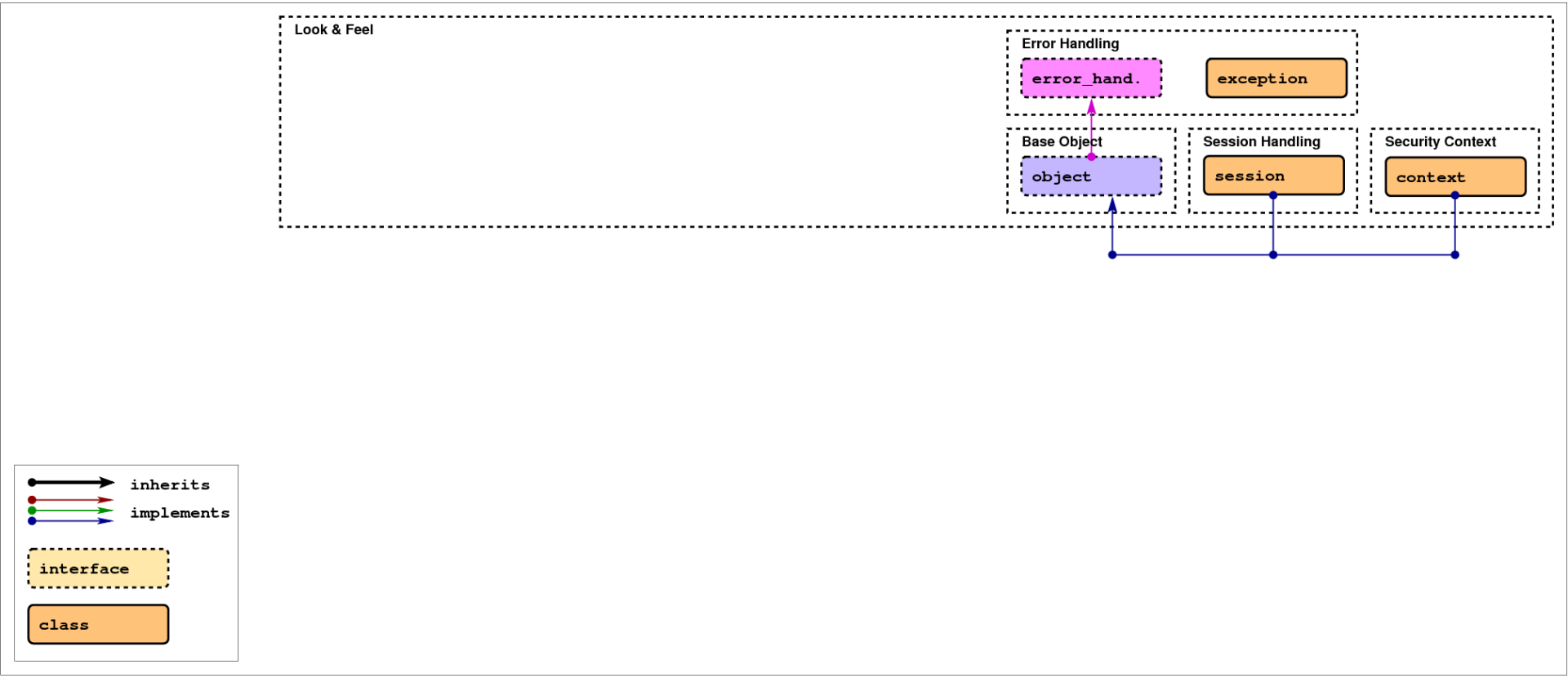


The common root for all SAGA classes.
Provides unique ID to maintain a list of SAGA objects. Provides methods (e.g., get_id()) that are essential for all SAGA objects

Errors and Exceptions

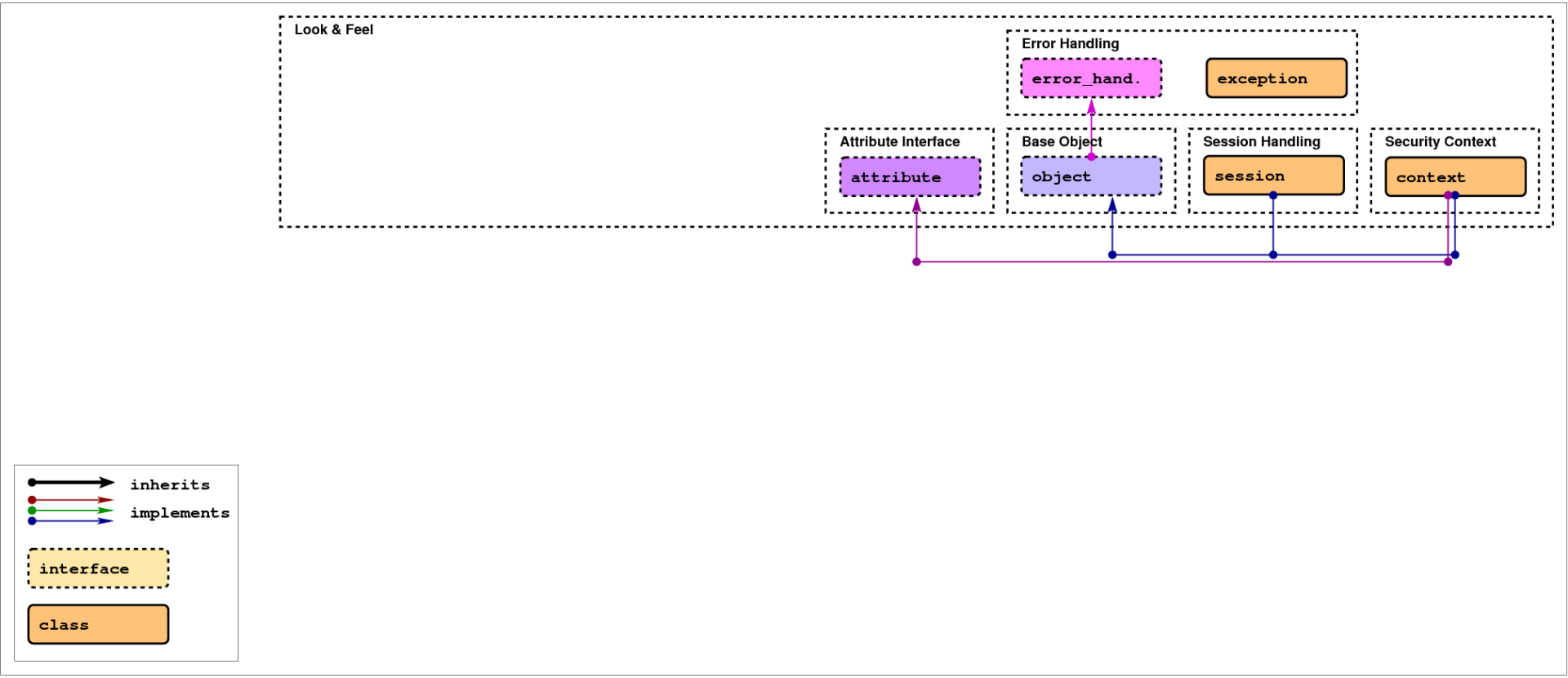
SAGA defines a hierarchy of exceptions
(and allows implementations to fill in specific details)

Session, Context, Permissions



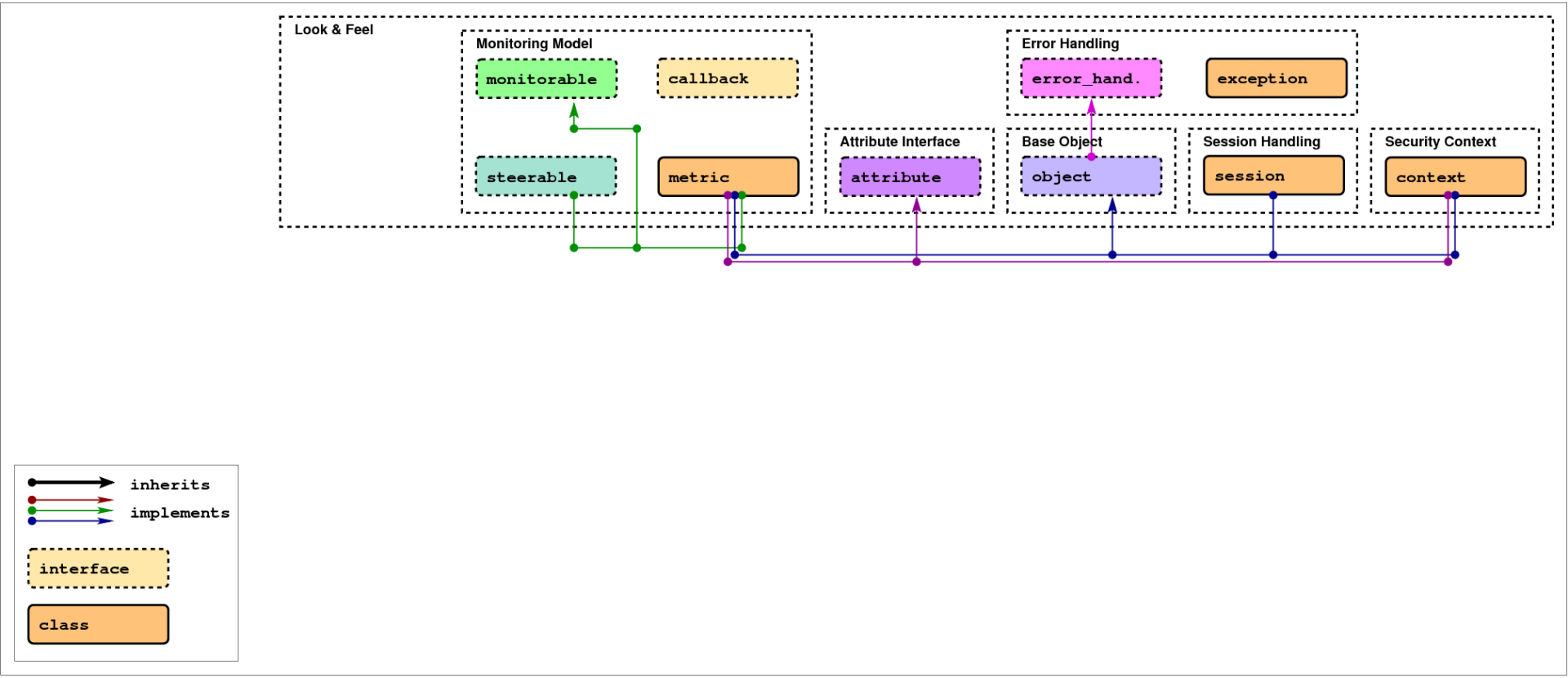
Context provides functionality of a session handle and isolates independent sets of SAGA objects. Only needed if you wish to handle multiple credentials. Otherwise *default context* is used.

Attributes



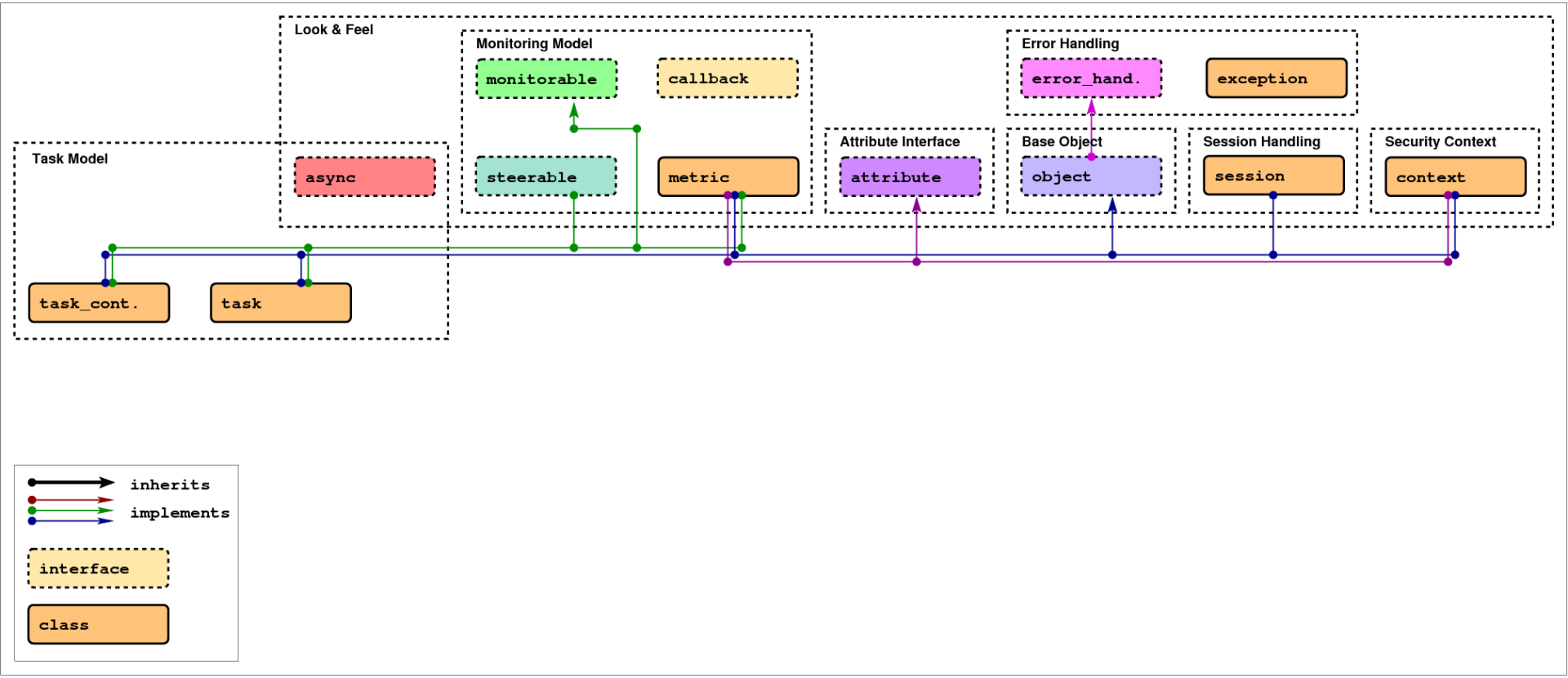
Where attributes need to be associated with objects, e.g. Job-submission. Key-value pairs, e.g. for resource descriptions attached to the object.

Application Monitoring



Metric defines application-level data structure(s) that can be monitored and modified (steered). Also, task model requires state monitoring.

Asynchronous Operations, Tasks

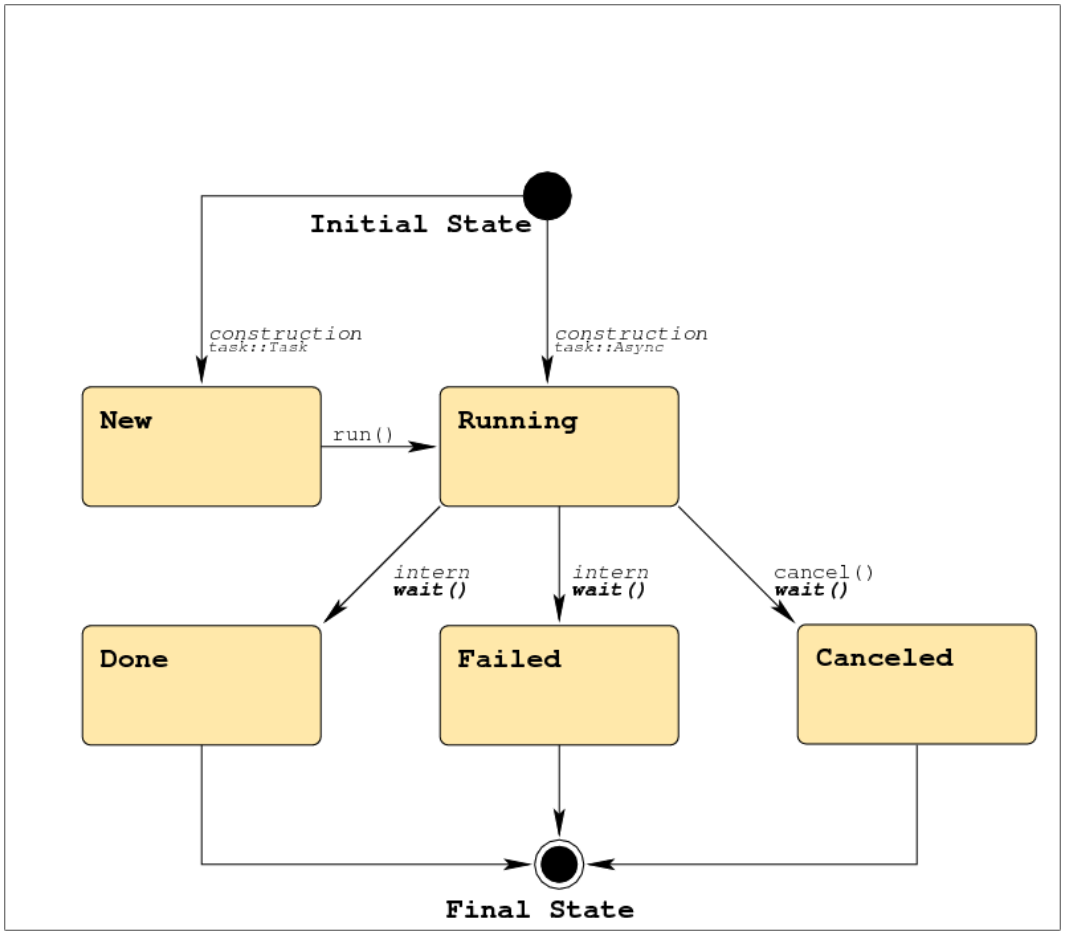


Most calls can be synchronous, asynchronous, or tasks (need explicit start.)

SAGA Task Model

- **All SAGA objects implement the task model**
- **Every method has three "flavours"**
  - synchronous version - the implementation
  - asynchronous version - synchronous version wrapped in a task (thread) and started
  - task version - synchronous version wrapped in a task but not started (task handle returned)

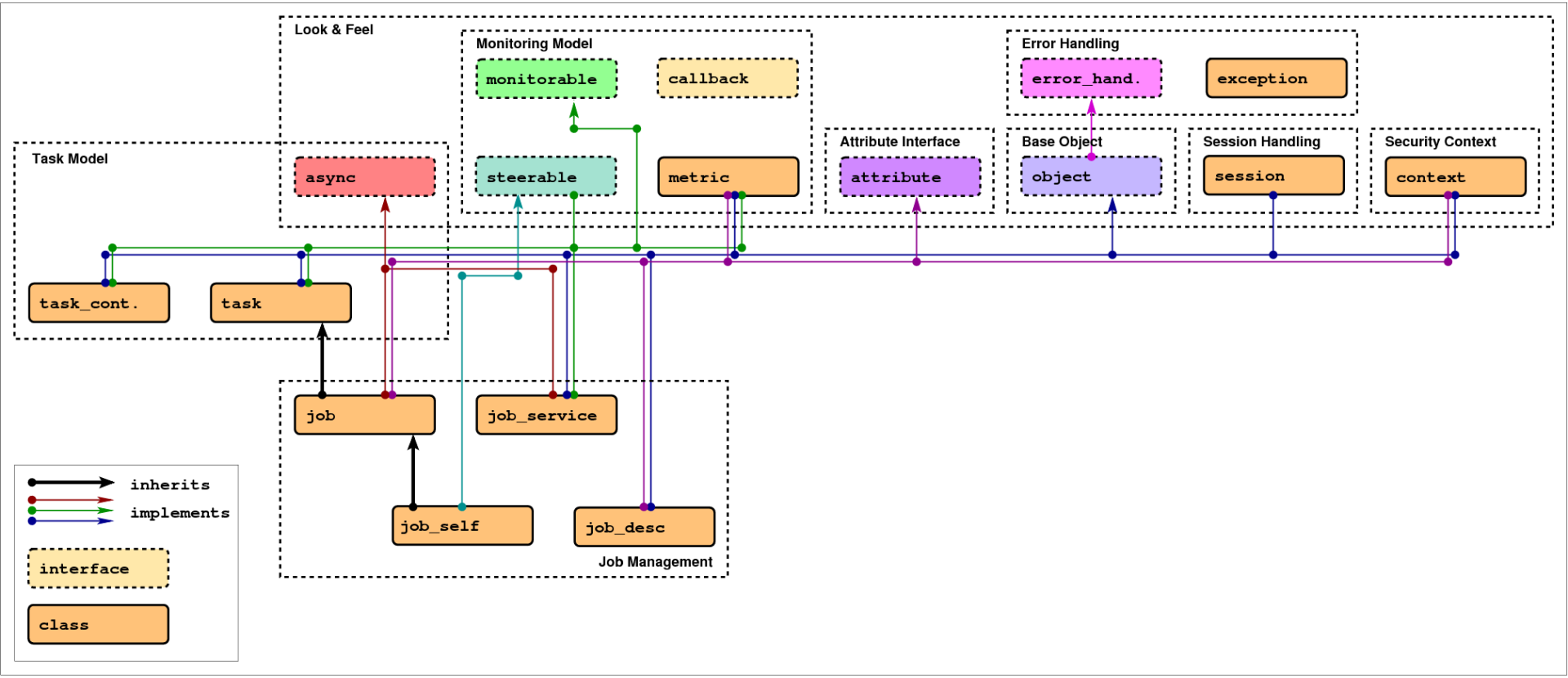
SAGA Task Model



SAGA Task Model

```
import org.ogf.saga.error.SagaException;
import org.ogf.saga.file.File;
import org.ogf.saga.file.FileFactory;
import org.ogf.saga.task.Task;
import org.ogf.saga.task.TaskMode;
import org.ogf.saga.url.URL;
import org.ogf.saga.url.URLFactory;

public class TaskModelExample {

  void foo() throws SagaException {

    URL src = URLFactory.createURL("any://host.net/data/src.dat");
    URL dst = URLFactory.createURL("any://host.net/data/dest1.dat");

    File f = FileFactory.createFile(src);

    // normal sync version of the copy method
    f.copy(dst);

    // the three task versions of the same method
    Task t1 = f.copy(TaskMode.SYNC, dst);  // in 'Done' or 'Failed' state
    Task t2 = f.copy(TaskMode.ASYNC, dst); // in 'Running' state
    Task t3 = f.copy(TaskMode.TASK, dst);  // in 'New' state

    t3.run();

    t2.waitFor();
    t3.waitFor();
  }
}
```

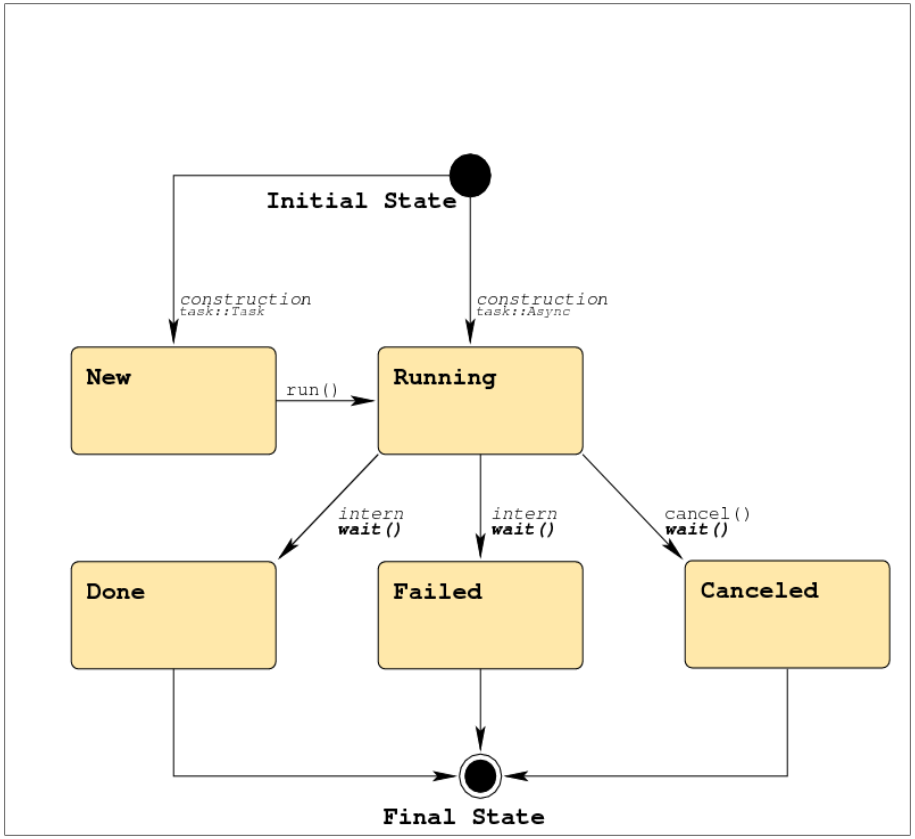
Jobs



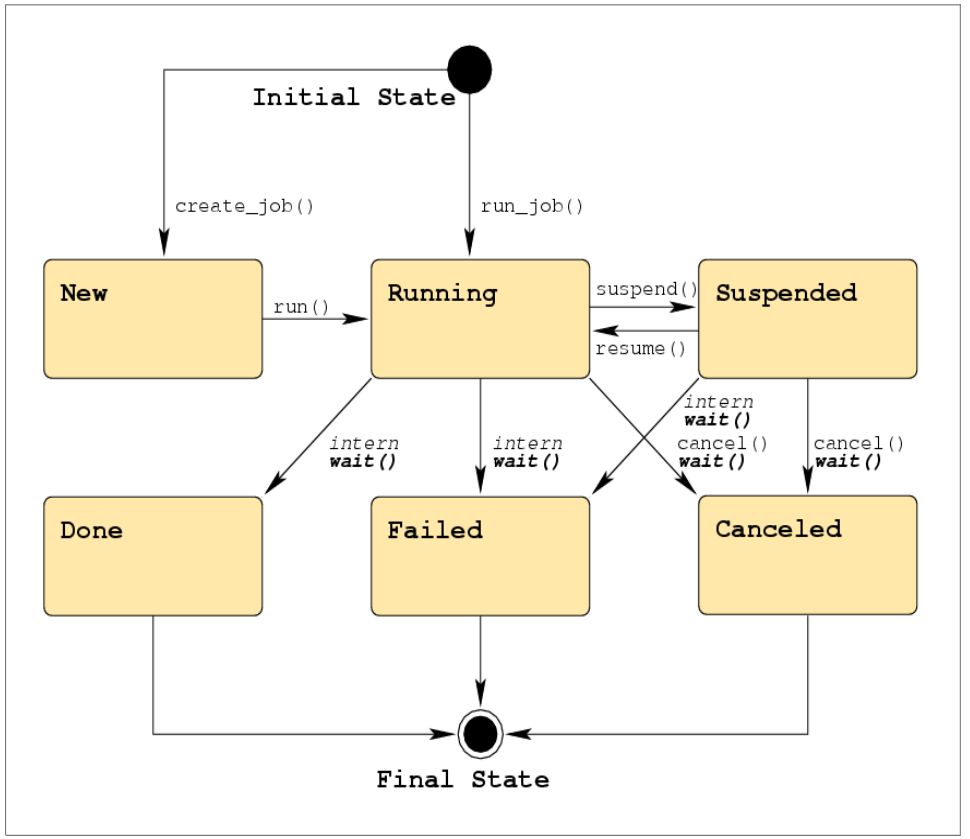
Jobs are submitted to run somewhere in the grid.

SAGA Task and Job States

Tasks:



Jobs:



Job Submission API

```
import org.ogf.saga.job.Job;
import org.ogf.saga.job.JobDescription;
import org.ogf.saga.job.JobFactory;
import org.ogf.saga.job.JobService;
import org.ogf.saga.task.State;
import org.ogf.saga.url.URL;
import org.ogf.saga.url.URLFactory;

public class JobSubmissionExample {

  void foo() throws SagaException { // submit a simple job and wait for completion

    JobDescription d = JobFactory.createJobDescription();
    d.setAttribute(JobDescription.EXECUTABLE, "job.sh");

    URL u = URLFactory.createURL("any://remote.host.net");
    JobService js = JobFactory.createJobService(u);

    Job job = js.createJob(d);
    job.run();

    while(job.getState().equals(State.RUNNING)) { // polling example
        String id = job.getAttribute(Job.JOBID);
        System.out.println("Job running with ID: " + id);
        Thread.sleep(1000);
    }
  }
}
```

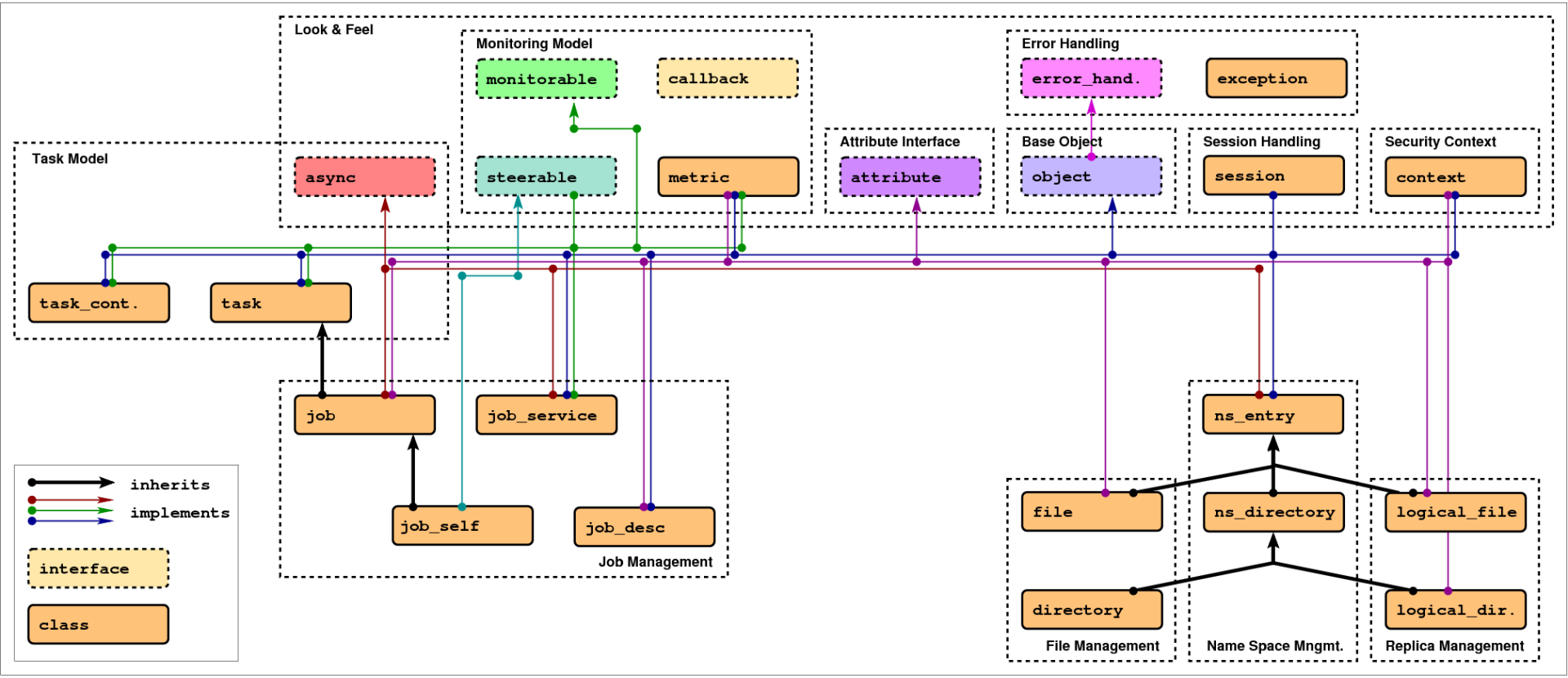
Job Submission API

`job_service` uses `job_description` to create a `job`

- **`job_description` attributes are based on JSDL [OGF, GFD.56]**
  - **JSDL files can be imported/exported separately**
- **State model is based on OGSA BES [OGF, GFD.108]**
- **`job_self` represents the SAGA application**

XtreemOS
Enabling Linux for the Grid

Files, Directories, Name Spaces



Both for physical and replicated ("*logical*") files

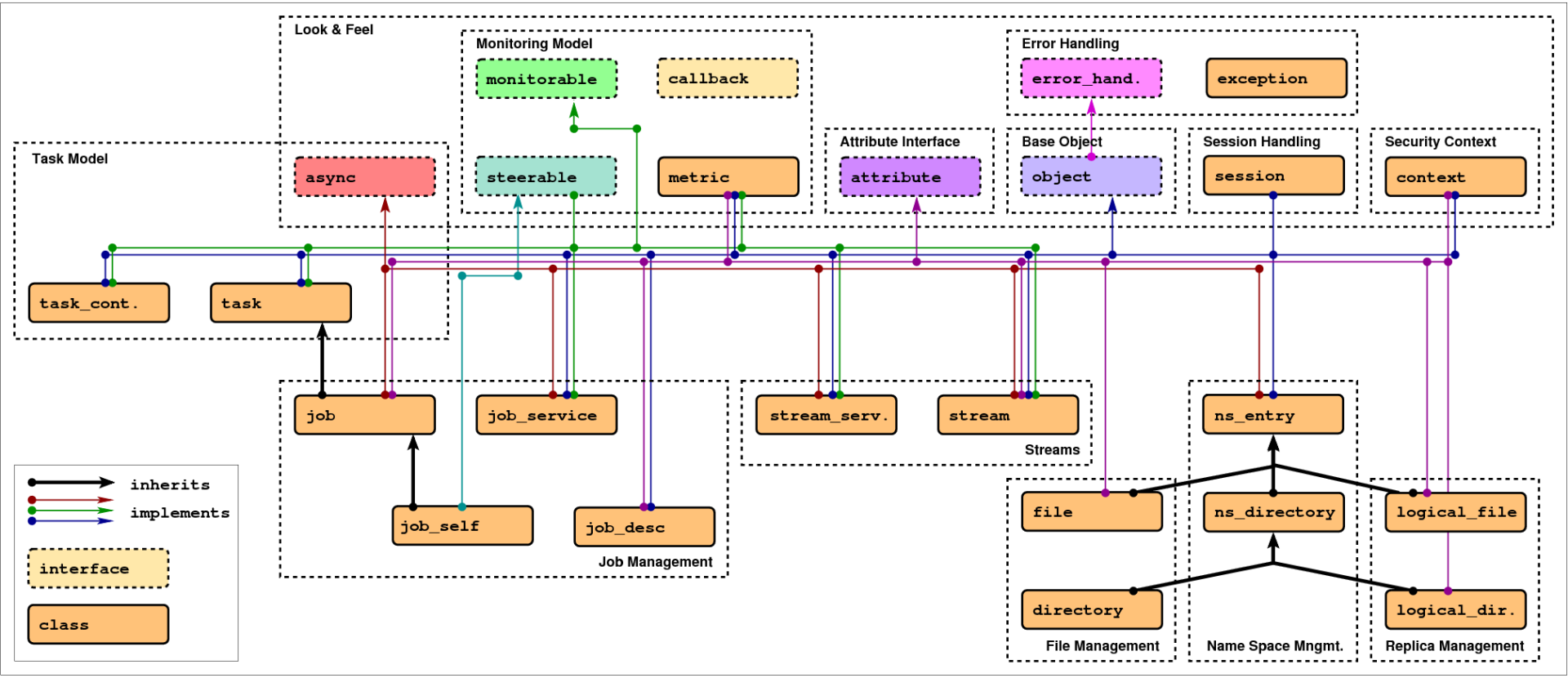
File API Example

```
import org.ogf.saga.buffer.Buffer;
import org.ogf.saga.error.SagaException;
import org.ogf.saga.file.FileFactory;
import org.ogf.saga.job.JobDescription;
import org.ogf.saga.job.JobService;
import org.ogf.saga.url.URL;
import org.ogf.saga.buffer.BufferFactory;
import org.ogf.saga.file.File;
import org.ogf.saga.job.Job;
import org.ogf.saga.job.JobFactory;
import org.ogf.saga.task.State;
import org.ogf.saga.url.URLFactory;

public class FileAPIExample {
    void foo() throws SagaException {
        // read the first 10 bytes of a file if file size > 10 bytes

        URL u = URLFactory.createURL("file://localhost/etc/passwd");
        File f = FileFactory.createFile(u);
        long size = f.getSize();

        if (size > 10) {
            Buffer buf = BufferFactory.createBuffer(10);
            int readBytes = 0;
            while (readBytes < 10) {
                readBytes += f.read(buf, readBytes, 10 - readBytes);
            }

            String s = new String(buf.getData());
            System.out.println(s);
        }
    }
}
```

FileReadExample.java

```
import org.ogf.saga.buffer.Buffer;
import org.ogf.saga.buffer.BufferFactory;
import org.ogf.saga.error.SagaException;
import org.ogf.saga.file.File;
import org.ogf.saga.file.FileFactory;
import org.ogf.saga.url.URL;
import org.ogf.saga.url.URLFactory;

public class FileReadExample {
    public static void main(String[] argv) {
        if (argv.length < 1) {
            System.out.println("usage: java FileRead <URL>");
        } else {
            try {
                Buffer buf = BufferFactory.createBuffer(64);
                URL u = URLFactory.createURL(argv[0]);
                File f = FileFactory.createFile(u);
                int readBytes = 0;
                do {
                    readBytes = f.read(buf);
                    String s = new String(buf.getData(), 0, readBytes);
                    System.out.print(s);
                } while (readBytes > 0);
            } catch (SagaException e) {
                System.err.println(e);
            }
        }
    }
}
```

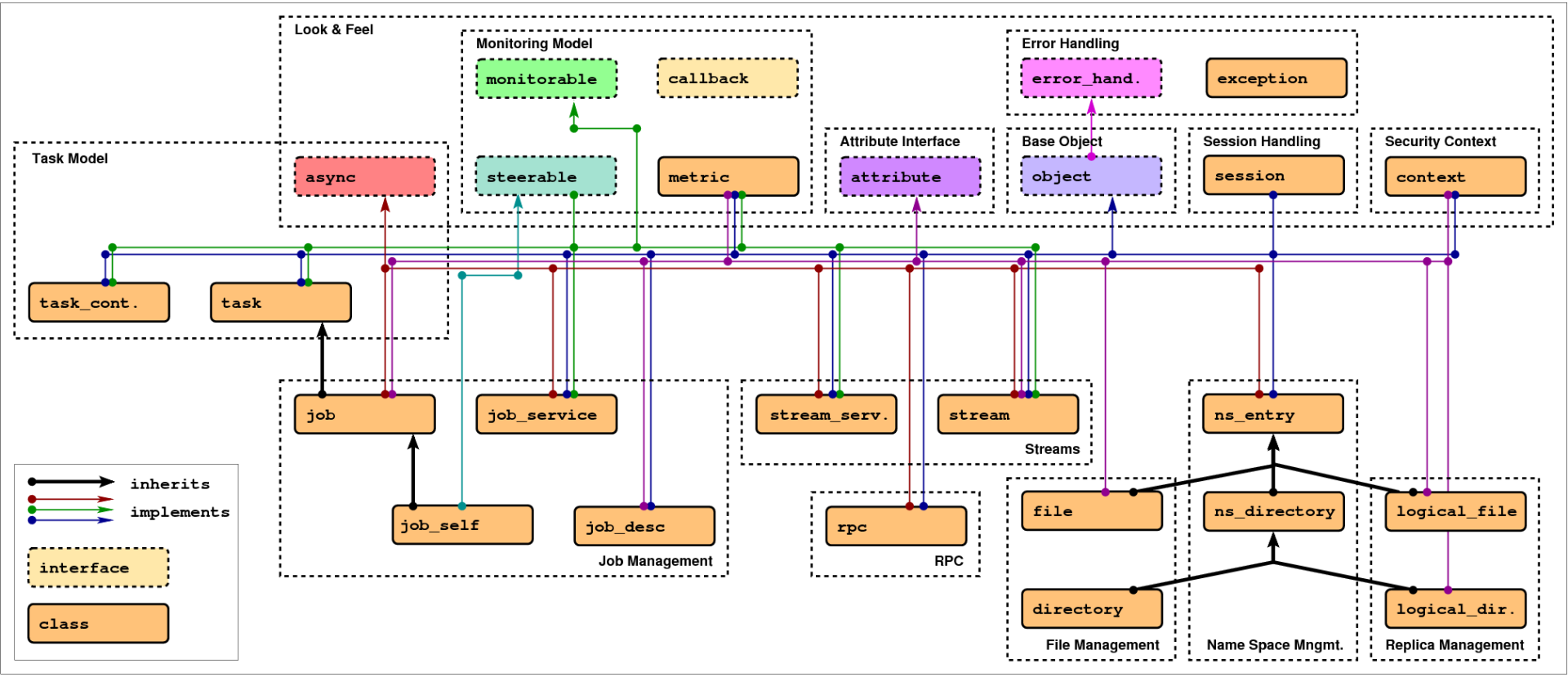
Streams



Simple, data streaming end points

XtreemOS
Enabling Linux for the Grid

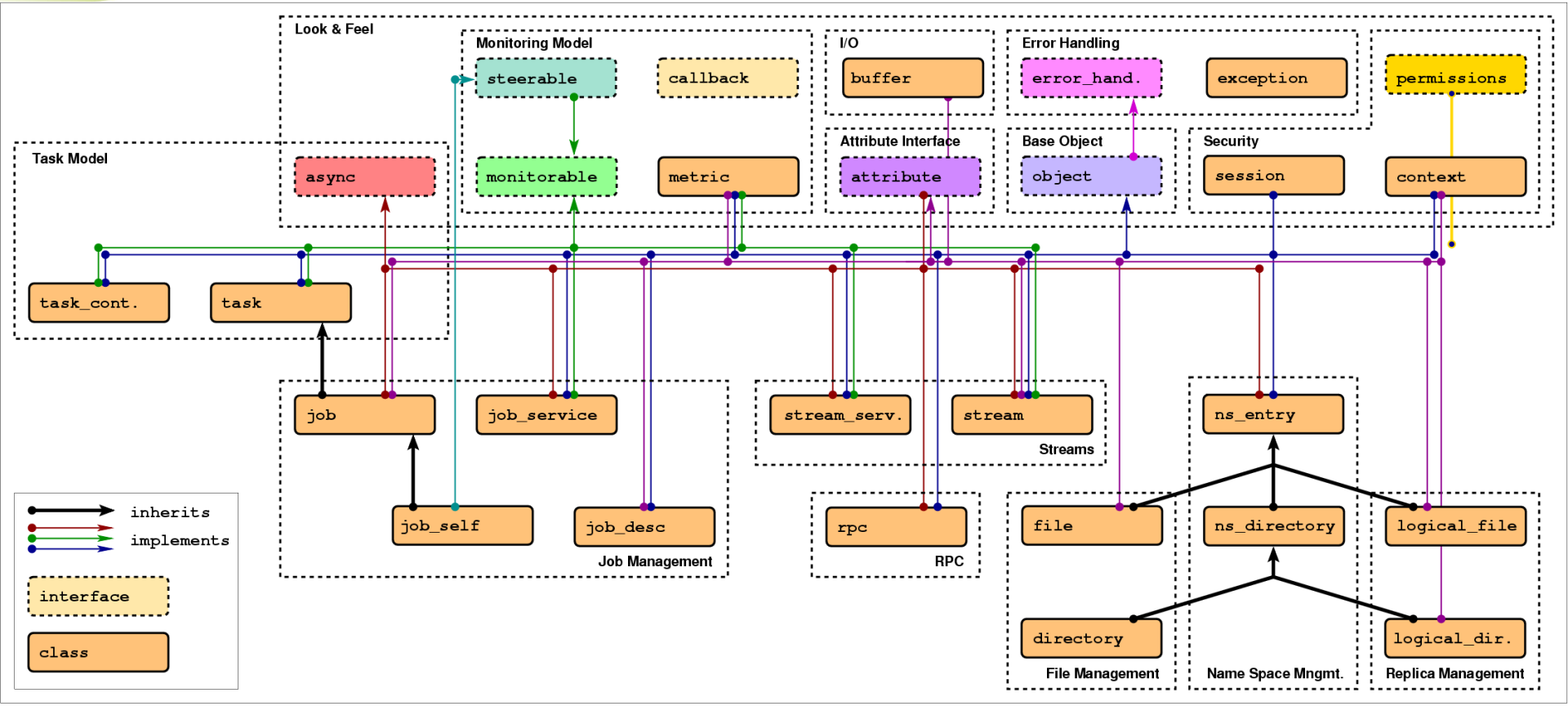
Remote Procedure Call



A rendering of GridRPC [OGF, GFD.052]

XtreemOS
Enabling Linux for the Grid

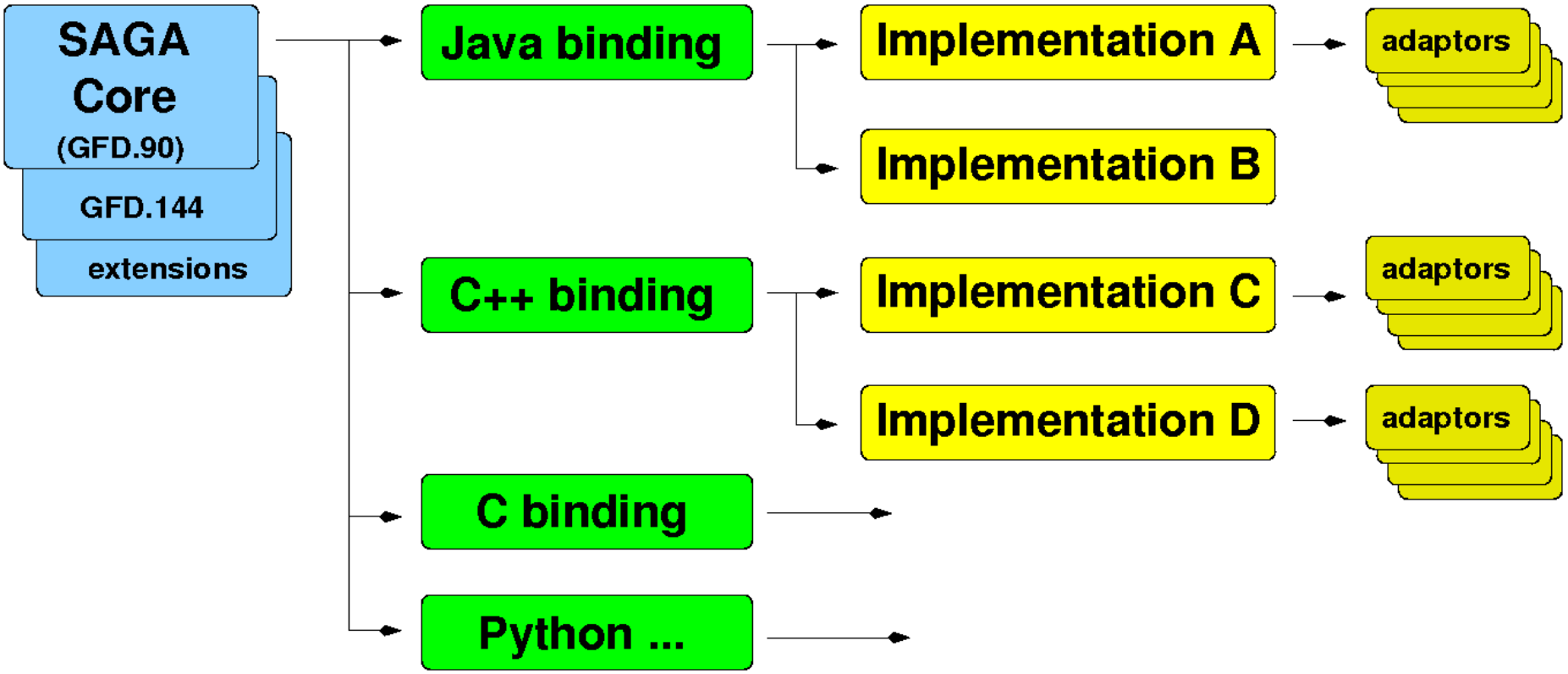
Permissions, I/O Buffers



Permissions for access rights

Buffers for I/O operations

The SAGA Landscape



SAGA Language Bindings

- **For C++, the binding currently is implicitly defined by the reference implementation**
- **For Java, a language binding has been defined**
  - used by the VU reference implementation
- **For Python, a language binding has been defined**
  - same story as with Java...

- **Language bindings currently are the "weak spot" in the standardization process (work in progress)**

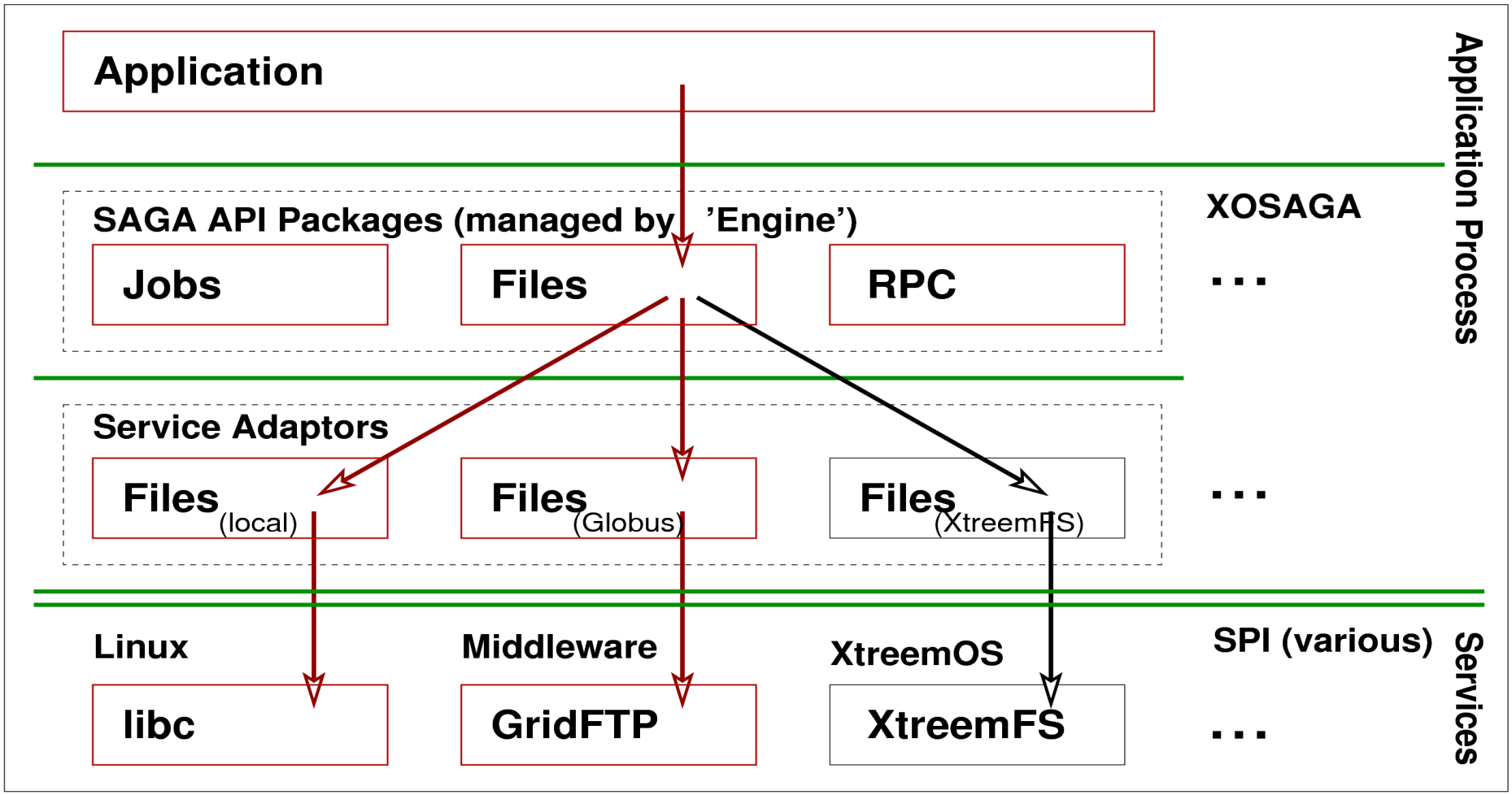
Outline

- **The Simple API for Grid Applications (SAGA)**
  - Motivation & Scope
  - SAGA as an OGF Standard
- **The SAGA Landscape**
  - Interfaces
  - Language Bindings
- **SAGA Implementations**
  - Engine with Adaptors
  - C++, Java, Python

Implementation Requirements

- **Non-trivial set of requirements:**
  - Allow heterogeneous middleware to co-exist
  - Cope with evolving grid environments; dynamic resources
  - Future SAGA API extensions
  - Portable, syntactically and semantically platform independent; permit latency hiding mechanisms
  - Ease of deployment, configuration, multiple-language support, documentation etc.
  - Provide synchronous, asynchronous & task versions

Typical(?) SAGA Implementation



Implementations

- **VU: Java**
  - Part of XtreemOS and the OMII-UK Project
  - Builds on JavaGAT
- **LSU: C++**
  - Developed (originally) with/at VU
- **VU/LSU: Python**
  - Wrappers on top of C++ and Java SAGA

Supported Middleware (Adaptors)

- **C++**
  - local, XtreemOS, Globus 3 and 4, OMII-UK GridSAM, GridFTP, Globus RLS
- **Java**
  - local, XtreemOS, Globus (up to GT4.2), gLite, ssh, OMII-UK GridSAM, XMLRPC
- **Python**
  - via C++ or Java

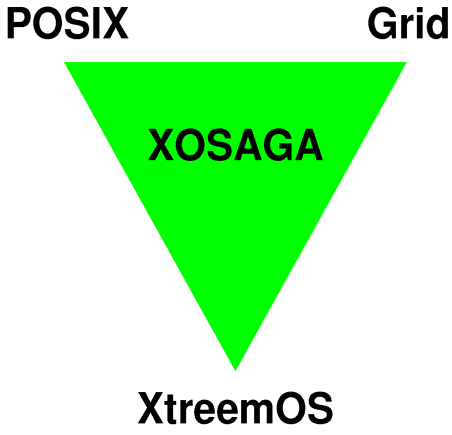
Finally, is SAGA Simple?

- **It depends: It is certainly not simple to implement!**
  - Grids are complex and the complexity needs to be addressed somewhere, by someone!
  - Pain using the middleware goes into the SAGA engine and adaptors.
- **But it is simple to use!!**
  - Functional Packages (specific calls), Look & Feel
  - Somewhat like MPI - most users only need a very small subset of calls

Application Programming Interfaces for XtreemOS

Design Challenges:

- POSIX look-and-feel (for Linux apps)
- grid look-and-feel (for grid apps)
- provide XtreemOS functionality



Approach:

- Start from OGF standardized *SAGA* API
  - POSIX look-and-feel
  - accepted grid API
- build XtreemOS-specific extensions (*XO*SAGA)

API Target Feature Coverage

- Features:
  - standard SAGA (as of OGF document GFD.90)
  - XOSAGA extensions (first set)
    - VO Management (XtreemOS credentials)
    - Application Execution Management (mostly reservation)
    - XtreemFS (URL schema)
  - XOSAGA extensions (second set)
    - Distributed Servers (hand-over sockets)
    - Data sharing (OSS, Scalaris publish/subscribe)
- Languages:
  - C++, Java, Python
  - User GUI (Xterior)

Side Track: XOSAGA for IaaS (EC2)

- The XOSAGA package for resource management is *very* close to an IaaS interface
  - starting a VM means implicitly creating a resource
- Ongoing work: transform XOSAGA RM package to become an official SAGA extension package
  - Ongoing work at OGF
  - Simplifying the package
    - Remove reservations (it +/- always is “now”)
    - Streamline attributes with JSDL and the GLUE schema

Conclusions

- **Today's and tomorrow's computing platforms are heterogeneous, dynamic, and error-prone**
- **Applications have to address scalability, elasticity, heterogeneity, faults, ... into account**
- **Programming models and interfaces must abstract from underlying middleware and service platforms (use SAGA underneath)**
- **SAGA enables**
  - programming grid/cloud-aware applications
  - providing higher-level programming models
    - e.g. map/reduce, divide-and-conquer,...

XtreemOS
Enabling Linux for the Grid

Acknowledgements

- **The SAGA Team, at and with OGF:**
  - Andre Merzky, Shantenu Jha, Pascal Kleijer, Malcolm Illingworth, Hartmut Kaiser, Ole Weidner, Stephan Hirmer, Ceriel Jacobs, Kees Verstoep
- **The European Commission via grants to**
  - The **CoreGRID** network of excellence
  - The **XtreemOS** project
    - Mathijs den Burger, Ana Oprescu, Emilian Miron, Manuel Franceschini, Tudor Zaharia, Pravin Shinde, Paul van Zoolingen
- **The Dutch VL-e project, OMII-UK, CCT LSU**