

The SAGA Standards Landscape

Thilo Kielmann
Vrije Universiteit Amsterdam

SAGA: In a Nutshell



- SAGA: The **S**imple **A**PI for **G**rid **A**pplications
- Applications must be able to utilize Infrastructure
 - “e-Infrastructure that will enable novel and different research” linked with e-Science
- But (how) can applications be developed to utilize complex, dynamic – qualitative and quantitative – infrastructure?

SAGA: A Quick Tour



- **Simple, integrated, stable, uniform** and **high-level** interface
 - Simple: 80:20 restricted scope
 - leave out corner cases for preserving simplicity
 - Integrated: Similar semantics & style across commonly used distributed functional requirements
 - Stable: Standardization
 - Uniform: Same interface for different distributed systems
- SAGA: Provides the high-level abstraction that application developers need that will work across different distributed systems
 - Shields details of lower level middle-ware and system issues
 - Enables the details of distribution to be left out

Copy a File “before SAGA”: 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);
  globus_gass_copy_attr_set_io (&io_attr);
  globus_gass_copy_handleattr_set_ftp_attr (&gass_copy_handleattr,
  globus_gass_copy_handleattr (&ftp_handleattr);
  globus_gass_copy_handle_init (&gass_copy_handle,
  globus_gass_copy_handleattr (&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,
  globus_gass_copy_attr (&source_ftp_attr);
}
else {
  globus_gass_transfer_requestattr_init (&source_gass_attr,
  globus_gass_copy_attr_set_gass (&source_gass_copy_attr,
  globus_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,
  globus_io_handle (&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);
}
```

SAGA Example: Copy a File

High-level, uniform



```
#include <string>
#include <saga/saga.hpp>

void copy_file(std::string source_url, std::string target_url)
{
    try {
        saga::file f(source_url);
        f.copy(target_url);
    }
    catch (saga::exception const &e) {
        std::cerr << e.what() << std::endl;
    }
}
```

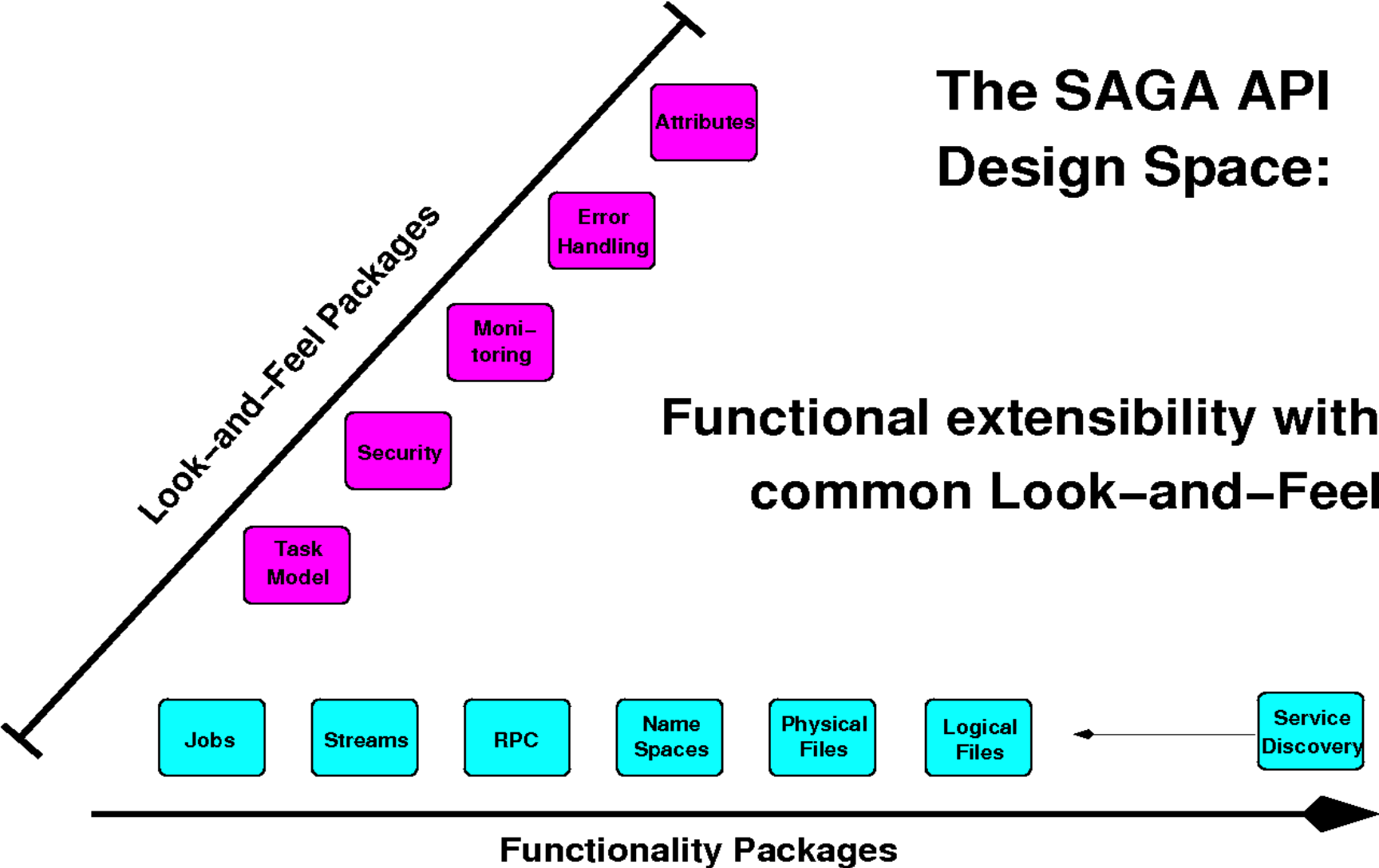
SAGA Interface

Job Submission API

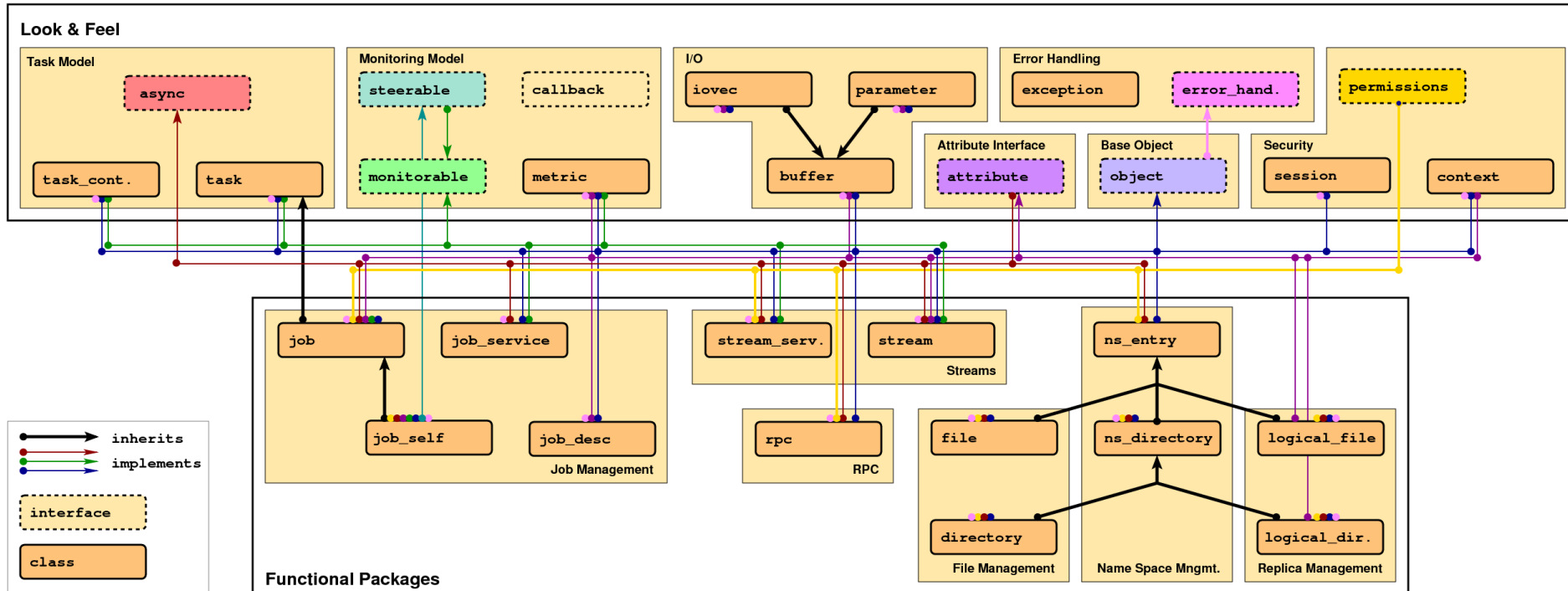


```
01: // Submitting a simple job and wait for completion
02: //
03: saga::job_description jobdef;
04: jobdef.set_attribute ("Executable", "job.sh");
05:
06: saga::job_service js;
07: saga::job job = js.create_job ("remote.host.net", jobdef);
08:
09: job.run();
10:
11: while( job.get_state() == saga::job::Running )
12: {
13:     std::cout << "Job running with ID: "
14:               << job.get_attribute("JobID") << std::endl;
15:     sleep(1);
16: }
```

SAGA API Design Overview



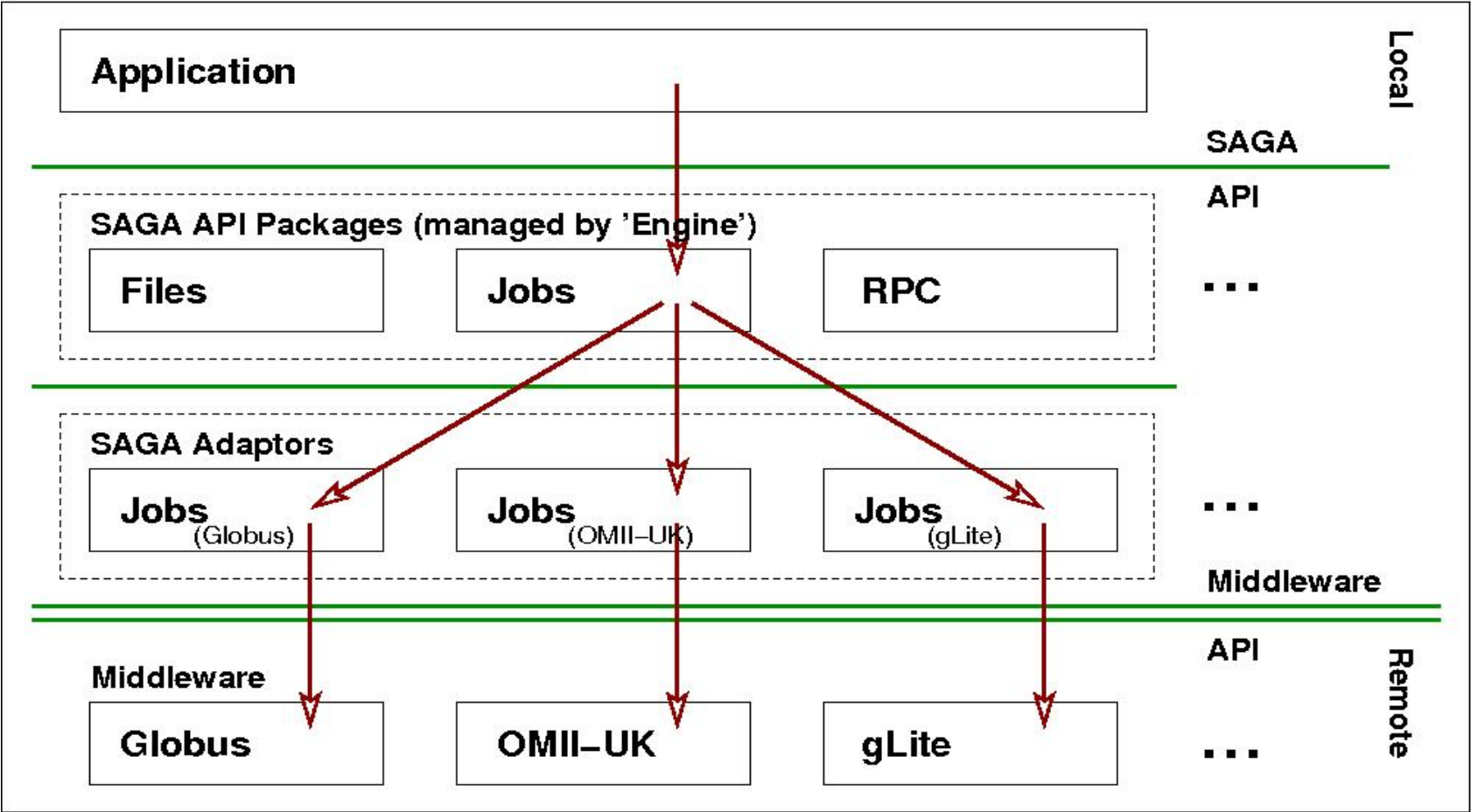
SAGA: Class Diagram



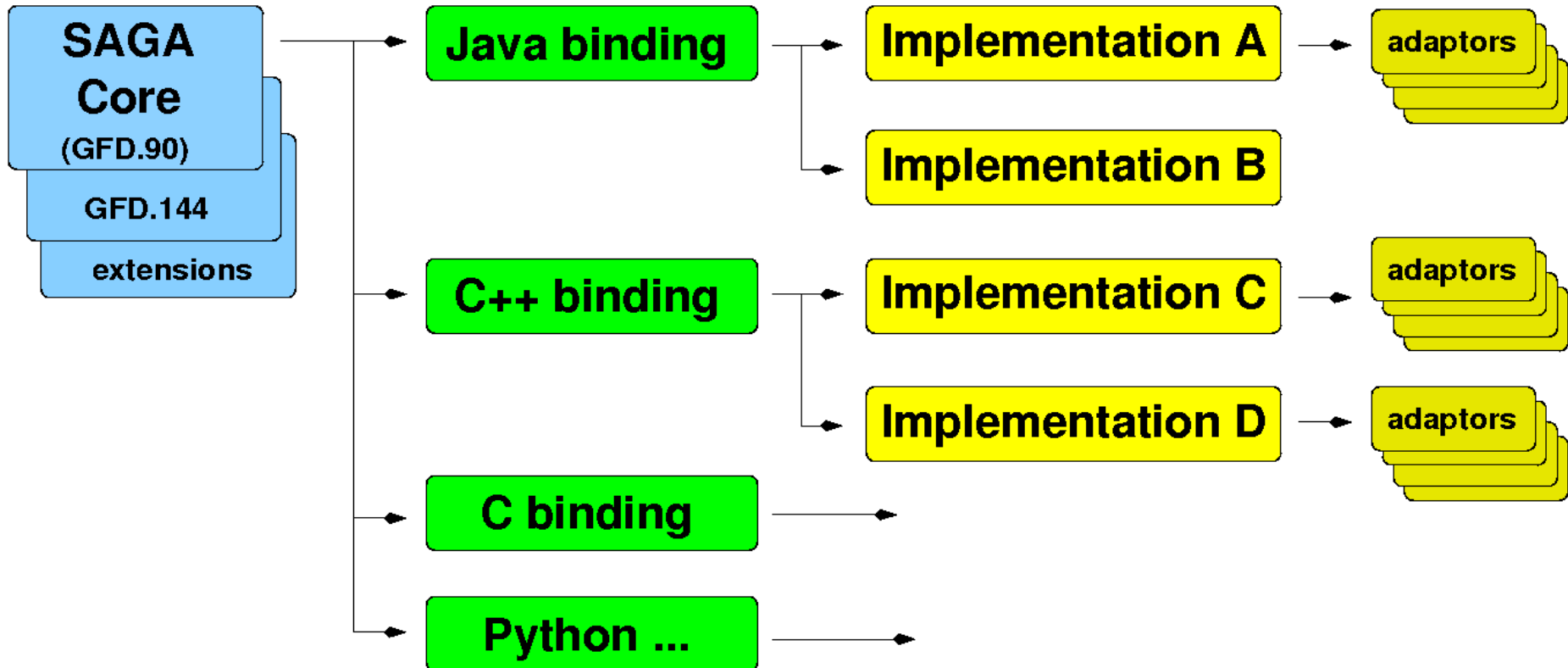
In the works: CPR, Information Services, Messaging....
 Service Discovery has just been added

Typical(?) SAGA Implementation

Role of Adaptors (middleware binding)



SAGA: The Landscape



What's in it for you?

- Reference implementations (open source) available from ***<http://saga.cct.lsu.edu>***
 - C++
 - Java
 - Python coming soon
- Showcasing SAGA applications:
 - Talk by Shantenu Jha