Workflow application development with Web-Service on EGEE

Tristan Glatard

EGEE and SEE-GRID Summer School on Grid Application Support
June 27th

Grid computing

Medical imaging

Agenda

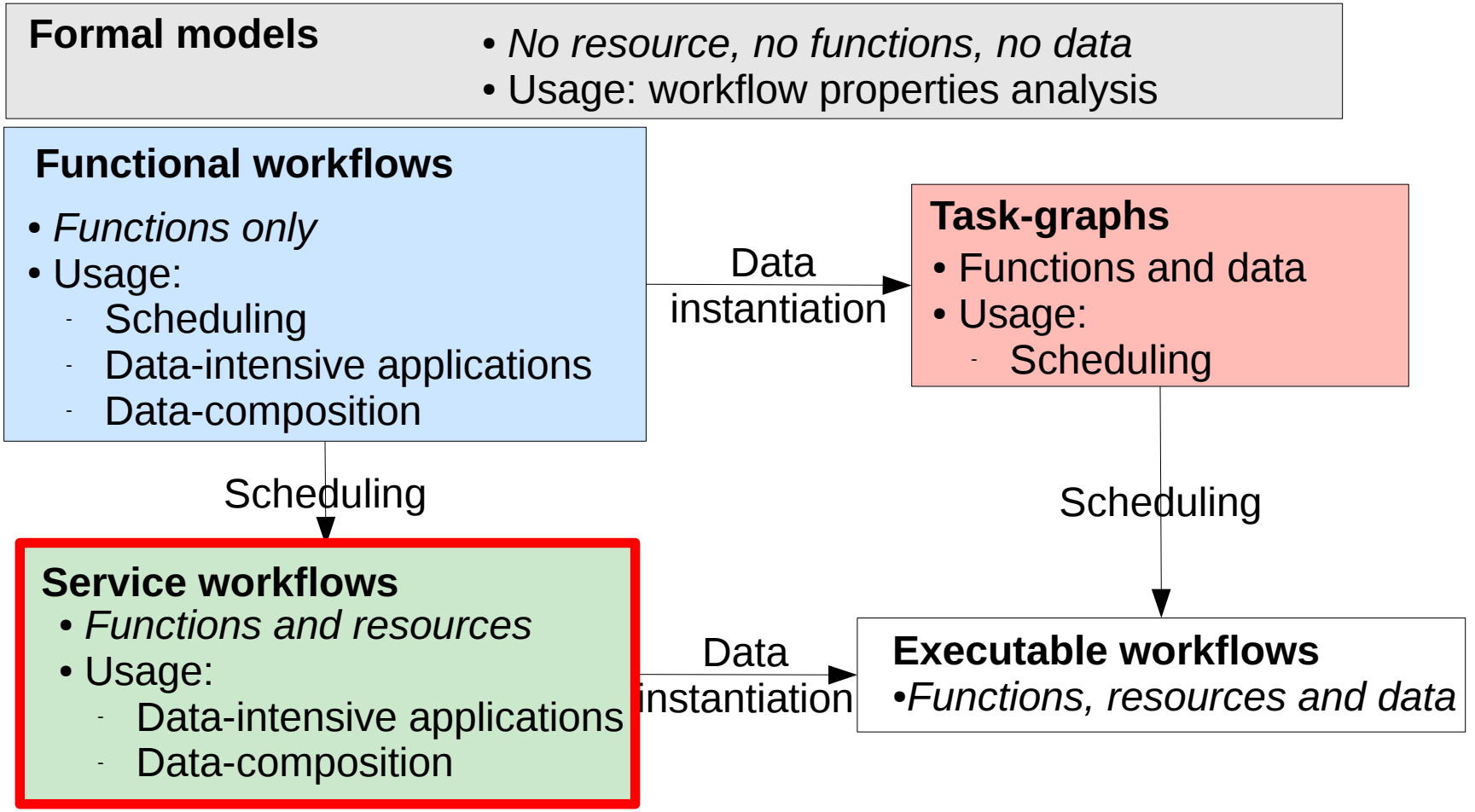
- 14h00 → 14h15: brief introduction
- 14h15 → 15h45: workflow prototyping:
  - Design with Taverna (65 min)
  - Efficient execution with MOTEUR (30 min)
- 15h45 → 16h00: coffee break
- 16h00 → 16h40: Web-services deployment
- 16h40 → 17h20: interface with EGEE
- 17h20: end of the practical

Why workflows ?

- Definition of a workflow (Mayer *et al)*:
  - *Organization of a structured application in an abstract fashion, such that the implementation of the atomic tasks being organized is independent from the organization itself*
- Component programming model:
  - Fosters code reusability
  - Platform independence
- Simple graphical language
- Natural parallelization of an application
  - Coarse grain parallelism

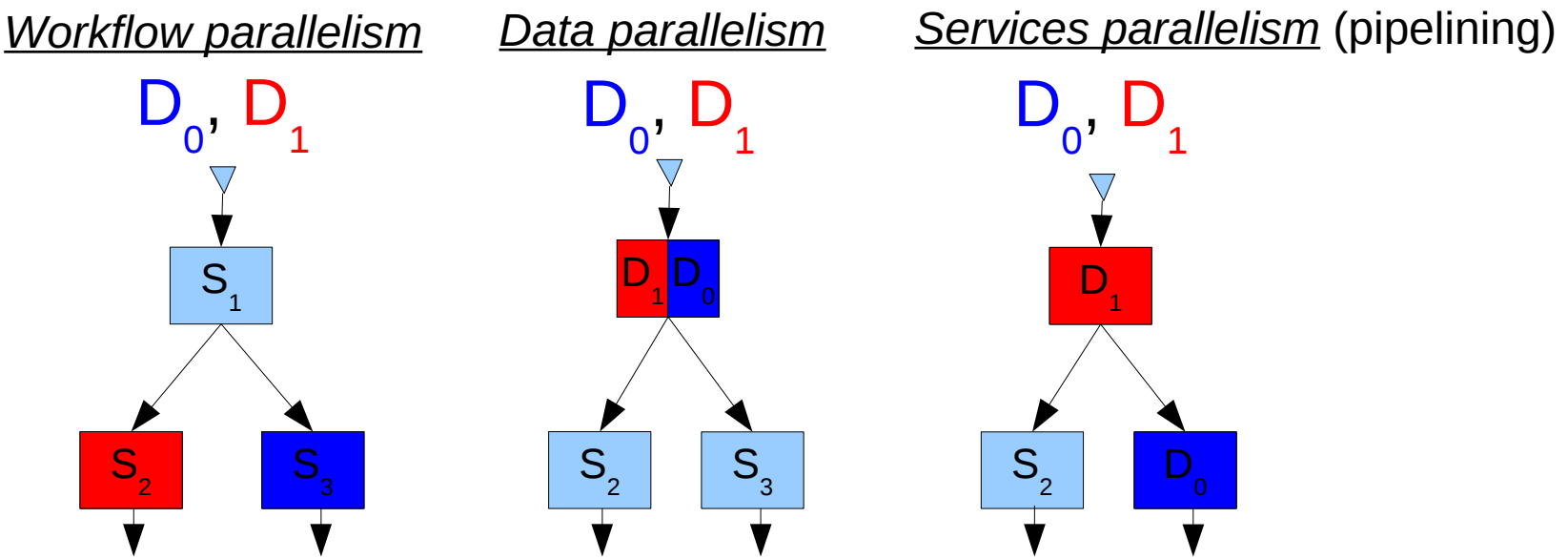
Workflow classification

- Classification w.r.t the presence/absence of functions, data and resources in the description:

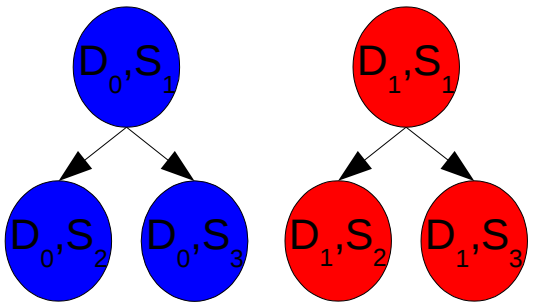


Parallelism in service workflows

- 3 kinds of parallelism can be exploited:



- Data and service parallelism are intrinsic in task graphs:



Let's go !

- 14h15 → 15h45: workflow prototyping:
  - Design with Taverna (65 min)
  - Efficient execution with MOTEUR (30 min)

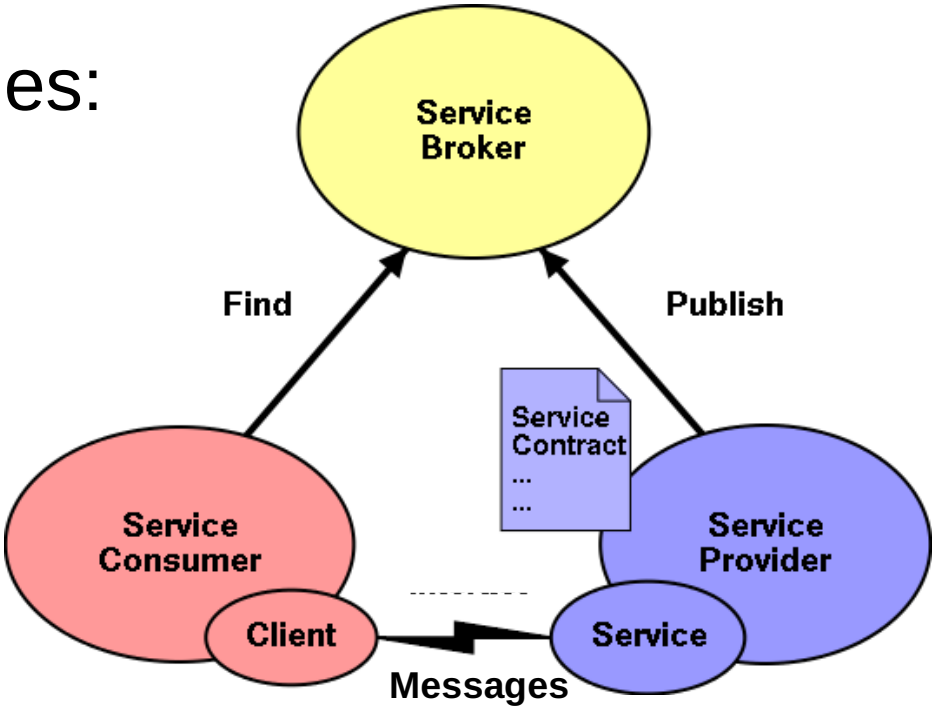
- Practical instructions available from http://colors.unice.fr/tutorialBudapest/tutorial.pdf

Agenda

- 14h00 → 14h15: brief introduction
- 14h15 → 15h45: workflow prototyping:
  - Design with Taverna (65 min)
  - Efficient execution with MOTEUR (30 min)
- 15h45 → 16h00: coffee break
- 16h00 → 16h40: Web-services deployment
- 16h40 → 17h20: interface with EGEE
- 17h20: end of the practical

SOA in a nutshell

- Basic principles:



- A service is an exposed piece of functionality with 3 properties:

  (1) The interface contract to the service is **platform-independent**

  (2) The service can be **dynamically located** and invoked

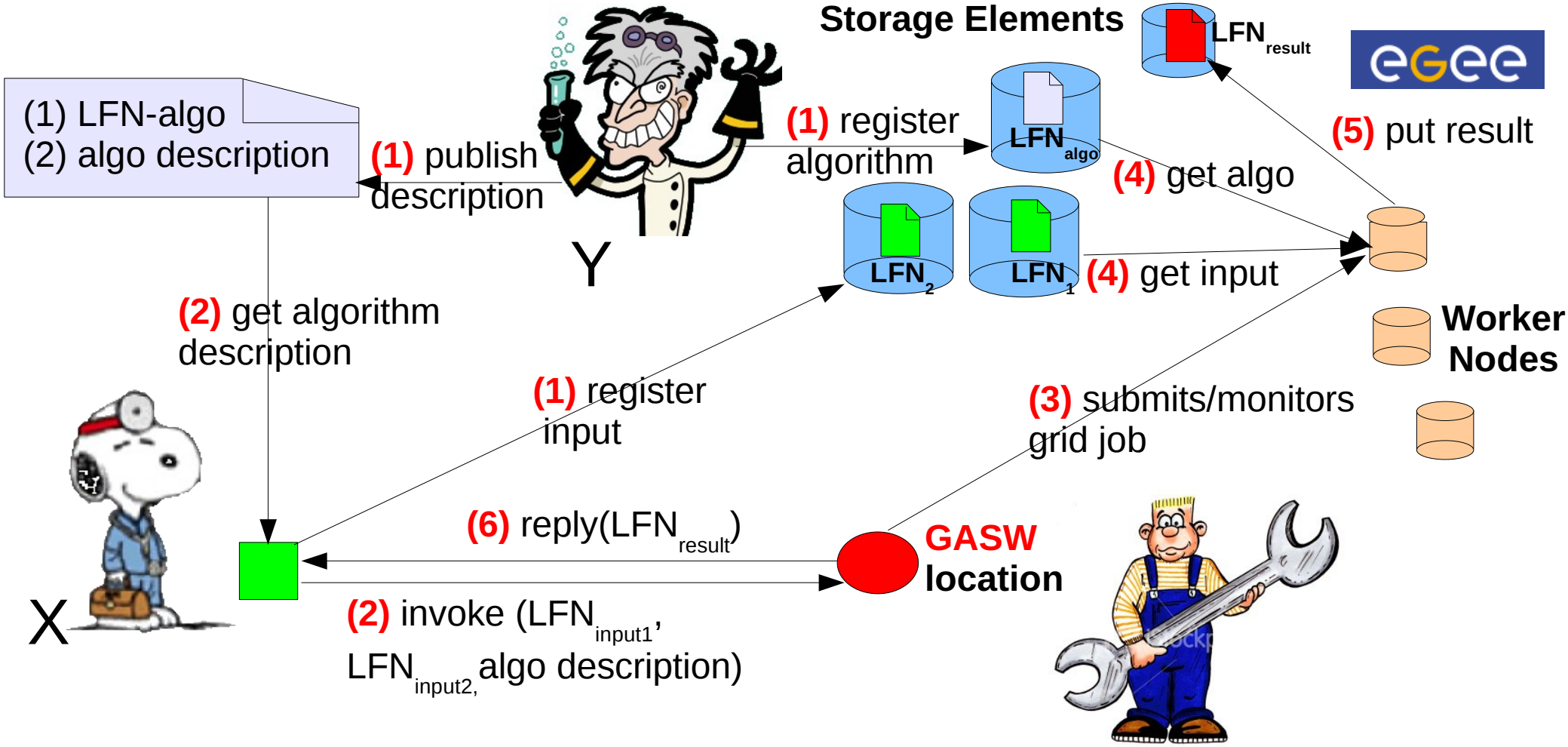
  (3) A service does not call another service (**loosely coupling**)

Web-Services

- Originally from IBM and Microsoft
- Standardized by the W3C:
  - Contract format: **W**eb **S**ervices **D**escription **L**anguage
  - Messages format: **S**imple **O**bject **A**ccess **P**rotocol
  - Discovery format:
    - **U**niversal **D**escription **D**iscovery & **I**ntegration
- Based on XML (platform/language independence)

Generic Application Service Wrapper

- The grid job handling can be **decoupled from Y**



Let's go !

- 16h00 → 16h40: Web-services deployment
- 16h40 → 17h20: interface with EGEE (GASW)

- Practical instructions available from http://colors.unice.fr/tutorialBudapest/tutorial.pdf

The end !

- Thank you for your participation
- More information from:
  - http://www.i3s.unice.fr/~glatard
  - glatard@i3s.unice.fr

WSDL

[http://w3.org/TR/wsdl]

- Describes the interface of the Web-Service
- Current version: 1.1 (15th march 2001)
- 7 basic XML tags:
  - **Types**: complex types made from basic types
  - **Message**: set of parts (Input/Output parameters)
  - **Operation**: set of messages (≃method)
  - **Port type**: set of operations (≃class)

  What ?
  - **Binding**: protocol used to invoke the service (*e.g*: SOAP/HTTP)
  - **Port**: internet address and port
  - **Service**: set of ports

  How ?
- Example on an image registration service:
  - Input: 2 images ; output: 1 transformation

WSDL structure: <*types*>

```
<types>
    <complexType name="image">
          <sequence>
           <element name="file-name" type="xsd:string" />
           <element name="modality" type="xsd:string"/>
          </sequence>
  </complexType>
</types>
```

WSDL structure: <*message*>

```
<message name="registrationRequest">
    <part name="reference-image" type="ns:image"/>
    <part name="floating-image" type="ns:image"/>
</message>

<message name="transformation">
    ...
</message>
```

WSDL structure: <*portType*> and <*operation*>

```
<portType name="registrationPortType">
    <operation name="register">
        <input message="registrationRequest"/>
        <output message="transformation"/>
    </operation>
</portType>
```

WSDL structure: <*binding*>

```
<binding name="registrationB"
   type="registrationPortType">
 <SOAP:binding style="rpc" transport="http"/>
</binding>
```

WSDL structure: <*port*> and <*service*>

```
<service name="registration_service">
   <port name="registration1" binding="registrationB">
    <address location="http://rigid.registration.com:1234"/>
   </port>
  <port name="registration2" binding="registrationB">
    <address location="http://bestAlgos.fr/registration"/>
   </port>
</service>
```

GASW algorithm descriptor

- Executable access method:
  - URL
  - Grid file
- Input/Output
  - Command-line options
  - Access methods (for files)
- Sandbox files access methods

```
<description>
  <executable name="CrestLines.pl">
    <access type="URL">
      <path value="http://somewhere.eu/"/>
    </access>
    <value value="CrestLines.pl"/>

    <input name="image" option="-im1">
      <access type="LFN" />
    </input>
    <input name="scale" option="-s"/>
    <output name="crest_lines" option="-c2">
      <access type="LFN" />
    </output>

    <sandbox name="convert8bits">
      <access type="URL">
        <path value="http://elsewhere.dk/"/>
      </access>
      <value value="Convert8bits.pl"/>
    </sandbox>
  </executable>

</description>
```

GASW algorithm descriptor

- Executable access method:
  - URL
  - Grid file
- Input/Output
  - Command-line options
  - Access methods (for files)
- Sandbox files access methods

```
<description>
  <executable name="CrestLines.pl">
    <access type="URL">
      <path value="http://somewhere.eu/"/>
    </access>
    <value value="CrestLines.pl"/>

    <input name="image" option="-im1">
      <access type="LFN" />
    </input>
    <input name="scale" option="-s"/>
    <output name="crest_lines" option="-c2">
      <access type="LFN" />
    </output>

    <sandbox name="convert8bits">
      <access type="URL">
        <path value="http://elsewhere.dk/"/>
      </access>
      <value value="Convert8bits.pl"/>
    </sandbox>
  </executable>

</description>
```

GASW algorithm descriptor

- Executable access method:
  - URL
  - Grid file
- Input/Output
  - Command-line options
  - Access methods (for files)
- Sandbox files access methods

```
<description>
  <executable name="CrestLines.pl">
    <access type="URL">
      <path value="http://somewhere.eu/"/>
    </access>
    <value value="CrestLines.pl"/>

    <input name="image" option="-im1">
      <access type="LFN" />
    </input>
    <input name="scale" option="-s"/>
    <output name="crest_lines" option="-c2">
      <access type="LFN" />
    </output>

    <sandbox name="convert8bits">
      <access type="URL">
        <path value="http://elsewhere.dk/"/>
      </access>
      <value value="Convert8bits.pl"/>
    </sandbox>
  </executable>

</description>
```