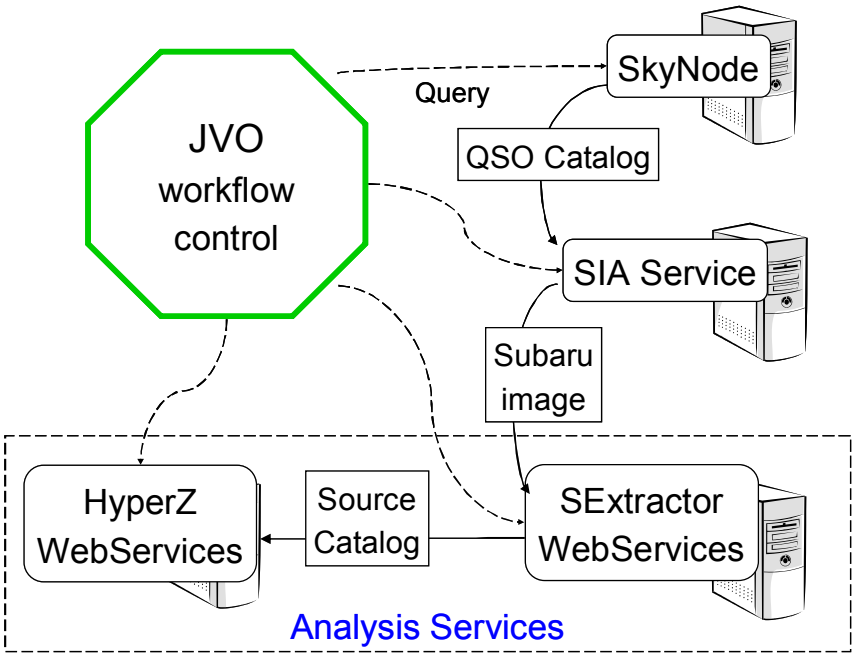
JVO technical overview
- Workflow -

Masahiro Tanaka

(NAOJ)

Purpose

- Federate Services
  - Catalog Search
  - Analysis Services
- Automatic execution
- Flexible construction of workflow by users.

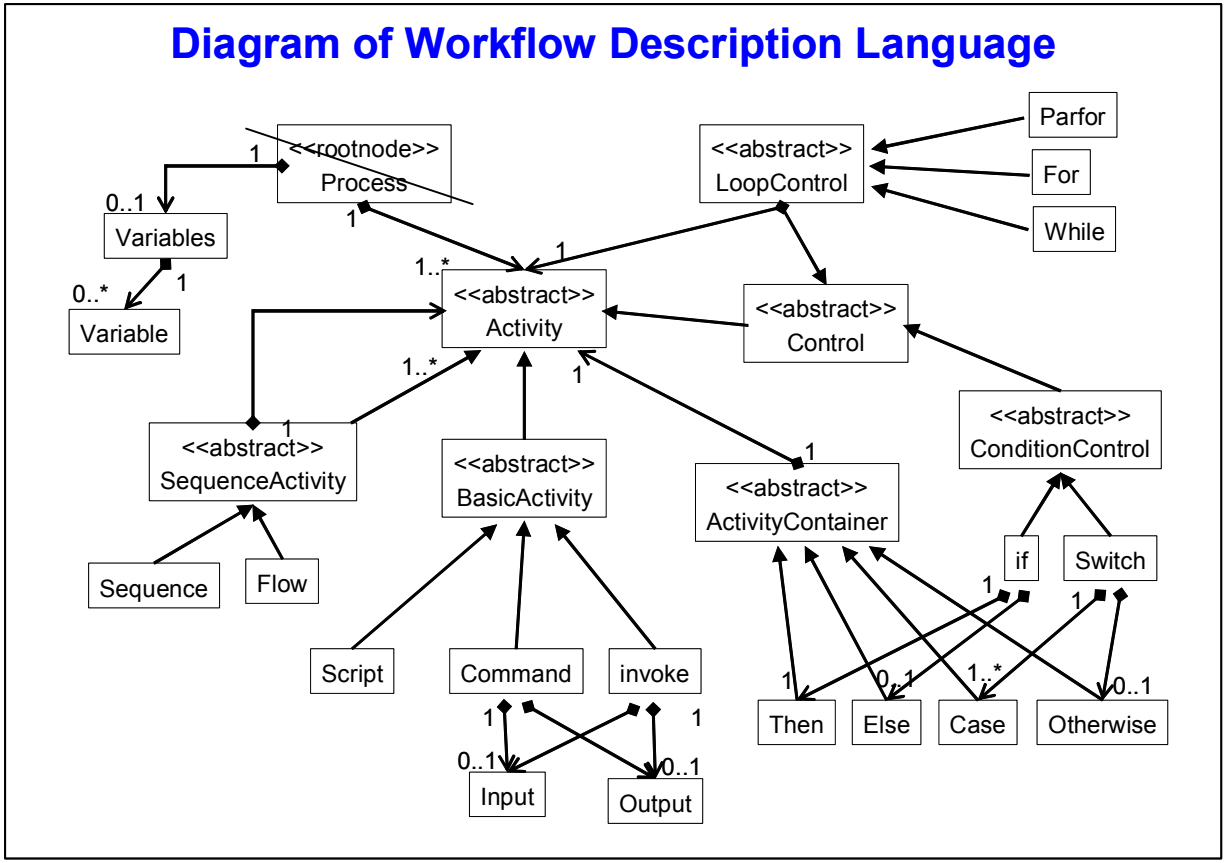


Design of Workflow

- Described in XML. (and more human-friendly language?)
- Available services can be found from VOResource.
- Usage descriptions of a service are also found from VOResource.
- Service Provider can register their services easily.

Workflow Schema

- Design of **BPEL4WS** is reflected in JVO workflow.



Variable definition

- variables
- variable
- value
- set

```
<variables>
      <variable name="imageSize" type="double">
          <value>0.02</value>
      </variable>
      <variable name="imageTable" type="String">
          <value>ivo://jvo/subaru/spcam:spcam_mos_view</value>
      </variable>
      <variable name="votableForImage" type="ArrayOfVOTABLE"/>
      <variable name="listOfURL" type="ArrayOfString"/>
      <variable name="catalog" type="VOTABLE"/>
</variables>

<set name="imageSize" literal="0.1">
```

Variable types (TBD)

- Simple types
  - int
  - double
  - string
  - VOTable
  - ADQL
  - DataHandler
- Array types
  - ArrayOf*
    - (* is a simple type)
- Collections
  - Map
  - List

invoke

- invokes **Web services**

  (SkyNode, SIAP, Analysis services etc.)

- Specify **identifier**, then resolve services from VOResource

Example:

```
<invoke identifier="ivo://jvo/tools/sextractor"
        url="http://jvoc.dc.nao.ac.jp:8080/services/SExtractor?wsdl"
        operation="performForURL">
    <input>
        <varRef>listOfURL[_count]</varRef>
    </input>
    <output>
        <varRef>catalog</varRef>
    </output>
</invoke>
```

command

- executes internal functions
- 2 types:
  1. classMethod
  2. builtin

command(1): class method type

type="classMethod"

- invoke Java class method

```
Example:
<command xsi:type="classMethod"
          class_name="QSOStudy"
          method="prepare">
      <input>
         <varRef>result_file1</varRef>
      </input>
      <output>
         <varRef>listOfURL</varRef>
      </output>
</command>
```

command(2): built-in type

type="builtin"

- pre-registered, frequently-used operation
  - executeQuery
  - copyToVOStore
  - …

Example:

```
<command xsi:type="builtin" name="executeQuery">
    <input>
        <varRef>jvoql</varRef>
    </input>
    <output>
       <varRef>votableForImage</varRef>
    </output>
 </command>
```

SequenceActivity

- holds List of activities
  - sequence : sequential execution
  - flow : parallel execution

- example:

```
<sequence>
  <command type="builtin" name="…">
  <command type="classMethod" class_name="… >
  <for …>
  …
</sequence>
```

Loop & Condition

- LoopControl:
  - for
  - parfor
  - (while)

    Nested loop is not currently supported.
- ConditonalControl
  - (if)
  - (switch)

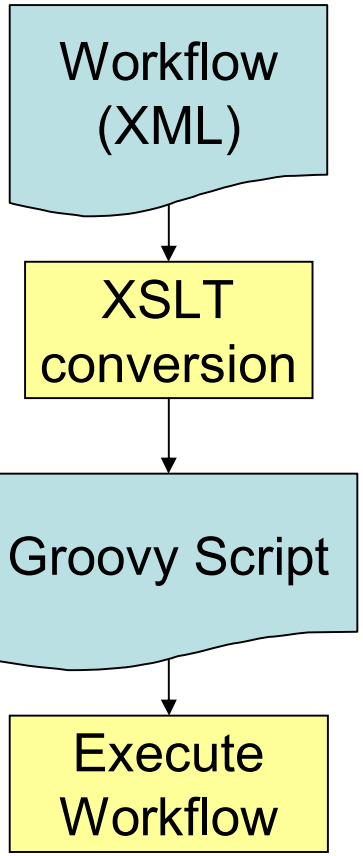
Script support

- **Groovy** script is supported.

  Example:

```
<script>
   <input>
       <literal>
           pattern = “[^abc]”;
           text = “d”;
           println text==~pattern
       </literal>
   </input>
</script>
```

Current implementation



Web Service Interface (1)

- Conventional use of Web Services:
  - generate Java Stub code from WSDL.
- But it needs re-compilation.
  - JVO workflow must be restarted when new services are added.

Web Service Interface (2)

- **DII (Dynamic Invocation Interface)** of JAX-RPC is used for workflow executer.
  - Benefit
    - Register new Web Services dynamically.
    - Avoid re-compilation of JVO System.
  - Restriction
    - Only primitive and pre-registered classes can be used for arguments.

Workflow Builder for users

- Simple workflow editor is implemented.
- More user-friendly workflow builder will be developed.
- Graphical workflow builder
  - use of **JFLOW** by Andre Schaaff et al. @CDS?

Development timeline

- Jun 2005 – design started
- Dec 2005 – implementation started
- Jan 2006 – first prototype appeared
- current – under development

Summary

- Workflow Description Language is defined.
- Workflow is converted to Groovy script.
- DII is required for dynamical registration of Web Services.