Astronomical Data Analysis Software and Systems XV **P.107**
ASP Conference Series, Vol. XXX, 2005
C. Gabriel, C. Arviset, D. Ponz and E. Solano, eds.

Web Service Interface and Workflow Mechanism for JVO

Masahiro Tanaka, Yuji Shirasaki, Satoshi Kawanomoto, Satoshi Honda, Masatoshi Ohishi, Yoshihiko Mizumoto

National Astronomical Observatory of Japan

Naoki Yasuda

Institute for Cosmic Ray Research, The University of Tokyo

Yasuhide Ishihara, Jumpei Tsutsumi

Fujitsu Ltd.

Hiroyuki Nakamoto, Yuusuke Kobayashi, Michito Sakamoto

SEC Ltd.

Abstract.
Federation of astronomical data analysis services is one of main issues of JVO development in 2005. We employed the Web Services as an interface of analysis services, and designed the interface for various astronomical analysis tools that require efficient data transfer. We also designed a workflow mechanism for JVO on a basis of BPEL4WS, etc. This mechanism enables users to construct their data query and analysis sequences by combining multiple VO services.

1. Introduction

Japanese Virtual Observatory (JVO; Ohishi et al.) aims at a federated system of distributed servers which provide catalog, image and spectrum searches and analysis services. As a science use case of JVO, Shirasaki et al. describe a study of QSO environment. This use case is a flow of multiple Web Services; image extraction, catalog query, source extraction, calculation of photometric redshift and clustering analysis. Previous JVO prototypes had "Scheduler" to generate workflows from JVOQL (Tanaka et al. 2005). With such a mechanism, however, users cannot flexibly construct their own workflows including analysis services. In order for users to build workflows flexibly, we are designing a workflow language and developing a system to execute workflows written in the workflow language. With this mechanism, users are able to build workflows combining data query and analysis services available in the VO framework. In this paper, we describe the current design of the JVO workflow language and discuss the interface of Web Services used for JVO workflow system.

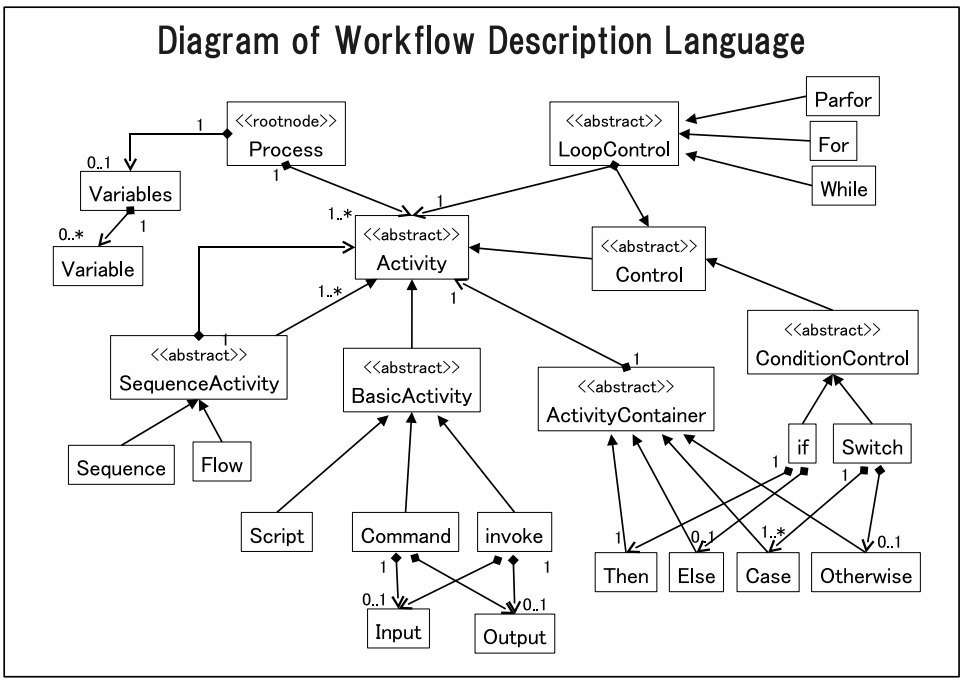

Figure 1. XML Schema diagram of JVO Workflow Language

2. Workflow Description Language

We are developing JVO workflow language based on the design of a standard workflow description language, Business Process Execution Language for Web Services (BPEL4WS). JVO workflow language is defined in XML. The XML schema diagram of JVO workflow language is shown in Fig. 1. This workflow language has the following types of elements:

- Variable definition (variable)
- Loop (for, while)
- Condition (if, switch)
- Sequential execution (sequence)
- Parallel execution (flow, parfor)
- Invoke external services (invoke)
- Invoke built-in Java classes (command)

JVO workflow language is equipped with a subset of the BPEL4WS functionality required for a basic sequence of astronomical tasks. In addition, this language has the capability to execute built-in Java class methods. This capability is useful for efficient processing using Java classes. JVO workflow execution system is under development and its first version is planed to be finished at the end of 2005.

3. Web Service Interface

In the course of workflow development, we considered the following technical issues of Web Service interface.

3.1. DII (Dynamic Invocation Interface)

Web Services are usually accessed by creating Java proxy classes from WSDL. This method, however, requires compilation of Java proxy code when Web Services are used for the first time. In order to register new Web Services to JVO system dynamically, we adopt Dynamic Invocation Interface (DII) mechanism of JAX-RPC for workflow executor. Although the execution time using DII is slightly longer than that using static proxy, it is negligible in total time of Web Service calls. Another disadvantage of DII is that only primitive and pre-registered classes can be used for arguments. We consider this is not a serious problem if major data types used in VO are registered in advance.

3.2. Document Metadata for Web Service

When users build a workflow which invokes Web Services, they have to know the meaning of argument and return values of the services. Although WSDL includes interface information used by computer systems such as data types, it is not intended to include meaning and usage information required for general users. Therefore we developed a mechanism to distribute document metadata of Web Services.

In our design, document metadata are embedded in the `portType` section of VOResource metadata. Thus the document is seachable by JVO Registry. Since the `portType` section comes from WSDL, embedding document metadata in WSDL is a simple way to distribute them. This method, however, is not practical because in most cases WSDL is automatically generated under the frameworks of standard tools such as Apache Axis. For this reason we adopt separate registration of WSDL and document metadata as follows; (1) Service provider registers the URL of WSDL to JVO portal. (2) JVO portal extracts information from WSDL and creates a template of VOResource. (3) Service provider edits the template and registers it to JVO Registry. We plan to develop a GUI registration tool to automate this process.

Acknowledgments. This work was supported by the JSPS Core-to-Core Program and Grant-in-aid "Information Science" (15017289 and 16016292) carried out by the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

Ohishi M. et al. this volume, [O7.9]

Shirasaki Y. et al. this volume, [P.105]

Tanaka M. et al. 2005, will be given at a later time, [P1.1.24]