

# Distributed Scientific Workflow Management for Data-Intensive Applications

S. Shumilov, Y. Leng, M. El-Gayyar, A.B. Cremers  
Computer Science Department III, University of Bonn, Germany  
shumilov, leng, elgayyar, abc@cs.uni-bonn.de

## Abstract

*Grids and service-oriented technologies are emerging as dominant approaches for distributed systems. Rising complexity of grid applications places new requirements and increases needs to improve the design and reusability of grid workflow systems. In particular, scientific data-intensive workflows require new approaches for integration and coordination of distributed resources. Traditional centralized approaches for workflow execution can be quite inefficient for such workflows. In order to clarify these issues, the paper surveys existing workflow management systems evaluating them practically on some use cases for management of natural resources from the multidisciplinary research project GLOWA Volta<sup>1</sup>. Subsequently, the most important obstacles are identified and a new approach facilitating semantic oriented composition, reuse and distributed execution of workflows is proposed.*

## 1. Introduction

Much scientific and industrial collaboration is increasingly employing grid and service-oriented technologies to manage their enterprises. The reason for this can be explained by the fact that grid systems are becoming more multidisciplinary and dependent upon specialized heterogeneous applications and databases. Particular jobs in these systems are specified with the help of workflows - collections of tasks that are processed on distributed resources in a well-defined order to accomplish a specific goal.

Several use cases from the GLOWA Volta project are considered in this study as examples of such workflows. The project focuses on development of decision support tools for sustainable management of natural resources in the Volta Basin in West Africa [26]. Management of natural resources is a complex task raising questions touching many different

disciplines (biology, hydrology, etc.), which cannot be solved within a single application. Therefore, a multidisciplinary team of professionals is required. Often team members are distributed geographically and use diverse highly specialized applications exchanging large data volumes. Working together on decision-making problems, team members need to invoke these applications individually or as parts of workflows. Such workflows are called *scientific*. They differ from *business* workflows in having long lasting execution tasks with heavy data flows and utilizing heterogeneous and dynamic resources from multiple organizations whereas business workflows involve short, transaction processing tasks with small amounts of data.

For example, the use case "Best source of irrigation water" [23] integrates two simulation systems. The main system is an economic optimization model coded in GAMS [4] which seeks to maximize the economic value of available water resources through choice of a crop, size of irrigated cropping, choice of irrigation water source and irrigation schedule. To be able to deal with complex hydrological systems the model is coupled with a physical hydrology model, which was made in WaSiM-ETH [10]. The model-generated data provides the boundary conditions for the economic model in GAMS, which output is examined to determine whether the boundary conditions are still valid, e.g. if groundwater levels have not fallen below acceptable bounds. If conditions have been violated, a new water-use-schedule is generated and the GAMS model is re-executed. Therefore, a decision making here typically requires at least two runs of both systems. The spatial data to be transmitted per iteration is quite heavy and when the number of iterations increases, it can be a real performance issue.

The best way to integrate these applications and compose workflows is to use grid technologies [6]. In order to find a system suitable for our project requirements we have started with some surveys [30]. However, trying to employ the selected systems in the practice we have found that some of the published claims are hardly justified by the real implementations. In order to clarify these issues, the paper makes a

---

<sup>1</sup> GLOWA Volta is financed by the German Federal Ministry of Education and Research (BMBF) as part of the GLOWA research initiative: Global Change in Hydrological Cycle.

practical survey of existing grid workflow systems evaluating them on some project's use cases. The survey identifies the most important obstacles and proposes a new approach facilitating semantic oriented compositions, reuse and distributed execution of workflows. The rest of the paper is organized as follows: Section 2 states the grid-based workflow requirements. In section 3, some selected workflow environments will be evaluated according to the stated requirements. A proposal for a workflow system architecture fitting our requirements is presented in section 4. The final section 5 summarizes the paper.

## 2. Workflow Requirements Analysis

In this section, we state a set of the most important requirements for workflow management systems gained through the analysis of the project's use cases exhibiting many typical features of scientific workflows. Gathered requirements in this study could be classified as *build time requirements* concerned with definition and modeling of workflows or *run time requirements* dealing with the overall management and execution of workflows.

### 2.1. Workflow Design and Definition

This section analyzes requirements covering design and definition of workflows. The specification of a workflow can be understood from four perspectives [25]: an *operational perspective* dealing with the elementary actions executed by tasks, a *control-flow perspective* covering activities controlling the flow of execution, a *resource perspective* providing a view from the sight of available human and hardware resources and a *data perspective* responsible for data flows. We extend this categorization by an *annotation perspective* providing semantic descriptions of workflows and resources.

From the *annotation perspective*, it is clear that many of the challenges for building scientific workflow specifications arise from their integrative nature and heterogeneity of resources used. Many scientific projects like GLOWA Volta deals with heterogeneous applications that have to be integrated. This generates a strong demand for instruments helping to semantically describe, discover and match these applications. Therefore, the workflow description language should support semantic annotations of services and interfaces as well as other characteristics of services/components. Since parts of existing workflows can be treated also as reusable components, semantic annotations should include descriptions of workflow functionalities and information about its

input/output data. Consequently, the system can provide means for easy discovery and matching of resources and reusable workflows.

From the *operational perspective* and in order to allow scientists to describe workflows in a simple way without knowing their implementation details, the workflow language should span abstraction level by providing models to map abstract workflows into concrete ones [13]. Therefore, the system should support a global concept model composed of high-level abstractions for all application components and data, which can be mapped into a concrete workflow referring to actual processing and data resources. This easy to use concept model is especially important for multidisciplinary projects, where participation of domain specialists or stakeholders who do not have deep knowledge in computer science is required.

Evaluating workflow systems from the *control-flow perspective* we noticed that current workflow control logic lacks some characteristics needed for running long-term processes. In the GLOWA Volta project, simulation systems typically have quite different running times from the order of minutes (GAMS model) to the order of weeks (MM5 models). Therefore, insertion of these systems in real-time workflows is quite problematic and requires additional local services decoupling execution of such resources from the main stream.

Taking into account the dynamic structure of scientific workflows, from the *resource perspective* the system should be able to create variants of stored workflows adjusting them according to available resources specifications (e.g. CPU speed, amount of memory). In order to get the best performance on heavy long-term computations, workflow specifications should include resource requirements for every computational task. Later, during execution this information allows the workflow management system to select the most suitable resources.

The *data perspective* describes requirements for data flow management within workflows. From this perspective the heterogeneities mentioned earlier can take several forms: attribute level incompatibilities, entity definition incompatibilities including naming and schema isomorphism conflicts and abstraction level incompatibilities [15]. For example, in the "Best source of irrigation water" use case, some semantically similar entities have different measurements, time scales and precisions, e.g. WaSiM produces "reach inflows" in *mm* per hour, while GAMS needs it in *Ha-m* per week. The proposed semantic annotations should help to resolve such heterogeneities. Propagation of this information to the following stages of workflow development should simplify discovery and generation of necessary data transformation components.

Due to the fact that in our project, most of the integrated simulation models are data intensive, optimization of *data flows* becomes an issue. To avoid unnecessary data movements the language should allow techniques like reference-based data handling, direct peer-to-peer data movements and parallel data flows. However, data flow abstraction used in scientific workflow systems typically means only simple parallel execution of independent jobs and lacks expressions of the control logics such as merge, branching or iterations [27]. Therefore, it would be reasonable to enhance the abstraction by supporting "Data Parallel Execution Patterns" representing a form of single instruction multiple data parallelism, or "Pipelined Execution Patterns" defining sequences of tasks applied sequentially to a vector of data [19].

## 2.2. Workflow Execution and Control

This section continues the analysis of requirements while focusing on the execution and control of workflows. A workflow specification generated in the previous steps is passed to a *workflow enactment service* for execution. The major functions of such service are scheduling, executing, fault management, data movement and monitoring.

Scientific workflows usually consist of many distributed computationally intensive tasks that can be executed in parallel. Thus, the execution environment should support distributed parallel execution. Furthermore, in order to balance the work over a set of identical resources, the execution environment should have a load balancing mechanism. The environment should be able to dynamically discover available services fitting the task specification (semantically-based) and switch execution. This feature also can be used to prevent the need for "hard-coding" of service endpoints into the workflow specification (abstract constructs) or to improve the reliability of the system providing automatic fault handling. Moreover, to be able to integrate heterogeneous systems, the execution environment needs some services for automatic data transformation between tasks. These services should provide the environment in run-time with all necessary mediators.

The requirements for decoupling of long-running processes discussed in the previous section are also an issue on the execution level. To cover them, the execution environment should make use of a caching mechanism, which stores pre-calculated outputs in a local database. Scientists often rerun a certain workflow while changing only few input parameters of one model (e.g. for the GAMS model). Using caching mechanisms, such workflow can be repeated in a faster

and more efficient manner where only modified processes will be actually re-executed.

Another important issue is to allow users to monitor and steer the execution of workflows. Possible actions may include performing diagnostics and gathering summary reports of workflows in progress. Workflow steering provides the capability to stop the workflow execution and change its intermediate values at run-time. Moreover, to improve reusability and simplify design of new workflows the environment should be able to expose workflows as grid services.

## 3. Evaluating Workflow Systems

In this section, we are going to compare a set of *workflow management systems* (WFMS) against the requirements previously discussed. There are many systems currently available in the market. Here we are focusing only on some of them, which seem to be the most suitable in our case. Table 1 presents a summary of the overall comparison.

### 3.1. An Overview of WFMS

**Taverna** [18] is a software tool for designing and executing scientific workflows. *XScufl* is the workflow description language used in Taverna. It annotates the input/output data with three types of metadata: A MIME type, RDF limited to the bioinformatics domain [14] and a free textual description. The control flow is specified in XScufl with the help of conditional branching. Additionally, Taverna provides implicit iterations, which occur when a process expects fewer inputs than it receives. To overcome limitations of soap-based data movement for large datasets, Taverna provides "data proxies" for the reference-based data movement. Data transformations can be achieved by composing existing "shim" services or by using "*Beanshell*" scripts to build specific transformation components. Taverna's workflow execution service is based on the *freefluo* engine [7]. Semantic-based service discovery can be achieved in Taverna through the *Feta* plug-in. However, it is limited to services provided by the Feta engine. Taverna supports fault handling through a configurable mechanism. On the task level, users can specify number of retries and alternate tasks. On the workflow level, users can determine non-critical tasks where execution can be continued even in case of failure. Taverna has no capability to expose workflows as grid services. However, GRIA infrastructure [8] provides tools for deploying and running Taverna workflows as grid services. Moreover, workflows monitoring and steering are supported.

**Kepler** [12] is a scientific workflow environment based on the PtolemyII system [20]. Kepler uses the MoML language [11] to build workflows as clustered graphs of entities. Compared with other languages, data-flow based MoML does not provide any control flow constructs. Flow controls are supported by additional components (Actors, Directors). Due to its data-flow oriented structure, MoML provides support for variations of pipelined execution patterns, such as streaming, blocking and buffered [19]. For the workflow authoring, Kepler uses both semantic metadata standards (EML and OWL-DL) to capture the domain semantics. To optimize data flows Kepler allows working with remote data in three ways: *GridFTP*, *Storage Resource Broker (SRB)* and the *scp*, which is a shell command that helps users to copy files between systems. Two mechanisms for data integration are applied in Kepler. One is "shims" like Taverna; the other is to convert data into a common data model (EML), which is only used for the ecological domain. Kepler provides a centralized workflow execution service through the Ptolemy engine. Semantic annotations provided by Kepler allow semantic-based search of components. Kepler has no fault handling mechanism over the task level. However, on the workflow level, the workflow rescue mechanism is supported. In some of its actors Kepler also provides the output caching mechanism. Kepler workflows can be monitored by users. However, there is no way to edit the intermediate result of a paused task.

**Triana** [24] is a grid aware workflow management system. Workflows in Triana consist of tasks interconnected via data and control links. *TaskGraph* in Triana provides a WSFL-like representation of processes. Same as MoML in Kepler, WSFL-like language has no explicit control constructs. Besides the common context information, workflows in Triana can be annotated by unique pipe names, which will be given for each remote connection during distributing workflow components on remote services. In this way, abstract workflows can be converted into concrete workflows with extended annotation about resources and message channels. Triana Service is responsible for executing the incoming TaskGraph where partial TaskGraphs can be distributed as P2PS services to other Triana services. Methods for advertising and locating other peers/services are also available; however, they depend on the individual bindings. For example, in the JXTA binding, service discovery is done through the JXTA Discovery Service, while UDDI discovery service is used for Web Services binding. Until now there is no available mechanism for semantic-based search. Workflows in Triana can be exposed as web services or P2PS services. Triana workflows can be monitored, but not steered.

**Unicore 6** [3] is a grid computing technology, based on the *Open Grid Services Architecture* (OGSA) specifications, that provides seamless access to distributed grid resources. BPEL and JSDL [1] languages are used to describe workflows in UNICORE. JSDL is used to describe requirements of computational jobs for submission to a specific resource. Unlike other workflow description languages, it does not attempt to address the entire life cycle of a job or the relationship between individual jobs. In Unicore, data can be integrated using user-defined *Gridbeans*. The central component of the Unicore server side is the *Network Job Supervisor* (XNJS). It is a multi-threaded workflow enactment system. Fault tolerance in Unicore is handled internally by the XNJS, which is able to recognize data movement, input availability, task failures and user-defined exceptions. To accomplish failure recovery, the XNJS is used to retry failed jobs on the same resource. Unicore integrates the *Collaborative Online Visualization and Steering* framework (COVS) [21] in order to improve the work of scientists by providing online schematic visualizations of complex parallel simulations.

**NextGrid** [16] is a project aimed to develop a common architecture for next generation grids, which should create a dynamic market place for new services and products. OWL-WS framework is used in the project for managing adaptive grid service compositions. OWL-WS [2] is an extended OWL-S semantic language defining a workflow-centric model used to annotate services by properties involving input/output, constraints capability and execution information. Workflows acting as composite services are represented by adding process information to service descriptions. Besides, both services and workflows can be described as abstract or concrete depending on the availability of execution information. Like Unicore, OWL-WS is based on business domain and does not support parallel computing patterns. Unfortunately, until now there is no workflow engine encountering NextGrid specifications.

The **Knowledge-based Workflow System for Grid Applications (K-wf Grid)** [5] assists users to compose powerful workflows by means of a rule-based expert system. *GWorkflowDL*, a workflow definition language developed in the K-wf Grid project, describes workflows as high-level Petri Nets. The language consists of two parts: a generic part defining the structure of a workflow and a platform-specific part specifying how the workflow should be executed. WSRF2OWL-S components are responsible for converting the WSDL to OWL-S referring WSDL inputs/outputs to the domain ontology concepts. Since *GWorkflowDL* is based on Petri Nets, such translation-token language is able to model parallel branches and

loops. However, K-wf Grid lacks resource descriptions and utilizes a resource definition language *GResourceDL* for this purpose. Therefore, during the concretization of workflows in the *Workflow Execution Service* (GWES) *GWorkflowDL* specifications will be supplemented by the *GResourceDL* descriptions representing all available resources and their dependencies [9]. The K-wf Grid embeds the *Knowledge Assimilation Agent*, which facilitates the discovery of services through interactive semi-automatic ontology alignment (the OnTal tool). Fault tolerance is an important feature of the K-wf Grid. At the task level job retry is supported. At the workflow level the system provides robust checkpoint-restore functionality and the workflow rescue mechanism. In order to monitor workflows, the K-wf Grid provides a generic monitoring infrastructure (GEMINI) allowing end users to monitor and steer workflows.

### 3.2. Comparison

In the context of our requirements summarized in Table 1, we can see that the OWL-DL used in Kepler can help to capture semantic domain metadata and support the retrieval of components/services. Ontologies can only indicate semantic similarities, but not solve the heterogeneities between two semantically alike entities. The OWL-WS framework in NextGrid is a good candidate for annotation of workflows while it lacks provenance information. Kepler and Taverna provide better support for parallel data flows. Currently, most systems realize data movement via SOAP-based means. Considering performance problems of SOAP for data-intensive applications, the reference-based data transfer such as "data proxies" in Taverna will be a better alternative. Shims are used for data integration in all environments except Triana. Another integration approach is to use a common data model, which is supported by Kepler and Triana. However, this model limits the cross-domain integration. NextGrid and Triana introduce means for mapping abstract workflows specifications into concrete, but they lack resource specifications which are only supported by Unicore and K-wf Grid.

Some of the requirements for workflow execution environments such as load balancing, automatic data transformation, and data caching are not or weakly supported by any of the compared WFMS. Decentralized execution of workflows is partially supported only by Triana where the main workflow control of distributed TaskGraphs remains centralized. Even in the last version, Unicore still lacks a service discovery capability while Triana needs to extend existing services by semantic facilities. Taverna provides promising capabilities for fault handling

while it still needs a check-pointing capability, which is introduced only in K-wf Grid. Only Triana and Taverna workflows can be shared as grid services. Means for workflow steering are still missing in Kepler and Triana. Summarizing, we can state that none of the reviewed systems is fitting well our requirements.

## 4. Workflow System Architecture

To close the gap in workflow management discussed in the previous section, a *four-layered architecture* and a *new data model* for a workflow management system supporting integration of heterogeneous data-intensive applications are proposed. Our architecture is based on *Web Services* technology, since it provides a widely accepted basis for loosely coupled distributed systems (Figure 1). The first three layers from the bottom are responsible for mapping abstract workflows into semi-concrete workflows, which are passed to the fourth layer where they can be executed in a distributed and efficient manner over the underlying grid. In the following, we will give more details of the proposed data model and functionality of each layer.

### 4.1. Scientific Data Modeling

The presented data model is dedicated to support the concretization of abstract workflow specifications and provide means for annotation of components/services and specification of requirements for computational tasks. As illustrated in Figure 2 the model has two main entities. First, *AbstractWorkflow* entity defines abstract workflows with *Service* and *Link* entities. They provide a high-level description of services and show how they are related to each other. SAWSDL format [22] is employed for services definition since it is a good completion to traditional WSDL enhancing it with semantic annotations without a binding to any concrete type of semantic models. Second, *ConcreteWorkflow* entity refers to concrete workflows involving *Hardware* entity which points to the requirements of computational resources and *DataStaging* entity which refers to input/output data and *Link* entity which defines the relationship between services. The *Link* entity has two subclasses that support required parallel computing patterns: *dataflow* and *controlflow*.

### 4.2. Workflow Composition Layer

Workflow composition layer enables users to compose workflows using available services. Users proceed either by creating workflows from scratch or

by modifying existing workflow templates. For every service users can specify a set of resource requirements, which will be used later for the generation of semi-concrete workflows. As a result, a description of an abstract workflow will be composed. It is based on the proposed data model and contains references to the corresponding ontologies needed by the mapping layer.

### 4.3. Mapping Layer

On the mapping layer, abstract workflow descriptions and *ontologies* will be used in order to generate *mediators*, which are responsible for data transformations. This layer contains three main components. First, the *semantic matching component* is trying to find corresponding pairs between ontologies either by keyword and synonyms matching using *domain* and *functional ontologies*, or by searching in the pre-defined *matching ontology*. If the matching information is already available in the matching ontology, the associated mediator can be retrieved directly from the mediator catalog. Otherwise, this matching information will be passed to the *semantic mapping component* where rules for transforming from one schema to another are generated and added into the *rules ontology*. The system allows users to build their own data transformation rules manually with a GUI. These rules can be used by the *mediator generation component* to create new mediators (shims) which are stored in the *Mediator Catalog*.

### 4.4. Semi-Concrete Wf. Generation Layer

This layer acts as a bridge connecting the *mapping layer* with the *workflow execution layer*. It is responsible to generate *semi-concrete workflows*. The "semi-concrete" here means that workflows still lack information about which concrete computational resources will be used. This information can be obtained later in run-time. This decision is taken due to the dynamic nature of grid environments and for long running processes early allocation of resources can become obsolete. Semi-concrete workflows combine descriptions of abstract workflows and resource requirements obtained from the *workflow composition layer* with mediators information retrieved from the *mapping layer*.

### 4.5. Workflow Execution Layer

The *workflow execution layer* provides a new approach for distributed execution of semi-concrete workflows. The basic idea of this approach is to

separate workflow control and execution flows. The proposed approach can be realized as a deployment of a bundle of WSDM services [28] for workflow management in every grid node. Such bundle of services contains four main services. First, the *resource information service*, which can be used to retrieve information about resources involving both, relatively static information such as system configuration and more dynamic information such as instantaneous load. Second, the *scheduler service* which coordinates the execution of a workflow instance. It breaks the instance into small tasks, submits each task to a specific grid node, monitors their execution and gathers the final outcome of the submitted workflow. The selection of proper resources is done by contacting the *Service Catalog* to retrieve for each task a list of currently available resources. This list will be ordered depending on the similarity between the task's resource requirements and the resources' current state. The scheduling of each task is done only after the completion of all its predecessors. Third, the *task service* which is responsible for the actual execution of the submitted task, caching the task's output, and notifying the submitting scheduler about the task status. Finally, the *data management service*, which is dedicated for reference-based data movement between nodes and data provenance. Here OGSA-DAI service [17] exposes the generated service/application data as web services, which allows direct remote access to the data. The remote interaction of services located in different nodes is based on *publish and subscribe notification events* supported by the *Web Services Notification* framework [29].

According to this principle, execution control of workflows remains centralized, but their actual execution is moved to local grid nodes where the data and services are located. Consequently, many advantages can be obtained. First of all, this will avoid unnecessary movements of data through the network. Second, with help of mediators obtained in run-time from the *Mediator Catalog* automatic data transformations can be applied directly where the data is located. Third, smart rerun can be easily achieved through data caching and provenance mechanism. Last but not least, distributed fault handling and load balancing can be accomplished through the use of scheduler services in different grid nodes.

## 5. Conclusion

Much research has already been done and more research is still under way to provide an efficient scientific WFMS. However, additional efforts are needed to make these systems really usable for

building complex distributed systems integrating several heterogeneous data-intensive applications.

This paper tries to gather and filter requirements that are typical for such applications. The presented list of requirements was collected from several use cases of the GLOWA Volta project. The practical evaluation of the most potentially suitable scientific workflow systems showed that each of them still lacks some of the stated requirements. Therefore, a new approach for scientific workflow systems targeted to bridge these gaps was presented. The proposed architecture provides means for efficient execution and composition of data-intensive workflows integrating heterogeneous computationally intensive applications. Using principles of decentralized execution of workflows and reference-based data movement, the proposed architecture should be able to reduce communication traffic between services sufficiently. As the execution of workflows is performed by the local workflow execution services that lie in the same host as the target application, full control over the application and all necessary data transformations is possible. Additionally, means for easier composition of workflows and integration of applications are provided. Through recognition of semantic information in early stages of workflow composition it is possible to partially automate the generation of concrete workflows and other supplementary components like shims and data transformers. Moreover, abstract workflow composition means provide an easier and intuitive way for description of workflows. It is practically important for multidisciplinary projects where participation of non-IT people, for example, decision-makers is required. Workflow composition tool allows these people to build workflows on the abstract level without deep knowledge of the underlying complex grid infrastructure.

The implementation of the presented architecture is now in early stages and we expect to receive the first working version over the next few months.

## References

- [1] A. Anjomshoaa et al. Job submission description language specification. Technical report, Global Grid, 2005.
- [2] S. Beco et al. Putting semantics in grid workflow management: the OWL-WS approach. Technical report, University of Southampton IT Innovation Centre, 2006.
- [3] K. Benedyczak et al. UNICORE as uniform grid environment for life sciences. In *Proc. of European Grid Conference*, Springer (LNCS 3470):364–373, 2005.
- [4] A. Brooke, D. Kendrick, A. Meeraus, and R. Raman. GAMS: A user's guide. *GAMS Development Corp.*, 1998.
- [5] M. Bubak et al. K-wfgrid - the knowledge-based workflow system for grid applications. In *Proc. of CGW'06*, Vol. II:74–81, 2007.
- [6] I. Foster et al. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal Supercomputer Applications*, 15, 2001.
- [7] Freefluo workflow engine. <http://freefluo.sourceforge.net>.
- [8] GRIA Service oriented infrastructure. University of Southampton IT Innovation Centre, <http://www.gria.org/>.
- [9] A. Hoheisel. Grid workflow execution service - dynamic and interactive execution and visualization of distributed workflows. In *Proc. of the Cracow Grid Workshop*, 2006.
- [10] K. Jasper and J. Schulla. Model description WaSiM-ETH. Geography Department, ETH Zürich, 1999.
- [11] E. Lee and S. Neuendorffer. MoML: A modeling markup language in XML. Technical report, Technical Memorandum ERL/UCB M 00/12, 2000.
- [12] B. Ludäscher et al. Scientific workflow management and the Kepler system. *Concurr. Comput.: Pract. Exper.*, 18(10):1039–1065, 2006.
- [13] B. Ludäscher et al. Scientific process automation. Technical report, Scientific Data Management Integrated Software Infrastructure Center, 2007.
- [14] myGrid project. <http://www.mygrid.org.uk/>.
- [15] M. Nagarajan et al. Semantic interoperability of web services - challenges and experiences. In *Proc. of IEEE International Conference on Web Services*, 2006.
- [16] NextGRID project. <http://www.nextgrid.org/>.
- [17] OGSA-DAI project. <http://www.ogsadai.org.uk/>.
- [18] T. Oinn et al. Taverna: lessons in creating a workflow environment for the life sciences. *Concurr. Comput.: Pract. Exper.*, 18(10):1067–1100, 2006.
- [19] C. Pautasso and G. Alonso. Parallel computing patterns for grid workflows. In *Proc. of the 15th IEEE International Symposium on High Performance Dist. Computing*, 2006.
- [20] PtolemyII project. <http://ptolemy.eecs.berkeley.edu/>.
- [21] M. Riedel et al. Requirements and design of a collaborative online visualization and steering framework for grid and escience infrastructures. In *Proc. of German e-Science Conference*, 2007.
- [22] SAWSDL. Semantic WSDL. <http://www.w3.org/>.
- [23] S. Shumilov et al. First steps towards an integrated decision support system. In *Proc. of the 20th International Conference on Informatics for Env. Protection*, 2006.
- [24] Triana project. <http://www.trianacode.org/>.
- [25] W. van der Aalst and A. ter Hofstede. YAWL: Yet another workflow language. *Information Systems Frontiers*, 30(4):245–275, 2005.
- [26] GLOWA Volta project. <http://www.glowa-volta.de/>.
- [27] W. Johnston et al. Advances in dataflow programming languages. *ACM Computing Surveys*, 36(1):1–34, 2004.
- [28] WSDM. Web services distributed management. <http://docs.oasis-open.org/wsdm/>.
- [29] WSN. Web services base notification. <http://docs.oasis-open.org/wsn/>.
- [30] J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *Journal of Grid Computing* 34(3), pages 44–49, 2006.

**Table1: Summary of workflow environments comparison**

Requirement	Taverna	Kepler	Triana	Unicore	NextGrid	K-wf Grid
Language	XScufl	MoML	WSFL	JSDL/BEPL	OWL-WS	GWf-S
<b>D1:Service/Component Description</b>	- MIME - RDF - Free text	- EML,FGDL - Dublin Core - OWL-DL	Simple WSDL	JSDL	OWL-S	OWL-S
<b>D2:Workflow Annotation</b>	-Input/output - Provenance (LSID,KAVE)	- Context - Input/Output - Index	- Resource - Message channels	Context	- Wf profile - Wf grounding - Wf model	- Context - Semantic description
<b>D3:Control Flow Patterns</b>	- Implicit iteration - Conditional Branching (driven by control links)	Supported by Directors and Actors: - Conditional Looping	Supported by components: - Dynamic distributing - Conditional Looping	Business workflow patterns (Defined in BEPL)	Business workflow patterns (Defined "Process" by OWL-WS)	Parallel branching (Defined in Petri nets)
<b>D4:Handling Scientific Data</b>						
Parallel Patterns	Dynamic Parallel Execution	- Streaming - Blocking - Buffered	- Buffered	N/A	N/A	N/A
Data Movement	- Soap based - Data Proxy	- GridFTP - SRB - SCP	File based	Soap based	Soap based	GridFTP
Data Integration	- Shims - Beanshell	- Shims - Common data model	Common data model	Gridbeans	Shims	Shims
<b>D5:Mapping to Concrete WF</b>	Concrete	Concrete	Task annotation	Concrete	Workflow substitution (Profile matching)	Workflow execution services
Engine	Freefluo	Ptolemy	TrianaService	XNJS	N/A	GWES
<b>E1:Parallel Execution</b>	Centralized Multi-Threading	Centralized Multi-Threading	Decentralized (P2PS)	Centralized Multi-Threading	N/A	Centralized Multi-Threading
<b>E2:Loading Balancing</b>	N/A	N/A	N/A	N/A	N/A	N/A
<b>E3:Automatic Data Trans.</b>	Only for simple data types	N/A	N/A	N/A	N/A	N/A
<b>E4:Service Discovery</b>	- Scavengers - Semantic search	Semantic search	- JXTA Discovery - UDDI server	N/A	N/A	Semi-automatic semantic search (KAA)
<b>E5:Fault Handling</b>						
Task-Level	- Retry - Alternate resources	N/A	N/A	-Retry - User defined Exceptions	N/A	-Retry - Alternate resources
WF-Level	WF rescue	WF rescue	N/A	N/A	N/A	-Checkpoints -WF rescue
<b>E6:Caching Mechanism</b>	N/A	Limited to some actors	N/A	N/A	N/A	N/A
<b>E7:Shared WFs</b>	WFs as GRIA Job services	N/A	WFs as Web or P2PS services	N/A	N/A	N/A
<b>E8:WF Monitoring/Steering</b>	Monitoring and Steering	Monitoring	Monitoring	Monitoring and Steering (COVS)	N/A	Monitoring and Steering

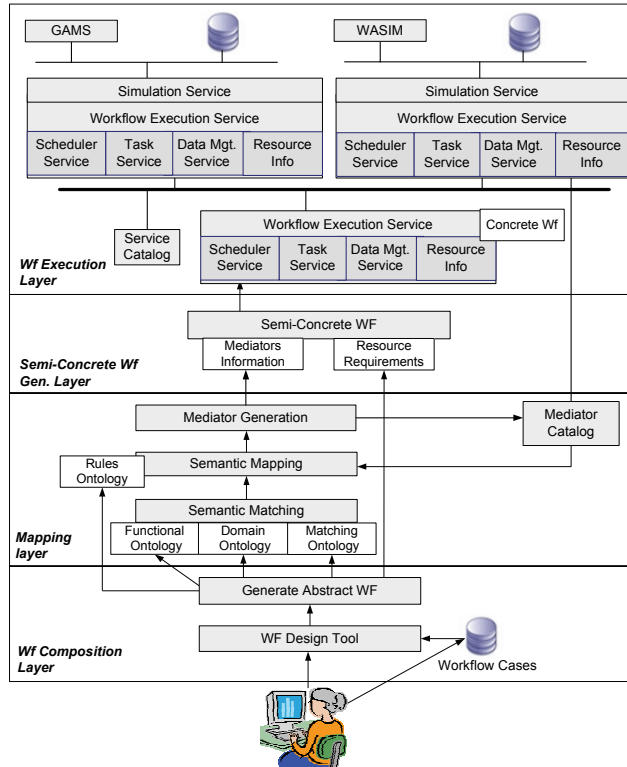


Figure 1: Four-layered architecture for workflow management

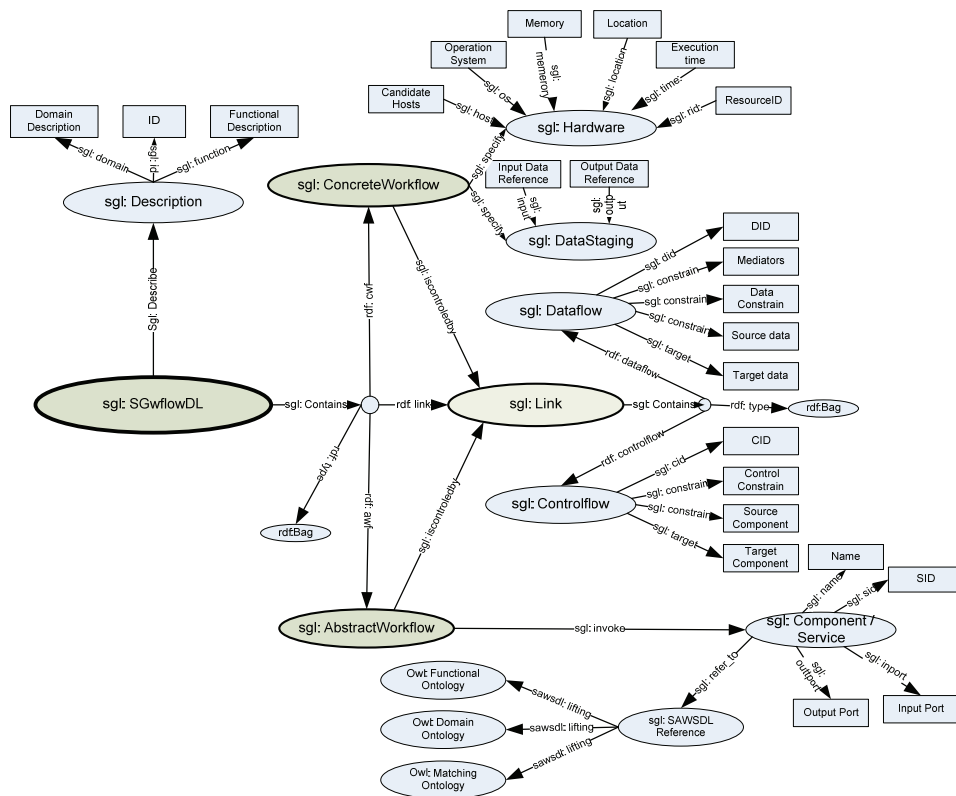


Figure 2: RDF-based data model