A Service Infrastructure for e-Science: The Case of the ARION* System

Catherine Houstis, Spyros Lalis, Vassilis Christophides, Dimitris Plexousakis, Manolis Vavalis, Marios Pitikakis, Kyriakos Kritikos, Antonis Smardas, and Charalampos Gikas

Institute of Computer Science, Foundation for Research and Technology – Hellas
P.O. BOX 1385, GR-711 10 Heraklion, Greece
{houstis,lalis,christop,dp,mav,pitikak,
kritikos,smardas,hargikas}@ics.forth.gr

Abstract. The ARION system provides basic e-services of search and retrieval of objects in scientific collections, such as, data sets, simulation models and tools necessary for statistical and/or visualization processing. These collections may represent application software of scientific areas, they reside in geographically disperse organizations and constitute the system content. The user, as part of the retrieval mechanism, may dynamically invoke on-line computations of scientific data sets when the latter are not found into the system. Thus, ARION provides the basic infrastructure for accessing and producing scientific information in an open, distributed and federated system. More advanced e-services, which depend on the scientific content of the system, can be built upon this infrastructure, such as decision making and/or policy support using various information brokering techniques.

1 Introduction

ARION is a service-based infrastructure designed to support search and retrieval of objects in scientific collections, such as, data sets, simulation models and tools necessary for statistical and/or visualization processing. It also actively supports dynamic and distributed scientific data processing workflows, in interactive and batch mode. The computational grid used in ARION is composed of geographically distributed and heterogeneous resources, namely, servers, networks, data stores and workstations with GUI displays, all resident to the member organizations that provide the scientific content and resources. ARION provides the means [1], [2] for organizing this ensemble so that its disparate and varied parts are integrated into a coherent whole. Hence, ARION can be viewed as the middleware between users, the data they wish to process and the computational resources required for this processing.

Central to ARION are ontologies and workflows. They are the main mechanisms for recording expert knowledge, for information representation/navigation and for expressing computation processes over the grid.

* ARION is supported by the European Commission under the 5th Framework Programme, IST-2000-25289, Key Action 3: Digital Heritage and Cultural Content.

Ch. Bussler et al. (Eds.): WES 2002, LNCS 2512, pp. 175–187, 2002.

The ARION system is built using results of two recently initiated research activities, the Semantic Web and the Semantic Grid. Within the former activity, RDF/S [17], promotes semantic interoperability by making use of ontologies and associated tools (such as RDFSuite [3]) at the information modeling level, whereas within the latter, a computational framework for the system is defined. In addition, specifications of metadata standards and syntactic interoperability standards, like OpenGIS [20], are direct and very valuable contributions for the implementations of basic computing/visualization services.

The rest of the paper is structured as follows. In section 2, we present the system architecture. Although the design of the architecture is complete, the limited space permits only a high-level description. In section 3, we explain the use of recent developments in the architecture, focusing on the use of ontologies and workflows and associated software tools for their implementation. Finally, in section 4 we conclude our presentation by suggesting the use of the ARION system as a scientific service tool for more advanced information services.

2 ARION a Lightweight System Integration Architecture

The principle architecture of the ARION system consists of three independent parts. The *Search Engine* allows users to pose queries to the knowledge provided by ARION. The *Workflow Data Base System* contains the workflow specifications and handles the preparation of execution specifications to be sent to the workflow runtime. The *Workflow Runtime System* is responsible for the execution of workflows and the management of the information produced during this execution from the distributed nodes. The architecture is shown in Fig. 1.

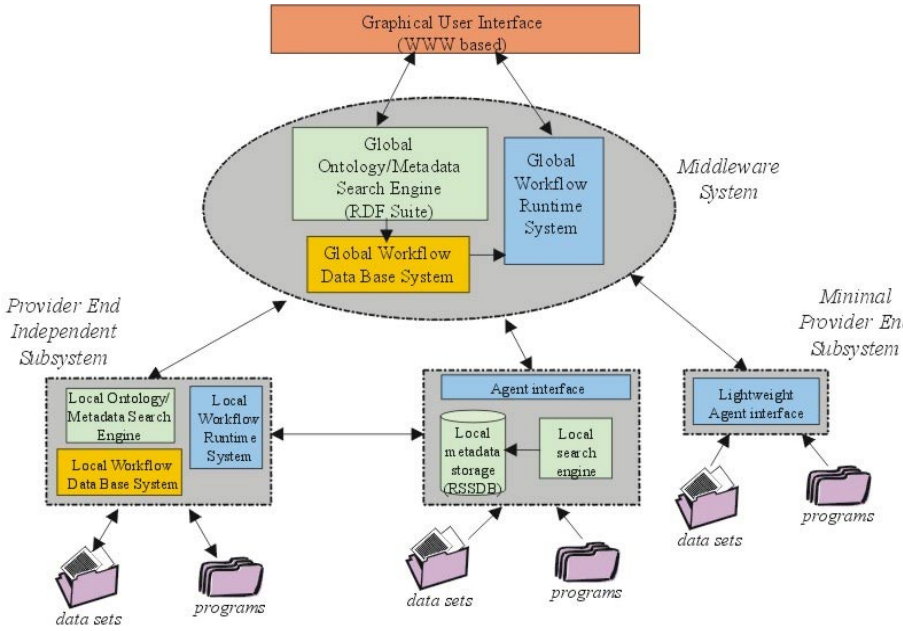

Fig. 1. The ARION architecture

All three parts of ARION are conceptually linked by an ontology and the corresponding (computational) workflows, the overall structuring principles of the whole system.

ARION is composed of a set of distributed nodes containing different data sets and programs (scientific collections). These nodes interoperate using an agent platform, and provide the basic services for a workflow execution. Workflows typically rely on distributed and autonomous tasks, and are controlled by a centralized server (ARION main server). The agents installed on each node execute workflow tasks (mainly computations) and monitor its services.

2.1 The Search Engine Subsystem

The *Search Engine*, shown in Fig. 2 (a), is mainly based upon RDFSuite [3], which will be described in section 3. An editor is used to enter metadata and ontology information, which are loaded to the ORDBMS of the ARION system. A global storage located in the ARION main server contains all the metadata descriptions submitted to or created by the system (centrally or locally). An update tool is responsible for collecting locally (on each ARION provider-end independent subsystem) the submitted metadata.

A RQL [4] query engine executes any search demand that is originated from the user interface or the *Workflow Data Base System*. Queries can be posed for either the metadata information or the concepts and properties of the ontology.

2.2 Workflow Data Base Subsystem

The *Workflow Database System*, shown in Fig. 2 (c), consists of the following components: Workflow Editor, Update Tool, Workflow Storage System, Workflow Database Server and Statistical Database.

The Workflow Editor is used to define a workflow specification or to alter an already existing one. The specifications go through the Update Tool and finally stored in the central Workflow Storage System.

The Workflow Database Server is the most important part of the Workflow Database System. It communicates with the Graphical User Interface, the *Search Engine* and the *Workflow Runtime System*. To illustrate the functionality of the Workflow Database Server, suppose that a user issues a data set query. The Workflow Database Server contacts the *Search Engine* to determine if the requested data set is stored in our system. In case the answer is negative, it searches the Workflow Storage in order to find a workflow specification that can produce the specific data set and if such a workflow exists, it prepares a workflow execution specification. This workflow execution specification is eventually sent to the *Workflow Runtime System* to be executed.

During a workflow execution process, the *Workflow Runtime System* communicates with the Statistical Database and stores runtime related information like execution time, resources occupied, network nodes passed, number of execution errors, warnings etc.

2.3 Agent Runtime Subsystem

The *Workflow Runtime System*, shown in Fig. 2 (b), consists of two main components, namely the Workflow Manager and the User Monitoring System.

The Workflow Manager has seven sub-components. The Workflow Engine is responsible for the execution of a workflow instance. For each workflow definition received from the *Workflow Data Base*, an execution environment is initialized and a Task Scheduler is created. The Task Scheduler makes decisions about the order of execution of tasks and assigns blocks of tasks to the Task Manager, which is responsible for the execution. The Task Manager cooperates with the Agent Management System where all the objects related with the agent platform are located, including agent generation mechanism, communication objects and proxies to Grasshopper objects. Grasshopper [21] is the basic agent used in the ARION runtime subsystem. A Task Execution Agent is created by the Agent Management System, containing all the workflow-related information required in order to migrate to a remote node and execute the designated tasks.

The Gateway is the sub-component that is used for connecting with the runtime system and facilitating remote management (e.g., from the Web). The last sub-component is the Logger, which keeps track of all the steps taken during the execution phase of a workflow instance.

The User Monitoring System is divided into the Workflow Execution Monitoring, where a user can view or even alter the progress of a workflow execution, and the Notification System, where user input is requested.

2.4 User Authentication/Authorization

The authentication process, during which the user proves his/her identity, is implemented by the system via an authorization mechanism, which grants or denies permission to access a digital object (data or resource). In order to define these permissions, we have adopted a role-based access control mechanism [18] to handle the users and objects. A user that is assigned to one role has all the access privileges of that role. These roles are shared inside the distributed system by using a hierarchical structure (depending on trust domains). When a user initiates a workflow, the agent runtime system acts on behalf of that user and carries all the required authentication and authorization information.

2.5 User Interfaces

The user interface of a system like ARION has three main responsibilities. Firstly, it has to provide mechanisms to access the domain ontology and metadata information, publish them, manage them and of course to query them. Web-based interfaces allow users to navigate through an ontology efficiently and easily (e.g., via hyperbolic trees). This provides a powerful “see and go” interaction style that retains the simplicity of “point and click”. A Metadata Editor inserts new metadata information into the system or modifies existing ones. The ontology can also be used by the *Search Engine* user interface to provide guidance when formulating queries. This has been proved to be a more effective way to retrieve accurate search results than a simple keyword-based query.

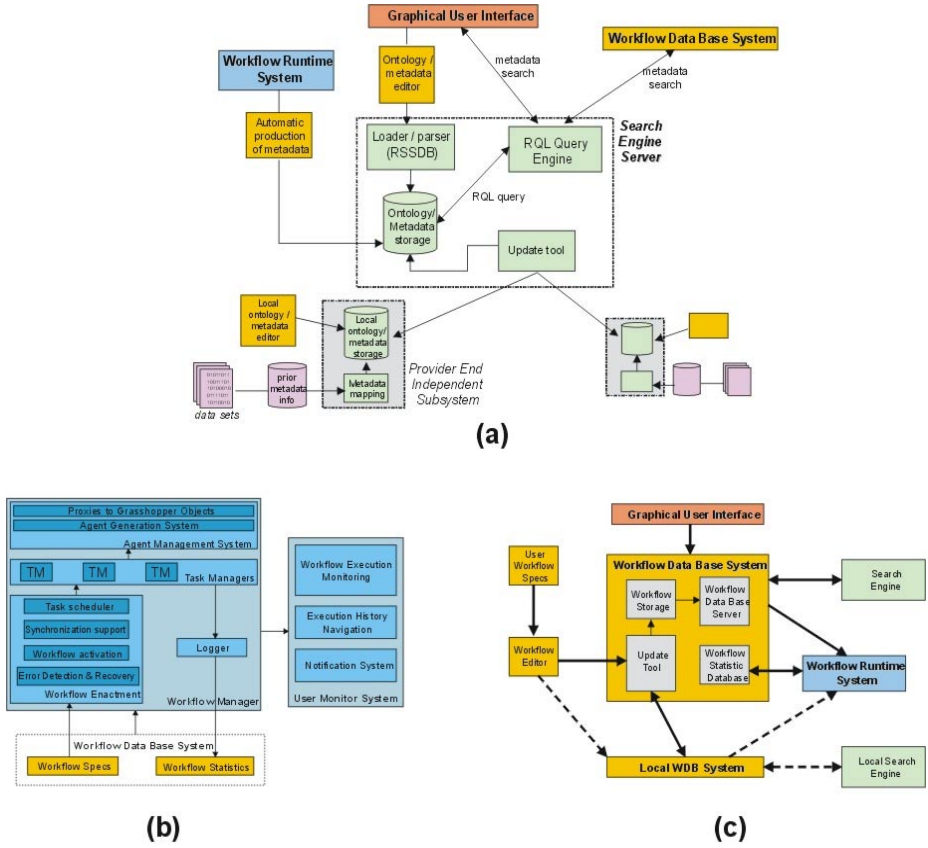

Fig. 2. (a) The Search Engine subsystem architecture, (b) The Agent Runtime subsystem architecture, (c) The Workflow Data Base subsystem architecture

Secondly, it has to visualize produced or retrieved scientific data sets. In our case, these are mostly geographical information, so it has to provide Geographical Information System (GIS) capabilities in order to visualize them over maps. For example, the results of a workflow execution may be graphically displayed. In fact, Java applets could be used to visualize any kind of scientific information.

Thirdly, it has to provide ways to interact and control a workflow execution. We have adopted three different approaches in user interface design for these purposes. In the Workflow Overview approach, the user conceives the whole process of a workflow execution and is aware of the various steps that comprise the workflow. The user may interact with the runtime system through a Web-based interface. This is implemented by dynamically produced Web pages based on Java Servlets. In this case, the user is able to:

- Monitor the workflow execution and retrieve statistical and other information about the execution of the current as well as of previous phases.
- Suspend / resume the workflow execution.

- Provide the input parameters required for a task to go on with its execution.
- At certain points (checkpoints) to choose whether the workflow will continue its execution or whether it should return to a previous stage.

There is no need for a user to be constantly connected in order to control a workflow that is being executed. The user is notified (by e-mail, SMS etc.) when a predefined stage (checkpoint) in the workflow execution is reached or when (data or control) input is required for the workflow to continue execution.

In the Application View approach, the user views a workflow execution as a standalone application that hides the complexity of working with several different processes and is implemented as a Java applet. The user may program the execution of the whole workflow process and get the final output. That implies that input parameters to tasks will have to be pre-specified and will be regarded as input parameters to the application. It is expected that this approach will be used from those wishing to program a large number of workflow executions (batch execution) and obtain the final output without any intervention during their execution.

The third approach is a combination of the previous two. The user will be able to program the execution of a number of workflows, providing all the input parameters from the beginning. However, the system will request verification (through a Web interface), when, for example, a parameter is used.

ARION also includes a collection of tools addressing expert users (providers). Tools, like Protégé-2000 [19], can be used by domain experts to develop ontologies. A Workflow Editor has been implemented, which enables users to graphically create, modify or delete workflow specifications. These specifications are transformed into XML files compliant with our XML-based workflow specification language. The editor uses graphical constructs / symbols that correspond to language constructs (workflow specification entities) of our workflow specification language, and specialized forms and templates that automate the creation procedure. The workflow entities are combined according to the semantics / restrictions of our workflow specification language and no other combination is allowed. The users should be acquainted with the workflow language in order to understand the basic semantics of the editor. With this knowledge, users can comprehend the graphical constructs and combine them to generate a complete workflow specification.

2.6 Tailored Deployment

ARION's three-tier distributed architecture enables a high level of reliability and scalability. Not all nodes (provider end nodes) need to install all three ARION subsystems locally. Because of the flexibility of ARION's architecture, multiple deployment options are available. Through analyzing each provider's environment, determining general administrative needs and resources, the deployment option best suited for a provider can be effectively chosen.

In a minimal deployment configuration, only the agent platform is deployed in a provider side local server node. Neither the search engine (there is no local metadata storage) nor the workflow database are installed. In this configuration, the ARION main server connects directly to this node, utilizing software agents, and executes workflow tasks.

The other two configurations, shown also in Fig. 1, present a more autonomous and self-managing deployment. Local queries could be performed and even the execution of a completely local workflow would be possible. The ARION architecture can be easily extended to handle any number of clients and manage provider local server nodes.

Additional features incorporated in the ARION system include on-line data set production and automated metadata generation for these data sets. ARION offers tools (editors) for easy publication of workflow specifications and metadata information. All the above features contribute to provide minimal administration of the system.

The user-end required platform is kept to a minimal, that is, the use of any web browser is sufficient to access ARION. In order to achieve this, a gateway is provided to every ARION server, which hides the agent runtime system complexity from the user web browser.

3 Ontology Based Knowledge Representation

Very often, there is a need to share the meaning of terms in a given domain. Achieving such a common understanding is accomplished by agreeing on an appropriate way to conceptualize the domain. The result is a domain-oriented ontology, a "formal specification of a conceptualization" [5], which is either domain specific or generalizes or reconciles domains.

An ontology, apart from the navigational benefits it brings, provides common semantics that can be used to improve communication between either humans or computers. Ontologies may be grouped into the following three areas, according to their role: to assist in communication between people, to achieve interoperability among computer systems, or to improve the process and/or quality of engineering software systems [6].

We identify five important classes of benefits that may result from the use of ontologies:

- **Useful queries.** An ontology is used for searching a metadata information repository for resources (e.g., data sets). The main benefit of this approach is faster and intelligent access to relevant information resources, which leads to more effective use and reuse of knowledge resources.
- **Sharing knowledge.** More generally, an ontology helps to integrate many overlapping pieces of information. Ideally, people will contribute to a shared, global and well-organized knowledge base of information. Of course, this requires a lot of effort in both the technical and institutional realms.
- **Extensibility.** An ontology can be enriched with new classes in every branch and every level. However, the ontology must be designed carefully in order to be sufficiently general (and thus extensible).
- **Reusability**. The ontology is the basis for a formal encoding of the important entities, attributes, processes and their inter-relationships in the domain of interest. A common ideal for an ontology is that it could be a re-usable and/or shared component in a software system [6].

- **Identifying rights.** In particular, agent-based systems such as ARION will need ontology-based metadata to compute the licenses required to provide various services, to pay the appropriate copyright fees, and so on.

In the ARION system, we focus on common semantics for scientific collections. We have worked on an environmental (ocean wave) ontology consisting of a collection of different facets. For instance, facets may describe data sets, production methods including mathematical modeling, parameters used, etc. A reason for the combination of several facets is the modularization of a potentially large monolithic ontology. In addition, facets may be formulated according to core queries users may be interested to formulate. A facet-based engineering of an ontology scales well with large scientific ontologies. New information may be appended in accordance to user/provider needs. The ontology definition contains an "IS-A" hierarchy of relevant domain concepts, relationships between concepts and properties of concepts. There are two main entities in our ontology, consisting of different facets that describe the scientific data and scientific models respectively. This approach provides another level of granularity. The basic structure of an ocean-wave ontology is shown in Fig. 3.

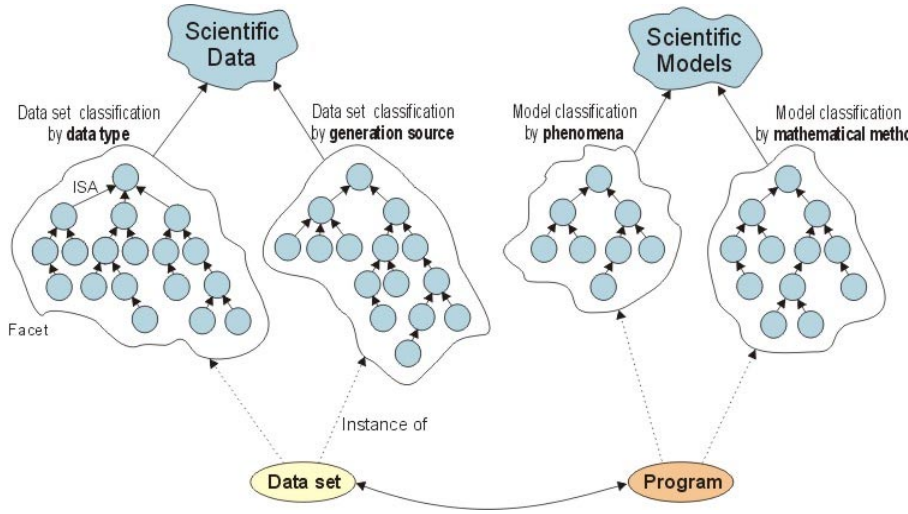

Fig. 3. The structure of our environmental (ocean wave) ontology

The representation of metadata in ARION is realized by RDF [16]. RDF schemas (RDFS) [17] provide a basic type schema for RDF. Objects, classes, and properties can be described. In relation to ontologies, RDF/S provides two important contributions: a standardized syntax for writing ontologies and a standard set of modeling primitives like instance-of and subclass-of relationships.

The expressive capabilities of RDF and RDF Schema suffice for the purposes of ARION and are used as the basis for modeling our domain of knowledge. In particular, metadata description is ontology-driven, in the sense that the construction of the metadata information is carried out in a top-down fashion, by populating a given ontology, rather than in a bottom-up fashion. Every scientific object (data set or model) is described by a collection of attributes (properties), inherited from its parent-class or native to the specific object.

3.1 Conceptual Querying

The creation of RDF raises the prospect of a widely accepted standard for representing knowledge on the Web. However, just representing knowledge and information is not enough; query languages and tools are needed to enable the creation of RDF-aware applications. Such a solution is RDFSuite[1].

RDFSuite is a suite of tools for RDF metadata management providing storage and querying both RDF descriptions and schemas [3]. It consists of three main components: a RDF validating parser (VRP), a RDF schema-specific storage database (RSSDB) and a query language (RQL) [4].

The RDF Schema Specific Database (RSSDB) loads recourse descriptions into an object-relational DBMS. The representation of the RDF schema and the metadata information is done separately, avoiding the monolithic table approach of representing RDF triples. This provides flexibility to the underlying ORDBMS and allows easier manipulation of schema information. After the validation of RDF metadata and the consistency check of schema descriptions, a loader stores them in the DB.

The RDF Query Language (RQL) [4] is used to uniformly query RDF metadata information and RDF schemas. Thus, we can exploit this ability to implement schema browsing, since large RDF schemas carry valuable information themselves due to class refinement.

There are various ways in which an ontology assists searching data collections. In the case of ARION, it is used as a basis for semantically structuring and organizing the metadata information repository, and to assist in query formulation. We use our ontology in three distinct ways:

- as a conceptual framework to help the user think about the information repository and formulate queries
- as a guide to understand the ontology-driven metadata
- to drive the user interface for creating and refining queries

Ontology based search uses an intuitive relationship between concepts to provide intelligent access to information. By using RQL, the queries can either use semantic concepts (RDF schema) or just be word-based (RDF metadata). The semantic richness of the ontology can be an important factor here. A richer ontology can improve search.

4 Active Data Generation and Retrieval

In a repository of data sets it is not particularly useful to store the data sets alone. Storing the computational tasks necessary to produce output data sets is even more valuable especially when the user is allowed to customize the computation according to his/her needs. The ARION middleware provides (a) tools to publish computations of various data sets according to an XML based workflow specification and (b) a runtime platform to execute the workflows. Thus, ARION is an active repository that also stores workflows for scientific data set computing.

[1] http://139.91.183.30:9090/RDF/

4.1 XML-Based Workflow Specifications

In ARION workflows provide abstractions of natural process models. An important portion of a process model (workflow) is the definition of the process logic. The definition of the process logic is expressed by the usage of *a workflow specification language*. Our workflow specification language is based on the XRL language [14], an XML-based workflow specification language. Our language extends XRL to match our model of scientific workflows. The fundamental idea is that a workflow can have the same construct that a programming language has and that derives from the effort to model the flow of scientific tasks in order to produce a data set. This flow can be sequential, parallel, conditional or a combination of such sub-flows. For this reason we equip our language with constructs like sequence, parallel-sync (tasks are executed in parallel), conditional if-else and while-do that can be combined and attached to a workflow definition. These constructs are sufficient to model the scientific logic of a workflow. A task is defined of having input and output (just like a program execution), and some properties like location (URL) and type. The input and output are usually data sets that also have properties like location, type etc. This information is needed in order for the *Runtime System* to actually execute a task.

Workflow specifications are generated from our XML-based workflow specification language. XML [15] is a world-wide standard file format with a specific tree-like internal structure. With the help of a DTD (a prolog-formatted file), that defines its vocabulary and structure, an XML file can describe the semantics of our workflow specifications. XML files are easily transported, exchanged, stored and occupy little space on disk. Additionally, several tools have been developed by the wider CS community that can be used to manipulate XML files and to transform them into other data-representation formats.

4.2 Agent Runtime Technology

Workflows are considered as a kind of multi-agent cooperation, in the sense that software agents may be used to perform tasks (computational processes), and the workflow can be used to orchestrate or control the interactions between agents. To be more specific, a workflow specification [9] is defined by the following elements:

- activities to perform (tasks)
- sequence of activities (control flow)
- data sets
- data flow

A workflow consists of several tasks and the relations among them are managed by the control flow. The runtime system enables the integration of each task's application-specific logic into a large application that combines the knowledge of separate tasks. The specification of a task contains a description of the required input (i.e., data sets and initialization parameters) as well as the produced output. It may also describe execution rights / privileges for users, groups of users, machines or computer programs. Tasks are usually executable programs installed on remote machines and therefore the definition of a task also includes remote host's related information. The

data flow states how data sets move between different tasks. Examples of data sets are files of scientific content and database entries.

The objective of the agent runtime system is to take care of the execution of workflows and it is a part of a Workflow Management System (WfMS). Requests for workflow execution originate from the Workflow Database and are managed by a Workflow Engine, which is responsible for interpreting the workflow definition and interacting with a Task Scheduler. These requests include the workflow specification that contains all necessary information required in order to facilitate data set(s) production.

A Task Scheduler determines the order of execution of applications on host machines and provides all the necessary initialization parameters and input data sets required. The runtime system guarantees that all relations among tasks described in the workflow specification will be preserved while trying to achieve the highest possible level of parallel task execution.

The runtime system supports the notion of backtracking. This workflow feature offers the option of returning back to a previously executed step rather than continuing the normal execution flow. Our aim is to support the need for calibration of input parameters in order to achieve a desired result. The system can be programmed to execute repeated executions of the same workflow with slight changes in the input parameters of some tasks so that the user won't have to request their execution one at a time. This feature is very important since parameter calibration is part of scientific computing and takes place interactively with the user. As pointed out at the user interface section three different interfaces are supporting it. Interactive, batch and intermediate interfaces are tailored for user interactivity.

ARION's runtime system implementation takes advantage of mobile agents' technology to facilitate the execution of remote tasks. A software agent is defined as a computer program that acts autonomously on behalf of a person or organization [10]. Furthermore, the properties proactive, autonomous, intelligent [11], and mobile are often used to characterize agents.

Mobile agents are generally used in order to reduce the network load, to overcome the network latency and to encapsulate protocols. In addition, their use has been widely adopted for the following reasons:

- Agent technology is suitable for workflow execution. The notion of execution of tasks residing in physically separated machines fits well with the mobile agents' execution model. It also provides features for the enhancement of complex information retrieval and workflow services.
- It allows flexibility for the runtime system in terms of task scheduling. An agent may be provided with variable level of knowledge and/or authorization to decide/collaborate with other agents during its execution. So, part of the task scheduling process may be entrusted to mobile agents.
- Agents can perform complex tasks and communicate/co-operate with other agents on behalf of the user. They are also capable of operating without additional user input and act independently, even if the user is disconnected, which makes them ideally suited for the fulfillment of automated tasks.
- The user doesn't have to be constantly connected to the system. Agents represent a user and operate on behalf of him without the user being required to be on-line for

the whole period of the workflow process. The user may connect and disconnect several times and be able to monitor the evolvement of the workflow.

- It addresses the limited scalability faced with the RPC-based model architecture [12]. Due to the inherent characteristics of the RPC model a two-step communication for assigning tasks and obtaining results is essential and the workflow engine solely takes charge of scheduling and assigning tasks. Furthermore, communication overhead is concentrated to the workflow engine. It is common for most organizations to have massive amount of workflows to process simultaneously resulting in an ever-increasing demand for better performance and scalability. Mobile agent technology can be used to overcome these limitations. The primary difference of this model to the RPC model is that the scheduling and assignment of tasks are not the sole responsibility of workflow engines. Since mobile agents may carry the whole or a part of the workflow definition they can also decide the next tasks to perform without resorting to the help of centralized workflow engines. That implies that the computational overhead is distributed among workflow engines and task performers. In addition, the communication overhead for assigning tasks is also distributed.

A mobile agent's operation is supported by an agent platform that has to be running at every host that participates to workflow execution to facilitate the agent migration. The platform chosen (Grasshopper agent platform [21]) apart from the basic support of mobile code also provides communication services that enhance coordination abilities of the runtime system.

5 Conclusion

An advanced lightweight architecture of the ARION active repository has been presented. It provides the infrastructure for a scientific knowledge service for different domains of science. The present content of the repository is data sets, simulation models statistical and visualization tools concerning wave data. The repository provides ontology based search service and workflow computational service. The user is capable of producing advanced queries and interacting with the computational service to customize his/her computations. Rich interfaces provide a very user-friendly environment.

Advanced services, which depend on scientific information, can be easily built. For instance policy making can be supported via appropriately building of workflows according to a policy-dictated scenario. Likewise decision support, which requires scientific information production according to many sources and rules, can be easily accommodated perhaps via mediation or other known information brokering techniques. The system incorporates tools for publishing descriptions of information and publishing of computational structures (workflows) to compute information. Thus, the ARION system can also be used as a tool to store and compute information on line and upon user demand for decision making and/or policy support.

References

1. C. Houstis and S. Lalis: “ARION: An Advanced Lightweight Architecture for Accessing Scientific Collections”. *RTD Information Society Technologies*, III.1.4, 2000.
2. C. Houstis, S. Lalis: "ARION: A Scalable Architecture for a Digital Library of Scientific Collections" ,8th Panhellenic Conference on Informatics, November 2001
3. S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, K. Tolle: The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases, 2nd International Workshop on the Semantic Web, WWW10 (2001)
4. RQL: A Declarative Query Language for RDF, G. Karvounarakis, S. Alexaki, V.Christophides, D. Plexousakis, Michel Scholl: The Eleventh International World Wide Web Conference (WWW2002), Honolulu, Hawaii, USA, May 7-11, 2002, pages 592-603.
5. T. R. Gruber: A Translation Approach to Portable Ontology Specifications. In: *Knowledge Acquisition*. vol. 6, no. 2, 1993. pages 199-221
6. Uschold, M., Healy, M., Willamson, K., Clark, P., & Woods, S.: Ontology reuse and application. In Guarino, N., (Ed.), *Formal Ontology in Information Systems*, 1998, pages 179–192, Trento, Italy.
7. I. Foster, C. Kesselman (eds): “The Grid: Blueprint for a New Computing Infrastructure”, Morgan Kaufmann, 1998.
8. Benjamins, R., Fensel, D. and Gomez Perez A.: Knowledge Management through Ontologies. In U. Reimer (editor), *Proceedings of the Second International Conference on Practical Aspects of Knowledge Management.* 29-30 October, 1998, Basel, Switzerland
9. H. Stormer : *A Flexible Agent-Based Workflow System*. University of Zurich
10. Crystaliz, Inc., General Magic, Inc., GMD Focus, and I. Coorp.: Mobile Agent Facility Specification. Technical report, OMG, 1997.
11. J. Ferber : Multi-Agent Systems: An Introduction to Artificial Intelligence. Addison-Wesley Publishing Company, 1999.
12. Jeong-Joon Yoo, Doheon Lee, Young-Ho Suh, Dong-Ik Lee : *Multi-Agent Systems: Scalable Workflow System Based on Mobile Agents*
13. F. Leymann and D. Roller: “Production Workflow : Concepts and Techniques”. *Prentice* Hall, 2000.
14. W.M.P. van der Aalst and A. Kumar : “XML Based Schema Definition for Support of Inter-Organizational Workflow”. *University of Colorado and University of Eindhoven report*, 2001.
15. The XML specification web page, http://www.w3.org/XML.
16. Ora Lassila and Ralph R. Swick.: Resource description framework (RDF) model and syntax specification. Technical report, W3C, 1999. W3C Recommendation. http://www.w3.org/TR/REC-rdf-syntax.
17. Dan Brickley and R.V. Guha: Resource description framework (RDF) schema specification. Technical report, W3C, 1999. W3C Proposed Recommendation. http://www.w3.org/TR/PR-rdf-schema.
18. RBAC. http://csrc.nist.gov/rbac
19. Protégé-2000, http://www.smi.stanford.edu/projects/protege/
20. OpenGIS, http://www.opengis.org
21. Grasshopper 2 Agent Platform, http://www.grasshopper.de