Scientific Information Management in Collaborative Experimentation Environments

Kaletas, E.C.

Citation for published version (APA):
Chapter 7

Conclusions

This concluding chapter provides an overall view of the research work presented in this thesis. Specifically, the chapter first presents an overview of the proposed approach in this thesis followed by summary of the thesis achievements. The evaluation and comparison of the proposed approach is provided next. Then this chapter focuses on the future work, which describes improvements to the presented work and mentions some advance collaborative extensions.

7.1 Overview of the Proposed Approach

There are a number of driving forces for research activities in supporting scientific experimentations and applications. The revolutionary technological advances in laboratory instruments and techniques that generate massive amounts of data and enable more complex experimentations can be named among the pulling forces, while the technological advances in the ICT can be named among the pushing forces.

The pulling forces certainly open the way for newer experiments, newer solution methods and techniques, and hence new results. However, the pulling forces at the same time result in highly automated but still very complex experiments, which generate large amounts of data and require advanced solutions for the management and analysis of this data.

On the other hand, although the pushing factors do already play an important role in supporting scientific experiments and applications, a concerted use of available technologies will certainly help scientists to better exploit their experimental results and obtain new insights on the problem domain.

A collaborative experimentation environment (CEE) provides such a concerted use of the available technologies, including both pulling and pushing technologies. Several issues must be considered in a complex environment like a CEE; for instance, representation and management of experiments, modelling information related to these experiments, and functionality for the management of experiments and related information. Management of scientific information can be named among the most
important issues, since it is involved in any and all steps during an experiment.

Information management in a CEE introduces several challenges, such as modelling and management of complex experimental procedures and heterogeneous data. In this context, the main objective of this thesis work was to analyze the requirements of scientific experiments from different disciplines, and to propose generic and reusable solutions for information management related problems that scientists face when performing these experiments.

In order to achieve this goal, first a comprehensive study of different experiments and applications from experimental science domains has been performed. The study included microarray and confocal microscopy experiments from life sciences, material analysis experiments from physics, and traffic simulation applications. Furthermore, several other experiments and applications have been studied from the literature, including medical applications and other physics applications. This study led to the identification of the information management related problems that scientists face during experimentation.

Then, based on the study results and the identified problems, a requirements analysis has been performed to identify and categorize the requirements that need to be fulfilled to overcome the problems of scientists. The categorization resulted in two groups of requirements, namely user requirements and base ICT requirements. The former group includes the needs and expectations of users of the CEE, while the latter group is more solution-oriented and includes the requirements for the necessary base ICT infrastructure in order to properly fulfill the user requirements.

The other step taken before designing and developing an information management platform for CEEs was a thorough study and analysis of the state-of-the-art in this field. This study was divided into two parts:

1. Study of existing and emerging technologies, paradigms, and standards that enable the development of support environments for complex experimentations (i.e. enablers). The study covered information models and standards, distributed information management systems, resource management technologies, and other related technologies, paradigms, and tools such as workflow management and virtual organizations. These areas have been chosen based on the identified requirements.

2. A comprehensive survey of several support environments for complex experimentations and their scientific information management approaches. The survey provides an overview of each support environment and presents their functionality with a specific focus on information management. Furthermore, an evaluation of the surveyed support environments has been preformed with respect to different criteria, including the identified requirements, provided functionality, applied enablers, and employed implementation technologies. The survey and the evaluation provide a snapshot of the state-of-the-art in scientific information management.

Study of scientific experiments, survey of existing and emerging scientific information management approaches, and the identified requirements form the basis of defin-
7.2 Summary of Thesis Achievements

The main achievement of the work presented in this thesis is the modelling, design, and development of an information management platform that is open, flexible, and extensible, which provides models and mechanisms for the storage and management of heterogeneous scientific information. Other achievements related to the management of scientific information in a collaborative experimentation environment are enumerated below:

1. A framework for CEE covering aspects related to experiment management, information management, and user environment in the CEE.
2. An experiment model covering different aspects of scientific experiments; namely procedures, contexts, and computational processing of experiments. The experiment model is capable of uniformly representing multi-disciplinary scientific experiments.

3. A framework for user environments, covering aspects related to provided facilities and graphical representation of experiment-related information. Uniformity and reusability are the main features of the user environment framework, which are mainly achieved through the uniformity of the experiment model.

4. Several data models representing different types of data/information generated and handled within a CEE. The major types of data/information represented by these models are experiment procedures, experiment contexts, and computational processing of experiments; user information and access rights; and information about available resources. Experimentation Environment Data Model (EEDM) is the data model designed for experiment contexts. EEDM is generic to model information related to experiments from different domains, customizable to easily modify existing types/attributes, extendible to model application specific information, and open for future extensions. Application data models extend the base constructs of EEDM for modelling domain-specific experimental information. Furthermore, EEDM supports the modelling and representation of multi-disciplinary experiments.

5. Design of the functionality required for storing and managing diverse types of information that are represented by the data models.

6. Support for object sharing and reuse, through formal definition of semantic relationships among elements in an experiment procedure. Sharing policies ensure semantic consistency of shared information.

7. A methodology for adding new applications to a CEE. Availability of such a methodology prevents the re-invention of the wheel for each domain to be supported by step-by-step guiding of the developers during the process of adding new applications. The methodology considers the approaches followed and the data models developed in this thesis.

8. Architectural design and implementation of VIMCO – Virtual Laboratory Information Management for Cooperation. VIMCO is a realization of the experiment model, data models, and functionality design. VIMCO models a wide variety of experimental information and provides mechanisms for the management of such information. It is scalable, flexible, and open for adding new users and resources; open for adding new types of information and components to support the management of these new information types; and does not depend on any third party products or solutions. The flexibility, openness, and extensibility of the proposed approach is proven by the incremental design and development of VIMCO.
9. An *application case* demonstrating the applicability of the model and its implementation to a real life experiment. The DNA micro-array application serves as both the verification/validation proof, as well as a realization of the methodology defined for adding new applications to a CEE.

### 7.3 Evaluation and Comparison of the Proposed VLAM-G/VIMCO Approach

In order to compare the proposed information management platform in this dissertation with the information management approaches of the related systems, VIMCO is evaluated using the same evaluation criteria applied to those related systems. Furthermore, the proposed platform is compared to the related systems to identify the main similarities and differences.

#### 7.3.1 Evaluation of VLAM-G/VIMCO

VIMCO is evaluated using the same evaluation criteria described in Section 3.2. Evaluation and comparison results of VIMCO can be found in the tables presented in Appendix C, together with the evaluation and comparison results of the other related systems that were studied. Similar to the other evaluated systems, the entire VLAM-G is used for the evaluation including its information management platform VIMCO, hence it will be referred to as ‘VLAM-G/VIMCO’ in this section.

The VLAM-G/VIMCO approach was classified under ‘other systems’ in Section 3.3, because it provides a broad range of functionalities which covers functionalities provided by context-oriented systems, computation-oriented systems, and metadata-oriented systems. Thus, VLAM-G/VIMCO is too extensive to be classified into any of the other groups, and hence classified as part of the ‘other systems’.

VLAM-G/VIMCO is evaluated with respect to the same criteria: identified requirements, provided functionality, applied enablers, and employed implementation technologies. Each table in Appendix C includes a row for VLAM-G/VIMCO.

#### Identified Requirements

Unlike the other evaluated systems, VLAM-G/VIMCO identifies all the requirements that are selected as evaluation criteria in Section 3.2. This is mainly because the other systems have a specific target domain to support (e.g. Cactus aims to support simulation applications in numerical relativity and PELLPACK focuses on solving partial differential equations), or because they target at only one aspect of experimentation (e.g. Gateway and UNICORE aim at providing easy to use interfaces to high-performance computing resources). For earlier systems, however, this is because certain technology was not available at the time of development (e.g. Grid technology was not available when Zoo, SoftLab and Ecce were developed).
Provided Functionality

VLAM-G/VIMCO provides all functionalities given in Tables C.4 and C.5, except replication, different execution modes (e.g. batch mode), cooperative working environments, and collaboration management. These are planned in VLAM-G as future work. Please refer to Subsection 3.2.2 for the descriptions of the functionalities considered for the evaluation.

Experiment management. VIMCO allows domain experts to define and manage templates for any type of experiments. These templates allow scientists to capture any relevant information about an experiment. Novice users create instances of these templates using workflow-like graphical user interfaces. The user interface provides context-sensitive assistance to users during template instantiation using the information embedded in the templates. Studies (template instances) are descriptions of experiments and steps in experiments, and users can create, modify, or delete studies. Studies cover the design, history of activities, and metadata for an experiment. Users can load an existing experiment description by issuing a query on the properties of overall experiments or on the properties of individual steps in an experiment. Users define their experiment’s computational processing (called topology) based on the set of available software tools (called modules) using a graphical editor. Descriptions of modules and the topologies are stored in a VIMCO database. A topology is always associated with a study. Processes included in a topology are executed on the Grid environment by the VLAM-G Run Time System.

Data/information management. Large data sets are stored in high-capacity servers as files. VLAM-G provides generic file manipulation mechanisms as modules. In addition, descriptions of experiments and experiment steps, metadata (i.e. information about the large data sets) are stored in VIMCO databases. There is one database in VIMCO for each application/domain. Users can query information stored in these databases. Because multiple databases are maintained for different VLAM-G applications, VIMCO provides mechanisms to uniformly access these databases and integrate information that is distributed among these databases. VIMCO supports managing multiple versions of experiment templates.

Resource management. Information about available storage resources (e.g. about application databases) and software resources (e.g. about software entities/modules) in VLAM-G are stored in the VIMCO databases. The VLAM-G Run Time System uses the Grid services to schedule the modules in the topology of an experiment to the available computing resources. It also allows users to monitor the execution of their topologies and steer the execution through parameter changes.

User management. Every user has an account in VLAM-G/VIMCO. Different user roles have been defined for different types of VLAM-G/VIMCO users, where every user assumes one role.
Collaboration. VLAM-G/VIMCO allows users to share hardware and software resources through the Grid, and to share information through VIMCO. Cooperative working environments (e.g. chatboxes) and application of virtual organizations are planned as future work.

Security. Grid-based authentication/authorization mechanisms are used in VLAM-G. Single sign-on is supported; users provide their username/passwords only once at log-in time, subsequently the access is provided to all VLAM-G/VIMCO resources including databases without presenting their username/passwords again. In VIMCO, the access rights to the stored information are defined for each user role and enforced at run-time.

Applied Enablers

VLAM-G/VIMCO makes use of all enablers described in Section 3.1, except virtual organizations which is planned as future work. Distributed information management mechanisms are used for managing information stored in multiple VIMCO databases. Several data models are developed for experiment and information modelling. Dublin Core standard is used in the experiment model, while using ODMG standard is planned as future work. Grid technology is used for resource management, including file management. Studies and topologies are modelled in VIMCO and presented to users through user interfaces in workflow-like presentation forms.

Employed Implementation Technologies

Different components of VLAM-G/VIMCO are implemented using different programming languages: VIMCO, Session Manager, and user interfaces are implemented in Java while Run Time System and modules are implemented in C/C++. The Matisse ODBMS is used in VIMCO. VIMCO provides several interfaces, including HTTP(S) and RMI. XML is used for data exchange among VLAM-G components.

7.3.2 Comparison of VLAM-G/VIMCO with Other Systems

This section provides a brief discussion of similarities and differences between the proposed system and some of the evaluated systems.

- **High-level definition of an experiment.** Experiment definition of OPM is similar to VIMCO experiment definition. Experiments consist of a number of steps. The steps can follow a specified order, or can be aggregate steps consisting of sub-steps. In addition, VLAM-G/VIMCO defines a higher-level 'project' that contains a set of ordered/unordered experiments. In OPM, rules/conditions for transition between experiment steps can be specified and enforced. Currently, VIMCO does not support this functionality.

- **Experiment models.** Sieve/Symphony and VLAM-G/VIMCO have similar experiment models, based on 'instantiation'. In this model, templates are provided for typical experiments, which are instantiated for description and design
of actual experimentations. These designs are then executed. Zoo also employs a similar approach that is based on the instantiation of a hierarchy of schemas (i.e. schema, meta-schema, meta-meta-schema, etc.).

- **Composing experiments.** In Gateway, Sieve/Symphony, SCIRun/BioPSE/- Uintah experiments are composed of executable components. Similarly, VLAM-G/VIMCO allows its users to compose their experiments' processing by attaching executable modules to each other. In addition, similar to Sieve/Symphony, VLAM-G performs a type checking during composition.

- **Experiment execution.** Although experiment models and definitions of Zoo, OPM and VIMCO are similar, view of these systems on experiment execution differs significantly. In Zoo, experiment execution is considered as derived relationships based on related input classes, while in VIMCO experiments are considered as data flows between executable modules that are connected to each other. OPM, on the other hand, does not support experiment executions.

- **Information about an experiment.** Sieve/Symphony, SoftLab, Zoo, Chimera and VIMCO all provide experiment databases, containing all related information about an experiment that is necessary and sufficient for the reproducibility of the experiment results.

- **Implementation approaches.** Implementation approaches of evaluated systems differ considerably. For instance, Sieve/Symphony uses JavaBeans components and SCIRun employs a stand-alone shared memory execution. However, many systems developed a modular architecture (e.g. Zoo, UNICORE, Gateway, VLAM-G/VIMCO).

### 7.4 Future Work and Advance Collaborative Extensions

Characteristics of emerging scientific experiments have been outlined in Section 1.1, and the information management requirements for managing scientific information in a collaborative experimentation environment have been described in Subsection 2.6.2. One of the major characteristics of scientific experiments, and hence one of their major requirements, is related to collaboration. Chapters 4 and 5 have focused on the modelling, design and development of an information management platform for scientific information within an experimentation environment. However, in order to propose an environment where scientists can collaborate with each other in a *controlled* manner, extensions are needed to the design and development presented in these chapters.

In this section, first the future work describing the possible improvements to the existing information management platform is discussed, which mainly consists of minor improvements but of general nature. Then two possible extensions are described
that involve the application of federated information management and virtual organizations to the existing platform to further support the collaborative aspects of scientific experiments.

7.4.1 Future Work

The following can be mentioned among the future work to improve the existing information management platform:

Supporting different formats for data exchange. The current design and implementation of the information management platform uses XML as the data exchange mechanism. Existing tools are used for the serialization of data objects into XML documents. However, an XML document generated by a specific tool complies to the XML schema specified by the tool, hence, it becomes complex and costly for other tools to understand this XML document. Furthermore, this prohibits the usage of standard XML formats defined for scientific domains. In order to overcome this problem, a highly generic XML tool is needed for the serialization of data objects into XML documents and vice versa. Given an XML schema, the tool must be capable of serializing data objects into an XML document which complies with the specified XML schema. Development of this tool may require the mapping of data types to different elements in an XML schema. Availability of such a generic tool will enable the import and export of data as XML documents that comply to the already defined standards, hence, improve the collaboration and cooperation between the CEE and other centers.

Cross-database links. Cross-database links are required to support high-level data integration. Currently, the information management platform provides data integration to a certain level (e.g. user information and project/experiment information that are distributed over multiple databases). The best approach would be the ability to define virtual integrated schemas.

Integrating Data Grid tools. As described in Subsection 3.1.3, several on-going research efforts exist in the area of Data Grid. Developments in this area, such as replica managers for large data sets, need to be integrated within the CEE.

Definition of views. Currently, the access rights are represented in the form of restrictions. A better approach, however, is the ability to define views on existing data types and specify the access rights with finer granularity on these views.

Automated linking of studies and topologies. A study may include elements that correspond to modules in a topology (e.g. image analysis step in microarray experiments). Currently, users must specify the parameter values for this step twice, first in the PFT Viewer when providing the description of this step and then in the Topology Editor when providing the information about the module representing this step. This duplication of work is difficult for the user, and may be error prone. To solve this problem, a component is needed that will automatically make the parameter values used in the PFT Viewer also
available in the Topology Editor, and allow users to map the parameter values in the study to parameters in the module.

**Customizing the experimentation environment.** Currently, experiment procedures can only be defined by experts in the domain. Although this approach is useful to assist inexperienced users, it may have a restrictive effect on the experienced users in the long term. For instance, extending the application database schemas with *customized* (user-defined) data types is needed when modelling user-specific experimental steps and results, especially during ad-hoc experimentations. Similarly, *user-defined experiment procedures* are needed to represent more specialized experiments, which will be only used by a certain group of experienced users. *Ontology management systems* can be used to assist this task. An ontology provides a dictionary for formal definitions of concepts and entities (e.g. data elements, processes, and their attributes) in a specific scientific domain, and serve as the base for achieving a common understanding of these concepts and entities for *inter-disciplinary research*. Customizing the experimentation environment is illustrated in Figure 7.1. In this figure, a customized data type (namely BioSample) is defined using the definitions in two different ontologies (namely Target from the Molecular Biology Ontology and Sample from the Material Analysis Ontology), and this data type is in turn used when defining a new, customized experiment procedure.

**Enforcement of update policies.** In the current information management platform, the necessary constructs are already available for the specification of the update policies on data objects. However, the proper means for getting update policies from users and for the enforcement of these policies is still to be designed and developed.

**Graphical and automated administration facilities.** Although it is designed and implemented by the information management platform, the user interfaces (i.e. the VLAM-G Front-End) does not support the administration functionality. Automated mechanisms would further ease the administration of the platform; for instance, automatic creation of user accounts and specification of privileges on the underlying DBMS.

### 7.4.2 Advance Collaborative Extensions

As outlined in Section 1.1, collaboration is one of the main characteristics of e-science domains and applications. Emerging scientific experiments and applications are evolving towards collaborative efforts involving several partners from different disciplines, different organizations, and different countries. With the increasing complexity and cost of scientific experiments, sharing expertise and sharing resources have become two of the most important motivations for collaboration. However, proper management of such large collaborations is still an issue.

The information management platform described in this thesis already addresses some of the main issues related to collaborative experimentation of several scientists,
when the shared information for a joint experiment is not duplicated and is entirely stored within the VLAM-G. These are summarized below:

- **Multi-disciplinary projects are supported.** Scientists from different disciplines can collaborate to perform a multi-disciplinary project, which can include experiments from these disciplines.

- **Sharing complete experiments or parts of experiments is supported.** Scientists can share experimental procedures, contextual information (i.e. scientific information about experiment steps), or processing designs. These do not only allow sharing of information, but also allow the transfer of expertise and knowledge.

- **Basic mechanisms for controlling the collaboration are supported.** Sharing policies are defined and enforced to ensure the semantic consistency (i.e. with respect to semantic object boundaries) of the shared information. Furthermore, basic access control mechanisms allow scientists to define who can access what information and at which level (e.g. complete studies, software descriptions, experiment steps).

In addition, the VLAM-G experimentation environment described in Subsection

![Diagram of Molecular Biology and Material Analysis Ontologies](image)

**New PFT definitions**

Figure 7.1: Customizing the experimentation environment
1.5.1 addresses other collaborative aspects (e.g. sharing of software and hardware resources and the Grid security credentials needed to use these resources). These functionalities form the base collaboration functionality in the VLAM-G/VIMCO. In addition to these, cooperative working environments (e.g. chatboxes) are planned as future work.

However, some of the collaboration requirements that were identified in Section 2.6 still need to be fulfilled. These requirements are integration of heterogeneous data from autonomous sources, and setting and enforcing rules and regulations for a proper collaboration among partners within the context of a virtual organization. Figure 7.2 shows the initial ideas on extending the VLAM-G architecture with two collaborative components, namely Archipel and VO Manager, which respectively apply federated information management and virtual organizations to VLAM-G to fulfill the remaining advance collaboration requirements. In the remaining of this subsection, these two components will be further described.

### 7.4.3 Archipel Federated Information Management System

A generic federated information management framework is being designed within the context of the VLAM-G project. The research and development activities within this framework cover the fundamental data management infrastructures and mech-

---

**Figure 7.2: VLAM-G architecture extended with Archipel and VO Manager**
anisms necessary to support the advanced applications of the VLAM-G. Archipel is an object-oriented information management system infrastructure, supporting the storage and manipulation of inter-related data objects distributed over multiple sites. It provides uniform access to a variety of heterogeneous and distributed information sources while preserving the autonomy of the individual information sources, through cross-institutional data sharing and exchange, (semantic) integration of diverse information from heterogeneous data resources, and management of integrated data. Please note that federated databases support access to distributed resources, where there is no centralization (or duplication) of data/control, and all data is accessed at the time it is needed directly from its source.

A preliminary design for the Archipel architecture is given in Figure 7.3. Below, brief descriptions of the main components involved in this design are presented:

**Schema Manager.** Data resources within the Archipel system are autonomous, meaning that each resource defines what part of their schema is available to which other users. Based on the agreement to create a federation, an administrator creates a schema for the federation. This schema contains mappings to the data resource schemas. *Schema Manager* provides the functionality to the administrator to view data resource schemas (component schemas), define the federated schema and create the mappings between the federated and component schemas. The mappings are formulated through a specific schema mapping.

![Figure 7.3: Preliminary design of the Archipel architecture](image-url)
and derivation language. The federated schema definitions are stored in the *Schema and View Catalogue*. The mappings are then used by other Archipel components, such as the *Federated Query Processor* and *View Manager*.

**Ontology Manager.** As introduced earlier, an ontology provides a dictionary for formal definitions of concepts and entities in a specific domain, and serves as the base for achieving a common understanding of these concepts and entities for inter-disciplinary research. Ontology plays an essential role when designing a federated schema. It provides descriptions of entities as well as the semantic relationships between these entities, hence it helps the administrator to resolve semantic and syntactic conflicts during federated schema design. The *Ontology Manager* allows the *Schema Manager* (hence the administrator) to access the definitions in the ontology. Furthermore, the *Ontology Manager* and *Schema Manager* can be used to automate the federated schema design. For instance, during the schema design, the *Schema Manager* can read the schemas of the data resources, for each entity in the schemas contact the *Ontology Manager* to obtain its relationship to other entities, and build a similarity graph based on the ontology entries. This similarity graph can be used to automatically resolve the conflicts. Any remaining conflicts can be manually resolved by the administrator.

**View Manager.** The *View Manager* allows individual users as well as administrators to define views on their part of information for other users, and associate access rights with these views. *View Manager* uses the schema definitions and mappings generated by the *Schema Manager* and stored in the *Schema and View Catalogue*. The view definitions are also stored in the same catalogue. Any access request to the federated schema is evaluated against the views defined.

**Resource Manager.** An Archipel federation may consist of multiple data resources. Information about these resources are stored and maintained by the *Resource Manager*. Such information includes, for instance, name of the data resource, communication information to the host of the data resource, and information about the drivers that can be used to access this data resource. Drivers are software libraries that actually realize the mapping between the federated data model (i.e. object-oriented) and the data model of the resource (e.g. relational) and between the federated queries (i.e. in OQL) and the query languages of the resources (e.g. SQL).

**Federated Query Processor.** The Archipel *Federated Query Processor* (FQP) receives user queries on the federated schema, identifies the resources involved, optimizes the query according to the resources and *Query Processors*. It then divides the federated query into sub-queries, assigns the sub-queries to *Query Processors* for execution, and merges the results of these queries. During query decomposition, the FQP makes use of the schema and view definitions, and the information about the resources. During query execution, it makes use of a number of query processors.
Query Processors. *Query Processor* (QP) is the Archipel component that is responsible for processing queries on a certain platform (e.g. Grid-based query processor, mobile-agent based query processor). Therefore, there is one QP for each of the supported platforms (represented as QP-A to QP-N in Figure 7.3). A QP receives a portion of the federated query and executes it on a certain platform using the resource drivers. The query executed by a QP may still span multiple data resources, thus it makes use of multiple drivers.

**Scheduler.** All user queries will be inserted into a queue. *Scheduler* constantly checks this queue, and schedules the execution of the queries based on predefined rules, such as priority assigned for each request, or quality-of-service agreed with the user. This way, reservation in advance can also be supported. Forcing each Archipel component to offer a getStatus() or getWorkload() method allows the development of a monitoring component. The *Monitoring Agent* will be used by the *Scheduler* for performance monitoring of each Archipel component.

### 7.4.4 Supporting Virtual Organizations

The necessary supporting information management infrastructure for Virtual Organizations is currently at the design stage [171] and is outside the scope of this thesis. This infrastructure will contain a catalogue and a federated framework. The catalogue will store information related to VOs; namely the VO information (id, partners, coordinator contact info, etc.) and enterprise information (business area, services and data provided, contact information, etc). The VO support infrastructure, called as VO Manager in Figure 7.3, will make use of the other VLAM-G components; for instance, it will use the Archipel for sharing data resources and controlling access to these resources, or Grid for enforcing the sharing and access policies on hardware resources.

With the existence of a VO support infrastructure, a number of organizations can join together, sharing their resources and skills towards reaching common goals. As applied to the scientific collaboration domain, VOs can assist organizations in pursuing a common goal, for instance, tight collaboration towards solving scientific problems, where the sub-tasks are distributed among different organizations and the distributed multitasking is coordinated by the VO Manager. As the base necessity in the VO, it will be possible to share (access) privileges on all kinds of resources, from hardware and software to data and information. Furthermore, this sharing and collaboration is regulated by pre-defined set of rules and policies, which are agreed upon by all collaborating partners. These agreements in form of contracts will further increase the trust among partners, and help them to advance their collaborations.