Scientific Information Management in Collaborative Experimentation Environments

Kaletas, E.C.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 2

Analysis of Requirements for Virtual Laboratories

Challenging characteristics of emerging scientific experiments and applications in e-science domains were described in Chapter 1, which include complexity and diversity of experiments, size and heterogeneity of data generated by these experiments, and need for collaboration when performing these experiments. Moreover, in Chapter 1, some of the major difficulties that scientists face when performing these experiments were outlined. Challenging characteristics of experiments and difficulties that scientists face when performing these experiments impose a number of requirements to be fulfilled. These requirements include both the needs and expectations of the scientists and other VL users (i.e. user requirements), and the requirements for the underlying base infrastructure (i.e. base ICT infrastructure requirements). In order to propose sound solutions, these requirements need to be clearly identified and analyzed.

A virtual laboratory solution was proposed in Chapter 1 as a support environment for experimentation in e-science domains. In order for a VL to help scientists tackle the challenging characteristics of experiments, it must properly address their needs and expectations. In a VL environment, however, there are different types of users that perform different activities. Consequently, each of these users has different needs and expectations from the VL environment that mainly reflect the major activities they perform within the VL. Detailed characterization of the proposed VL solution allows for the identification and characterization of both different types of the VL users and activities that they perform within the VL environment. Such a characterization leads to identification of 'user requirements'. Furthermore, a proper fulfillment of user requirements in turn puts a number of 'ICT requirements' on the necessary base VL infrastructure, which also need to be carefully identified and analyzed.

In this direction, this chapter first presents an approach for requirements analysis that is adopted in this thesis research for the identification and analysis of both user and ICT infrastructure requirements. Following this approach, a detailed characterization of the proposed virtual laboratory solution is realized, and a use-case analysis is
performed to identify and characterize main user activities in VL, as described next in this chapter. The remaining two sections outline and describe the user requirements and the ICT infrastructure requirements, respectively. The latter section further focuses on the information management related aspects and provides an analysis of the information management requirements.

2.1 Approach for Requirements Analysis

A systematic approach is adopted for the identification and analysis of both user requirements and base ICT infrastructure requirements. This approach consists of four steps, as described below:

1. **VL characterization.** Characterize the proposed VL solution at a level of detail that is necessary for identifying its constituents (e.g. users of the VL, data/information handled in the VL, etc.) and their characteristics. Careful characterization of these constituents leads to the identification of requirements (e.g. user requirements, requirements for managing the data/information, etc.).

2. **“Use case” analysis.** Focus on different actors of the VL environment, which were identified during the characterization of the VL, and identify and characterize main activities that each of these actors performs within the VL.

3. **Analysis of the user requirements.** Based on the use case analysis, identify and analyze the requirements of VL users for each of their use cases and for activities involved in these use cases.

4. **Analysis of the base ICT infrastructure requirements.** Identify and analyze the requirements for the necessary base ICT infrastructure in order to properly fulfill the user requirements. Identification and analysis of the base ICT infrastructure requirements constitute a step towards providing a solution to specified user demands.

Steps involved in this approach are further described in the remaining sections of this chapter.

2.2 Virtual Laboratory Characterization

In this section, characteristics of a virtual laboratory are provided. This characterization is made by distinguishing the major constituents of a virtual laboratory; namely *experiments, users, data, functionality, and infrastructure*. In the remaining of this section, different aspects of these constituents are identified and described, which need to be carefully considered and supported in a virtual laboratory.
Experiment Characterization

The main characteristics of several experiments and applications from different e-science domains were provided in Subsection 1.1.1. Here, the focus will be on the characterization of the life-cycle of a typical e-science experiment.

The life cycle of a scientific experiment consists of three ‘recursive’ phases, which are depicted in Figure 2.1 and described below:

- **Design phase.** During this phase, the aim of the experiment is usually formulated as a question, and the methodology to answer this question is mapped to an experiment design. The question can be general and high-level such as profiling the expressions of all genes in an organism under excessive nutrition, or very specific such as calculation of the energy barrier for electron transfer of a compound by measuring the emitting life-time from room temperature to low temperature. The experimental designs addressing the former type of questions generally have an ad-hoc nature and are generally driven by ‘what-if’ questions. The designs addressing the latter type of questions are generally better understood and sometimes already fixed.

Input for the experiment design process includes the objective of the experiment, available information about the experiment subject gathered from various resources, knowledge in the laboratory about the subject, and a survey of possible (instrumental) techniques that can be used to achieve the experiment objectives. Then the experimental procedure is designed considering all these input.

At this stage, knowledge and expertise of the scientist play an important role. The scientist must be able to choose the correct procedure to follow and the protocols to apply. Furthermore, considering that experiment definitions are very complex and future references to experiment definitions are highly possible, archiving becomes also very important.

- **Execution phase.** In this phase, the experimental procedure designed in the first phase is executed. The procedure may include laboratory activities (e.g.
sample treatment), using an instrument (e.g. sample scanning), or data gathering (e.g. images generated by the scanner). Laboratory activities can be performed off-line by a technician, or automated. Furthermore, steps in a procedure may also correspond to simulated activities. All these activities are performed based on the parameters, conditions, etc. as designed in the procedure.

The procedure designed in the design phase constitutes the input for the execution phase. The output is data generated by instruments (e.g. images obtained by scanning the sample) or by simulations and software tools (e.g. numerical values obtained by quantifying the image). Note that some experiments may produce a physical entity as output (e.g. a sample covered with a protective layer). In this case, either a new experiment is designed to further work on the physical entity (e.g. polishing the protective layer on the sample) or the execution phase continues with other activities that may generate data as output.

The important points in this phase are the access to and operation of various instruments, using several hardware and software tools, and the organization of the experimental information.

- **Result analysis phase.** During this last phase of experiments, the data generated by the experiments are analyzed and interpreted by scientists. Several analysis tools can be used during this phase, for instance, visualizing the results in different ways. The interpretation of results involves correlating results of different experiments to each other to make a comparative analysis.

The input to this phase is the data generated in the execution phase, and the output is the information obtained after the analysis. This information can contain, for instance, the characteristics of the sample under certain conditions, or the verification of a previously observed behavior of a biological specimen under certain circumstances.

The analysis phase is of ad-hoc nature and intuitive. Learning from the analysis results, the scientist may decide to use different analysis methods/algorithms on the same experiment results, or may decide to make a new experiment.

The recursive nature of the experiment life-cycle comes from the last point. Scientists may recursively continue with different analyses or with design and analysis of a new experiment.

**User Characterization**

In order to characterize a system, it is useful to identify its different types of users and the typical activities that they perform with the system. Note that these are the ‘users’ of the virtual laboratory; developers of the base VL infrastructure are not considered as users of the system, hence are not characterized here.

The four target user groups distinguished for the VL include the following (see Figure 2.2):
- **Scientists.** Scientists are the actual users of the VL. A scientist is typically associated with an e-science domain (e.g. molecular biology).

Scientists need (graphical) user interfaces for data entry and retrieval. These interfaces must provide a high-level representation of available resources to scientists in order to hide the details of these resources from them. Furthermore, scientists may require assistance, for instance, for using an instrument with a valid set of parameters. Inexperienced scientists typically follow a pre-defined procedure when making their experiments, while experienced scientists can also define new, customized procedures.

- **Domain experts.** Domain experts are scientists who have extensive knowledge and experience on a given e-science domain and on the experiments being performed in that domain. They are responsible, for instance, for the modelling of the data resources used to store experimental information, designing experiment procedures, defining protocols to be used for certain activities in the laboratory, and defining the parameters to be used for certain hardware and software.

In order to perform these tasks, domain experts need interfaces for editing the working environment for the domain, such as the design of experimental procedures, modelling the necessary information and metadata, etc.

- **Tool developers.** Another type of user for the VL constitutes the developers of support tools. A number of tools are used in a VL environment, from tools for data analysis to tools for controlling laboratory equipment.

A repository is needed for the storage of the binary codes of the tools. Furthermore, a developer requires interfaces for entering the description of the tool that s/he develops, e.g. its function, run-time requirements etc., so that scientists can choose the correct tool to use when executing an experiment or when analyzing the experiment results.

- **Administrators.** Administrators are responsible for the tasks related to the proper management and operation of the VL, such as resource management, infrastructure maintenance, and user management.

![Diagram](image_url)  
*Figure 2.2: Different types of VL users*
They require interfaces and monitoring tools for the administration of the VL environment and infrastructure.

**Data Characterization**

In VL, a wide range and variety of data/information are handled in scientific experiments within different e-science domains, which need to be properly managed. Important aspects that need to be considered for the management of data/information are outlined below:

- **Size.** As mentioned earlier, one of the most common characteristics of e-science experiments and applications is the large amounts of data that they generate. However, the composition of generated results differs for each experiment. For instance, a microarray experiment measures the expression levels of thousands of genes organized on multiple glass-slides (microarrays), where intensity, background intensity, error, p-value, confidence level etc. are quantified and/or calculated for each gene. A material analysis experiment, on the other hand, measures/records the FTIR spectra of the sample cross-sections, generating a typical FTIR data cube of 4096 spectra (i.e. 64 X 64 images). The total data size for a single microarray experiment is in the order of megabytes, while this value is in the order of hundreds of megabytes for a single material analysis experiment. However, microarray experiments are performed more frequently than material analysis experiments (e.g. time-series, comparison, profiling microarray experiments).

- **Storage.** Size of the data, its structure, and how it is used are the main factors that have an influence on the way that this data is stored. Large amounts of data are generally stored on disks as flat files. In some cases, the files are of a specific format. For example, in the case of DNA sequences in molecular biology, text files are formatted as key-value pairs, where the keys are defined as tags. However, there are also cases where the data is stored in a DBMS. This is the case, for instance, for microarray data, where the usage of a DBMS is both feasible with respect to data size and necessary with respect to the complexity of queries issued against the data. Furthermore, data with a structure is generally stored in a DBMS while unstructured (or semi-structured) data is stored in files.

- **Manipulation.** During its life-cycle, scientific data/information is created, updated, and queried. In a research organization, scientific data is either generated in a step-wise manner over time (e.g. for describing experiment steps) or generated by an instrument at once (e.g. LHC experiments). Access to scientific data can be on-demand basis for only a single element (e.g. to retrieve the descriptions of all treatments applied to a given sample), in the form of aggregate queries (e.g. to see the average intensity of a gene through a time-series experiment), or as data scanning (e.g. extracting the spectrum of a sample at certain coordinates). SQL and similar languages are the main tools used for accessing and manipulating data stored in DBMSs (e.g. for single element or aggregate
queries). On the other hand, specific file manipulation tools are used for data sets stored in files. For instance, NetCDF library [74] is used to manipulate n-dimensional data cubes stored as files.

- **Modelling.** A variety of information needs to be modelled in an experiment, including the experimental procedure description, resulting data, and annotations/metadata. Today, every organization models the experimental information in a different way. Typically, the model is developed based on the current types of data generated in the organization, with minimum effort (quick-and-dirty approach). Neither the possible future extensions nor the compatibility with the data models used by other (international) organizations are considered. Furthermore, in many cases, no standards have been developed for the generated data. For microarray data for example, although a standard is currently being developed for defining the required minimum information about an experiment, this effort does not define any standards for the modelling of this information. Hence, while still complying to this standard, several microarray databases use different data models for their microarray data.

- **Heterogeneity.** Scientific data is heterogeneous by nature. Among different types of heterogeneity, one can mention model/paradigm heterogeneity (i.e. relational models, object-models, XML-based models, hierarchical models, etc.), data definition and/or manipulation language heterogeneity (i.e. SQL, OQL, XQL, etc.), semantic heterogeneity (i.e. different meanings can be assigned to the same entity), and system heterogeneity (i.e. different DBMSs, etc.).

- **Interoperability.** In order to support sharing and exchange of data among collaborating centers, data manipulation mechanisms and data models must support interoperability. In line with the different types of heterogeneity, interoperability must address syntactic interoperability (e.g. defining standard formats for data such as XML), semantic interoperability (e.g. supporting a common understanding of entities through ontologies), and system interoperability (e.g. through deploying middleware technology such as CORBA, Grid, JDBC).

**Required Functionality Characterization**

Following outlines the main functionality required from a virtual laboratory:

- **Experiment management.** Scientists must be supported during all three phases of an experiment. It must be possible to manage an experiment as an atomic unit. This in turn requires a clear definition of an experiment, as well as definition of the necessary models (i.e. data model, functionality model, presentation model, usage model). Considering these, the VL must develop the necessary experiment management functionality, such as storage, retrieval and modification of experiment designs; discovering, allocating, and managing the resources needed for experiment execution and analysis; and managing the results of analysis.
• **Data/information management.** Mechanisms must be provided for manipulation of the wide variety of information handled in the VL. These include mechanisms for storage, querying, retrieval, and modification of experimental procedures, results, and annotations/metadata.

• **Resource management.** Several resources are needed and used during scientific experiments in VL, such as computing, storage and networking resources, instruments, data/information, and software. VL must manage all these resources in an *efficient and coordinated* manner, in order to achieve utmost fulfillment of user requirements.

• **User management.** In order to properly address the requirements of different types of users, VL must support definition of different user roles, assignment of roles to users, definition of access rights for roles, and dynamic manipulation of users, roles, and their access rights.

• **Security provision.** VL must provide the necessary mechanisms for the authentication of users and authorization of their requests. Security must be provided at different levels, for instance authorization rules can be defined for accessing a hardware resource (e.g. logging-on to a server), accessing to a data resource (e.g. reading a file on the server), or performing an operation (e.g. executing software programs on the server). Furthermore, security on the communication channels must be addressed.

• **Collaboration support.** VL must support the collaborative activities among distributed scientists. Such activities include sharing of resources (e.g. hardware, software, data/information resources), knowledge and experience sharing, and cooperative work among remote users (e.g. simultaneously visualizing a data set, video-conferencing, etc.).

### Required Infrastructure Characterization

Characteristics of e-science experiments, data generated and handled within these experiments, and the required functionality imposes a set of requirements on the supporting infrastructure. Following is a characterization of the 'minimum' required infrastructure within the VL.

• **Computing facilities.** For the analysis of large data sets, high-performance computing infrastructure is needed. The trend has been moving from supercomputers to clusters of personal computers, which is currently moving towards virtual clusters. The VL environment must provide the computational capacity required by its applications.

• **Networking facilities.** In line with the computational trend, networking is becoming more important. Today, computing facilities are geographically distributed, and data and/or processes are transferred among distributed nodes. This requires high-bandwith, high-speed networks. Furthermore, as described
earlier, cooperative work (e.g. interactive simulation performed at a number of sites using CAVE), transfer of data from a source (e.g. instrument) to a storage facility (e.g. archive), or from a storage facility to high-throughput analysis software requires high-speed network connections.

- **Software environment.** The required software environment must be open for adding new resources, scalable to work with increasing number of users and workload, and easy to customize. It must allow the development of specific software tools for experiments, such as analysis tools, instrument control packages, simulation software, etc.

### 2.3 Use Case Analysis

Use case modelling is a technique used to describe what a system should do. Following are the primary purposes for use cases, as defined in [75]:

- to decide and describe the functional requirements of the system in agreement with the users,

- to give a clear description of what the system should do to provide the basis for further design modelling that delivers the required functionality, and

- to provide a basis for performing system tests.

The primary components of a use case model are use cases, actors, and the system modelled [75]. An *actor* is a person that interacts with the system. *Use cases* correspond to the main activities that an actor performs when interacting with the system. The *system* here corresponds to the virtual laboratory.

Users constitute the major actors in a system. Four different types of VL users were identified in Section 2.2, namely *scientists, domain experts, tool developers,* and *administrators.* Another 'actor' of a VL environment is the **ICT developer.** ICT developers are not users of the virtual laboratory, rather they are developers of the base infrastructure. ICT developers contribute to the development of different functionalities of the base VL infrastructure, e.g. for storage of data or for monitoring the experiment execution.

In the remaining of this section, use cases identified for different VL actors are described. Note that the use cases presented in this section are high-level, mainly because e-science experiments are of ad-hoc and intuitive nature (especially the results analysis phase), where scientists may follow different routes and perform different activities within a use case. In the case of use case analysis for scientists, such alternative activities are also included in the descriptions of the use cases. Also note that all figures in this section are UML diagrams, hence they follow the standard UML notation.
2.3.1 Scientists Use Cases

The main interaction of scientists with the VL is to perform experiments. Three use cases are identified for scientists: design experiments, execute experiments, and analyze experiment results (Figure 2.3). These use cases match with different phases of experiment life cycle, as described in Section 2.2. These three use cases are further detailed below.

‘Design Experiments’ Use Case

In this use case, aim of the experiment is formulated as a question, and the methodology to answer this question is mapped to an experiment design. Typical activities involved in this use case are depicted in Figure 2.4 and enumerated below:

- *Formulate the experiment question.* The objective of the experiment is stated.

- *Search for an experiment template suitable for this objective.* A template may have already been designed that addresses the same or a similar question.

- *Design experiment by instantiating the template with actual input, protocols, conditions, and parameters.* If such a template is available, it must be instantiated by replacing the variables in the template with the actual values for the parameters, conditions, and perhaps even the protocols. At this stage, the experiment design is ready.

- *Search database for existing similar experiment designs.* If no such template exists, experiment database can be searched for experiments with the same or similar objective.

- *Retrieve the template.* If a similar experiment is found, the template used for that experiment can be re-used. This way, scientists may discover alternative procedures/experimental techniques to address the same experiment question.

![Figure 2.3: Use cases for scientists](image-url)
Figure 2.4: Activities involved in ‘design experiments’ use case for scientists

- **Access diverse data sources (for experiment, instrumentation, and analysis techniques, conditions, protocols, and parameters).** If no similar experiments are available, then the scientist can access multiple resources for designing a new experiment from scratch. For this purpose, several data sources can be accessed that contain information about available experimentation, instrumentation, or analysis techniques, as well as information about some best-practices (e.g. best set of parameters to use a certain instrument). This will lead to a new design.

- **Save experiment design.** The experiment design will be saved in the experiment database for future references.
Note that these activities may involve collaboration with other scientists/organizations; for example to share experiment templates or experiment designs.

‘Execute Experiments’ Use Case

Execute experiments use case corresponds to the execution of the experiment design. All activities in this use case are performed by following the protocols defined, parameters and conditions specified in the experiment design. Typical activities involved in this use case are described below (see also Figure 2.5):

- **Use an instrument/run a simulation (following the experiment design).** The main activity in this use case is to acquire data, either from an instrument or from a simulation.

- **Save any specific run-time info.** If any specific run-time variable is set during the operation of the instrument or execution of the simulation, which is not specified in the experiment design, these are saved for future references. An example of such variables is a machine-specific environment variable for the computer on which the simulation is running. Similar to parameters and conditions, such information is important for the reproducibility of the experiment.
2.3. Use Case Analysis

Figure 2.6: Activities involved in ‘analyze experiment results’ use case for scientists

- **Store raw data.** The data generated by the instrument or simulation is saved to a storage system for further analysis.

- **Perform a laboratory activity (following the experiment design).** Alternatively, the scientist may perform a laboratory activity, such as crystalizing a compound for further analysis in chemistry.

Note here again that the execution of experiments may require sharing of resources among collaborating partners. For instance, a unique instrument, expensive simulation software, or even entire laboratory facilities can be shared.

‘Analyze Experiment Results’ Use Case

In this use case, scientists process the raw data that they have generated, analyze it, interpret the results, and save their conclusions. Activities in this use case have an ad-hoc and intuitive nature, and may include recursive actions. Typical activities involved in this use case are depicted in Figure 2.6 and outlined below:

- **Process raw data.** In some cases, raw data generated by an instrument or a simulation needs to be pre-processed (e.g. fourier-transforming raw data generated by an FTIR spectrometer) to prepare it for further analysis. Usually special processing software is used for the processing, depending on the type of the data. Such processing is generally well-defined and applied to all acquired data of the same type.

- **Analyze processed data.** During this activity, processed data is analyzed by using several different analysis tools, being mainly visualization tools. Examples are visualization of spectra images for material analysis samples, or cluster diagrams of gene expression data.
- \textit{Interpret results.} At this stage, results are analyzed in different ways, and possibly correlated with results obtained in earlier experiments. The results might be sufficient to properly answer the experiment question; otherwise it might be necessary to make further analysis using different techniques/methods, or even new experiments might be needed.

- \textit{Save metadata.} The conclusions drawn from the interpretation are saved in the database.

Similar to the previous use cases, analysis of experiment results may involve collaborative activities. For example, analysis results can be correlated with results from other experiments made by collaborators, or findings about an image can be discussed with partners using real-time video collaboration, etc.

2.3.2 Domain Experts Use Cases

Domain experts are scientists that have extensive knowledge and experience in the domain that they work. In this sense, besides making experiments, domain experts deal with higher-level activities as well, such as designing experiment templates for novice scientists or modelling information handled in these experiments. The use cases identified for domain experts are given in Figure 2.7, which are further detailed below.

'\textit{Model Domain-Specific Information}' Use Case

As they possess extensive knowledge in their scientific domains, domain experts may help administrators to model information handled within a certain type of experiment, or even model the information themselves. One step further is designing experiment database schemas, which contains the type definitions for experimental information.

![Figure 2.7: Use cases for domain experts](image-url)
'Define Protocols' Use Case

In an experiment, several steps are performed according to certain protocols, which describe for instance chemical reactions to be done, conditions to be set, amounts of material to be used, etc. For instance, the hybridization protocol used in microarray experiments define molarity of the solution used, temperature for reactions, and directions to actually do the reactions. Protocols are generally standardized by experts with the optimal setting of different parameters, and applied to all laboratory activities of the same type.

'Design Experiment Templates/Procedures' Use Case

While the protocols define what to do and how to do in a single step in an experiment, templates/procedures define how to make an experiment to address a specific question. Domain experts define the required activities to perform a specific type of experiment, their order, protocols to be used for the activities, and an optimal setting for parameters. These templates are then used by novice scientists as guidelines during experiment design.

2.3.3 Tool Developers Use Cases

As can be seen from the use cases for scientists and domain experts, a wide variety of software tools are used during experimentation, ranging from data processing tools to analysis/visualization tools. Although some of these tools can be generic (e.g. visualization of 3D images), they are generally specific to a domain (e.g. clustering tools for gene expression data). Developing a software tool specific for a domain requires certain knowledge in that domain. At the same time, tool developers must have a certain level of knowledge on the base infrastructure for which the tool is being developed. Hence, tool developers work in close cooperation both with scientists and domain experts and with the ICT developers. Use cases identified for tool developers are presented in Figure 2.8, and described below.

![Figure 2.8: Use cases for tool developers](image-url)
‘Develop Software Tools’ Use Case

Software tool development for e-science domains involves the well-known base activities in software development: Design, implementation, testing. However, one important aspect to consider here is interoperability. The developed tools will be running as part of a VL infrastructure. Furthermore, these tools will be possibly used in combination with other tools available in the VL. Hence, a developed tool must be interoperable both with components of the VL infrastructure and with other tools it may be used with. Activities involved in this use case are described below.

- **Design.** One of the main ideas behind the VL is reusability. Hence, functionality and architecture of a tool must be designed as generic as possible to allow its reuse. Furthermore, design of the tool must consider programming interfaces and user interfaces of the VL, data sources it uses, and other software tools that are using this tool or being used by this tool.

- **Implementation.** This activity must consider the technologies used or supported by the VL infrastructure. Furthermore, any functionality required by the VL must be incorporated into the tool (e.g. implementation of a VL interface for monitoring).

- **Testing.** The developed tool must be tested in the VL environment. This requires the existence of a test environment within the VL, which allows interaction of the tool with the base VL infrastructure and with the resources available through the VL.

‘Deploy Software Tools’ Use Case

When the tool is implemented and tested, it must be deployed in the VL environment. At this stage, executable binaries for the software tool are made available to users through a software repository, and documentation about the tool is provided along with the binaries. Documentation includes textual descriptions about the purpose of the tool (what it does), algorithm implemented by it (how it does), and run-time requirements (hardware requirements, system requirements, environment requirements, etc). This information is necessary for a user to choose the most suitable tool for her/his needs among the available tools.

2.3.4 Administrators Use Cases

Administrators are responsible for maintaining the proper operation of the VL environment. For this purpose, they regularly monitor usage and status of the resources that are available through the VL, and take the necessary actions to maintain their availability and performance. Another responsibility of administrators is the management of VL user accounts. The main use cases for administrators are described below (see also Figure 2.9).
‘Monitor/Manage Resources’ Use Case

In a VL environment, several resources are provided to users, including hardware resources (e.g. instruments, computer clusters, storage systems), software resources (e.g. analysis tools, base tools like file management tools, etc.), and data/information resources (e.g. databases for application-specific information).

In VL, resource monitoring and management is the responsibility of administrators. An administrator constantly monitors the available resources to meet certain criteria like maintaining high availability or high performance. Hence, an administrator may have to take a resource out of the VL environment, add a new resource, or re-start/re-initialize a resource during the operation of VL to meet such criteria.

‘Manage User Accounts’ Use Case

User account management is one of the requirements to support authorized access to VL resources. Account management involves creating/deleting a user account, and assigning/revoking a role to/from a user account.

It is important to note here that managing user accounts does not include access rights management. Access rights for a resource are assumed to be defined by the owner of the resource. For instance, a scientist defines who can do what with her/his scientific data, while a domain expert defines who can use the experiment templates that s/he designed. Hence, access rights management functionality is required by all VL users.

2.3.5 ICT Developers Use Cases

As mentioned earlier, ICT developers are not users of the virtual laboratory, rather they are developers of the base VL infrastructure. ICT developers contribute to the development of base VL functionalities, e.g. file manipulation, user authentication and authorization, or monitoring experiment executions. One use case is identified for ICT developers: develop base infrastructure functionality (see Figure 2.10).

Similar to all software development processes, this use case includes design, implementation, and testing activities. As in the case of developing a software tool for VL, ICT developers must consider genericness and reusability of the functionality or

![Figure 2.9: Use cases for administrators](image-url)
components that they develop. Furthermore, the design of the functionality must be compatible with the overall design philosophy of the base VL infrastructure.

2.4 Classification of Requirements

Five different actors of VL were identified during the use case analysis and described in the previous section. First four of these actors are users of the VL (namely scientists, domain experts, tool developers and administrators) while the last actor (namely ICT developer) is the developer of the base VL infrastructure.

Figure 2.11 depicts the interaction among the actors of the VL with respect to their requirements from each other. To further clarify the figure, consider the following example: A scientist needs a virtual reality environment to visualize her/his medical simulation results, and the required virtual reality environment uses a special visualization software. This need of the scientist puts a number of requirements on other actors in the form of a chain, where each actor requires support or availability of certain features from another. Support/features required from each actor in this example are as follows:

- **From the domain expert.** Order the virtual reality environment and its special visualization software, and design an experiment template to use the environment;

- **From the tool developer.** Develop the special visualization software for the virtual reality environment and deploy it as a VL tool;

- **From the administrator.** Register the virtual reality environment as a VL resource;

- **From the ICT developer.** Acquire the virtual reality environment, set up the necessary computing and networking facilities, develop the required high-performance data transfer functionality.

Note that this list provides the required support/features from each actor, but not the order of actual occurrence of these support activities. For instance, the two

![Diagram](image_url)

**Figure 2.10:** Use case for ICT developers
required support activities from the domain expert are to order the virtual reality environment and its software, and to design a template for using it. In this case, the domain expert first orders the environment, and only when it is available, s/he designs the template.

Based on the use case analysis and considering the interaction among VL actors with respect to their requirements from each other, analysis performed in this chapter classifies all these requirements into two categories, namely user requirements and base ICT infrastructure requirements (see Figure 2.12). The former group is further classified into four groups corresponding to the different types of VL users, namely scientists’ requirements, domain experts’ requirements, tool developers’ requirements, and administrators’ requirements. The latter group is further classified into two, namely general VL requirements and information management requirements.

As mentioned earlier, identification and analysis of the base ICT infrastructure requirements constitute a first step towards providing a solution to user requirements. Therefore, in Section 2.7, base ICT requirements are matched to user requirements, to indicate the required solution for each user requirement. In Figure 2.12, this matching is represented with block arrows from user requirements to base ICT requirements. In order to present the matching between base ICT requirements and user requirements in the final section of this chapter, first letters of each requirements category in Figure 2.12 are printed in bold fonts. The 'requirements' in the following sections are enu-
Chapter 2. Analysis of Requirements for Virtual Laboratories

2.5 Analysis of User Requirements

In this section, requirements analysis for different VL users is presented. Requirements analysis is based on the use case analysis, in which the main activities that different types of VL users perform in the VL were described.

The requirements analysis presented in this section aims to answer the following question:

*What do the different VL users require to properly perform their activities within the VL?*

Remaining of this section presents the requirements of each of the VL users.

2.5.1 Scientists Requirements

The list of scientists' requirements, as enumerated below, are grouped based on the use cases as described earlier, followed by a list of general requirements. Latter are the scientists' requirements that are not limited to a use case but valid for all activities of scientists in a VL environment.
Requirements for Designing Experiments

Following are the identified requirements of a scientist when designing experiments:

S.1 Ability to clearly and sufficiently formulate the experiment objective/question in an experiment design

S.2 Availability of a repository (or repositories), which contain(s) templates for the most common types of experiments, designs for earlier experiments, most common techniques and protocols used during experiments, etc.

S.3 Ability to search this repository (or repositories), based on keywords

S.4 Ability to compare experiment designs

S.5 Ability to design a diversity of experiments with any required detail, with varying length and complexity, with different experiment flows (e.g. optional steps, alternative paths, etc.)

S.6 Ability to uniformly represent and manage these different types of experiment designs

S.7 Ability to store, query, retrieve, modify, delete experiment designs

Requirements for Executing Experiments

Requirements of a scientist during experiment execution are presented below:

S.8 Ability to execute experiments with varying characteristics (length, complexity, flow, etc.)

S.9 Ability to (remotely) control/operate an instrument or run a simulation using the conditions/parameters specified in the experiment design; retrieve the measurement data (raw data) from the instrument or from the simulation process; transport the data to a storage facility

S.10 Availability of sufficient network bandwidth for transferring measurement data from an instrument/simulation to a storage system; sufficient storage space for storing the measurement data; sufficient computing power for processing and analyzing the measurement data

S.11 Availability of mechanisms for accessing and managing data, such as for reading, saving, moving, deleting files

S.12 Availability of mechanisms for ensuring the high availability of data (e.g. back-ups, replication)
Requirements for Analyzing Experiment Results

Identified requirements of scientists during analysis of experiment results are as follows:

S.13 Flexibility in the analysis phase; support for ad-hoc and intuitive nature of analysis

S.14 Ability to store, retrieve, modify analytical strategy/process descriptions

S.15 Availability of both generic and most commonly used problem solvers; availability of descriptive information and help about these problem solvers

S.16 Availability of high-performance infrastructure for the processing/analysis of data; ability to discover, allocate, and (transparently) use resources in this infrastructure required for the data processing and analysis (e.g. computing units, networks, storage systems, visualization devices, analysis/processing tools, etc.)

S.17 Ability to submit a background process or to schedule a process for a specified time

S.18 Ability to process/analyze data using a combination of tools and techniques; ability to monitor data processing/analysis

S.19 Ability to correlate and compare different data sets (possibly obtained from different types of experiments, using different analytical techniques)

S.20 Ability to access/query information in files or in DBMSs; ability to access a diverse set of heterogeneous information resources; ability to integrate information from multiple information resources

S.21 Ability to capture and record all observations, inferences, and conclusions drawn during processing/analysis of data sets (i.e. annotations and metadata), which can be structured or unstructured

General Requirements

In addition to the requirements of a scientist for a specific use case, some requirements are more general and apply to all activities of a scientist. These general requirements are enumerated below:

S.22 Easy management of information about experiments, at the level of an entire experiment or parts of an experiment

S.23 Ability to store multiple versions of experimental information

S.24 Availability of graphical, easy and convenient to use, uniform interfaces for designing experiments, searching and instantiating experiment templates, entering or querying data; availability of flexible user interfaces that are easy to customize to personal preferences
S.25 Availability of assistance/help at any stage during an experiment, for instance availability of descriptions, guides for usage, possible values for variables, alternatives, etc.

S.26 Ability to share and exchange data and information, instruments, or other resources with collaborating partners in a secure environment

S.27 Availability of various technologies for cooperative work, such as video-conferencing or display sharing

S.28 Ability to define access rights on the resources owned by the scientist

2.5.2 Domain Experts Requirements

List of identified requirements of domain experts is provided in this subsection. Since domain experts are scientists with extensive knowledge and experience in their domains, their requirements are in addition to those of scientists. Following are the requirements of domain experts, that are grouped based on the use cases. Similar to scientists, a list of general requirements is also provided.

Requirements for Modelling Domain Specific Information

Following are the requirements of domain experts when modelling domain specific information:

D.1 Administrative access to databases

D.2 Availability of a methodology for modelling highly heterogeneous experimental data

D.3 Availability of easy to use, easy to understand, evolvable base schemas that enable modelling domain-specific information and developing schemas that are compatible with the (data modelling and management) philosophy adopted by the VL

D.4 In case of a need for accessing multiple heterogeneous data sources, availability of information about the syntax and semantics of objects deployed by other data sources that need to be accessed

Requirements for Defining Protocols

Requirements of domain experts when defining protocols are enumerated below:

D.5 Availability of a repository for the most common techniques, protocols, etc. that are used in different types of experiments

D.6 Ability to model standardized activities as protocols and to store them in the repository

D.7 Ability to define and enforce usage rules for techniques/protocols, for instance by specifying the range of valid parameter values
Requirements for Designing Experiment Templates/Procedures
Requirements of domain experts when designing experiment templates/procedures are as follows:

D.8 Availability of a template repository for the most common types of experiments

D.9 Ability to model standardized experiment designs as templates and to store them in the repository

General Requirements
Following are the general requirements that apply to all activities of a domain expert:

D.10 Availability of graphical, easy to use, customizable user interfaces for data modelling, schema design, technique/protocol modelling, experiment template design, and for data entry and retrieval

D.11 Availability of mechanisms and graphical interfaces to define access rights on the data structures, protocols/techniques, and experiment templates

2.5.3 Tool Developers Requirements
Following is a list of requirements of tool developers during the two main activities that they perform in a VL environment, namely developing software tools and deploying these software tools.

Requirements for Developing Software Tools

T.1 Availability of a design philosophy adopted by the VL for software tools; availability of a methodology for developing software tools

T.2 Availability of well-documented and easy to understand VL APIs

T.3 Availability of an environment for testing the tools

Requirements for Deploying Software Tools

T.4 Availability of a repository for storing the binary code

T.5 Availability of mechanisms/interfaces for deploying the software tool

T.6 Availability of graphical user interfaces for entering the tool description and providing the documentation

2.5.4 Administrators Requirements
Similar to the other VL users, requirements of administrators correspond to their use cases, as presented below:
Requirements for Monitoring/Managing Resources

A.1 Ability to easily add new resources when needed; ability to stop, re-start, disconnect a resource when needed; ability to enable/disable access to resources for a specific user or for a group of users when needed

A.2 Availability of monitoring tools for the resources, for instance to read real-time status information

A.3 Availability of tools and graphical interfaces for easy administration of resources

Requirements for Managing User Accounts

A.4 Ability to add, modify, remove user accounts; ability to define, modify, delete roles; ability to assign roles to users and to revoke roles from users

A.5 Availability of graphical interfaces for managing user accounts

2.6 Analysis of Base ICT Infrastructure Requirements

In the previous section, needs and expectations of different VL users when performing their activities in the VL were presented (i.e. user requirements). A proper fulfillment of these user requirements in turn puts a number of requirements on the base ICT infrastructure for the VL. Requirements analysis for the base ICT infrastructure presented in this section aims to answer the following question:

*What functionality and facilities must be provided by the base ICT infrastructure to properly fulfill requirements of different VL users?*

In this section, requirements for base ICT infrastructure for VL are addressed in two groups: The first group of requirements includes the general requirements that apply to the overall VL infrastructure, while the second group focuses on the information management requirements. Following two subsections present the analysis of base ICT infrastructure requirements corresponding to these two groups.

2.6.1 General VL Requirements

Analysis of general VL requirements is performed by classifying them into the following categories:

1. Infrastructure requirements
2. Interface requirements
3. Functionality requirements
4. Architectural/technological implementation requirements
In the remaining of this subsection, general VL requirements are presented based on this classification.

**Infrastructure Requirements**

The main infrastructure requirement for a VL is the availability of a high-performance infrastructure for the transfer, storage, and processing/analysis of large experimental data sets. In specific, following can be mentioned among the infrastructure requirements:

**Requirements for storage facilities.** In order to store large data sets generated by scientific experiments, VL must provide storage facilities, that:

- **G.1** have sufficiently large storage space available at all times;
- **G.2** ensure high availability of data, e.g. through backup or replication;
- **G.3** provide high-performance mechanisms for the storage of real-time acquisition of data, and to support high-throughput/high-performance data analysis tools.

**Requirements for computing facilities.** For the analysis of large data sets, high-performance computing infrastructure is needed. The trend has been moving from supercomputers to clusters of personal computers, which is currently moving towards virtual clusters. Thus, the VL computing infrastructure must:

- **G.4** support different platforms, ranging from workstations to PC clusters and virtual clusters;
- **G.5** support transparent and uniform use of computational facilities;
- **G.6** ensure availability of the sufficient computing power at all times, through mechanisms such as dynamic resource discovery and allocation.

**Requirements for networking facilities.** In line with the computational trend, networking is becoming more important. Today, computing facilities are geographically distributed, and data and/or processes are transferred among distributed nodes. Thus, the VL networking infrastructure must:

- **G.7** provide high-bandwidth, high-speed network connections sufficient for the transfer of data acquired in real-time;
- **G.8** ensure high availability;
- **G.9** sustain a certain level of quality-of-service when demanded.
Requirements for instrumentation facilities.

G.10 In order to cope with the increasing use of automation in experiments, the VL infrastructure must allow connecting ‘networked’ instruments. This in turn requires user interfaces, programming interfaces, and functionality/mechanisms for (remote) controlling and operation of an instrument, and availability of high-performance networking and storage infrastructure for the transfer and storage of the data it generates.

Requirements for software environment. The required software environment must be open for adding new resources, scalable to work with increasing number of users and workload, and it must be easy to customize. Furthermore:

G.11 in order to ensure and maintain the consistency of the VL environment and to ease the development and testing of tools, a philosophy and methodology must be defined and adopted by ICT developers for software design and development;

G.12 a set of generic or most commonly used software tools must be incorporated within the VL software environment.

Interface Requirements

User interfaces in general act as the entry-point of users to the underlying environment, and users interact with the underlying environment only through these interfaces. They present the functionalities and resources provided by the virtual laboratory infrastructure to users. Similarly, programming interfaces act as the entry-point of applications to the underlying VL infrastructure, and applications make use of functionalities provided by the VL only through these programming interfaces. Hence, interface requirements are among the most important type of requirements for the base ICT infrastructure. Requirements for user interfaces and for programming interfaces are described below.

Requirements for user interfaces. An integral part of a VL infrastructure is its user interface. Considering that experiments are performed by scientists (possibly with limited computer knowledge), user interfaces must hide the technical details of the underlying experimentation environment while supporting any possible usage of functionality provided by this environment. Thus, the VL infrastructure must provide user interfaces that:

G.13 are graphical, easy and convenient to use, flexible, and easy to customize to personal preferences;

G.14 are uniform, problem-oriented, and that can be used in diverse activities during all stages of an experiment (e.g. experiment design, template search and instantiation, data entry, query, and retrieval, etc.);
G.15 allow an organized working environment which eases the management and usage of diverse data and available resources, and which help the scientist to easily find what is where;

G.16 are suitable for providing assistance and help to scientists during experimentation, for instance, by providing context-sensitive help at each step of the experiment, or by suggesting a suitable analysis method to use.

Requirements for programming interfaces.

G.17 The VL infrastructure must provide platform independent, generic, and uniform programming interfaces. These interfaces must be well-documented and easy to understand. Furthermore, the interfaces must support the interoperation of domain-specific software tools both with the VL software environment and with other tools.

Functionality Requirements

The required functionalities from the base VL infrastructure are presented in this subsection.

Requirements for experiment management. Scientists must be supported during the entire life-cycle of experiments. Thus, the VL infrastructure must develop the necessary experiment management functionality. In specific:

G.18 A clear definition of an experiment must be provided, along with the necessary models (e.g. data model, presentation model, usage model) that are capable of sufficiently representing complex experiments. Different components of an experiment must be clearly identified (e.g. design, template, metadata, etc.). It must be possible to work with experiments both at the level of an entire experiment or at the level of its components (steps).

G.19 Necessary storage facilities (i.e. databases), user interfaces, and functionality/mechanisms must be provided for the definition, storage, query, retrieval and update of descriptions, purpose, usage rules, etc. for experiment templates and designs, as well as techniques and protocols used in experiments.

G.20 Necessary storage facilities, user interfaces, programming interfaces, and functionality/mechanisms must be provided for the definition, storage, retrieval, modification and deletion of experiment designs; for the storage, query, retrieval and manipulation of both information generated by experiments and metadata/annotations.

G.21 Necessary user interfaces, programming interfaces, and functionality/mechanisms for experiment execution must be provided, such as for template instantiation, enforcing usage rules for techniques and protocols, and discovering, allocating, monitoring, and managing hardware and software resources used during experiment execution.
G.22 Version management mechanisms must be provided for the different components of experiments.

G.23 Necessary models and mechanisms must be developed both to represent complex relationships between different experiment components and between different versions, and to maintain consistency within an experiment.

Requirements for information management. Information management requirements are elaborated in details in Subsection 2.6.2.

Requirements for resource management. Several resources are needed and used during an experiment in a VL environment, such as computing, storage and networking resources, instruments, data/information, and software. The base VL infrastructure must manage all these resources in an efficient and coordinated manner, in order to achieve utmost fulfillment of user requirements.

Necessary user interfaces, programming interfaces, and functionality/mechanisms must be provided:

G.24 for the discovery, allocation, usage and monitoring of the resources required for data acquisition, processing and analysis during experiment execution and analysis of results (e.g. computing units, network, storage, visualization devices, analysis/processing tools, etc.); a repository for available software resources;

G.25 to support various execution modes (e.g. background /interactive processing), and scheduling of processes for a specified time;

G.26 for the definition, storage, query, retrieval and update of descriptions of resources as well as up-to-date information about their current status, performance, and availability;

G.27 to support integrated usage of a number of heterogeneous resources (for example piping the data acquired from an instrument directly to a storage facility or to a computing facility);

G.28 for transparent usage of multiple resources (e.g. mechanisms like single sign-on);

G.29 for defining and enforcing access rights and usage rules for the resources.

Requirements for user management. In order to properly address the requirements of different types of users, the base VL infrastructure must support definition of different user roles, assignment of roles to users, definition of access rights for roles, and dynamic manipulation of users, roles, and access rights. In specific:

G.30 Necessary data models for the representation of entities in the VL environment (such as data objects, functionality, resources, etc.), user roles, permissions/access rights must be developed.
G.31 Necessary user interfaces, programming interfaces, and functionality/mechanisms must be provided for creation, update, and deletion of user accounts; definition, modification, and deletion of user roles; definition, modification, and deletion of permissions/access rights; assignment and revocation of roles to users and permissions/access rights to roles.

G.32 Necessary user interfaces, programming interfaces, and functionality/mechanisms must be provided to support creation and maintenance of user profiles, for instance, for the purpose of capturing user preferences.

Requirements for collaboration. The base VL infrastructure must support the collaborative activities among scientists. Such activities include sharing of resources (e.g. hardware, software, data/information resources), knowledge and experience sharing, and cooperative work (e.g. simultaneously visualizing a data set, video-conferencing, etc.). In specific, following collaboration-related requirements must be fulfilled by the VL infrastructure:

G.33 Necessary infrastructure, software environment, user interfaces, programming interfaces, and functionality/mechanisms must be developed to enable sharing of resources, such as the availability of a (distributed) resource management system, information system for up-to-date status and performance information about shared resources, mechanisms for adding/removing a resource to/from the pool of shared resources, definition and maintenance of user account mappings, and definition and enforcement of usage rules.

G.34 Necessary user interfaces, programming interfaces, and functionality/mechanisms must be provided to make resource sharing easy for users, including for instance abstract representations of resources (e.g. representing an instrument with an icon in a drag-and-drop capable graphical editor window), or discovery, allocation, and usage of resources (e.g. interfaces to specify the query parameters for required resources).

G.35 Necessary data models, user interfaces, and functionality/mechanisms must be provided to enable and ease the transfer of knowledge and experience sharing; for instance by making experiment templates, designs, protocols defined by expert users available to novice users, or through on-line (virtual) discussion environments among scientists.

G.36 Necessary software, hardware and networking infrastructure, user interfaces, programming interfaces, visualization environment, and functionality/mechanisms must be developed to enable the usage of various technologies to collaborate with partners: synchronous such as video-conferencing, display sharing/simultaneous visualization, joint sessions, etc. or asynchronous such as data exchange, sharing an instrument in an other organization, e-mail, etc.

G.37 A proper infrastructure must be provided for coordination of joint distributed activities and for secure and authorized sharing of resources, which also considers the autonomy of collaborating organizations.
G.38 Necessary data models, user interfaces, programming interfaces, functionality/mechanisms must be developed for the definition, management, and enforcement of collaboration rules.

**Requirements for security.** The base VL infrastructure must provide the necessary mechanisms for the authentication of users and the authorization of their requests. Furthermore, following must be considered when addressing the security requirements:

G.39 Security is at different levels: For instance, authorization rules can be defined for accessing a hardware resource (e.g. logging-on to a server), accessing to a data resource (e.g. reading a file on the server), or performing an operation (e.g. executing software programs on the server).

G.40 For data/information operations, access granularity can be finer: Accessing certain portions of a file, rows or columns in a table; or performing certain operations (read/write/delete). Furthermore, operations on data descriptions (schemas) require administrative privileges.

G.41 Security on the communication channels must also be addressed. Different encoding mechanisms can be employed.

G.42 Necessary environment for logging user activities must be provided. This can be used also for accounting purposes.

G.43 Security system must support transparent access to multiple resources (e.g. single sign-on).

G.44 Necessary user interfaces, programming interfaces, and functionality/mechanisms must be provided to support granting and revoking access permissions to other users, or delegate authority to a third party.

**Architectural/Technological Implementation Requirements**

Architectural design plays an important role on the functionality provided, and on the scalability, openness, flexibility and manageability of the overall system. Following is a list of architectural/technological requirements related to the implementation of the base VL infrastructure:

G.45 The VL infrastructure must adopt a technical and architectural design philosophy for software development, information management, and resource management. The philosophy must be complemented with well-defined methodologies for each of these activities.

G.46 The VL architecture must be open, flexible, and scalable to support interfacing with other systems (e.g. existing DBMSs, data acquisition and data analysis software, resource managers and schedulers), to improve, extend, or customize the provided functionality when needed, to sustain a certain level of performance
even with increasing number of users, to support collaboration, and to develop a number of monitors for managing the system and maintaining its performance.

**G.47** The system implementation must exploit the existing and emerging standards as much as possible, such as common Grid technologies and Web based technologies. However, compatibility of a technology with the VL philosophy and methodologies, and its openness for any future improvements/extensions must be considered beforehand.

### 2.6.2 Information Management Requirements

In this subsection, the focus will be on the information management requirements for the base VL infrastructure. Characterization of data handled in the VL environment (presented in Section 2.2), and the analysis of user requirements (presented in Section 2.5) are the two important inputs for the identification and analysis of information management requirements. Information management requirements identified in this subsection complement the general VL requirements.

Similar to the analysis of the general VL requirements, information management requirements will be analyzed by classifying them into categories:

1. Modelling requirements
2. Storage requirements
3. Manipulation requirements
4. Collaboration requirements
5. Security requirements
6. Interoperability requirements
7. Implementation requirements

In the remaining of this subsection, each of these requirement categories is further elaborated. Please note that every such category provides a list of the “main” requirements identified.

### Modelling Requirements

Following can be named among the modelling requirements for information handled in a virtual laboratory:

**I.1** Data models must be developed that are capable of properly representing the various types of information handled in the VL environment. The following can be mentioned among the different types of information handled in a VL: Experiment templates; experiment designs; descriptions of activities (i.e. history of events); experimental data/information (i.e. results); metadata
2.6. Analysis of Base ICT Infrastructure Requirements

(i.e. descriptions of data sets, annotations and conclusions); user information (e.g. profiles, roles); information about available resources (e.g. descriptions of hardware/software resources, and experiment templates and protocols); and security-related information (e.g. access rights).

I.2 The developed data models must support modelling and representation of various types of experiments (e.g. time-series, comparison, ordered/unordered, etc.). It must be possible to incorporate any level of detail about experiments. The experiments can also be of varying lengths and complexities. The data models must also support modelling of different experiment flows (e.g. optional steps, alternative paths, etc.) as well as representing complex relationships among experiments and components (steps) in experiments.

I.3 The developed data models must be generic to achieve *uniformity* in representing both heterogeneous experiment types and heterogeneous data types. Similarly, off-line activities (e.g. laboratory activities) and on-line activities (e.g. computations) must be uniformly represented.

I.4 The developed data models must enable the representation of semi-structured information (e.g. inferences and conclusions drawn during processing data sets) as well as structured information.

I.5 The developed data models must be of high expressive power to sufficiently model complex experiment types and data types. They must support clear and sufficient formulation of the experiment objectives/questions in experiment designs.

I.6 Schemas in the developed data models must have a succinct representation so that they are easily understood by scientists.

I.7 Schemas in the developed data models must be evolvable. They must be flexible for future changes, extendible for future extensions, open for customization to specific domains. Furthermore, the schemas must be compatible with the (data modelling and management) philosophy adopted by the VL infrastructure.

Storage Requirements

I.8 While the general storage requirements for the base VL infrastructure focused on availability of high-performance and high-capacity storage systems, storage requirements here focus on the availability of databases for different types of information about experiments. Thus, databases are required for storing templates for the most common types of experiments, descriptions of previously made experiments, and descriptions of the most common techniques, protocols, etc. used in different types of experiments.
Manipulation Requirements

Requirements related to the manipulation of diverse types of information handled in a virtual laboratory include the following:

I.9 Storage, access and manipulation mechanisms for the various types of information handled within the VL must be developed. These mechanisms must be uniform within and across disciplines.

I.10 The provided mechanisms must efficiently utilize the generality and expressiveness of the developed data models. Mechanisms must support the manipulation of experimental information of any length, complexity and heterogeneity, as well as the inter-relationships among different parts of an experiment. For instance, it must be possible to link an experiment template to its instantiation (i.e. an experiment design), or link an experiment design to information about its execution (i.e. run-time information and/or experiment results).

I.11 Provided mechanisms must support manipulation of experiment descriptions at different levels. It must be possible to manipulate the description of an entire experiment as well as the description of one of its components (steps).

I.12 Mechanisms for high-level searches must be provided, such as searching for an experiment template or for existing experiments based on keywords.

I.13 Mechanisms for arbitrary queries must be provided.

I.14 Mechanisms must be provided for managing multiple versions of experimental information (i.e. version control).

I.15 Administrative mechanisms are needed, including mechanisms for schema design, user account management, access rights management, monitoring the status of the servers, etc.

I.16 All mechanisms and functionality must be made available to tool developers, users, and to other components of the VL infrastructure through uniform, generic, easy-to-understand programming interfaces that are compatible with the general VL interfaces.

I.17 Graphical and easy-to-use user interfaces must be developed supporting all the mechanisms provided.

Collaboration Requirements

Mechanisms must be provided to support the collaboration needs of VL users, related to the secure sharing and exchange of information among collaborating partners. In specific:

I.18 Distributed/federated information management and query processing mechanisms must be provided to access heterogeneous data sources and to inter-link/integrate information from these resources.
I.19 Secure mechanisms for data/information sharing and exchange must be provided that consider and enforce the access rights and any other collaboration rules.

I.20 Data models and mechanisms are needed to provide information about the syntax and semantics of objects deployed by other data sources that need to be accessed (in case of a need for accessing multiple heterogeneous data sources).

Security Requirements

Requirements related to the security of information are enumerated below:

I.21 Mechanisms to define access rights for data security and information visibility must be provided. These mechanisms must be available to any user that is the owner of some information stored in the VL environment.

I.22 All provided information management mechanisms must consider and enforce the access rights that are defined for the information that they manipulate.

Interoperability Requirements

I.23 Available standards must be followed where applicable, for instance in information modelling, information exchange, data manipulation, and database implementation.

I.24 In case of a need for accessing multiple data sources, mechanisms to help/assist administrators and domain experts to resolve model/paradigm heterogeneity or semantic heterogeneity must be provided. Furthermore, mechanisms for resolving data definition/manipulation language heterogeneity must also be provided.

Implementation Requirements:

General VL requirements for implementation (i.e. G.45 – G.47) are also applicable for the implementation of the information management platform.

2.7 Conclusions

In this chapter, the proposed VL solution was characterized in Section 2.2 and different types of VL users and the activities that they perform with the VL were identified and described in Section 2.3. Each of these users performs different activities in the VL environment, hence they have different needs and expectations from the VL environment. In addition to user requirements, requirements for the base ICT infrastructure underlying the VL environment were presented, with particular attention on information management requirements. As these requirements point out, there are several different aspects that need to be addressed related to the management of experiment-related information, including among others data and functionality modelling for representing and managing information about scientific experiments.
### Table 2.1: User requirements (for scientists) and the base ICT requirements targeting their proper fulfillment

Analysis of requirements presented in this chapter showed that many requirements are related to each other. User requirements in turn impose a number of requirements on the base ICT infrastructure for VL. These ICT requirements in fact represent a first step towards providing a solution to user requirements. ICT developers must address these requirements to provide the necessary environment and functionality to VL users.

In this direction, Tables 2.1 and 2.2 match the user requirements to those base ICT requirements that target their proper fulfillment and represent a solution. Requirements are represented based on their enumeration values used in the previous sections.
<table>
<thead>
<tr>
<th>User Reqs.II</th>
<th>Base ICT Infrastructure Requirement(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.1</td>
<td>I.15</td>
</tr>
<tr>
<td>D.2</td>
<td>G.45</td>
</tr>
<tr>
<td>D.3</td>
<td>I.6, I.7</td>
</tr>
<tr>
<td>D.4</td>
<td>I.20</td>
</tr>
<tr>
<td>D.5</td>
<td>I.8</td>
</tr>
<tr>
<td>D.6</td>
<td>I.1, I.2, I.5</td>
</tr>
<tr>
<td>D.7</td>
<td>G.21</td>
</tr>
<tr>
<td>D.8</td>
<td>I.8</td>
</tr>
<tr>
<td>D.9</td>
<td>I.1, I.2, I.5</td>
</tr>
<tr>
<td>D.10</td>
<td>G.13, G.15, I.17</td>
</tr>
<tr>
<td>D.11</td>
<td>G.13, G.15, I.17, I.21</td>
</tr>
<tr>
<td>T.1</td>
<td>G.45, G.11</td>
</tr>
<tr>
<td>T.2</td>
<td>G.17</td>
</tr>
<tr>
<td>T.3</td>
<td>G.11</td>
</tr>
<tr>
<td>T.4</td>
<td>G.24</td>
</tr>
<tr>
<td>T.5</td>
<td>G.13, G.24</td>
</tr>
<tr>
<td>T.6</td>
<td>I.17, I.9</td>
</tr>
<tr>
<td>A.1</td>
<td>G.24</td>
</tr>
<tr>
<td>A.2</td>
<td>G.24, G.26</td>
</tr>
<tr>
<td>A.3</td>
<td>G.13</td>
</tr>
<tr>
<td>A.4</td>
<td>G.30, G.31, G.32</td>
</tr>
<tr>
<td>A.5</td>
<td>G.13</td>
</tr>
</tbody>
</table>

Table 2.2: User requirements (for domain experts, tool developers and administrators) and the base ICT requirements targeting their proper fulfillment