

VCLab as an Example of GRIDifying Virtual Scientific Experiments

Piotr Szczytowski
Institute of Automation and Computer Control, Ruhr-Universität Bochum
pszczytowski@atp.rub.de

Abstract

This paper describes and summarizes the current state of the development for making the Virtual Control Laboratory (VCLab) a GRID application within the ELeGI project. It introduces shortly into GRID techniques and shows how this GRIDifying process is performed to offer virtual scientific experiments (VSE) on the GRID. The architecture and some technical details of the prototype implementation are presented. The next steps in the development are outlined concerning the collaboration aspects of conducting VSEs. The authoring of VSEs is sketched by some proposals of authoring tools. A vision into the future emphasizes the VCLab as a component in a learning environment with a new learning paradigm.

1. INTRODUCTION

The Virtual Control Laboratory (VCLab) [1] has been originally developed as a tool to support students in control system design using professional simulations of automation processes. Because of its generic character, there are no restrictions to make use of it also in other scientific domains. VCLab uses a 3D virtual user environment to recreate and to visualize experimenting plants. One can interact with a displayed scene in a similar fashion like with real devices. The dynamical behavior of the plant is generated by a simulator driven by simulation models.

The first implementation of this system to perform Virtual Scientific Experiments (VSE) was running locally on a computer inside an internet browser, with the content being delivered from the web. Similar solutions are quite common [2] [3]. They have obviously drawbacks: the user is required to pose the machine capable of running demanding simulations and has to develop or install simulation software.

From the system type of viewpoint, VCLab is not a learning system, it should be rather considered as a technological system, which can deliver components for a learning system with VSE. One task within the European Union funded ELeGI project [4] aims at the redesign and reimplementation of the current solution in order to make it GRID aware. Implemented as a GRID service it will be later integrated into the Intelligent Web Teacher (IWT), which is a learning system currently also under reimplementation within the ELeGI project.

This paper presents the first results from the VCLab reimplementation within the ELeGI project. At first, for the inexperienced a short introduction into the terminology of GRID computing [5] is given and into the software platforms used in the context of the VCLab reimplementation. The second chapter is of technical character and is devoted to the discussion of the new VCLab implementation in a GRID environment. This chapter reflects the current implementation state. The following chapters are less technical and concern about future plans. In the focus are works for adding new GRID-based collaborative features to VSEs. The fourth and fifth chapter covers proposals for future implementations of learning modes and for developing GRID-based tools supporting the authoring of the content to be used by VCLab.

1.2. The GRID

GRID computing means applying resources of many computers in a network to solve larger problems – commonly seen as computational problems. In the case of our project, we consider computer resources as services delivered by participants of the ELeGI project, which are used for enriching and improving the e-learning environment.

1.3. OGSA, OGS.NET, WSRF.NET and GrASP

OGSA stands for Open Grid Service Architecture. It is a standard defining which requirements must be met by a service implementation in order to function as a GRID service.

OGSI.NET [6] (Open Grid Service Infrastructure) and its successor WSRF.NET [7] (Web Service Resource Framework) are two implementations of the OGSA. Both were developed at the Virginia University and both, as their names indicate, are based on the Microsoft .NET Framework. They deliver software base classes and port-types to implement user defined GRID services.

WSRF.NET is a successor of OGSI.NET and the conversion of the service program code from OGSI.NET to WSRF.NET is a pure mechanical task. The WSRF tends to extend the idea of Web Services by adding statefulness and management capabilities to them.

GrASP [8] (Grid-based Application Service Provision) is a GRID middleware for hosting and management of GRID services compatible with the OGSI.NET implementation. GrASP allows the dynamic instantiation of services, their accounting and publishing in web catalogs. GrASP is currently considered to be converted from OGSI.NET to WSRF.NET, because only the second option is under further development.

2. VCLAB AS A GRID APPLICATION

VCLab in its new GRIDified version has been split into:

- GRID services, namely the Computation Service responsible for the execution of simulations and for streaming results, and the DataSupply Service providing content delivery,
- Web Application consisting of a client and server side, which delivers an interface for the user and monitors the execution of the GRID services,
- GRID middleware responsible for the management of services, mainly their creation and disposal.

This modular architecture according to Figure 1 has some added value; it allows the reuse of the services created for VCLab within other applications and it preserves the scalability of the software. Such applications may run on many machines still being accessible from one central place.

In the following, the technical details of functioning and implementation of this architecture are described.

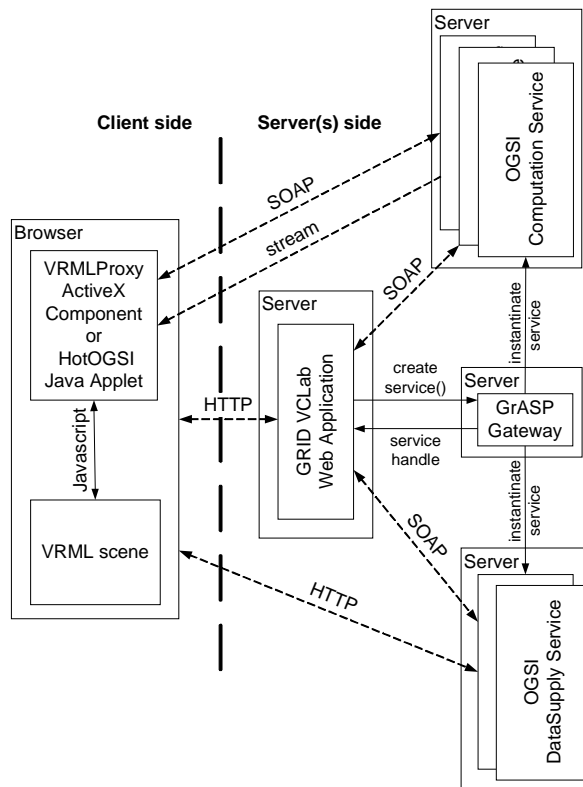


FIGURE 1. Overview about the VCLab GRID-based architecture

2.1. Computation Service

The Computation Service mainly delivers computational capabilities by means of the evaluation of mathematical expressions and allows running simulations of dynamical systems. Simulation output is delivered in a XML formatted stream. Mathematical expressions in symbolical or numerical form can be pushed to the service or received from it in LaTeX format.

It is implemented as an OGSi.NET-based service and is easily integrated in the GRID environment by GrASP, see Figure 1. From the implementation perspective, this service in fact mainly acts as a proxy between the client and the Computation Engine installed on the same server. It provides the required interface for integrating with the GRID, as shown in Figure 2.

Each server that hosts this service may run several instances of it. The communication between a client or a program that acts on behalf of the client and this service is performed by HTTP SOAP binding. This makes it mainly transparent to firewalls and NAT solutions. In order to avoid running the Computation Engine inside the service process itself and to move the management of the Computation Engine processes to a centralized location, the Computation Engine Proxy Server has been developed as an additional application. The communication with this application is performed using the .NET Remoting inter-process communication mechanism. The instances of this service make calls to this single proxy server. For all computational sessions the proxy hosts an instance of a COM control, which takes the responsibility for the direct communication with the Computation Engine. The COM approach is necessary, because the managed version of a runtime environment not always provides a mechanism for the interoperability with an engine. The use of the COM allows a greater flexibility of choosing the Computation Engine. In the case of switching to another engine, only a revision of the COM library is necessary and the GRID service and its interface remains still intact.

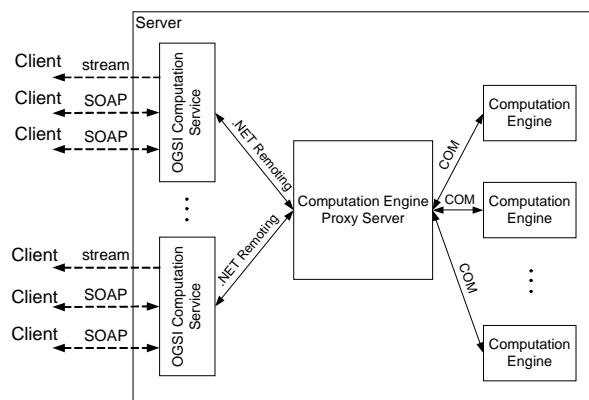


FIGURE 2. Computation Service architecture

Each instance of the Computation Service may instantiate multiple instances of the engine. The proxy server takes the responsibility of synchronized access to these instances. But the responsibility for the closure of orphaned instances of engines still remains on the service side. The proxy server acts also as some type of cache. This caching means that the server - instead of shutting down engines - clears only their environments and keeps them in an idle state. This guarantees a quicker response in case of new requests.

As shown in Figure 3, the execution of the Computation Service is performed within threads. The task of the Socket Threads - each running for a simulation session - is to retrieve results from the Computation Engine and to forward it to the clients reformatted in XML format. For streaming purposes the User Datagram Protocol (UDP) is used for the sake of efficiency. It is a connectionless protocol; therefore it does not carry transmission control data within packets. It is an information overhead because for streaming purposes there is no need for the detection or transmission of lost packets. Unfortunately, this protocol has limitations when it comes to establish this connection with a client within NAT networks. The Listening Thread helps to establish the connection with clients, which do not have a public IP address and therefore are unable to receive the default stream. In this case the client has to initiate a TCP connection on a port, which is retrieved by a call to the service. After establishing a connection the client sends a session identifier and the new connection is added to the connection pool of the proper Socket Thread. Moreover, the Computation Service needs one more thread called Monitoring Thread, which has to monitor for orphaned sessions and to close them in order to release the associated resources.

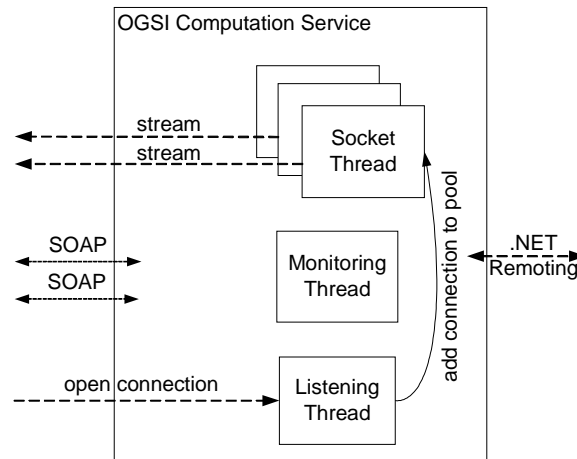


FIGURE 3. Details of the Computation Service

2.2. DataSupply Service

The tasks of the DataSupply Service are storing and managing simulation models. It provides the capabilities of uploading new simulation models, querying the database about its current content and deleting obsolete simulation data.

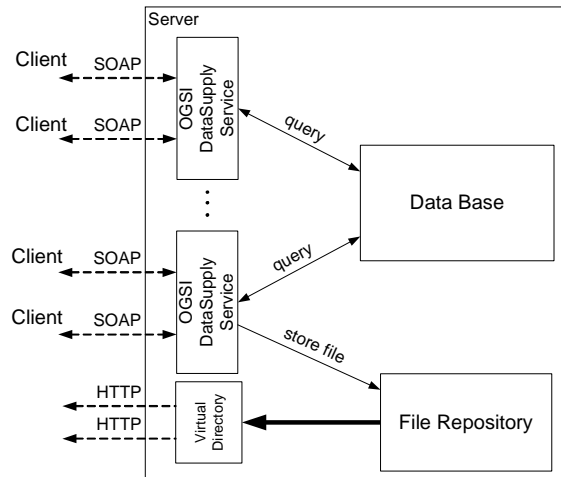


FIGURE 4. DataSupply Service architecture

The DataSupply Service, similarly to the Computation Service, is implemented as an OGSi.NET service. Its architecture is shown in Figure 4. The server may host several instances of the DataSupply Service. The communication with the outside world is using HTTP SOAP binding. The access to simulation models is provided in form of a Virtual Directory, which publishes the content of the File Repository. All instances inside the server share a common database and File Repository. Calls to this service cause the executions of queries to the database and their results are returned to the client. When uploading, new files are stored and the database is updated to point to their location.

There remain two problems for the DataSupply Service not yet solved. One of them is the user authentication. It is obvious that the access to such an option like uploading and deleting simulation data should be allowed only for trusted users. This problem can be solved by an own authentication service or by relying on the built-in GrASP security subsystem. The other problem requiring consideration is the synchronization of the content of the DataSupply Services spread across the GRID. One possible solution is a static internal service responsible for replication, which would be situated on each of the servers hosting a DataSupply Service.

2.3. VCLab Web Application and interaction with the services

In order to deliver the capabilities of the described services, the Web Application has been developed to control services and to provide an interface suitable for the user, see Figure 1. The Web Application consists of a component on the client and an application on the server side.

On the client side, inside the internet browser, the VRMLProxy ActiveX Component or the corresponding HotOGSI Java Applet is running. The task of these controls is to provide a binding between simulation results retrieved by a stream from the Computation Service to the 3D scene displayed in the browser. These controls also provide the communication in reverse direction. Actions taken by the user during interaction with the 3D scene are translated into commands for the Computation Service and sent to it using the HTTP SOAP binding. The installation of the client side components is done automatically. Therefore the user does not need to download the software manually, to install and to configure it.

On the server side the task of the Web Application is to call for the creation of the underlying services and their later coordination. So it is an extension of the GrASP middleware for visualizing the resources in a suitable manner for the user. The Web Application, the OGSI services and the GrASP Gateway may run on separate computers or share the same server or servers.

On start of the Web Application, it requests the GrASP Instantiation Service to create instances of those services required to run the application, in this case the Computation and DataSupply Service. The services to be created are described by a SLA document, which allows the customization of the Quality of Service aspects of newly created services. In the case of a successful instantiation of these services, GrASP returns their handles to the Web Application. GrASP can instantiate the service on any of the servers, which provide the corresponding factory for the service being created and which match the requirements specified by the requestor contained in the SLA. The Web Application uses handles to communicate directly with the services. When a user opens the main page of application, it invokes the DataSupply Service methods for listing accessible simulations and displaying them to the user. The communication between the application and the user takes place using the HTTP protocol. When a user decides to start a simulation, the Web Application calls the Computation Service to create a new instance of the Computation Engine and returns its session identifier. The Web Application renders the simulation page to the user. It provides for the component handle to the Computation Service and for the session identifier earlier obtained.

In case of more clients willing to run a simulation than a single instance of the Computation Service can handle, the Web Application requests the GrASP Instantiation Service to create an additional instance of the service. After receiving the new handle, the Web Application adds it to its pool of services. The same steps are undertaken in the case of the DataSupply Service. Thresholds are defined by the configuration file of the Web Application for adding new instances of the service.

3. THE COLLABORATION ASPECT OF VCLAB

VCLab in its original form provides a very limited support for experiments in a collaborative learning environment. Simulations running on the server can be observed by participants other than the simulation creator, but only in passive mode. This means that they cannot interact with running simulations. Nonetheless, this option provided by the Computation Service may be the basis for further developments of a fully fledged collaboration environment. Each simulated experiment can be made accessible to all participants, which will take part in it. Participants may pass control over the experiment to each other. This conception is described in the following sections.

3.1. Participants roles in the collaborative environment

One of the important aspects of defining collaboration in e-learning is the definition of roles of participants and the assign of these roles to them. In Figure 5, similarly to many pedagogical approaches, we propose to model four distinct roles within a collaboration environment. These roles are Author, Tutor, Learner and Experimental Plant.

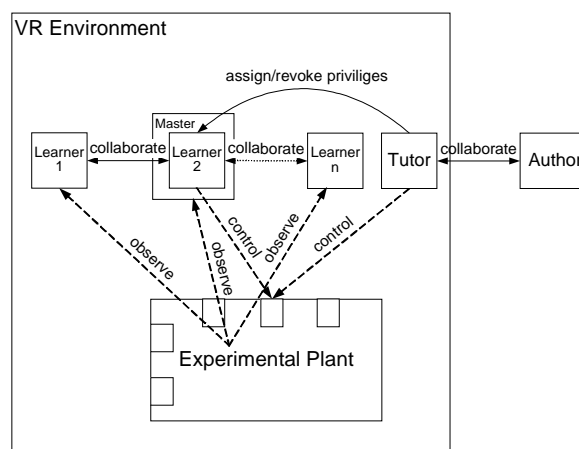


FIGURE 5. Simplified model of a collaborative environment

The Author does not directly take part in the collaboration activities. Its tasks are preparing the environment by means of defining experiments, tasks for learners, instructions, designing graphics and so on. The Author object does not have to represent a single person, it may be a team responsible for authoring the learning content (models, graphics, sounds, etc.)

The Tutor is a privileged participant of the collaboration activities. Its tasks are to provide the content to learners, to monitor their progress, to supervise experiments, to give hints, explanations and advice, to answer learners' questions, which may occur during the learning process.

The Experimental plant represents the modeled knowledge, which learners should gain during experimenting with it. The Learner role is described by a learner model. The Learner is allowed to interact with the experiment in active and in passive mode. Only one Learner is allowed to work with the plant at a moment. Such a Learner is marked in Figure 5 as a "Master".

Besides roles, there is also a need for relations between them to be defined.

The Author - Tutor relation requires the collaboration between these two figures. The Tutor has to be instructed about the created content and has to be provided with possible scenarios. The Tutor has to provide feedback related to the pedagogical aspect of the content.

Author - Experimental Plant, this relation may be described using term transpose. The Author models the university-level knowledge in form of the experimental plant. The plant is being described by a set of model equations using tools designed for this purpose. Such a created simulation model is being connected to the 3D visual representation of the experiment. The Author decides also how knowledge should be delivered to the learners. That means he/she creates scenarios for the usage of experiments.

The Author - Learner relation is not direct, an indirect connection is maintained by the knowledge.

The Tutor - Learner relation may be described using the terms guide and advice. The Tutor has the responsibility for providing the knowledge in understandable and accessible manner to the learners. The Learner may ask the Tutor for guidance in case of difficulties in solving a particular problem and may also confront their findings with expected results. The Tutor decides about assigning and revoking privileges to control the simulation to the Learners. He/She may also himself control the simulation to show an example or explain an experiment.

The Tutor - Experimental plant relation is based on the management of knowledge by the Tutor. The Tutor has to define the course in the frame of experiments and scenarios delivered by the Author.

The Learner - Learner relation is mainly collaborative. The Learners work with each other by communicating and passing among each other the control over the Experimental Plant.

3.2. Collaborative environment - architecture

Our approach for implementing a collaborative environment with the roles sketched above would incorporate existing GRID services by means of orchestration to build a new collaboration service as it is shown in Figure 6.

The task of the Collaboration Service is to provide a virtual environment, in which participants of a learning course could meet in a similar fashion to a classroom meeting. The service should act as a collaboration server, which communicates with all participants of a course/meeting/lesson both in visual and textual form, and controls their access to experiments. Experiments within the Collaboration Service are maintained by the orchestrated Computation and DataSupply Services.

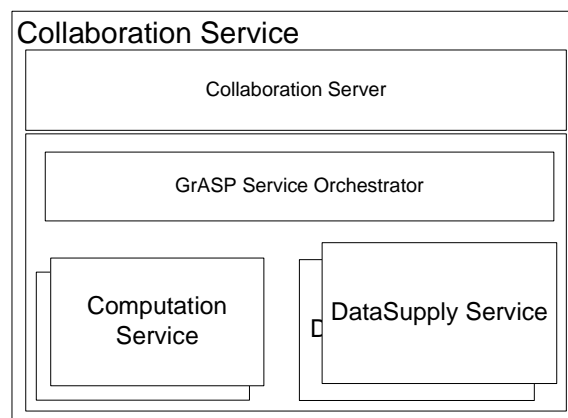


FIGURE 6. Collaboration Service architecture

The virtual world can be implemented using VRML technology in the same way as it is implemented in the current version of VCLab. Every instance of the Computation Service could be responsible for simulations connected to the experiment.

The participants of an experiment will be represented by their avatars capable of operating the plant, displaying mimics and gestures, which will simplify the communication by imitating a real world behavior. This is illustrated in Figure 7.



FIGURE 7. VR Collaborative environment

3.3. Collaborative learning techniques in the context of VCLAB

Implementing a 3D collaboration environment supported by the Collaboration Service, in which people can meet, being represented by avatars imitating persons, allows main techniques for collaborative learning.

One of them is the so called Jigsaw method. It is based on organizing learners in groups and specializing members of every group in one particular problem. After completing the tasks by groups, groups are reorganized in such manner that each new group consists of one member of every previously existing group. Learners within new groups share experiences gathered in the previous phase. We may easily imagine such a form of collaboration in the context of the proposed collaboration environment. In the first step, every group of learners may be assigned to one of each simulation. After completing the experiment the groups may be reformed in respect of the described technique and members of the group may meet in a virtual classroom for a chat session, during which they exchange their gained knowledge.

Similarly, the concept of learning circles may be realized. Each local learning group may be represented by an avatar in the 3D collaboration environment.

4. LEARNING MODES

Our proposal foresees three learning modes: guided, supported and free. Figure 8 shows a course in form of connected task nodes with possible paths. This kind of approach is called project oriented [9]. In the guided mode the learner follows a predefined path of tasks and subtasks, which must be followed and lead to the goal.

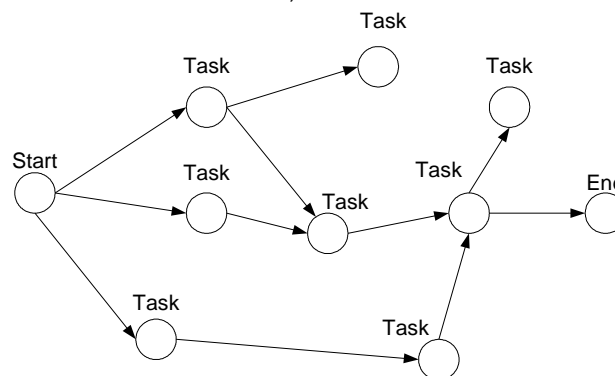


FIGURE 8. Example model of learning course

The supported mode means that the learner sees all direct connections to the next tasks and the recommended task is suggested, although the learner can choose the next step on his/her own.

In the free mode, all direct connections to the next task are visible and the user has to choose on his/her own, which path to follow. If a path is not chosen wisely it may lead to a dead end.

In order to implement the proposed learning modes, it is necessary to implement the software, which would handle the task of controlling learner doings. Such software is called Task Machine.

4.1. Task Machine

The Task Machine stores tasks and subtasks and associate with them paths. These data are being used to let the learners navigate through the content of the course. The combination of tasks and paths defines a finite amount of

states in which the user can find oneself. The Task Machine stores for every user the history of the progress and the current state. So by a new visit to the course site, the user may continue from the point he/she left.

The Task Machine will be implemented also as a GRID service. So course data will be stored on a server side assuring that they won't be temper with and won't be lost. Implementing the Task Machine as a service one important aspect must be taken into consideration, namely, whether the task machine can be created dynamically or there should be only one instance of this service. In the case of a dynamic creation of the service by each visit of the learner, the Task Machine could be instantiated on another server that causes problems localizing data associated with the user. If only one instance of this service existed, the problem would have disappeared. But such a solution would be probably too radical. Another approach would be, placing a few static instances of the service on several servers and add a replication capability to the service.

5. THE AUTHORING TOOLS

In order to take full advantage of the new environment a set of tools is required, which would aid in the creation of the necessary content for the GRID e-learning environment. There are a few tools, which we take into consideration, both for the learner itself and for the author.

5.1. The Simulation Authoring Tool

Tools such as Simulink provided within the Matlab environment are too complex for the purpose of creating the content for our environment. Such tools still can be used, but it is obvious that some additional intermediate tool is needed, which would simplify this process. Such a tool should deliver predefined elements for designing experiments. The repository of such elements could be stored and managed by a new GRID service or an extended version of the DataSupply Service. Newly built simulation models should be also stored back to the repository to become building blocks for further more complex simulations. The tool itself should be accessible through the web or even embedded into the virtual collaboration environment. So it could be used by the learner to build ad-hoc new experiments or to redesign existing ones to check new theories.

This tool should also allow automatically publication of a newly created content, so that it would be accessible for other tools.

5.2. The Visual Object Authoring Tool

Similarly to the Simulation Authoring Tool, the Visual Object Authoring Tool should consist of a repository of predefined elements. It should provide an easy way for interfacing simulation models with the virtual environment by mapping handles automatically to the scenes with the simulation variables. Newly built objects should be stored back to the DataSupply Service and then can be reused as elements for building larger environments.

5.3. The Task Machine Content Creator

Preparing the content for the Task Machine also requires additional support in the shape of the authoring tool. It is important that such a tool would be capable of the validation of the created learning paths. It should also make use of the repository of already created simulation models by the previously described tools. The main principle, on which such a creator would work, is to use author input to automatically generate the GRID service code and to compile it. Such a compiled service would be added to the GRID.

5.4. The authoring tools interdependency

There is a possibility or even necessity for some level of integration between the tools mentioned and the VCLab GRID infrastructure.

The authoring tools should be an essential part of a new GRIDified e-learning environment as shown in Figure 9. The Simulation Authoring Tool should make use of the DataSupply Service in its extended version to retrieve the collection of predefined elements and for storing created simulation models together with their meta-data description. Additionally, it should make use of the possibility of filtering objects only to required ones for a particular simulation. Also the Simulation Authoring Tool could be embedded into the Collaboration Service and would allow directly forming a virtual world to create new or to modify those provided with the course simulations. The Visual Objects Authoring tool should also make use of the DataSupply Service in the same fashion. Besides, it could make use of the Computation Service in order to test the binding of newly created objects with simulations. The Task Machine Authoring Tool potentially also would benefit from the central repository. Based on simulation resources it would aid in constructing learning courses.

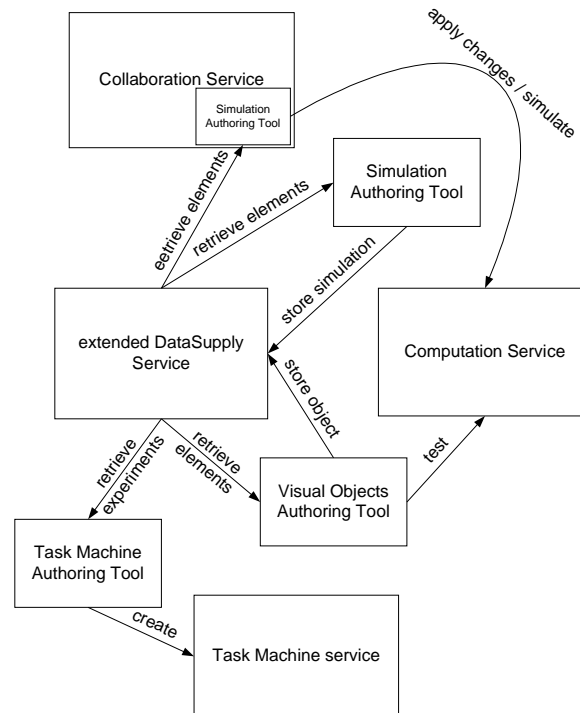


FIGURE 9. Authoring tools interdependency

6. CONCLUSIONS

This article describes in details problems and results of implementing VCLab as a GRID service. It may serve as a general introduction into GRIDifying currently existing applications. It has been shown how the currently available GRID-based version of VCLab is designed and implemented to conduct virtual scientific experiments. This prototype version is already available for users at [10] to be tested.

It is important to have a GRID-based support of the collaboration within the virtual scientific experiments. This has been addressed by the presentation of the next step in the ELeGI project for developing the Collaboration Service. It is also necessary to have a look into the far future when all those "executing" services discussed in this paper have become mature to be used. The authoring aspect and its GRID-based implementation as a natural complement for this environment have been foreseen.

REFERENCES

- [1] Chr. Schmid, "A Remote Laboratory Using Virtual reality on the Web", *Simulation, Special issue: Web-Based Simulation, Vol. 73, No.1*, 1999, pp. 13-21.
- [2] Bhandari, A. and M.H. Shor: Access to an Instructional Control Laboratory Experiment through the World Wide Web, Proc. 17th American Control Conference ACC'98, Philadelphia (USA), 1998, 1319-1325.
- [3] Karweit, M.: A Virtual Engineering/Science Laboratory Course. Department of Chemical Engineering, John Hopkins University. <http://www.jhu.edu/virtlab/virtlab.html>
- [4] ELeGI project web site, at: <http://www.elegi.org/>
- [5] I. Foster, and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure, Second Edition*, Elsevier, San Francisco, 2004.
- [6] OGS.NET project web site, at: <http://www.cs.virginia.edu/~humphrey/GCG/ogsi.net.html>
- [7] WSRF.NET project web site, at: <http://www.cs.virginia.edu/~gsw2c/wsrf.net.html>
- [8] GrASP project web site, at: <http://eu-grasp.net/>
- [9] M. Völker, A. Liefeldt, S. Engell and Chr. Schmid: "Learn2Control: A Project-oriented Approach to Teaching Control Engineering." *Proc. IEEE Conference on Computer-Aided Control Systems Design 2004*, Taipei, Taiwan, 2004, pp. 184-189.
- [10] VCLab GRID-based version web site, at: <http://quack.atp.rub.de/grid%20vclab>