

Web Based Simulations for Virtual Scientific Experiment: Methodology and Tools

Giovannina Albano*, Gerardo Iovane, Saverio Salerno*, Sandro Viglione

*Dipartimento di Ingegneria dell'Informazione e Matematica Applicata (DIIMA)
Università degli Studi di Salerno Via Ponte Don Melillo – 84084 Fisciano (SA), Italia

Centro di Eccellenza "Metodi e Sistemi per l'Apprendimento e la Conoscenza" (CEMSAC)
Università degli Studi di Salerno Via Ponte Don Melillo – 84084 Fisciano (SA), Italia

albano@diima.unisa.it, iovane@cemsac.it, salerno@unisa.it, viglione@cemsac.it

In this paper we want to present the tools and a methodology to realize web based simulations according to the didactic model defined for the Virtual Scientific Experiment. The strategy that we suggest intends to satisfy all the requirements for the actors involved in the learning process according to the adopted didactic model. These requirements provide that: the tools for building the simulations have to serve a teacher instead of a programmer of web based simulations; the simulation outputs showed at the learner have to focus his attention on the contents instead of the modalities to interact with the presentation; and last but not least, the simulation must not be static, but it evolves according to the different phases expected from the adopted didactic model. From a practical point of view the results reached by us until now are realized by IWT (Intelligent Web Teacher). IWT is a distance learning platform designed with the precise purpose to lay the foundation for the next e-learning generation. We extended IWT with the instruments required to support the new methodology.

These are the keywords. Web based simulation, Virtual Scientific Experiment, e-learning

1. INTRODUCTION

Until now, the adopted strategies for the implementation of the simulative and interactive contents in systems for the distance learning were based on the technologies and the available tools to develop web contents. This caused the development of didactic resources that were able to transmit information in a different manner than textual. The e-learning systems evolution from simple tools for the information transmission to devices for the knowledge transfer, requires the development of contents that are strongly dependent by the learning process in which they are inserted. Besides, these contents must interact with the e-learning environment in which they stay to provide all the information required by the learning process for its evolution towards the achievement of the planned didactic targets in agreement with the inputs provided by the learner.

Web based simulation represents the connection between the World Wide Web and the field of simulation. The term web based simulation emerged in the 1996 at the Winter Simulation Conference (WSC96) where was formally introduced. In one of the papers [1] presented, Fishwick describes several potential impacts of web technologies on simulation, giving particular attention to three areas: education and training, publications and simulation programs. Most commonly associated with the term web based simulation [2] is the web based access to simulation programs, this area includes both the remote execution of existing simulations from a web browser through HTML forms and appropriate scripts, and the development of mobile-code simulations (e.g. applets) that run on the client side. The concept of web based simulation describes a simulation practice that differs from "traditional" approaches in many ways [3] that conduct in the proliferation of digital objects and software standards,

in the use of multi-tier architectures and multi-language systems, in the model construction by composition, in the use of "trial and error" approaches and in the proliferation of simulation use by non-experts.

Numerous technological solutions have been developed to make web pages more interesting, interactive and dynamic containing embedded multimedia and applications like, for example, simulations. These new features are realized either at the client side as embedded features or plug-ins for web browsers, at the server side using different techniques or as a combination of both. The best solutions that have been developed to carry the simulations on Web, using the web browser features with and without the use of plug-in, were: Java applets, ActiveX components, VRML virtual worlds and Shockwave interactive movies. But all these solutions don't represent the best way to carry the simulations in e-learning systems in a simple way because everyone of these technologies has positive, but also negative features. For example, Java applets are secure and simply to add to web pages but who want to realize them must know Java language; the same problem have VRML virtual worlds; while Shockwave interactive movies can be developed using a graphical tool that simplifies the simulation building process, but the flow of simulation must be programmed by the user (e.g. movements of all the objects present in simulation) with a great effort. Moreover these objects are closed, in the sense that they don't interact with the e-learning environment that uses them.

To try to solve the problems researchers used other approaches. One of these is provided by the system [4] realized by Juan de Lara and Manuel Alfonseca. Their system provides an integrated environment for the construction of interactive simulations. Simulations are specified in a high level object-oriented language called OOC SMP and use different output formats, including VRML panels and multimedia elements that can be synchronized with the simulation by means of language primitives. The use of higher level languages increases notably the productivity of the document designer and avoids the need to know lower level languages such as Java, HTML, or VRML. Generally the design of these systems is driven by technology not by pedagogical aspects linked to the learning process that have to be provide later through these systems.

In this paper we want to present a methodology to define the necessary tools to realize web based simulations able to provide a concrete support to the learning process linked to Virtual Scientific Experiment (VSE). The suggested strategy starts from the analysis of the didactic model defined for VSE and then goes on to identify the features that the tools for creation, execution and visualization of web based simulations for VSE must have to satisfy the requirements of the actors involved in the learning process (teacher¹ and learner). Related to the aspects of the implementation, at last, we describe how we extended e-learning environment IWT [5] with the components needed to create and use simulation learning objects. The development of these components is not ended but, however, it is already possible to use them to test some of the methodology aspects discussed here.

2. VIRTUAL SCIENTIFIC EXPERIMENT SCENARIO

The learning process linked to VSE and to the didactic model defined for VSE is complex and does not finish considering the only simulation component. Surely, in this case, the simulations are the keys to realize an inductive-experiential situation. To carry out the whole didactic model are required even services and other typologies of didactic resources.

In this paragraph we show the context in which we are operating without be exhaustive, but we will try to underline the aspects linked to simulation components that characterizes our learning process.

The VSEs are objects that use the mathematical model description to simulate the real behaviour of some physical phenomena. With the simulation it is possible to observe the evolution in the time of the system or real process. An user can interact with the VSE object to modify its evolution. The user, by the direct observation of these changes, can acquire new knowledge on the system. If the VSE model provides to the user a 2D and/or 3D graphical interface like a real world scene where the system evolves, or the capacity to focus the simulation parameter keys, then the VSE catches the attention of the user who is encouraged to live the virtual experience.

The didactic model for VSE [6], presented in figure 1, is based on the a-didactic situations, as said in "Theory of Situations" of Brousseau. The a-didactic situations allow an experiential and inductive learning process, in this case carried out by simulation, facilitating the development of reflexive observation, critical reasoning, hypothesize capabilities, abstract conceptualization and generalization abilities.

The first phase of the model is the **presentation**, in this phase some general indications and characteristics are proposed: to whom is the experiment addressed, which are the necessary pre-requirements, the finalities, the targets, the motivations at the basis of that particular learning, so to give meaning and intentional value to the entire experience, involving actively and totally the student in the pursuing of the general cognitive, conscious and shared objectives.

¹ The term "teacher" is used in a general manner to indicate also other most specialized figures like: content expert, didactic engineer, content engineer, etc

The next phase is named **situation**. That is an active phase where the user is inserted into a context that belongs to him/her, allowing him/her to construct his own knowledge personally. Such situation is of a didactic type. The situation could be represented by a motivating and involving simulation that realistically recreates the conditions of a given operating context, pushing the user to get involved directly in the situation.

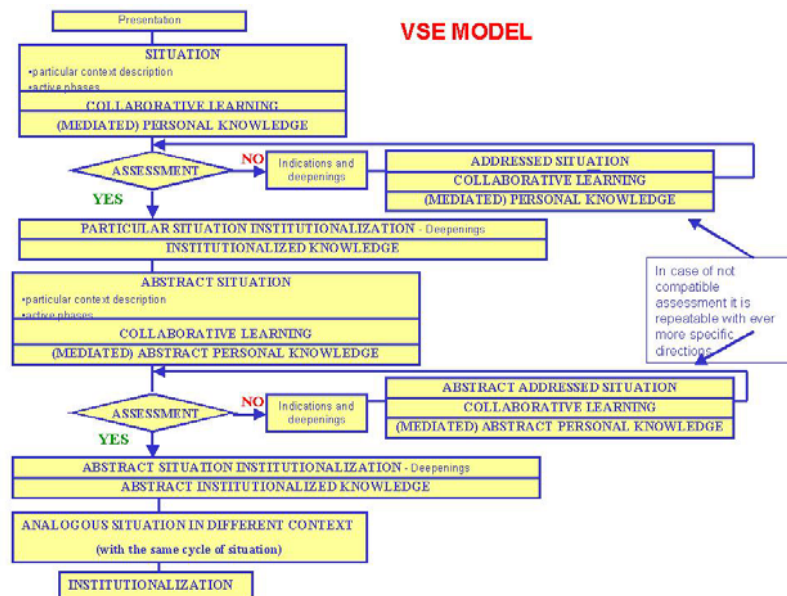


FIGURE 1: Didactic model for VSE

Then there is the **collaborative learning** phase that is the comparison phase with other users, where the personal single individual learning is validated and socialized.

In the **personal knowledge** phase (or mediated personal knowledge if the student collaborate with other users) the student works with the personal knowledge he has learned in the "situation" reflecting on what he/she has realized actively: filling tables, assuming a range of parameters, answering to questions, anticipating behaviours, choosing movements, etc. To encourage the learner to reflect on the experience, we could show him/her not only a 3D scene where the system evolve, but also the graphical 2D with the evolution of a particular parameter. Moreover to enter the value of same input parameters, the learner could use a slide bar control to underline the parameter range values. In this phase we use a simulation having the same mathematical model than situation phase, but the simulation outputs and input parameters are showed to learner in a different way.

The proposal compatibility or incompatibility assessment then follows: based on the "(mediated) personal knowledge distance" constructed actively from the user you will be passed to the "particular situation institutionalization" phase if the performance results "compatible" with the proposal, otherwise you pass to the "addressed situation" phase.

The **addressed situation** phase is foreseen in case of proposal incompatibility and represents the same initial situation, but this time mainly led by the author, in order to have that the following "(mediated) personal knowledge" phase can result more compliant with the user's requirements. In this case it is possible to think that the simulation plays only a demonstrative role in which the less meaningful input parameters aren't showed anymore to learner and some of them take a constant value while the others take a value that must be chosen among prefixed values. In the scheme of the didactic model, when the performance results and "(mediated) personal knowledge" are compatible with requisite knowledge for the understanding of experiment, the learner is introduced in the particular **situation institutionalization** phase, where the institutional knowledge involved in the particular situation is explicated.

If the experiment is not trivial, the didactic model foresees an **abstract** phases cycle, where the learner tries to extrapolate, from the "situation", an abstract model of the experience. In this phases the learner tries to understand, which are the laws that tie the fundamental parameters of experience. The learner interacts with a series of situations (abstract situation, collaborative learning, abstract personal knowledge etc.) where there are simulations that evolves inside the abstract cycle according to the prefixed didactic objectives and the adopted didactic model.

When the learner has completed also the cycle of learning produced by the "abstract situation", he is able to study an **analogous situation in a different context**. In this phase he is introduced in a new different experience, based on the same physic principles and laws that are the cognitive objectives of learning process. In this new situation the learner can use the learned knowledge and verify it.

Finally the didactic model foresees a conclusive theoretical and general **institutionalization** phase, that recalls the fundamental concepts used in the previous situations. In this moment the knowledge involved in the entire "cognitive situation" is explicated and formalized, also other particulars can be provided.

3. METHODOLOGY TO DESIGN A SIMULATION FOR VSE

In this paragraph we resume the whole previous analysis and introduce our guidelines that could be followed to realize web based simulations according to the learning process defined for VSE. Our aim isn't that of who want to examine how a simulation could be developed in each phase in agreement with the model, but we want define the set of properties that characterizes the simulation. The teacher, changing the value of this properties, can specialize the simulation for each specific phase of didactic model defined for VSE. In our considerations we decompose the simulation in three parts: mathematical model; input parameters, that characterize the simulation and whose analysis encourages learners to understand anything anyone want to teach them; and simulation outputs, considered as graphical presentations of whether the whole simulation (e.g. 3D scenes) or the evolution of its important output parameters.

Concerning the input parameters we can define for everyone also four attributes: starting value; read only, to denote that the parameter can't be changed by the learner; range, to identify the set of the parameter allowed values (this attribute can be used whether to avoid that the student enters unacceptable physical values or to provide to an Intelligent Tutoring System useful information for the learner rating); and format, to denote how the learner could change the value of that parameter (e.g. entering a numerical value in a text box, choosing a value from a list choice or using a slide bar, and so on).

For each phase of the didactic model defined for VSE, the considerations to do with the development of a simulation are:

The used input and output parameters must be emphasized to allow the learner to interact with the simulation. The choice of these parameters is made taking account of the particular phase in which the model will be used. For example, a situation phase in which it's important only the whole simulation output and it's essential involve the user through a realistic simulation that doesn't need many input parameters.

As we stated, the simulation input parameters must focus the learner attention on the particular didactic goal that he have to reach in that phase of the didactic model. So it's possible to operate on the attributes that characterize the parameters to reach this task. For example, considering an addressed situation phase, it's reasonable that some input parameters are fixed by the teacher and are read only.

Also the simulation output presentation must be relevant to the particular phase of the proposed didactic model. For example, in a situation phase it's surely recommended a graphical output with a 3D scene representing the simulated real system. In a personal knowledge phase instead, it's recommended to provide in output also some graphs that show the evaluation of some key parameters during the simulation.

3.1. Tools for editing, execution and visualization of simulations for VSE

Once it has been identified the requirements that the simulations must satisfy to carry out the determined didactic model for VSE, it's necessary define the tools needed to realize the simulation.

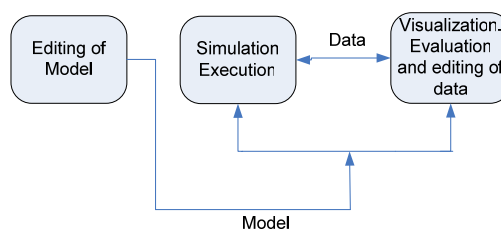


FIGURE 2: The development and use of scientific computing software

From a conceptual point view the development and use of scientific computing software goes through several stages[7] (see Figure 2), which can differ from application to application. Programming and simulation environments integrated with graphical user interfaces can help in different stages of model design and use. The visualization and editing tools are used for the representation of data structures (model) and data values (simulation outputs) on computer displays by means of two and three-dimensional graphical elements using an appropriate level of abstraction.

The 3D visualization is a representation of three-dimensional scenes mapped onto a 2D display. Scientific visualization is a special case of visualization which usually means visual presentation of high volumes of numeric data defined over some continuous domain, such as time and/or space. Often computational results of scientific computing are displayed by scientific visualization tools (e.g. AVS[8], Data Explorer[9], and Vis5D[10]). The information used in scientific computing falls into two categories: descriptions of mathematical models and descriptions of data. In our case we have a third category named learning process information. These information are necessary to characterize simulation according to a didactic aspect and they allow to specify the interesting parameters, how they must be offered to learner and how the simulation outputs must be presented (2D or 3D graph, etc.). When a mathematical model (at some level of abstraction) is represented graphically as a diagram by some tool it is usually not called visualization, but rather graphical model browsing and/or editing.

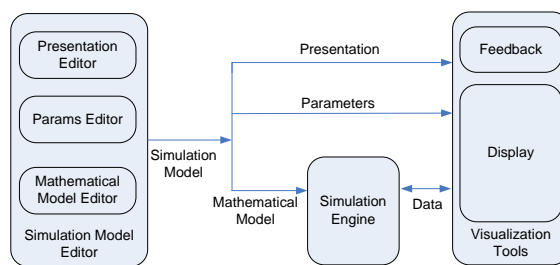


FIGURE 3: Conceptual architecture and tools to create, execute and visualize the simulations for the VSE

In the figure 3 is showed a conceptual architecture with the tools to create, execute and visualize the simulations for the VSE defined by us.

The Simulation Model Editor is constituted by three modules:

Mathematical Model Editor: Through this module the teacher can built the mathematical model of the simulation and select the input parameters that he want to present to learner. Model building can take place in a textual way or in a graphical way using a specific tool for the visual modelling. The output process can be transmitted to the simulation engine immediately.

Parameters Editor: it's necessary to characterize the simulation input parameters through the attributes previous defined.

Presentation Editor: it allows to combine the simulation outputs and the used display type when it's necessary. Display can show 3D scenes, 2D graphs or combinations of both. The Presentation Editor must also allows simulation parameters mapping with the objects that define the showing scene. For example, if the simulation define the Moon orbit round the Earth and the presentation is a 3D scene in which appear two objects Ob-Earth and Ob-Moon respectively, the mapping phase will be made up of the combination of simulation output parameters with the components that define the Moon movement in the 3D scene.

The Simulation Engine must deal with linear and nonlinear, discrete and continuous time dynamic systems; this feature is fundamental to control a bigger number of experiments. Simulation engine must have object libraries relating to hydraulic, thermodynamic, mechanical and electrical devices and further other devices used in various fields of Sciences. Besides it must allow the user to built own libraries using a particular object oriented description language. This language should interface to external functions written in other programming languages.

The Visualization Tools provide to show the student the simulation output through one or more Displays and in the same time the Feedback module traces the student actions and provides a feedback towards the e-learning environment. From what we stated appears clear that the Simulation Model is composed by the Mathematical Model and by the additional information that characterize parameters and the presentation definition.

4. IMPLEMENTATION SCENARIO

4.1. Relation with the WWW and distribution view

Next picture shows two aspects: the first concerns the teacher's and learner's modalities to enter the simulation; the second shows the adopted strategy to integrate the simulations tool in IWT.

The teacher, through a common web browser uses the modelling tool simulation. The editing phase take place in a visual way by means of a Java applet that act as a graphical interface. Teacher has a series of icons, organized in libraries, and a working area. Through simple drag and drop operations, teacher can drag the icons in the working area and link them each other to built the simulation mathematical model in a visual way. The modelling phase does not end with the building of the mathematical model, but goes on to characterize the simulation input parameters. Actually the teacher can fix only the starting value of an input parameter and settle it as a read only if he wants.

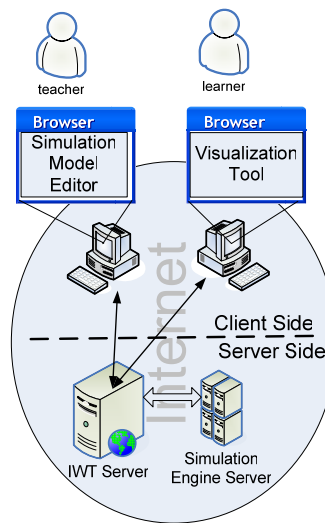


FIGURE 4: Distribution View

Learner enter the simulation through the Visualization Tool Display. The way to enter the simulation is thick server approach, that is every simulation run provided by the learner, the simulation is executed server side and then the results are sent to the Display. Generally, the simulation engine requires an own server and all the communications with the visualization tool go through IWT. Actually two simulation engines: Simulink [11] and Dymola [12] are integrated into IWT platform.

4.2. IWT e-learning Platform

IWT is an innovative software solution, (both at technologies and methods and functionalities level) it is modular and extensible; consequently it becomes the foundation for building up a virtually infinite set of applications for either traditional or innovative e-learning. IWT can be extended and customized at various levels:

1. Models and Strategies. IWT allows flexibility in the management and representation of all the e-learning entities process by using the following models: knowledge model, student model and didactic model.
2. Contents type and Aggregation modality. In IWT it is possible to use a wide range of Learning Objects (LO), managed by special object drivers. IWT also supports aggregate drivers managing simple LO aggregates which make use of their object driver functionality.
3. Functionality. In IWT you can add new functionalities to the platform by implementing the plug-ins. The new services can be implemented from scratch (internal plug-ins), or they can be imported from external engines (external plug-ins).

The IWT logical architecture is shown in figure 1. The Data layer uses two storage methods: a file system (Object Repository) and a relational Data Base (IWT DB). The object repository structure is based on the classical folder and file organization. The Infrastructure presents the base services to the up layers by means of API (Application Program Interface). New plug-ins can be add to the Infrastructure level to supply new functionalities. If the functionality is built from scratch then all the logical services are implemented in the Internal Plug-In. Instead, if the

services use the platform external functionality, to enable the new functionality in IWT we need to implement the Plug-In Driver/External Plug-In couple.

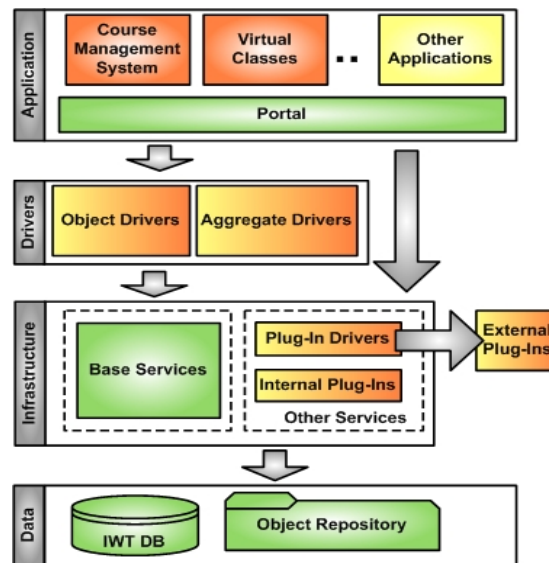


FIGURE 5: IWT Logic Architecture

The Object Drivers and Aggregate Drivers are situated in the Drivers layer. The Object Driver component manages a particular LO category (i.e. lessons, tests, simulations, etc.). On the LO request, the Object Driver can access to the base or other services in the Infrastructure layer. Through the Object Drivers and Plug-Ins, it can manage a virtually infinite LO types in IWT. In the platform, beyond the Object Drivers for the single LO control, there is another driver category to manage the complex objects derived from the LO aggregation process. An example of LO aggregation can be found in a course. The course is a LO sequence, where each LO is controlled by its Object Driver. To organize this aggregate objects in IWT, we have recourse to the Aggregate Drivers. The Application layer contains the specific platform application. The application can be realized by a Portal personalization. The portal is a dynamic panels container that supplies the Web-Based user interface to enable the services in the underlying layers. For complex application, where for example we need to compose more services outputs, the services can be called directly from the code portal by the specific API.

From the technology point of view, IWT has been entirely implemented in Microsoft .NET environment, using ASP.NET language (for the front-end) and C# language (for the back-end).

4.3. IWT for VSE simulations

In this paragraph we describe the IWT components developed to create and to deliver the simulation LO for VSE. We have developed two Internal Plug-In, two couples Plug-In Driver/External Plug-In and two Object Driver as showed in figure 6. The VMF (that is the Mathematical Model Editor) and the Display can be loaded in a browser web as plug-ins. The VMF component allows the Dymola [2] and Simulink [3] visual library access and their management. The teacher can build its own new libraries and icons. To use the Display in the model creation phase, by adding traditional simulation engines icons, the IWT user can work using the virtual sinks icons. A virtual sink icon is a symbol that is not understood by the simulation engines but it takes a meaning in IWT that, in the simulation phase, redirect the output towards the Display virtual sink. With the VMF the teacher builds the simulation model on line.

Dymola and Simulink are stand alone applications; they do not manage the multi-users access and it is not possible to run or build simulations from remote client. Libraries and models description are stored in the local simulation engine file system. In our approach, we have designed two Web Services (External Plug-In); they provide, as services towards the externals components, all Dymola and Simulink functionalities and the operations to manage the remote file system on the servers where the simulation engines are installed. The Simulink and Dymola Object Driver use the new services through Plug-In Drivers. Simulink and Dymola Object Drivers use the Infrastructure services to manage the simulation LO. IWT initializes the Object Driver instance with the teacher information (LO creator). Thanks to this information Dymola and Simulink Object Drivers can access to the IWT DB and pick the data up with the path of the teacher remote repository folders. With this information the simulation LO

Object Driver can correctly invoke the Plug-In services of simulation engine and VMF. The simulation LO Object Drivers are also responsible for the LO metadata management. Through the Base Services, the Object Driver can access to the IWT Repository and save/load the LO metadata information. The services for the simulation output visualization are grouped in the Display Plug-In. The Drivers in the up-level require to the Display Plug-In a web enabled component, to render the output into textual or graphics mode. Actually, we have developed a .NET Windows Forms component, that allows the textual and 2D graphics visualization, and we have a prototype of a component for 3D rendering based on managed DirectX 9 render engine.

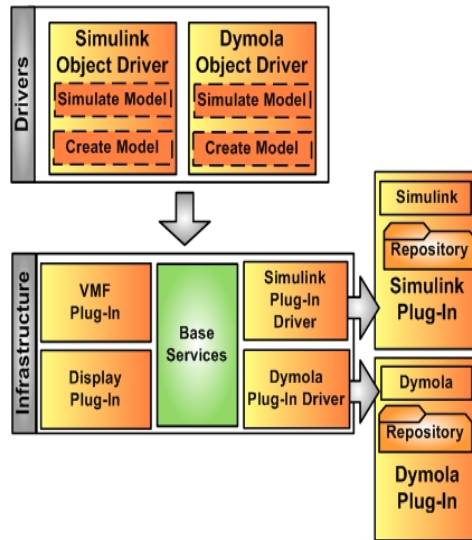


FIGURA 6: IWT and components for VSE

5. Conclusion

In this paper we presented a methodology that starts from the examination of the defined didactic model for VSE and identifies what features must have the building, executing and visualizing tools to support VSE. Didactic model analysis, done, phase by phase allowed us to define the tools that can produce the simulation contents that can interact both with e-learning environment and main actors involved in the learning process. To implement the proposed tools, we used the e-learning environment IWT. Taking advantage of its features of extensibility, the platform provide facilities to make easy the building of simulation learning objects to insert them in on-line courses immediately

6. REFERENCES

- [1] Fishwick, P.A. (1996). "Web-Based Simulation: Some Personal Observations", Proceedings of the 1996 Winter Simulation Conference, Coronado, CA, 8-11 December, pp. 772-779.
- [2] Page, E.H. (1998). "The Rise of Web-Based Simulation: Implications for the High Level Architecture", Proceedings of the 1998 Winter Simulation Conference, Washington, DC, 13-16 December 1998, pp. 1663-1668.
- [3] Page E.H., Opper J.M. (2000). "Investigating the Application of Web-Based Simulation Principles within the Architecture for a Next-Generation Computer Generated Forces Model", Future Generation Computer Systems, Volume 17 Issue 2, Amsterdam, Netherlands, October 2000, pp. 159 – 169, Elsevier Science Publisher B. V., Amsterdam.
- [4] Juan de Lara, Manuel Alfonseca (2003). "Visual Interactive Simulation for Distance Education", Simulation: Transactions of the Society for Modeling and Simulation International. January 2003, 79(1): pp. 19-34.
- [5] Capuano N et Al, PEG 2003, An Intelligent Web Teacher System for Learning Personalization and Semantic Web Compatibility, Proceedings of the 11th International Conference on Powerful ICT Tools for Teaching and Learning, St. Petersburg, Russia.

[6] Centre of Excellence "Metodi e sistemi per l'Apprendimento e la Conoscenza" Research Group - University of Salerno (2003). "Theoretical foundations for e-learning environments direct to Virtual Scientific Experiments", 3rd International LeGE-WG workshop - Online Educa Berlin 9th International Conference.

[7] Vadim Engelson, 2000, Tools for Design, Interactive Simulation, and Visualization of Object-Oriented Models in Scientific Computing, Linköping Studies in Science and Technology, Dissertation No. 627.

[8] Advanced Visual Systems Inc., AVS/Express. Reference Manual. <http://www.avs.com>

[9] IBM, Open Visualization Data Explorer, <http://www.research.ibm.com/dx/>

[10] Bill Hibbard, Vis5D, <http://www.ssec.wisc.edu/billh/>

[11] Dawn Tilbury et al, 22 September 1998, Control Tutorials for MATLAB and Simulink: A Web-Based Approach, Addison-Wesley Pub Co.

[12] <http://www.dynasim.com/>