Modeling and Simulation Framework for Development of Interactive Virtual Environments

Konstantin V. Knyazkov¹ and Sergey V. Kovalchuk¹

¹ITMO University, Saint-Petersburg, Russia
constantinvk@gmail.com, kovalchuk@mail.ifmo.ru

Abstract
The article presents a framework for interactive virtual environments’ development for simulation and modeling of complex systems. The framework uses system’s structural model as a core concept for composition and control of simulation-based scientific experiments not in terms of technological processes or workflows but in terms of domain-specific objects and their interconnection within the investigated system. The proposed framework enables integration and management of resources available within a cloud computing environment in order to support automatic simulation management and to provide the user with an interactive visual domain-specific interface to the system.

Keywords: simulation environment, workflow management, interactive workflow, system-level simulation

1 Introduction
Today computer-aided modeling and simulation serves as a powerful scientific tool, which can be used to investigate complex system within different problem domains. This stimulates development of modeling and simulation tools to be considered as important issues of e-Science (Hey, Tansley, & Tolle, 2009) area focused on technological support of scientific research. Contemporary simulation tasks are often characterized by complex structure and require different resources (software, hardware, data etc.) to be integrated within a single solution. Common approach for composite application development usually exploits the workflows (WF) concept to describe dependencies between different elements of the application (Gil, et al., 2007). To support the development and usage of WFs there exist a lot of workflow management systems (Yu & Buyya, 2005). Still the direct workflow development and support may become a rather difficult task for end users as they often are domain specialists with a lack of technological background (McPhillips, Bowers, Zinn, & Ludascher, 2009). As a result the development of conceptual and technological toolbox for support of simulation-based scientific experiments is one of the urgent issues of e-Science (see e.g. projects (McPhillips, Bowers, Zinn, & Ludascher, 2009) (Gil, et al., 2011) (Gubala, Bubak, Malawski, & Rycerz, 2006)), but the general approach for appropriate domain-specific support isn’t defined yet. On the other hand the simulation-based experiments often define the need of dynamic changes of simulation parameters.
This causes a concept of computational steering (Mulder, van Wijk, & van Liere, 1999) to appear as an approach to manage interactive simulation process. This approach is extended in direction of dynamic measurements' processing within the Dynamic Data Driven Applications Systems (DDDAS) paradigm (Daremä, 2004). This paradigm allows real-time measurements data to be dynamically incorporated into a simulation process. And moreover it allows application to dynamically steer the measurement process. Additionally the computational steering approach takes into account the ability of the system to provide the user with interactive visualization (see e.g. project (Ribicic, Waser, Fuchs, Bloschl, & Groller, 2013)) as a way to perform the user interaction within corresponding problem domain with a natural interpretation of visualized objects' meaning. The interactive visualization brings a set of additional issues to be managed (Doleisch, 2007) including focus and context analysis, iterative and interactive features' specification, multiple views' management etc. Nevertheless considering a scientific platform for simulation-based experiments, two coupled processes need to be developed in a tight relationship in order to provide the end user with full support: a) the simulation process, presented as a WF structure, is to be managed automatically, providing the user with domain-specific interaction toolbox; b) domain-specific interactive visualization needs to be managed in a general way to support semantically arranged data representation in a cognitive way which is easily understandable to the end user. In this paper we present conceptual and technological framework which enables development of such platforms using semantically supported description of the system. It allows synchronous management of the mentioned coupled processes to be performed. This framework is supposed to make a shift toward system-level exploration (Foster & Kesselman, 2006) instead of common used WF-based approach by allowing the user to manage interactively both system's structure, simulation process and analyze the visualized simulation results in a natural way without deep technological background required.

2 Interactive Framework for Modeling and Simulation

2.1 Requirements

Interactive virtual environment provides the user with the required facilities for deep immersion into simulation-based scientific experiments in a distributed computing environment. This statement defines the main goals of this work:

- Shift from imperative process-centric task description approach (which is represented in classic workflow formalism) to declarative system's model description.
- To hide complex technical issues from the user (caused by complexities of the e-Science domain, e.g. heterogeneity, large-scale data) by technical and domain knowledge acquisition and automating it's processing with semantic technologies.
- To provide the user with a convenient interactive environment exploiting all the benefits of contemporary human interface devices (HID).
- Give the user a rich cognitive visualization environment where system's elements are represented with abstract visual metaphors.

Considering the defined idea of the platform for visual interactive scientific experimentation several requirements can be defined:

1. **Framework structure.** The developed technology should be able to manage interactively the following artifacts during the simulation process: a) semantic description of the system's structure; b) composite application; c) data, obtained from external sources and produced during the simulation process; d) visual scene, representing the explored system.
2. **Semantic structure.** The system's structural model should be defined and managed by the end user. In order to support the user within these procedures the domain semantic should be used to hide the technological complexity of composite application development (in a form of WF). As a result the user should be able to change the system's structure during the simulation process.

3. **Dynamic experimentation.** The user should be able to change parameters of simulation process during the execution of constructed composite application a) by explicit definition of the parameters; b) by interaction within the visualized scene. These changes should have immediate affect onto the data which is used to build the visualization scene.

4. **Data representation.** Available data should be dynamically combined to build the proper visual scene presented to the user. The visualization space should give the user ability to interact with the scene i.e. explore the represented data; modify system's structure or parameters of simulation.

5. **User interaction.** User analysis should be supported by interactive visualization with ability to change scenario of system's development. Moreover actions performed by the user should be taken into account by the workflow management system in order to gain high reactivity on the next user's action (using prediction of the user activity on management of ensemble of possible scenarios).

### 2.2 Conceptual approach

The requirements mentioned above define the conceptual structure and general architecture considered within this and following sections.

In order to reach the goals and meet the requirement we propose a framework for interactive virtual environment development. Framework is a domain-independent set of technologies and tools. After it is deployed and filled with domain-specific elements (data, knowledge, software) it turns into interactive environment. Like a distributed operating system the framework provides an access to hardware, software and data through the user interface (UI). Using the UI the user can compose structural system's model and run it on the high performance computing hardware. User-side hardware includes different input and output devices connected to the environment. Structure of the virtual environment based on the framework is presented in the figure 1.

The framework core is composed of several layers.

1. **Formal knowledge layer** contains tools and technologies for environmental elements’ description and for description of theirs integration process. Knowledge represented in imperative and declarative form is processed at this layer. For example, before resources can be integrated into the environment, they should be described and all the information such as access credentials, functional characteristics should be provided in a formal form.

2. **Communication layer** of the framework provides capability to connect different elements of the environment (software at runtime and in batch mode), to deliver data from sources to consumers. Data representation (formatting, protocols) is performed here. At this layer it is possible to form workflows which can be run in batch or interactive mode.

3. **Execution layer** provides capability to use elements of the environment. Relying on the formal knowledge, the execution layer performs actions in order to harness computational resources and software. At this layer it is possible to run composite applications in distributed environment.

4. **Semantics layer** is the second descriptive layer along with the formal knowledge layer. Using the semantic declarations, different software and complex scenarios can be connected correctly to solve domain problems.

5. **Meta-level simulation engine** forms the central part of the framework. This is an active entity which orchestrates the work with system's model and execution of workflows. Using semantic data the engine glues all elements of environment together: it interprets system's model, interacts with user through UI, and manages execution of a set of workflows by communicating with WMS.
All of the presented layers together allow integration and consolidation of all the external resources into one distributed environment with unified interface. User’s resources are perceived: as ordinary resources, which can be used within a workflow’s execution to deliver interactive software to the user (e.g. execution of Matlab package on a user’s PC); as a connection point for input and output devices (to create a virtual environment). Number of user’s devices is not limited: user can view and interact with visualization process on a touch table, modify structural system model on a tablet PC and run interactive data processing software on a desktop.

2.3 User Interaction

All user interaction facilities within the framework are grouped into two perspectives. Structural system’s model perspective is a special environment for creation of system’s models. This perspective contains a workspace and a toolbox with virtual objects which were prepared in advance. Within the perspective it is possible to move objects from toolbox to workspace, specify values of their properties, and set connections between objects. Objects being instantiated became active, and actions associated with them became accessible.

The second perspective represents the visual virtual environment where all the domain-specific elements related to task can be projected. Each type of object or data gets a proper visual representation depending on their semantic description. For example, a data file with an urban road map will be presented as an image visualized automatically using certain visualization software.

Below the list of interaction methods accessible to user through both perspectives is presented:

1. Object’s properties modification before simulation and in runtime;
2. System’s structure modification before simulation and in runtime;
3. Controlling model time by changing speed of simulation, pausing, resuming;
4. Moving user’s focus in model time and in parameter space;
5. Changing the view scale in case of multi-scale models;
6. Applying domain-specific actions available in interactive simulation packages.
Some positions of the list require description. Domain-specific actions available to the user contain: (a) interactive capabilities already existing in visualization software packages, (b) interactive capabilities which exist only in simulation packages without a visual abstraction, (c) high-level complex actions. In the first two cases it is presupposed that the simulation software already has interactive interface and can be used in computational steering scenario. The last case is provided by formalization and automation of low-level actions. For example, “shut off a road” action in traffic modeling scenario could be already implemented in traffic visualization software (a). This action could be supported by simulation package and should be reflected in virtual environment during its implementation (b). And in case if we have separated batch simulation package and map editor, this action can be emulated by creation of automated map modification procedure which will at first invoke map editor and then simulation package.

User can engage in the simulation and change the scenario during runtime by changing parameters values, applying domain-specific actions in timescale of a model. For example, in application of urban flood simulation the user can investigate different scenarios of evacuation in a city by simply closing different groups of roads in time. It should be noted that this capability requires providing the advanced provenance service which will track all the user’s actions in order to restore all the history which led to certain simulation result. Other metaphor which can be implemented using the framework is taking control of model’s element by the user in time and space. For agent-based models this abstraction is native: the user projects his own behavior into virtual environment. E.g. in case of evacuation simulation the user can take control of one agent and try to evacuate.

![Figure 2: Branches of system’s state and a user’s focus (circle)](image)

Work with ensemble runs and control of parallel branches of parameter sweep are important features in the interactive simulation-based experiments. User’s actions at runtime change the state of the system. Moving backward in time or cancelling the actions are usually not supported in simulation packages. Fast switch between branches of parameter sweep will increase the environment’s interactivity. In this case automatic support can be provided in order to optimize reactivity of the system. Let the state of the model to be characterized by two parameters X and Y (see the figure 2). User can change values of the parameters interactively. At the moment of value’s change the system can automatically start two additional branches of simulation with other values of X from its domain. This will provide the capability for the user to move the focus in X space later without restarting the model. Advanced scenario of the resource allocation here can be the following. The user has a pool of resources which start running the model in parallel. The state of parallel models is the same. During the interaction models acquire different state in order to provide the user with wide space for parameter study.

The basic structure of the user interface within the interactive simulation environment is presented in the figure 3. On the left side of the scheme there are software packages being executed in a distributed computational environment. Software packages and data are projected to the visual environment.
2.4 Framework Architecture

**Technological background.** Interactive simulation framework is built on the top of several technologies developed in e-Science Research Institute with participation of authors:

1. Cloud computing platform **CLAVIRE** (CLoud Application VIRtual Environment), which enable execution of composite applications defined in a form of Abstract WFs (Knyazkov, Kovalchuk, Tchurov, Maryin, & Boukhanovsky, 2012). This platform provides high-level knowledge-based access to computational infrastructure and supports domain-specific definition of software calls which are translated into executable form automatically.

2. **Interactive WF (IWF)** (Knyazkov, Nasonov, Tchurov, & Boukhanovsky, 2013) is a technology which extends basic WF, executed within CLAVIRE with blocks which dynamically exchange data during the execution. This technology forms the basis for interaction between simulation and visualization tools as well as implementation of the user interaction scenarios.

3. **Virtual Simulation Objects (VSO)** (Kovalchuk, Smirnov, Kosukhin, & Boukhanovsky, 2012) is a concept and technology which uses system-level semantic description of the objects being simulated to generate WFs within CLAVIRE automatically. Being based on the conceptual hierarchy corresponding to the general structure of e-Science simulation tasks it can be used to develop high-level user-friendly representation of simulation tasks and obtained results.

Combination of such technologies allows developing knowledge-based solutions for simulation-based exploration of complex systems (Smirnov, Kovalchuk, & Boukhanovsky, 2013). Within the presented work we are trying to develop general purpose framework for building such systems. Figure 5 shows basic structure of main blocks of this framework based on technologies mentioned above. The framework support interaction with the user, visualization and control of external resources. The VSO
technology is extended to support dynamic (re-)building of system’s structural model, arrange a model state with related data according to domain semantics, and correct representation of system’s state during WF management. Extended IWF technology controls simultaneously the user interaction scenarios, composite application’s execution and external data sources.

![Figure 5: Basic technological architecture](image)

Architecture scheme of the framework is depicted in the figure 4. This scheme shows the multi-layered structure of the framework core. Components of the scheme are described below in order of attachment to the layers of the framework core.

**Formal knowledge layer.** Execution of software in a distributed computational environment requires formal descriptions of workflows, software packages and resources. Workflow description in the CLAVIRE platform is represented by the scripts written in EasyFlow language (DSL). Such description carries the defined domain-specific simulation scenario in a form of declarative knowledge. Software packages are described by scripts in EasyPackage DSL. The description includes general information about the package, declaration of the high-level input parameters and procedures of transformation to low-level format-dependent input files. Along with information about input parameters, description contains similar information on output parameters and a section of execution parameters (command line, parametric performance model) and communication ports of the package. So, all the formal knowledge on how to process a package and communicate with it is stored in the description. Resources are described according to the same approach – by text files. These three components are responsible for storing of knowledge represented in corresponding descriptions: Workflow Base, Package Base, Resource Base.

**Communication layer.** Communication is performed within the environment in two ways: interactive communication between elements through the network, information exchange in batch mode. The second way is based on the classical WMS facilities which allow to handle data and to run software in distributed environment. The CLAVIRE’s package description mechanisms enables the unified package interface declaration which allows to connect different packages together seamlessly at the workflow level. Another way of communication is based on the capability of software to communicate at runtime using network. It is provided within the extended IWF model. Interactive software packages can work in a network environment due to declared communication technologies in a package description.

The important feature of a virtual environment is flexible communications which are set up by dynamic switching. The role of connection manager is performed by meta-level simulation engine (see below).

**Execution layer.** Workflow management system forms a large part of the framework. It is represented in a scheme with three blocks. Resource Controller provides a direct access to different types of resources. Workflow enactor processes single tasks and control their execution on the resources through resource controllers. Workflow manager service interprets composite applications defined in terms of Abstract Workflows into Concrete form.
Semantic layer. The knowledge base contains a semantic description of the domain in form of VSO. Using the formal knowledge about software, scenarios and data, the semantic description integrates all of these parts into semantic structures – virtual objects.

Meta-simulation engine. Meta-simulation engine is a service providing backend functions to the user interface through session mechanism. Engine acts as a mediator between the user and a computational environment. It translates the user’s actions to technical procedures in terms of workflows. So, an action made in the user interface may require a single package run, a workflow run, and communication with an interactive package according to IWF technology. In this way one session may require many workflows to run. So, the meta-simulation engine is in charge of: VSO’s interpretation, tracking of user’s actions, running of workflows, management of currently running workflows using CLAVIRE API.

3 Application

The proposed approach has been tested in a task of agent-based urban traffic simulation. The test case application includes several software packages. There are several software utilities: map format converter, road graph separation software for distributed simulation case, batch map and population data visualization package, map editor with GUI. Traffic demand simulation package relies on a statistical data and generates agents with their individual daily activities. Agent-based traffic simulation package can be coupled with visualization package or run separately in distributed mode (in order to achieve high performance). Simulation package is configured by classes of agents participating in traffic and their amount. Interactive visualization software provides the user interface and supports touch screen devices. Visualization software coupled with simulation provides a function of road closing at runtime.

The set of accessible application scenarios contains: traffic dynamics simulation in a city with interactive closure of roads, investigation of traffic load in case of different scenarios of settlement, evaluation of different evacuation scenarios, etc.

The software packages were installed to the CLAVIRE platform and were semantically described using VSO technology with special tool – “VSO editor”. Structural system’s model for that application contains the following objects: City infrastructure, Virtual society, Vehicles, Traffic dynamics. City infrastructure is characterized by road network map and map of buildings, which are extracted from files in OpenStreetMap format. Virtual society object describes population living in city. This object requires statistics data about dwelling of the simulated area. Vehicles represent a hierarchy of agents which take part in an urban traffic. Each class of agent in a model is represented in a single object in VSO with its properties and models.

For interpreting of the system’s model “VSO Workspace” was used. It is a web-application which can be used to create a system from the existing set of objects. The scheme of structural system’s model for the traffic simulation application is depicted in the figure 6. After interpretation the workflow script in EasyFlow language is generated and transferred to the CLAVIRE platform. Simulation of traffic can run with interactive visualization software on a user’s PC or on a touch table.

The implemented system provides the following interactive capabilities:

- Modification of the structural system’ model by adding and removing of objects, changing connection between objects. For example, addition of new agents’ classes, change of a simulation scenario.
- Changing data by using of external editors, e.g. JOSM for OpenStreetMap files.
- Changing objects’ parameters and models. For example, traffic demand model can be changed in the object “Society”. Default daily activity-based agent behavior model can be changed in case of “bus” agents to timetable-based model.
• Visualization of input data using software packages in CLAVIRE. For example, visualization of population data with Gnuplot software.

• Interactive visualization coupled with simulation. Visualization supports: model’s time control actions (pause, change speed, restart etc.), closure of roads, changing the visibility for different layers of the map (map, road network, agents).

Figure 6: System’s model for a traffic simulation application
4 Conclusion

The shift to a system-level simulation concept based on the hybrid models integrated into one environment requires new tools and technologies for the end users. In this work we propose the framework which can be used to build interactive virtual environments for such simulations. The framework relies on several conceptual parts: workflow management with the support of interactive workflow model, semantic descriptions for the environment’s elements, visualization and interaction facilities. Implementation of the proposed framework applicable as a general purpose solution is an ongoing work. Nevertheless parts of the proposed framework were used for development of several research solutions including showed traffic application and marine objects application (Kovalchuk, Smirnov, Kosukhin, & Boukhanovsky, 2012). Implementation of the integrated visual environment and development of the advanced meta-simulation engine which provides transparent workflows’ management and intelligent user support will be a next step of our research.

Acknowledgements: This work was financially supported by the Government of the Russian Federation, Grant 074-U01.

References

Darem a, F. (2004). Dynamic data driven applications systems: A new paradigm for application simulations and measurements. In Computational Science - ICCS 2004 (Vol. 3038, pp. 662-669). Springer Berlin Heidelberg.

Doleisch, H. (2007). SimVis: Interactive visual analysis of large and time-dependent 3D simulation data. IEEE Winter Simulation Conference, (pp. 712-720).

Foster, I., & Kesselman, C. (2006). Scaling system-level science: Scientific exploration and IT implications. IEEE Computer, 39 (11), 31-39.

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., et al. (2007). Examining the Challenges of Scientific Workflows. IEEE Computer, 40 (12), 24-32.

Gil, Y., Ratnakar, V., Kim, J., Gonzalez-Calero, P., Groth, P., Moody, J., et al. (2011). Wings: Intelligent Workflow-Based Design of Computational Experiments. IEEE Intelligent Systems, 26 (1), 62-72.

Gubala, T., Bubak, M., Maławski, M., & Ryczek, K. (2006). Semantic-Based Grid Workflow Composition. Lecture Notes in Computer Science, 3911/2006, 651-658.

Hey, T., Tansley, S., & Tolle, K. (2009). The fourth paradigm: data-intensive scientific discovery.

Knyazkov, K. V., Kovalchuk, S. V., Tchurov, T. N., Maryin, S. V., & Boukhanovsky, A. V. (2012). CLAVIRE: e-Science infrastructure for data-driven computing. Journal of Computational Science, 3 (6), 504-510.

Knyazkov, K. V., Nasonov, D. A., Tchurov, T. N., & Boukhanovsky, A. V. (2013). Interactive Workflow-based Infrastructure for Urgent Computing. Procedia Computer Science, 18, 2223-2232.

Kovalchuk, S. V., Smirnov, P. A., Kosukhin, S. S., & Boukhanovsky, A. V. (2012). Virtual Simulation Objects concept as a framework for system-level simulation. IEEE 8th International Conference on E-Science (e-Science), 1-8.

McPhillips, T., Bowers, S., Zinn, D., & Ludascher, B. (2009). Scientific workflow design for mere mortals. Future Generation Computer Systems, 25 (5), 541-551.

Mulder, J. D., van Wijk, J. J., & van Liere, R. (1999). A survey of computational steering environments. Future generation computer systems, 15 (1), 119-129.

Ribicic, H., Waser, J., Fuchs, R., Bloschl, G., & Groller, E. (2013). Visual Analysis and Steering of Flooding Simulations. IEEE Transactions on Visualization and Computer Graphics, 19 (6), 1062-1075.
Smirnov, P. A., Kovalchuk, S. V., & Boukhanovsky, A. V. (2013). Knowledge-Based Support for Complex Systems Exploration in Distributed Problem Solving Environments. *Communications in Computer and Information Science, 394*, 147-161.

Yu, J., & Buyya, R. (2005). A Taxonomy of Workflow Management Systems for Grid Computing. *Journal of Grid Computing, 3* (3-4), 171-200.