A Framework For The Development Of Distributed Simulation Code Oriented To The Manufacturing Field

Antonio Grieco, Francesco Nucci, Massimo Pacella, and Alfredo Anglani
Dipartimento di Ingegneria dell'Innovazione, Università degli Studi di Lecce,
Via per Monteroni, 73100, Lecce - ITALY

ABSTRACT
Simulation models are becoming very complex software applications. This is mainly due to the increasing complexity of the analyzed systems composed by numerous groups of coordinated interacting elements. Nowadays simulation is shifting to a new paradigm: the DCC (Distributed Component Computing). The applications of this simulation paradigm are client/server running–programs using collaborating distributed components. These components can be located on different platforms under different operating systems, and heterogeneous simulation languages can be used to develop such components. In this paper, a framework for the integration of discrete event simulation components is presented. This framework is the focus of a research project whose aim is to design and develop an open-source distributed environment for the cooperation and integration of different simulation languages and tools.

Keywords: Discrete event simulation, distributed component computing.

1. INTRODUCTION

Discrete event simulation theory is a well-known necessary and profitable design tool to support activities of systems configuration and management. Simulation is used in the optimization of the system as well as in the evaluation of its performance and in the designing of new configurations [1,2,3].

The growing complexity of simulation projects involves the increasing both of the required simulation devices, and the heterogeneity levels among them. To address this complexity, a redesign of the future simulation software has to be performed. Several works are studying the application of component–oriented paradigms to develop simulation models in the manufacturing field [4,5]. In particular, the work performed in the Object Management Group (OMG) led to the definition of the Computer Integrated Manufacturing (CIM) framework, which is a distributed component–oriented for the manufacturing environment [6].

Different studies investigated in distributed component computing for discrete event simulation theory. In [7] the major combined simulation languages are classified into six approaches. The approaches are useful for understanding the historical development and current direction of combined simulation modeling. In [8] the focus is the efficiency of simulation experimentation for optimization. A framework for combining the statistical efficiency of simulation optimization techniques with the effectiveness of parallel execution algorithms is presented. In [9], a new C++-based framework in order to integrate discrete event simulation models is proposed. In [10], the analysis of the performance of a specific distributed simulation environment based on DEVS/HLA (discrete event simulation specification/ high level architecture) is reported. A clock synchronization algorithm for parallel road-traffic simulation is proposed in [11]

A promising approach to re–design simulation applications is to adopt the Distributed Component Computing (DCC) paradigm in which monolithic software systems are being replaced by a collection of different components [12]. In particular, heterogeneousness is a valuable feature of the DCC paradigm. Theoretically, heterogeneity permits to use the best combination of hardware and software elements. In the simulation field, DCC paradigm allows the opportunity to use the most suitable simulation tool in order to model each specific part of the analyzed system. The DCC paradigm enables the developer to implement complex simulation models by simply connecting a set of simulation modules [4].

Even if the potential of the discrete event simulation are evident, (e. g. by considering the number of the commercial tools and languages available on the market implementing the different aspects of discrete event simulation), a little attention has been reserved by the academic research to the integration of different simulation languages and methodologies.

In the last decade, in order different works focused on the combination of simulation models. For example, the High Level Architecture (HLA – IEEE standard 1516) [13] defines a framework that permits interactions among various simulation components. The aims of HLA are mainly to get an interoperability of the simulators and to reuse components over a large number of applications. HLA framework is able to establish the technical foundation for the combination of sub–models on the same planning level [14]. Nevertheless, HLA does not solve the problem arising when different simulation paradigms (e.g. object-oriented and transaction-oriented models) are coupled. For example, in the simulation of complex systems, the automatic cooperation among different simulation modules can be the way in which different languages and paradigms can be effectively integrated.

The Italian Minister of Education and Research (MIUR) has recently founded a research project (2001-2005 prot. RBNE013SWE) within the Investment Fund for Basic Research (PNR 2001-2003, FIRB art. 8, D.M. 199 Ric. 2001) concerning the designing and the developing of an open-source distributed environment for the cooperation and integration of different simulation languages and tools.

The main proposal goal is to design and develop an open-source distributed environment for the cooperation and integration of different simulation languages and tools. The objective of the project is to develop an environment for the integration and collaboration of different tools and packages with a significant time reduction in the building and validation phases of complex distributed simulators.

This paper presents a description and some preliminary results of such a project. In particular, the paper presents a General Simulation Framework (GSF) whose aim is describing each component produced during various designing/implementing phases. The environment definition for simulating complex systems requires to clearly identifying each activity occurring in the simulation model designing.

The definition of the GSF is based on a specific approach providing a "general" abstract notation, which is domain-independent, along with specific concrete notations, one for each domain. The abstract notation is defined as a simulation-oriented "reuse" of UML (Unified Modeling Language). Domain-specific notations are obtained by means of the customization facilities of UML in order to present the framework in a way suitable to the different domain experts. In this way, for each domain, it is possible to define a specific notation as a core transformation. Once such transformations are defined, users can work using their own notations, relaying on the core (hidden) notation to check and validate their models.

2. THE PROJECT

2.1 Project aims

The integration of different simulators, based on different languages, may be the solution for the study of complex systems. For example, cooperation between different simulation languages allows users to simulate complex systems, in which each system component is simulated with a proper simulation language. Hence, it is possible to use closed software simulation units (that do not allow to identify implementation details) to be integrated within complex modeling scenarios. These units (i.e. black boxes) can reach high degree of detail.

The aim of the project is to reduce costs, using existing simulators, while applying software engineering methods, as well as code re-usability and interoperability, within the modeling and simulation activities.

The arising issues are of two classes:

- *Homogenous:* dealing with the possibility to establish a tight connection between the system design phase and the simulation phase, in order to allow changes in the system design as a feedback from the following simulation activity. In other words, it is the possibility to have a homogeneous framework that drives the simulation phases (from the design activity to the actual simulation run)
- *Heterogeneous*: dealing with the possibility to interconnect existing simulators, in case, developed using different languages or tools and/or hosted on different workstations connected by LAN/WAN networks.

The second class is particularly important. Indeed, no unique monolithic simulator can completely fulfill the necessity to analyze complex heterogeneous real systems. This is valid especially when considering the simulation of systems characterized by a high degree of complexity. Relevant advantages can be achieved combining the results from several simulation blocks, each designed for a particular item of the overall system. Furthermore, the integration technique grants the possibility to exploit different simulation tools, each one chosen accordingly to the particular advantages that can be achieved. Hence, as new information technology tools are developed, several proposals for simulation support environment have been made, in which different components are. Therefore, an important role consists in the evaluation of the way these existing tools impact on such environments, in order to browse their capabilities and to test the feasibility of the interactions.

The main proposal goal is to design and develop an open-source distributed environment for the cooperation and integration of different simulation languages and tools. The components integration will be based on general and dedicated standard software (in particular the Common Object Request Broker Architecture CORBA), and it will provide all the necessary specifications and tools for the integration of different simulators in a unique environment. By means of the new environment, it will be possible the integration and collaboration of different tools and packages with an evident time reduction in the building and validation phases of complex distributed simulators. Although, as demonstrated in the following, this is a very interesting academic research goal, the future consequences in the industrial field may be very promising. In fact, many companies provide black box simulation packages concerning manufactured system components to deal with industrial patents related, for example, with implementing details. Only by means of new distributed and collaborative environment, it will be possible to integrate different component in complex system simulation.

2.2 Project structure

Four Research Units (RUs) are involved in the project.

The Research Unit of the University of Insubria Varese-Como is composed of Faculty members (Full Professors, Associate Professors and Researchers) and of personnel that will be hired for carrying out the activities foreseen in the project.

The main expected goals (in cooperation with other units) are:

1) the definition of the information model and of the reference architecture for the environment;
2) the design and the implementation of the communication infrastructure for simulators;
3) the design of the simulation sequence generator tool.

The Research Unit of Politecnico di Milano is composed by 8 professors and 10 researchers and young researchers. Main objectives of the RU are the following.

1) Definition of a descriptive model for representing real manufacturing systems.
2) Definition of a descriptive model of the functioning of the manufacturing systems' simulation software model.
3) Implementation of a standard language for the representation of management policies in the simulation of manufacturing plants.
4) A specification of a production system, chosen as a case study, will be developed. The specification, written in UML and Trio, may be simplified but overall realistic and will be used for the simulation activities.
5) A tool for execution sequence generation will be built and applied to the case study, in order to build and validate a simulation module. The developed tool will be portable, flexible, easy-to-extend and to be easily interfaced with other tools.

The research unit of the University of Calabria is established at the Department of Electronics, Informatics and System of the University of Calabria and collaborates with the Center of Excellence in High Performance Computing of the same University.

The aim of the research activity of is the design and implementation of optimization methods for manufacturing systems. The validity of such methods will be tested by means of simulation. In the design of modern manufacturing systems, particular attention has to be devoted to the selection of the factors affecting costs and performance. These factors can be related to the configuration of the physical systems (e.g., number of machines, the choice between several machine or a flexible one) or can concern management parameters (storage policies, rules of dispatching, kanban number).

The research unit of the University of Lecce is established at the Engineering for Innovation Department of the same University. The RU will be involved, in collaboration to the other participant RUs, on the development of every Work Packages. Nevertheless, the activities on which the RU of the University of Lecce will be mainly involved are the following ones.

1) Analysis and development of frameworks and models for the manufacturing domain.
2) Implementation of a development and management software environment for the distributed simulation.
3) Implementation of software communication tools for the distributed simulation.
4) Optimisation and management methods for resources and production policies in manufacturing environment by means of simulation.
5) Methods and tools for the HLA and web services integration.

3. THE ARCHITECTURE

The project aims at designing and developing an open-source distributed environment for the co-operation and integration of several simulation packages. The architecture is reported in Figure 1. The ORB module manages several Hw platforms, in case, by using Synchronous System Control (SSC) in order to endure temporal constrains [15] [16]. Time advance mechanism is a critical aspect in distributed simulation environments, because all the necessary simulation data have to be shared in a not-unique scenario. Furthermore, the temporal variable is a common data shared among all the simulation actors. For this reason the SSC play an essential role in the reported framework. Essentially, the GFS is based on three packages, namely: the *System Framework*, *Simulation Framework* and *Deployment Framework* (see Figure 2). The fundamental classes of the System Framework have been depicted by Figure 3. In practice, *System Architecture* can be considered as a hierarchical collection of *Architectural Elements*. The basic class of System Architecture is the *Object* representing a physical component of the real system to be modeled.

The *Simulation Framework* package, depicted by Figure 4, reports the environment resources, which can be used for simulation. Each *Simulator* is hosted by specific software *Platforms*, running on a specific hardware system (*HwPlatform*).

Finally, the *Deployment Framework* package (Figure 5) reports details of the simulation interfacing. In particular, each *Simulation Element* owns a proper Simulation Interface, which is based on a specific set of Simulation Gates. A *Simulation Gates* represents either the input (*Simulation In Gate*) or the output (*Simulation Out Gate*) exchange point for a specific *Simulation Element*. Two Simulation Gates might be connected by means of a Simulation Link in order to exchange input/output data. A special interface (Synchronous Simulation Interface) is designed for exchanging time data between Simulation Elements instead.

Figure 6 shows an overall view of the proposed framework. It can be noticed that users implement both System Architecture and Subsystems, which consist of several Architectural Elements (e.g. a simulation Object). Practically, when two simulation modules have been located on different platforms, they can exchange data by means of the Simulation Gates. From the user viewpoint, a Simulation Gate is a particular module of the environment in order to link an external simulation component although it has been implemented in a different language on several platforms. Once the simulation components have been fully implemented in the specific language and located on the specific platforms, user models the overall simulation architecture by defining the links among the different simulator components.

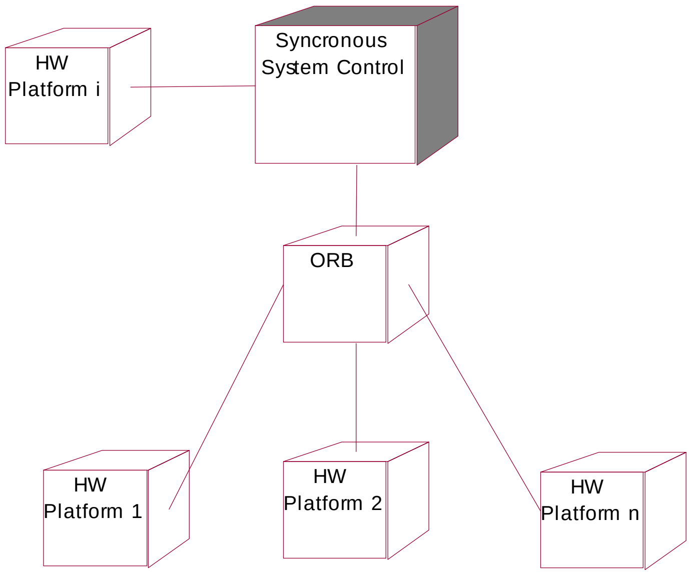

Figure 1: UML deployment view for a generic distributed component simulation

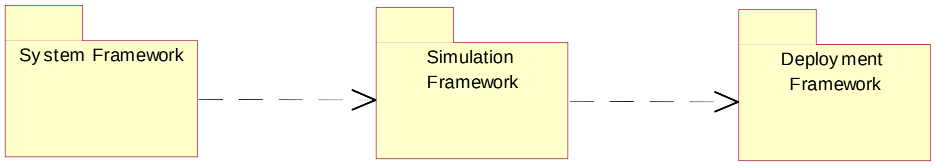

Figure 2: Main Packages Of Distributed Component Environment

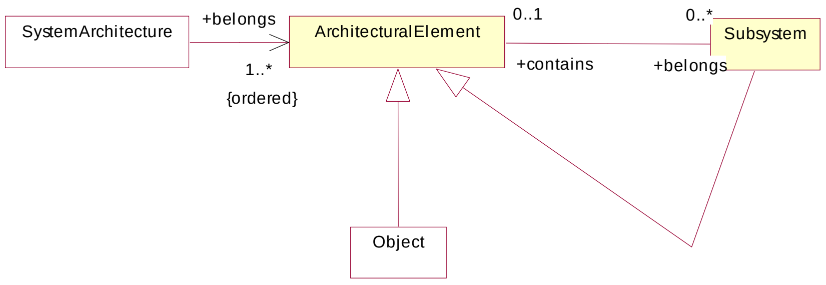

Figure 3: System Framework Package Details

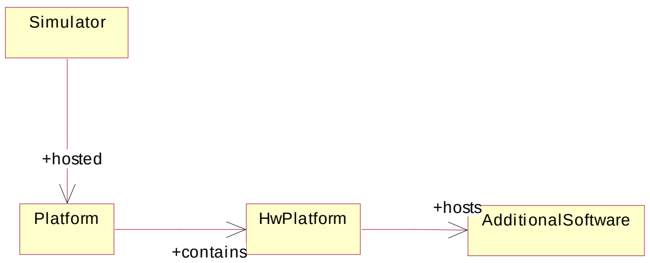

Figure 4: Simulation Framework Package Details

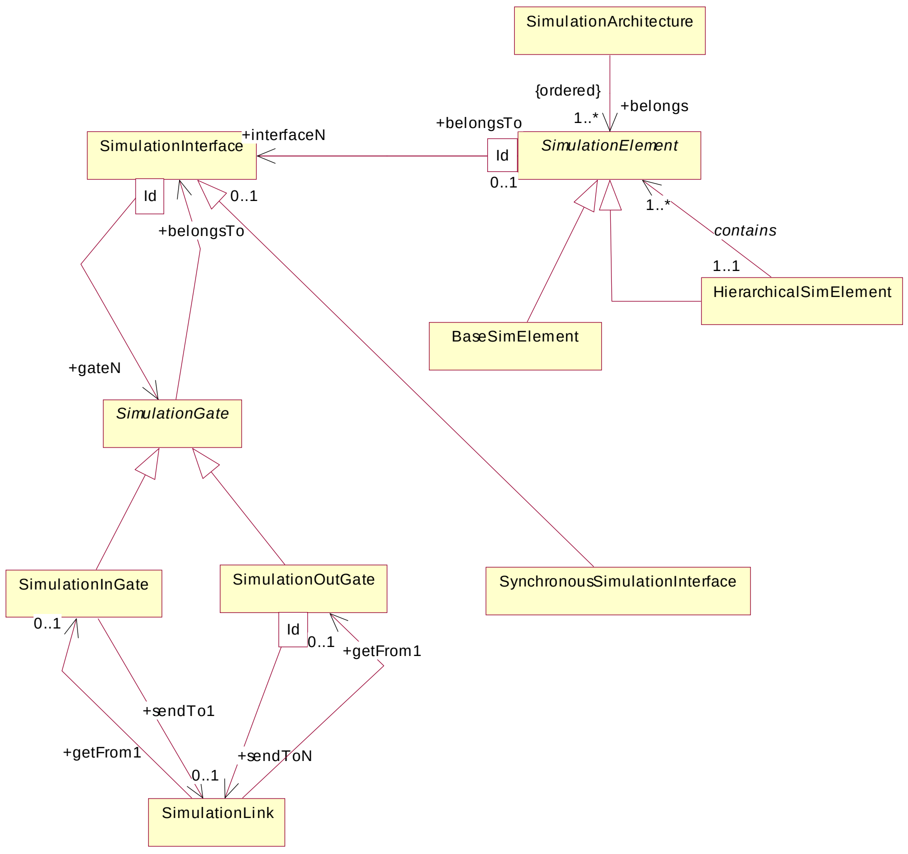

Figure 5: Deployment Framework Package Details

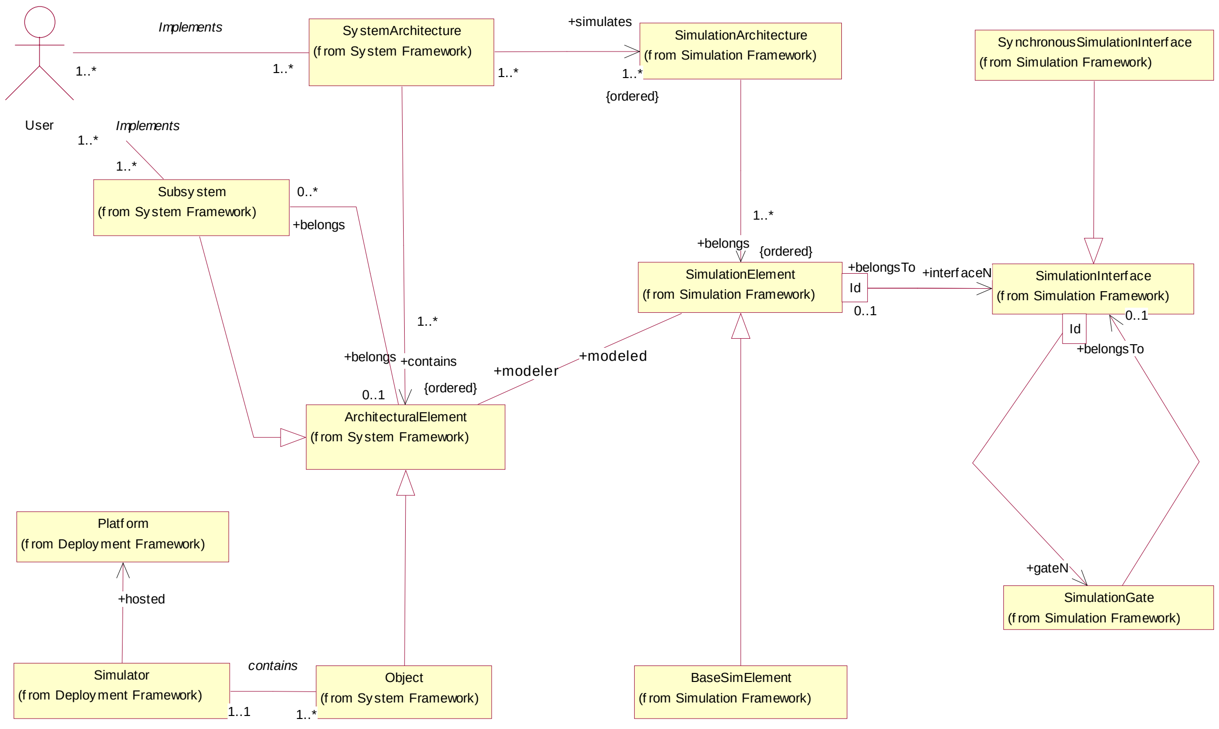

Figure 6: Class Diagram Of The Distributed Component Environment

4. SIMULATION FRAMEWORK IMPLEMENTATION

This section provides a description of *Simulation Framework* implementation that has been developed in a common object-oriented language. The implemented Simulation Framework can be referred as DEOS (Discrete Event Object Simulation). DEOS has been implemented in order to supply a C++ class hierarchy able to provide a tangible support for the development of discrete event simulation models.

The basic DEOS concepts - that have been developed in form of C++ class – are the timeline and the event. A simple framework has been created in which resources and entities of the simulation model can be described.

The class hierarchy is basically based on three types: event handling, timeline handling and entity/resources handling. The TZSimulationEvent class is the base to perform event management (see Figure 7a).

The handling of timeline consists in managing a priority queue containing scheduled events. The basic attribute for the extraction of a generic element from the queue is the occurring time. Fundamental classes for the timeline handling are reported in Figure 7b.

Classes standing for entities and resources have a common parent in the class hierarchy: the TZSimulationObject class. This one represents a generic simulation object and it is linked to the Timeline through the correspondent events. TZSimulationObject child classes - TZEntity and TZResource - manage entities and resources (see Figure 7c).

Moreover, class hierarchy is also made of support classes in order to manage special issues, such as simulation errors, statistical distribution properties, collection of simulation output, etc…. In order to use the above described hierarchy class, a visual framework is provided. In such a context it is possible to allocate and connect resources, set parameters, run simulations, watch and export simulation results. For this reason, special classes are developed (see Figure 8): TSimulationForm manages the graphical representation of the simulation model; TClassBox supplies a graphical representation of a resource (a box that can be moved or deleted - moreover class properties are available through it); TClassConnector connecting resources in order to establish simulation flows; TObjectInspector, for showing and setting class parameters.

This architecture enables the programmer to develop new classes to be used in the simulation model building. A basic set of classes has been provided for building a variety of simulation models. TZCreator (TZDestroyer) simulates the creation (disposal) of an entity. TZMachine stands for a machine with one input, one output and a given processing time. TZBranch, depending on a fixed condition, put the input on one of the two available outputs. Instances of the described classes can be used in a simulation model by dragging the correspondent graphical elements on the desktop, linking them each other and setting their properties. An additional custom component can be used by: (1) creating the new class as a descendant of the TZResources, (2) implementing the management of the attributes during the simulation running, the statistical information concerning a simulation run, (3) linking the class with the special ones mentioned above.

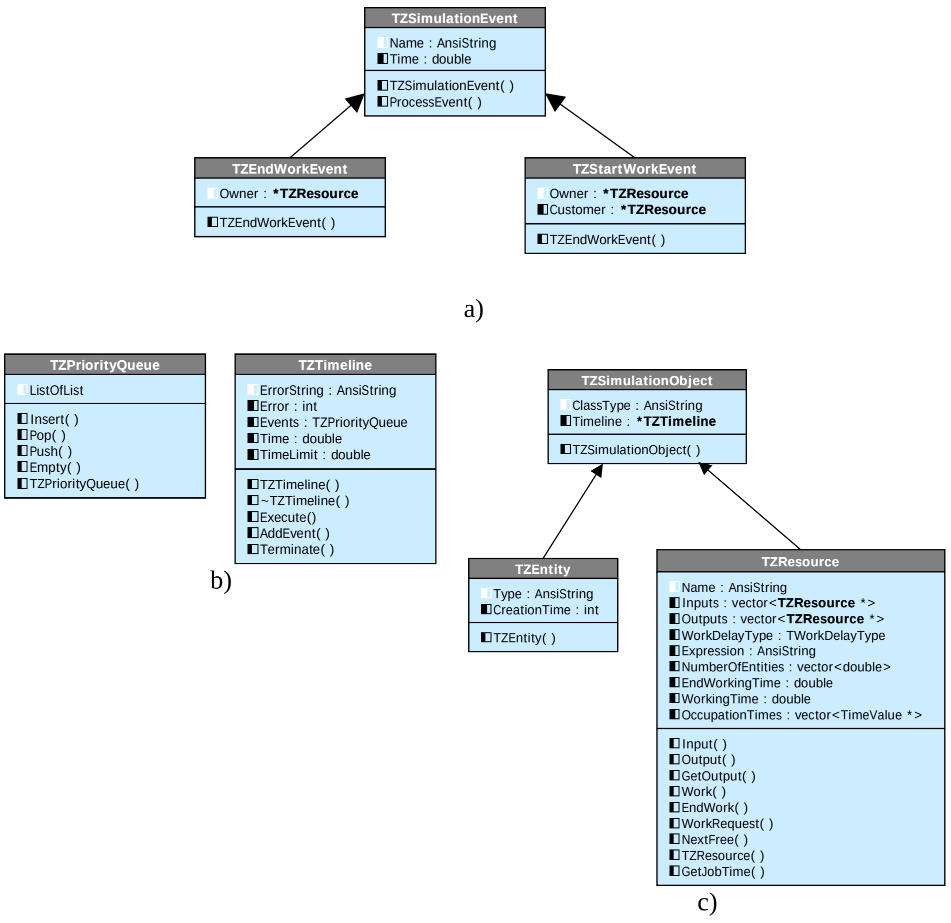

Figure 7: Fundamental Classes In DEOS

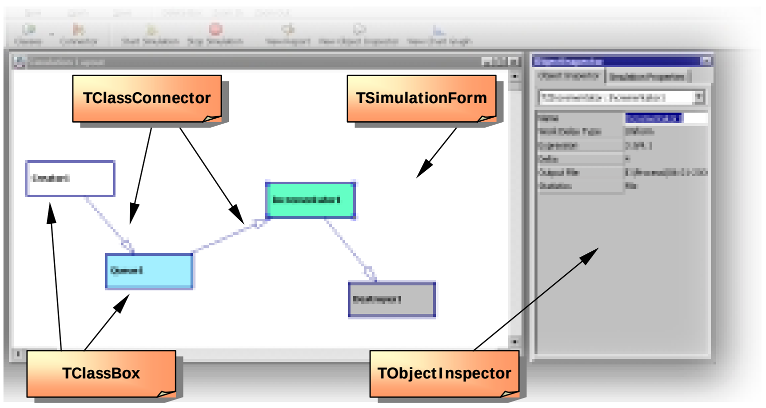

Figure 8: DEOS's Visual Framework.

5. CONCLUSIONS

The project is actually in its designing phase. Afterwards, the framework that has been briefly described in the paper will be implemented in an open programming system environment by using general-purpose programming language (as C++ and/or Java). The integration of the simulation components will be achieved by implementing a suite of abstractions and of simulation services, which may be based on the distributed component platform technology CORBA. This will allow the integration system also to work within the Internet. Finally, the system will be tested in order to validate it as a distributed simulation environment for the manufacturing field.

Acknowledgements: This research has been funded by Italian Minister of Education and Research (MIUR) whiting the project FIRB titled "Architetture e tecnologie informatiche per lo sviluppo ed evoluzione di software open-source per la simulazione a componenti distribuiti, orientate al settore manifatturiero" (2001-2005 prot. RBNE013SWE PNR 2001-2003, FIRB art. 8, D.M. 199 Ric. 2001).

6. REFERENCES

1. Kellert, P., Tchernev, N., and Force, C., "Object Oriented Methodology for FMS modelling and Simulation", *International Journal on Computer Integrated Manufacturing* 2, no. 6, 1997 pp. 405-434.
2. Klein, U., "Simulation-based distributed systems: serving multiple purposes to composition of components", *Safety Science* 35, 2000, pp 29-39.
3. Zhang, T., Dewey, A., Fair, R., "A hierarchical approach to stochastic discrete and continuous performance simulation using composable software components", *Microelectronics Journal*, 31, 2000, pp 95-104.
4. McArthur, K., Saiedian, H., and Zand, M., "An evaluation of the impact of component-based architectures on software reusability". *Information and Software Technology*. 44, 2002, pp. 351-359.
5. Maguire, L.P., MgGinnity, T.M., and McDaid L.J., "Issue in the development of an integrated environment for embedded system design. Part B: design and implementation" *Microprocessor and Microsystems*, 23, 1999, pp 199-206.
6. CIM Framework Architecture Guide 1.0., available at http://www.sematech.org
7. Yasser, D., and Chell, R.A., "A review and classification of combined simulation", *Computers & Industrial Engineering*, 32, no. 2, April, 1997, pp. 251-264
8. Enver, Y., Yuh-Chuyn, L., Chun-Hung, C, and Insup, L., "Distributed web-based simulation experiments for optimization", *Simulation Practice and Theory* 9, no. 1-2, October 15, 2001, pp. 73-90.
9. Gaujal, B., Jean-Marie, A., Mussi, P., and Siegel, G., "High speed simulation of discrete event systems by mixing process oriented and equational approaches", *Parallel Computing* 23, no. 1-2, April, 1997, pp. 219-233
10. Lee, J. S., and Zeigler, B. P., "Space-Based Communication Data Management in Scalable Distributed Simulation", *Journal of Parallel and Distributed Computing* 62, no. 3, March 2002, pp. 336 – 365.
11. Namekawa, M., Satoh, A., Mori, H., Yikai, K., and Nakanishi, T., "Clock synchronization algorithm for parallel road-traffic simulation system in a wide area" *Mathematics and Computers in Simulation*, 48, 1999, 351-359.
12. Sheremetov., L. B., and Smirnov, A. V., "Component integration framework for manufacturing systems re-engineering: agent and object approach", *Robotics and Autonomous Systems* 27, 1999 (pp. 77-89).
13. High Level Architecture. Available at https://www.dmso.mil/ public/ transition/ hla/, (September 2002).
14. Wilcox, P. A., Burger, A. G., and Hoare, P., "Advanced distributed simulation: a review of developments and their implication for data collection and analysis". *Simulation Practice and Theory* 8, 2000 (pp. 201-231).
15. Grieco, A., Pacella, M., and Anglani, A. "Integration of heterogeneous discrete event simulation tools by means of CORBA", *Annual conference of the Italian Society for Computer Simulation Proceedings*, Naples (Italy) December 2001, pp. 61-69.
16. Anglani A., Grieco, A., Pacella, M., and Colizzi, L., "A proposal of a distributed component environment for the integration of simulation models", *European Simulation Symposium (SCS-ESS02) Proceedings*, Dresden (Germany) October 2002, pp. 382-386.