Proceedings of the 1992 Winter Simulation Conference
ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson

REQUIREMENTS FOR A REPOSITORY-BASED SIMULATION ENVIRONMENT

Tuncer I. Ören, Douglas G. King, Louis G. Birta

Computer Science Department
University of Ottawa
Ottawa, Ontario K1N 6N5, Canada

Martin Hitz

Institute for Statistics and Informatics
University of Vienna
Vienna, Austria

ABSTRACT

Simulation tools and environments can provide several types of computer assistance. It is envisaged that this functionality is going to increase. Even though this increase of functionality is highly desirable, it is anticipated that there will be an interface problem between the simulation tools and environments, similar to the interface problems in computer-aided software engineering (CASE) tools and environments. The problem is diagnosed and a solution is proposed.

1 INTRODUCTION

Simulation environments already provide several types of assistance to the user in modelling, model-base management, specification of the simulation experiments, program generation based on high level specifications, execution and monitoring of simulation runs, other types of behavior generation, such as qualitative simulation, optimization, statistical and rule-based inferencing, as well as symbolic processing of models for model analysis and transformation Ören 1992a, b).

Simulation which contributed to other fields also benefits from the synergistic effects of their developments. Accordingly, knowledge-based simulation environments and other "intelligent" or "cognizant" simulation environments came into existence. In the first *knowledge-based simulation environments* the emphasis was on modelling and experimentation. The following generation is *comprehensive simulation environments* where more model-based functionalities are (or will be) provided. *Integrated simulation environments* provide (or will provide) functionalities germane to several fields of study such as computer-aided design(CAD), computer-aided software engineering (CASE), expert systems, computer-aided system theory (CAST), computer-based systems engineering (CBSE), qualitative simulation, and symbolic algebra.

Integrated simulation environments, though very desirable, will have two fundamental limitations: (1) They will be closed systems and therefore installation of new tools and/or environments will be difficult, if not impossible. (2) The complexity of the interface among several environments will increase with the number of existing tools and environments.

The relief of the user from the complexity issues raised by the use of application specific tools by individual users in an integrative simulation environment is the goal of this work. Systems engineering tools available today are dominated by those designed and built for the networked UNIX-based workstation market. As such, any proposed solution must be applicable to such an environment.

2 REPOSITORY APPROACH

A *repository* is a database system that supports the integration of CASE tools (Barton 1991, ECMA 1990, Thomas 1989). Already several repositories exist (Soufflet 1990, Blum and Kastner 1990). The integration of CASE and other tools can occur at two main levels of abstraction. In either case, the means by which the integration is realized should be hidden from the user. The tool interactions should be seamless and transparent.

The tools could be integrated at the *presentation level.* This means that they would be able to use the same presentation management system. This is easily accomplished if the tools are implemented using the same windowing system, e.g., X Windows, in such a way that interfacing with other tools is not prohibited.

The tools must be integrated at the *application level.* This means that they are able to share data, exchange messages, and otherwise cooperate to provide an integrated system. Ideally, the impact on existing source code should be minimal.

3 PROPOSED APPROACH

The proposed (and prototyped) system is based on the Platform Reference Model as described in the IEEE P1175 report (IEEE 1989, Poston 1989). The underlying architectural basis is as shown in Figure 1, with a platform consisting of hardware, an operating system, and various platform services (Xwindows, repository,...)

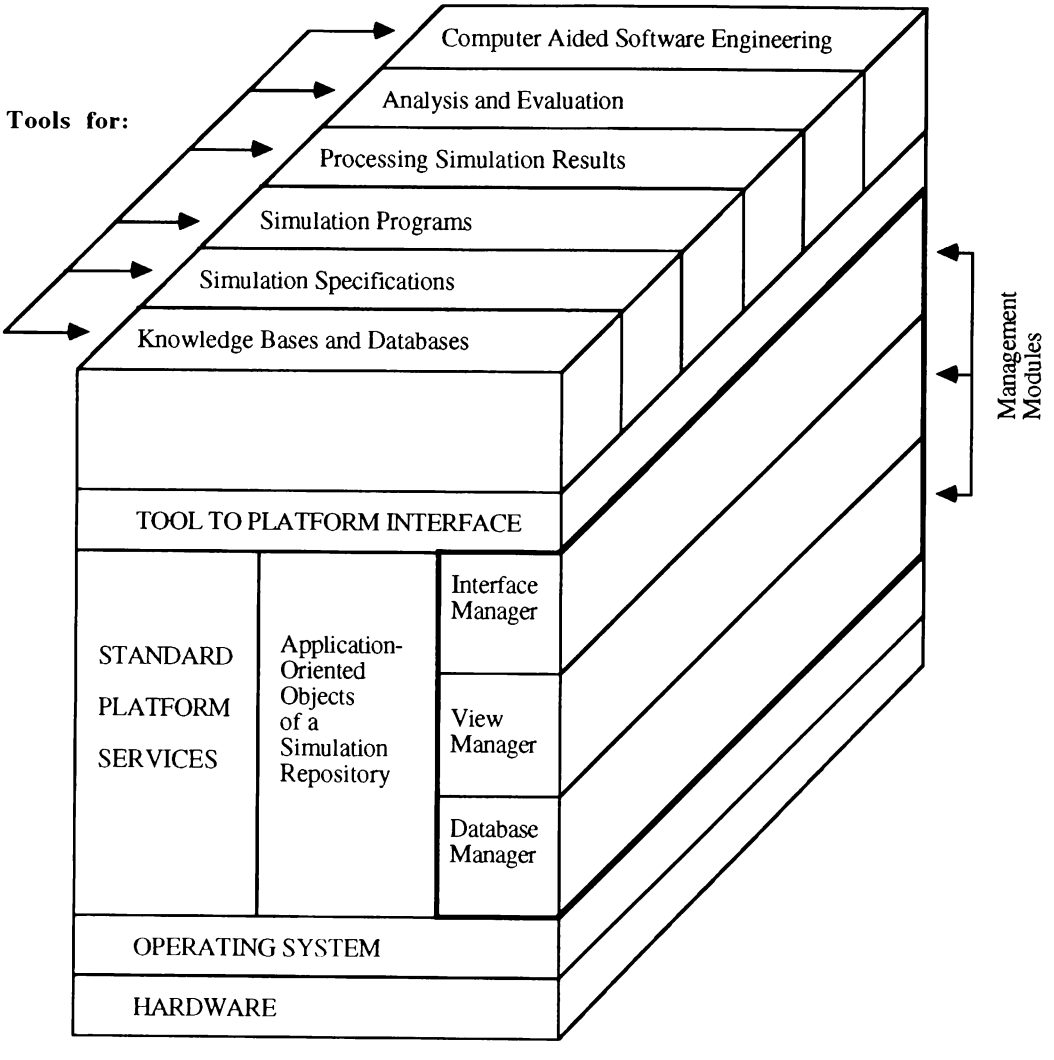

Figure 1: Platform Architecture for Repository-Based Environment

(Hitz et al. 1992, Ören et al. 1991). The tools which are to be integrated are considered as forming a layer which communicates with the platform using well-documented procedures, and communicate with each other via the platform.

This architecture implies that the Platform should provide the repository service. In our system the repository forms part of the high-level services provided by the platform.

The main advantage of our proposed architecture for a repository-based simulation environment, is its ability to provide multiple views for each managed object in the repository. Each tool which is installed in the repository defines its own representation of the objects which it will store/share by deposition/extraction into/from the repository. Each query to the repository is interpreted based on the identity of the requester tool. The request is processed with the identities of the receivers controlling the transformation from internal storage format into the toll-specific form.

In this way, tools and environments gain a significant advantage form automatic, transparent, seamless, view translation. The complexity of integration is greatly reduced.

The Platform Reference Model can be interpreted so that the layer of tools can be distributed throughout a network, and the underlying platform can also be distributed. This means that two or more users working at separate locations can work in a "you cut, I'll paste" relationship in which one user produces output and stores it in the repository, while another user takes the input from the repository and uses it. The storage associated with the repository could be at a third location.

The distribution mechanisms are provided in the Prototype so that the addresses of all participants are Internet Domain TCP/IP addresses. This allows complete transparency of the distribution.

This elegant distribution mechanism is provided using an underlying communication mechanism based upon the Client/Server model. The communication mechanism is provided such that each tool using the repository is considered a client with a unique identity. The client participates in a message passing protocol with one or more servers. Each of the servers can maintain storage, and each of the servers could be physically resident on separate machines. The client can start several overlapping sessions with various servers. In effect, the tool has connected to several repositories at once.

Each repository maintains a knowledge base with information about each tool which has been installed. Tools can be installed to each repository independent of the other repositories. Each tool is identified by a unique Tool ID which is used by the repository management system to determine the individual tool's representation of the objects it uses. Each access to the repository (deposit, extract, or query) is interpreted based on the identity of the requester.

Every effort has been made to allow easy installation of tools into the repository. The impact on source code is minimal. There is a requirement for changes to the source code which correspond to opening a session with a particular repository, the accesses themselves given by a single procedure calls, and the ending of the session with the repository. Typical source code changes are dependent on the number of objects to be managed using the repository. Accesses are explicit and are not transparent to the tool builder.

The definition of each tool's representation of objects is accomplished using schemas for the individual tool. The mappings between schemas (for data sharing of otherwise incompatible types) is provided using a set of view translators. The view translators are created during installation based on a specification provided by the installer (with on-line human-meditation). The view translators are used by the view manager together with the knowledge base of data schemas. The identity of the tools is used to trigger transformations between the internal storage format and the tool-specific forms. This is a very nice application for an embedded expert system.

The internally stored format is transparent to the user. The prototype uses the "first installed" form for each conceptual data type as its definition of internal format. All subsequent definitions require generation of view translators between the new forms and the internal format. This keeps the complexity level low. Future enhancements may provide buffering of data in a cache before translation to internal storage, version management, keeping of statistics on how many accesses are made using each schema, and performance of restructuring optimizations of the internal storage format.

5 CONCLUSION

The main advantage of the repository management system described herein is its ability to provide multiple views for each managed object in the repository. This allows automatic, transparent, seamless, view translation which can greatly aid in the production of integrative environments.

The complexity of integration of complex tools is greatly reduced, encouraging reuse of existing tools. Fine granularity of tool functionality is encouraged, thereby enhancing reusability of future tools by not encouraging the building of huge, monolithic, closed environments.

The next generation of simulation environments should be built using an integrative approach. This can be accomplished by using a repository integration mechanism with multiple views for each data object.

ACKNOWLEDGMENT

The work reported in this article has been carried out as part of a broad project funded by Bell Canada. The authors wish to express their appreciation both for this financial support and the encouragement and cooperation of Mr. F. Coallier and Mr. O. Tanır of Bell Canada. The support of the Austrian Research Council (Fonds zur Förderung der wissenschaftlichen Forschung) under contract number J0521-PHY made the contributions of Dr. Martin Hitz possible.

REFERENCES

Barton, R. 1991. Linking CASE tools with a repository. *CASE Trends*, 3:2, 14-15.

Blum, O. and A. Kastner. 1990. The ATMOSPHERE architecture. *PCTE Newsletter*, 3 (Feb.), 5-8.

ECMA. 1990. A reference model for frameworks on computer-assisted software engineering environments. ECMA TR/55. European Computer Manufacturers Association, Genewa, Switzerland.

Hitz, M., D.G. King, and T.I. Ören. 1992. A prototype of a UNIX-based repository management system. Technical Report TR-92-21, Dept. of Computer Science, University of Ottawa, Ottawa, Ontario, Canada.

IEEE. 1989. A standard reference model for computing system tool interconnections. IEEE P1175 Draft 4.4.

Ören, T.I. 1992a. Advances in knowledge-based simulation systems. In *Proceedings of the Symposium on Advances in Simulation '92.*, ed. A.R. Kaylan, M. Draman, and T.I. Ören. July 6-7, 1992, Istanbul, Turkey. Boğaziçi University, Department of Industrial Engineering, Istanbul, Turkey.

Ören, T.I. 1992b. Simulation environments: Challenges for advancement. In *Proceedings of the 2nd International Conference on System Simulation and Scientific Computing*, Oct. 20-23, 1992, Beijing, China.

Ören, T.I., M. Hitz, D.G. King, L.G. Birta, and O. Abou-Rabia. 1991. A repository based simulation environment: Rationale and a prototype architecture. Technical Report TR-91-17, Dept. of Computer Science, University of Ottawa, Ottawa, Ontario, Canada.

Poston, R.M. 1989. Proposed standard eases tool interconnection. *IEEE Software*, 6:11, 69-70.

Soufflet, D. (1990). Emeraude V12. PCTE Newsletter, 3 (Feb.), 4.

Thomas, I. 1989. PCTE interfaces: Supporting tools in software engineering environments. *IEEE Software*, 6:11, 15-23.