An environment for developing securely interoperable heterogeneous distributed objects

M. Berryman, C. Rummel, M. Papa, J. Threet, S. Shenoi
Department of Computer Science
University of Tulsa, Tulsa, Oklahoma 74104, USA
sujeet@utulsa.edu

John Hale
School of Electrical Engineering and Computer Science
Washington State University, Pullman, Washington 99164, USA
hale@eecs.wsu.edu

Keywords
Distributed objects, secure interoperability, high assurance, process calculus, execution model

PROJECT DESCRIPTION

The heterogeneity and volatility of open distributed systems make high assurance security an elusive goal. One solution is to provide developers with tools for designing and implementing robust object systems with verifiable behavior in open environments (Cleaveland *et al.*, 1994). The Meta-Object Operating System Environment (MOOSE) (Hale *et al.*, 1997) is intended to support the development, execution and verification of secure heterogeneous distributed systems.

MOOSE uses a layered architecture (see, e.g., Zhang *et al.*, 1995) and dual operational and verification frameworks to blend object technology with formal methods (Figure 1). The foundation of MOOSE is provided by the Robust Object Calculus (ROC), a process calculus (see, e.g., Milner *et al.*, 1989;

Database Security XI T.Y. Lin and Shelly Qian (Eds)

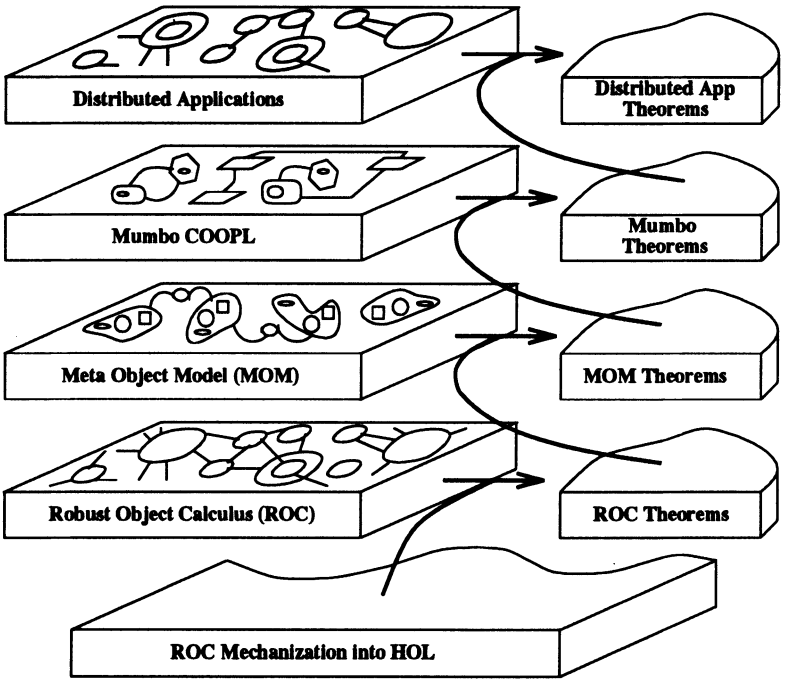

Figure 1 The Meta-Object Operating System Environment (MOOSE).

Nierstrasz, 1991) for modeling and reasoning about distributed objects. The Meta-Object Model (MOM), defined using ROC, forms the next layer of the operational framework. It is a primitive distributed object architecture (see, e.g., Agha, 1986; Houck and Agha, 1992) for constructing more sophisticated object models and programming languages that constitute the upper levels of the operational framework. MOM implements a capability-based security model of access control for distributed objects. Capabilities, which are unforgeable tokens, are modeled in ROC by unique names that are not visible and cannot be reproduced.

MOM is used to design Mumbo, a concurrent object-oriented programming language (COOPL) for orchestrating the secure interoperability of heterogeneous resources in open systems. Mumbo employs wrapper technology and abstract specifications to integrate native components, while translators provide mappings from high-level languages to ROC, permitting source-level integration. Mumbo uses MOM's security model to support discretionary access control (DAC) for software components. It also provides new language constructs for constraining class and object protocols, giving developers more control over component communication patterns.

The MOOSE verification framework complements the operational framework. The mechanization of ROC into Higher Order Logic (HOL) (Melham, 1992; Gordon and Melham, 1993) permits reasoning about distributed systems modeled in the operational framework. Each verification framework layer contains theorems about the corresponding layer in the operational framework.

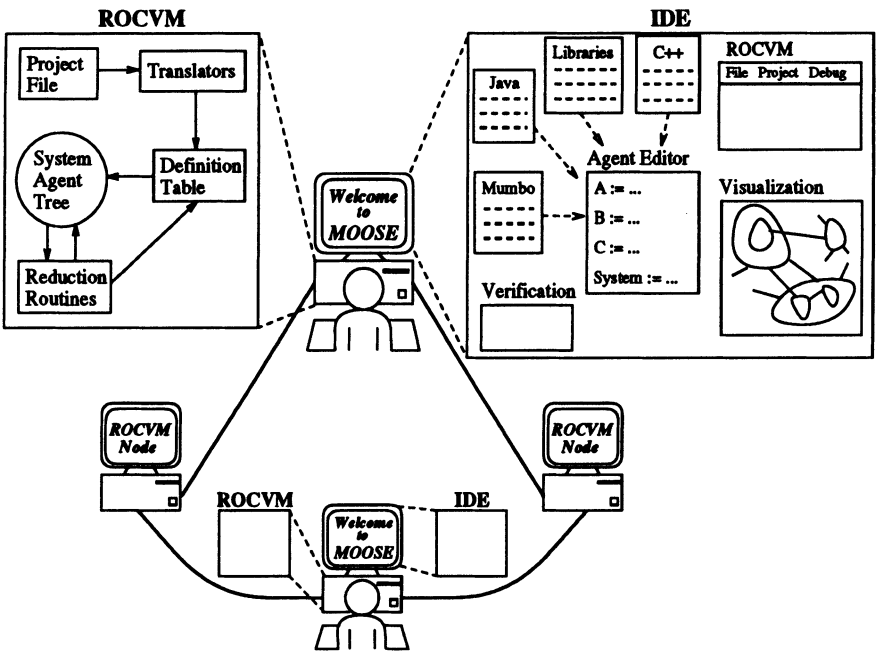

Figure 2 MOOSE implementation.

Figure 2 shows a schematic diagram of the MOOSE implementation. Implemented in Java to permit operation on heterogeneous platforms, MOOSE comprises two main components, a ROC Virtual Machine (ROCVM) and an Integrated Development Environment (IDE). ROCVM is designed to execute (reduce) ROC expressions, thereby simulating the execution of ROC-modeled applications in heterogeneous distributed environments (Hale *et al.*, 1997). The IDE provides an interactive graphical interface for intuitive visualization and analysis of distributed systems modeled with ROC.

Plans for future work include implementing the Mumbo coordination language and a distributed version of the ROCVM reduction engine that would permit multiuser engagement. The IDE will be extended to incorporate the HOL theorem-proving environment and tools for formal specification and verification. Also, popular object languages and architectures, e.g., Java and CORBA, will be mapped to MOM so that developers can use Mumbo to integrate these systems seamlessly and securely into their own applications.

Acknowledgement This research was supported by MPO Grants MDA904-94-C-6117 and MDA904-96-1-0115 and OCAST Grant AR2-002.

REFERENCES

Agha, G. (1986) *Actors: A Model of Concurrent Computation in Distributed Systems.* MIT Press, Cambridge, Massachusetts.

Cleaveland, R., Gada, J., Lewis, P., Smolka S., Sokolsky, O. and Zhang, S. (1994) The Concurrency Factory – Practical tools for specification, simulation, verification and implementation of concurrent systems, in *Specification of Parallel Algorithms* (eds. G. Blelloch, K.M. Chandy and S. Jagannathan), American Mathematical Society, Providence, Rhode Island, 75–90.

Gordon, M. and Melham, T.F. (eds.) (1993) *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, Cambridge, U.K.

Hale, J., Threet, J. and Shenoi, S. (1997) A framework for high assurance security of distributed objects, in *Database Security, X: Status and Prospects* (eds. P. Samarati and R. Sandhu), Chapman and Hall, London, 99–115.

Houck, C. and Agha, G. (1992) HAL: A high level Actor language and its distributed implementation, *Proceedings of the 21st International Conference on Parallel Processing*, 158–165.

Melham, T.F. (1992) A mechanized theory of the π-calculus in HOL. Technical Report 244, University of Cambridge Computer Laboratory, Cambridge, U.K.

Milner, R., Parrow, J. and Walker, D. (1989) A calculus of mobile processes. Technical Report ECS-LFCS-89-85&86, University of Edinburgh, Edinburgh, U.K.

Nierstrasz, O. (1991) Towards an object calculus, in *Proceedings of the ECOOP'91 Workshop on Object-Based Concurrent Computing* (eds. M. Tokoro, O. Nierstrasz and R.A. Olsson), Springer Verlag, Amsterdam, 1–20.

Zhang, C., Shaw, R., Heckman, M.R., Levitt, K. and Olsson, R.A. (1995) A hierarchical method for reasoning about distributed programming languages and applications. *Proceedings of the International Workshop on Higher Order Logic Theorem Proving and its Applications*.