# Melampus

IBM Almaden Research Center

## Personnel


| | |
|---|---|
| Luis-Felipe Cabrera | cabrera@ibm.com |
| Laura M. Haas | laura@ibm.com |
| Allen W. Luniewski | luniew@ibm.com |
| Joel E. Richardson | joelr@ibm.com |
| Peter M. Schwarz | schwarz@ibm.com |
| Jim W. Stamos | stamos@ibm.com |


## Contact


Luis-Felipe Cabrera
Computer Science Department
Mail Stop K55/803
650 Harry Road
San Jose, CA 95120-6099


## Project Description

The Melampus vision is to provide a significantly more powerful, more usable, and more flexible computing environment than those currently in existence. Our goal is to provide a common framework for describing, creating, querying, and evolving all of the objects in the system. In particular, we believe that providing a comprehensive query facility is necessary so that users can locate objects and derive interesting new relationships among them.

### Motivation and Description

In current systems, the lack of a common data model for all information inhibits integration of applications. Our initial goal is to define a data model so powerful and easy-to-use that it becomes the basis for all programming activities within the system, from shells and user-interfaces to applications to systems. Following the initial design effort, we plan to implement the model and to experiment with it in order to verify its strengths and discover its weaknesses. Our experiments will include using the system for day-to-day activities as well as coding interesting applications. Our system must support a community of real users, and it must be scalable, distributed, and multi-user.

A particular focus of our data modelling efforts is to address two major stumbling blocks in the use and maintenance of current computing systems: the fact that data is hard to find and the fact that systems evolve. Locating data is a problem at all levels of a computing system. From a programmer looking for an implementation of priority queues, to a user trying to locate an old news article, to a personnel administrator trying to find employees who earn more than their managers, searching is one of the most common (and often most frustrating) ways that humans use computers. Thus, one of the central themes of the Melampus data model is to provide associative access to all entities in the system. This includes entities created and owned by users (e.g., databases, mailboxes, etc.) as well as those managed by the system (e.g., disk queues, processes, etc.).

Like the problem of locating data, the problem of evolution occurs at all levels of the system. A software system itself evolves because of improvements in the implementation, changes in the stated requirements of the system, a desire to integrate existing applications, etc. Furthermore, a software system built in Melampus must be able to model entities that evolve. For example, a database system should be able to model individuals who migrate through many roles in a lifetime, e.g., student, employee, system-user, etc. A program development environment must be able to model software systems that evolve through different versions and configurations.

To support multiple users in Melampus, we plan to provide fine-grained access control based on access control lists; logically, there is an access rights check on every method invocation. While we believe that such fine control is critical in many cases, we also acknowledge that access checking is not needed for the vast majority of object accesses in a running system. For example, temporary objects that never escape a process's stack frame do not need protection. A major challenge, then, is to provide a comprehensive access control mechanism when it is needed, while avoiding an overhead penalty when it is not.

A second interesting implementation problem concerns storage management. The problems with explicit object deletion are well-known, i.e., dangling references and the creation of garbage. Thus, many object-oriented systems are garbage-collected systems. However, since we intend Melampus to be a fully distributed system, any garbage collection scheme must also be distributed. While there have been some efforts in this area, it is not yet clear how well any of them will scale. Moreover, garbage collection suffers from a problem that we call the "squirreling syndrome." That is, large numbers of objects are neither used nor garbage-collected because users squirrel away references to objects and then forget about them. We are currently considering a scheme involving expiration dates that would allow the system to reclaim expired objects, leaving a tombstone in its place.

## References

[1] L. F. Cabrera, L. Haas, J. Richardson, P. Schwarz, and J. Stamos, "The melampus project: Toward an omniscient computing system," Tech. Rep. RJ7515, IBM Almaden Research Center, San Jose, California, June 1990.

[2] J. Richardson and P. Schwarz, "Aspects: Extending objects to support multiple, independent roles," Tech. Rep. RJ7657, IBM Almaden Research Center, San Jose, California, August 1990.

[3] L. F. Cabrera and P. Schwarz, "Operating system support for an omniscient information system," in *Proceedings of the OOPSLA/ECOOP '90 Workshop on Object Orientation in Operating Systems*, October 1990.