We have implemented several kinds of object spaces with a common runtime library that supports object communication. The implementation is written in C++ and runs on top of Berkeley Unix. The initial target language is C++ so we can use Allegro on itself.

One of the object spaces we have implemented is a set of objects that perform incremental linking by maintaining a list of the absolute addresses that depend on each symbolic address. When a module changes, only the affected addresses are relocated. We have measured a 30-fold speedup over the Unix batch linker for a change to a single module in a large program. This implementation demonstrates the benefits of using object spaces for incremental processing.

We are putting the pieces of Allegro together into a usable environment. Although our current work is oriented to developing and maintaining code, object spaces can also be used to manage requirements, specifications, documentation, and the other information that makes up software development. Our goal is to bootstrap Allegro for C++ programs by the end of the year and then extend it to support other languages and other phases of software development.

The advantage of the Allegro architecture is that it supports multiple views, incremental processing, and dependency management for large software information databases. The disadvantage is that each object space is more complex than a traditional software tool because of the support needed for transparent object communication, locking, and query processing. Our experience to date is that this complexity can be shared in the runtime library and that incremental processing yields a significant improvement in turnaround time. Only when we are using Allegro for day-to-day development will we really know how practical and effective this approach is.

Acknowledgment

This research was supported by the SUNDEC project through a gift from Digital Equipment Corp.

References


Mark A. Linton is an assistant professor in the Electrical Engineering Dept. at Stanford University. His research interests include programming environments and user interfaces.

Linton received a PhD in computer science from the University of California at Berkeley.

Address questions about this article to Linton at Center for Integrated Systems, Rm. 213, Stanford University, Stanford, CA 94305.