Dynamic Encapsulation of C++ Objects for Distributed Object-Oriented Systems

Presented By

Frank E. Barrus

What: **Masters Thesis Defense**
When: **10-11AM; Thursday, June 19, 1997**
Where: **Conference Room 10-1124**

anyone interested is welcome to attend

ABSTRACT

Classes in C++ provide static encapsulation of objects, by generating code which contains specific knowledge about the internals of encapsulated objects. Static encapsulation occurs at compile time, and therefore cannot directly support the evolution of objects, since recompilation of source code is required if the internal layout changes. This also prohibits the use of distributed or persistent objects without first ensuring that the internal representations of the objects match the ones in the compiled code.

Dynamic encapsulation occurs at run-time, and allows the compiled code to exist without the knowledge of any particular object representation. Abstract base classes with C++ virtual functions support a limited form of dynamic encapsulation, but only for objects originally designated to inherit that class. Some languages, such as Smalltalk, support dynamic encapsulation, but without the performance of statically encapsulated languages.

An object model using dynamic type-binding is presented that allows the flexibility of dynamic encapsulation with much of the efficiency of static encapsulation. With this model, objects can potentially communicate and migrate across address space and network boundaries without specific prior knowledge of representations, and can invoke functions on local objects with no more run-time overhead than standard C++ virtual function calls.

This dynamic encapsulation model is incorporated into DC++, a set of dynamic extensions to the C++ language that allows for the dynamic encapsulation of existing C++ objects, and DECO (Dynamic Encapsulator of C++ Objects), a utility for converting DC++ source code to C++.

Faculty Advisors:
Professor Andrew T. Kitchen
Professor Warren R. Carithers
Professor Peter G. Anderson