#90-0005

Concurrency and real-time programming

Position paper, 22 February 1990

Dag M. Brück

dag@control.lth.se

Department of Automatic Control
Lund Institute of Technology
Box 118, S-221 00 Lund, Sweden

Proposal: To form a working group on concurrency and real-time programming.

There exists a considerable interest in concurrency in the C++ community, and the task library is indeed one of the few libraries distributed with the AT&T C++ compiler.

Different forms of concurrency can be identified: "soft" applications like discrete event simulation, and "hard" applications such as real-time control. The contradicting needs of these two examples (e.g., scheduling algorithms and mutual exclusion) will make it difficult to agree on a single form of concurrency in C++. I think Ada has failed in this respect: the definition of a single concurrency scheme has not been able to meet the demands of all potential users.

Concurrency in C++ should be realised through class libraries, with some very low-level support from the compiler and the standard run-time library. The work of X3J16 should concentrate on providing the minimum support for implementors of real-time libraries, not the high-level support required by end-users. The minimum support should be limited to primitives for

- Creating a new process.
- Transferring control from one process to another, similar to TRANSFER in Modula-2.
- Some mechanism for handling interrupts. I have no particularly good ideas in this case, but a possible source of inspiration could be the IOTRANSFER mechanism in Modula-2.
- Terminating a process (e.g., as a consequence of an uncaught exception).

The process data structure should only contain enough information for handling these operations. For example, there is no need to store a process identity or a priority. Defining the process data structure as a class will provide an easy extension path for library developers. Scheduling and mutual exclusion is deliberately left to libraries built on these primitives.

Real-time applications will impose constraints on other possible extensions, e.g., exception handling and garbage collection. The proposed working group must closely review other C++ extensions and suggested implementation strategies.

Weil (MS): leave "standard libraries" for concurrency to industry

Vilot: Beware: Ada Hassle! (change language soon)

Tiemann: don't mix up language with general OS & networking issues

Langley (DG): want support for object migration across networks