X3J16/96-0056
WG21/N0874
March 12th, 1996
John H. Spicer, Edison Design Group

Why I recommend removal of the "separation model"

There are many reasons to vote for removal of the separation model. In this paper I focus on what I believe to be the most important.

As a standards committee we have an obligation to ensure for any feature included in the standard:
- that it works (i.e., does what it was intended to do in a way that users will consider usable),
- that it is fully specified, and
- that a usable implementation is possible on a wide variety of systems employing a variety of different implementation strategies.

There is no consensus that the separation model fulfills any of these obligations.

This naturally gives rise to the questions:
- how long will it take to fully specify the model?
- when will we have sufficient implementation experience to know if the model works and can support reasonable implementations?

The committee has proven time and again that C++ is too complex for us to correctly make any significant change to the language without extensive implementation and usage experience. Every significant language feature has had to be modified based on implementation and usage experience.

1. Namespaces were approved in 7/93, underwent significant design changes as late as 7/95, required about a dozen changes and clarifications in 11/95 and still have interactions with templates that have not been fully worked out. This is despite the fact that two implementations of the proposal had been implemented prior to recommending it approval to the committee in 7/93. This illustrates that limited implementation experience with little user experience is insufficient to ensure that a feature is sufficiently specified and/or usable.
2. RTTI was added in 3/93. A number of issues still remained to be clarified in 11/95.
3. bool was added in 11/93 and usage issues are still being resolved.
4. Member templates were added in 3/94. Implementation and usage problems are only now starting to be discovered.

Is there any reason to believe that the separation model will somehow avoid the problems of previous language features that have been added? No. Instead it is likely that even more problems will be found in the separation model. The other language features did not involve the addition of any radically different implementation issues.

There is no pressing need for a template compilation model beyond the inclusion model. The inclusion model is used by most implementations and people are, in general, pleased with the model. In my opinion, what users want most is a model that will allow them to compile their templates quickly. The inclusion model is agreed to be faster than the separation model. In other words, we have little to gain by adding an unproven separation model and a great deal to lose if we add something that is later proven to be wrong.