No X3J17/96-0065
WG21/N0883
Date: March 12, 1996
Author: Michael S. Ball

Observations on the Template Compilation Model

Of all of the arguments made for and against the "Separation model" for template compilation, there are three that I find persuasive.

Name Access

Though the "separation" model does provide some advantages in limiting name access, I believe that this is the same sort of short-term advantage provided by the C preprocessor, and that it has many of the same longer-term liabilities that we have seen from the preprocessor. The advantage is that you can pay less attention to the structure of your interfaces and implementations. The liability is that the code so structured is less reusable, harder to change, and significantly more difficult to manipulate with advanced program development tools. We have tools to control interfaces and name access within the language, and using these tools will provide considerably more reusable code than an ad hoc structuring into files. File structuring and the preprocessor are reasonable compromises for dealing with limited resources, but the need to maintain compatibility with them has considerably slowed the development of more advanced tools.

Costs

The cost factor, though not overwhelming, is persuasive. The only examples that we have of the separation model have considerably higher costs than the inclusion model. Though we all have clever schemes that we think can bring the cost down, we have no evidence that these will work in practice. We estimate that the additional cost for separation will ultimately be small, but we have no evidence of this, and in existing practice the cost is quite high. Even the proponents of the separation model recognize this, and introduced the explicit instantiation syntax to allow the programmer to optimize the instantiation process. One justification for this was the large reduction in compilation time that judicious use of explicit instantiation could produce.

Lack of Experience

My major concern, however, is that the separation model has technical problems. I do not know what these are, but I know, as well as I know that the sun will rise tomorrow, that they are there. We will not discover them until we have a number of implementations in use by a large number of customers. We have a great deal of experience with the inclusion model, and we have a much better idea of its problems. If we issue the standard with such a large and complex area untested, we will guarantee incompatible implementation, exactly the situation that the standard is trying to avoid.