The Rationale for Software Wrapping
Panel Position Statement

Harry M. Sneed
SES Software Engineering Services
D-85521 Ottobrunn, Germany
100276.57@compuserve.com

1 Background

The pressure to replace the legacy software systems is growing. It is becoming increasingly difficult to fulfill the requirements of a heterogeneous, distributed business world with centralized, monolithic host-based software systems. A study by Deloitte and Touche of American civil service agencies showed that 75% of the operational information systems are of that type. In Europe the same situation exits throughout the public service as well as in the banking, insurance, transportation, trade and utilities sectors. Old batch and online COBOL or PL/I systems using hierarchical or network database systems dominate the scene. The cost of maintaining these systems is enormous. The benefits provided by them are sinking. The need to replace them is obvious to all. The question is how.

2 Strategies

It is hardly possible to rewrite all of the old code that has taken a decade to ripen in two or three years. In spite of all the modern CASE tools, it will require another decade. However, that is too long for impatient-users.

There are two alternatives to speed up the transition. One is reengineering, the other is wrapping. With reengineering the old code is transformed into a new form and transplanted into new distributed, object-oriented program frameworks. This sometimes works, but often the old code is too intertwined with its environment. It is impossible to take it out of its environment without destroying it. In any case, the transplantation operation is costly and risky.

With wrapping, the old code is left in its native environment and is connected to the new object-oriented software by an API or a CORBA type interface. It is possible to define business objects on the client side while using existing mainframe batch programs and online transactions on the server side as methods. This saves the effort required to transplant the code. It is similar to using a Kidney in a foreign body. It can work, companies like PAC-BELL have proved it, but it can only be viewed as a temporary solution. Not only is it inefficient, but it also restricts the design of the front-end applications. They have to be designed around the existing functions they want to reuse. On top of that, it does not solve the maintenance problem. The legacy software still has to be maintained in the old environment.

3 Summary

In summary, wrapping is a good technique for gaining time. One can construct object-oriented front-end applications with a modem user interaction without rewriting all of the functions, for the time being. Sooner or later though, the legacy functions must be reimplemented within the context of the new environment.