Integration

OverheadCAM is designed from beginning to end to be a well-behaved component of a larger system. The advantages in the paragraphs below can be roughly categorized as: ease of integration, ease of use, and availability of source code.

First, OverheadCAM is strictly a background component. It has no Graphical User Interface because it relies on the containing system to provide input and accept its output. For the user, this means that there is no need to learn new screen formats or data definitions. For the containing system, there is no need to sell clients on accepting another system developed by another vendor. The improvements will be part of the containing system in every sense.

Second, the function is very well understood and has clear boundaries. Every serious cost accounting text describes allocation. Having clear boundaries leads to good partitioning of functions into components. That leads to reduced testing cost and maintenance cost. When functions are mixed unnecessarily, the number of combinations and processing paths is increased without a corresponding benefit. When OverheadCAM has its data, it does not get any more data from other parts of the containing system. This simplifies testing. It also makes it simple to put allocations on another processor to run in parallel.

Allocations can be considered as much like a sort. For a sort, the function is well understood, the boundaries are clear, and no one would write their own. Specialized software with proprietary algorithms has better function, is cheaper, and is available now. These attributes describe OverheadCAM, too.

Third, the interface with the containing system is simple, efficient, and easily understood. The interface is just sequential flat files. As input, OverheadCAM expects a file of rules and a file of amounts to be allocated. It's output is a sequential file of allocated amounts. (Sizes and locations of fields are specified with configuration data.) This is an entirely appropriate interface for a batch process: a) It is much more efficient than other ways of transferring data. b) It is safe: neither OverheadCAM nor the containing system can touch the internal data or tables of the other. c) The approach is easy to understand: there is nothing fancy for programmers to learn.

Fourth, OverheadCAM can be used as a component for a variety of purposes. It can be used for GL financial close cost allocations. It can be used for more accurate calculation of activity pools for ABC/M. It can be used for budgeting with all-in costs. It can be used for BI and BPM profitability studies where use of all-in costs is important. Profitability studies based on incomplete costs can be expected to be misleading. OverheadCAM's outstanding speed and scalability make it practical to do allocations in all these contexts. Finally, OverheadCAM can allocate anything at all. For example, risk or investment income could be allocated according to rules.

Fifth, OHC is portable. It has been tested on Windows 98, Windows 2000 Professional, Windows XP, Windows 2003 Server 64 bit, HP-UX on Itanium and PA-Risc, and Linux. In every case, it compiled and ran correctly the first try. This is due to using Standard C and to avoiding the use of tricky coding or proprietary APIs. All use of OS functions, such as I/O and memory management, is through Standard C constructs. These approaches make it easy to move from platform to platform. Using only Standard C constructs also provides immunity from changes in OS versions. Not using an RDB or providing a GUI contributes to portability. Since OverheadCAM is so fast and does not even use an RDB, there is no need for stored procedures. Stored procedures are a maintenance problem, especially when multiple platforms are supported

Sixth, Necessary maintenance is minimal. The code does not have many execution paths because the algorithm is straightforward and because user input is not used during the batch process. This helps with testing, programmer training, and making changes. Everything that makes OverheadCAM portable contributes to low maintenance cost and effort.

Source code and a test file generator are provided. If a licensee wishes to make changes, the changes can be made within the component, rather than being attached someway

Seventh, OverheadCAM can be readily packaged in different ways. This is because it is implemented in layers, separating the core algorithm from the specifics of how data is acquired. It can be more tightly coupled, if that is desirable. It can be packaged as Service Oriented Architecture. It can be used to allocate transactions in “real time”, as transactions are received at a server. It can be packaged to do remote allocations, either in batch or transaction by transaction.

Eighth, on the business side of what makes a good component, use of OverheadCAM would enable the using system to have the best allocations in the industry. Since the algorithm is patented, copycat competitors would not be able to copy the function without paying anything. The owner of patent rights could license the technology or deny its use to others.