In-house technical expertise must be developed if a company wishes to control its own identity.

TransForge

# Managing Outsourcing – Why You Really Should

**by Campbell Cairns**

**TransForge corp.**

***The cleaner wrasse nibbles parasites off other fish. The sabre tooth blenny mimics the wrasse but bites a chunk of flesh from the unfortunate fish that allows it close enough.***

***Like the sabre tooth blenny, outsourcing might advertise a mutualistic relationship but deliver a parasitic one.***

Managing complex technical projects requires expertise and often implies a steep learning curve in unfamiliar technologies, some of which require years of advanced study allied with innate aptitude. In competitive business environments, where failure and success leverage vast amounts of money it is generally considered safer and easier to employ a large consulting house with the required skills than managing complex projects in-house — especially if the project is somewhat divorced from core business and expertise. Middle and upper-level management have every incentive to employ external consultancies rather than manage such projects themselves; they avoid many of the penalties associated with failure while increasing the scope of their authority without the burden of responsibility.

Reinforcing this perception was a wave that swept through companies about ten years ago: Each element of a company was to be financially isolated and profitable — if it wasn't essential or profitable it was dismantled, sold, or outsourced.

The financial restructuring implied expensive changes to legacy software systems, and management became acutely aware that the code they had taken twenty years to develop was inferior to many of the generic packages available. Programmers at that stage were highly paid and for tax reasons many of the them were already operating as 'consultants'. It made good financial sense to outsource software maintenance, development and operations. A vast outsourcing industry evolved about developing, maintaining, and servicing code.

Since then things have changed. Almost every system today has a microprocessor at its heart that communicates with other systems. Its functions and behavior are largely resident in its software –- its software could be said to serve as its *identity*. This 'identity code' has become kernel to all systems — even ostensibly non software-based systems.

It has become relatively easy to create and manipulate identity code: Standard architectures, integrated development environments, unit testing, and interactive languages have simplified code development enormously. The academic theory at the heart of good software is becoming progressively less critical as it becomes encapsulated in standards. Processing power has ceased to be a significant constraint and cross-platform languages and systems are talking seamlessly with each other. Encapsulated code can easily be replaced or improved without major system surgery. Large business systems have become either 'open' or 'configurable' enough to be installed by third-party implementors. Engineering and re-engineering a system has become largely a matter of writing a few lines of code.

While the benefits of outsourcing depend largely on context, the penalties are best understood from the vantage of a consultancy's modus operandi:

- Identify the characteristics of the host management structure, and interface at the highest possible level
- Identify key people in the host and co-opt them
- Identify problem areas and make them time-and-expense extensions to any contract. Subcontract experts for these problem areas
- Make sure that there is an escape clause in case the project tanks

This procedure ensures that the consultancy maximizes its influence while minimizing its exposure

and accountability. Ungoverned, a consultancy will extended the scope of a project until the project bears little resemblance to its original intent. While this analysis might appear somewhat cynical, it is merely the natural evolution of a service industry if allowed unregulated access to client systems. Unfettered and unconstrained, a project can consume so much money and expand so far beyond its original scope that it becomes its own *raison d'etre*.

Seen in this light the current trend towards outsourcing software development is a dangerous one, for the software employed by a company constitutes a large part of its corporate identity, which can lie in unexpected places. Where one company's identity might lie embedded in a huge enterprise package, another's might lie in a simple spreadsheet.

While a host company might believe it understands and controls its own functional identity, if it doesn't understand the technical mechanisms underlying that identity it cannot manipulate them. If a company cannot manipulate its identity it cannot adapt optimally to change or opportunity.

A company should therefore aim to build the in-house management skills and technical base it requires to create, maintain and manipulate its own identity systems. Some software is perhaps better entrusted to other more expert hands, and consultancies certainly have a role to play, but, for the reasons outlined above, that role must be technically managed by the host. This imperative extends beyond software, but software is the most accessible and malleable component of most companies.

In summary, in-house technical expertise must be developed if a company wishes to control its own identity.

.