Software Engineering Management: strategic choices in a new decade

Barbara Farbey & Anthony Finkelstein
University College London,
Department of Computer Science,
Gower St. London WC1E 6BT, UK
{b.farbey|a.finkelstein}@ucl.ac.uk

1. ABSTRACT

This paper discusses the strategic choices faced by software engineering managers and sketches a framework for analysing those choices. It draws attention to the idea of a software supply network which is novel in the context of software engineering and has significant implications for economics of software development.

2. INTRODUCTION

The late 1990s saw a proliferation of technical and organisational advances in software engineering and in its application. For the software engineering manager these advances provide both a range of opportunities *and* a considerable challenge in defining a software strategy. That strategy must now include:

- linking the software organisation's work to corporate strategic priorities
- understanding the market for the software organisation's products, its nature and composition
- defining the distinctive capabilities of his/her own software organisation *vis a vis* outside software houses and commercial off-the-shelf packages
- understanding the nature of the networks of supply that feed software production.

This is a more complex picture than it was in the early 90s. The fundamental issues regarding alignment with corporate strategy and achieving value for money have not changed. But the context and the options have changed, and the issues have broadened to include questions of purchasing and supply. At the very least, management needs a guide to the new options and issues and how they fit together. This paper assembles a framework for the development of such a guide. The framework places particular emphasis on the impact of software strategy on requirements management.

3. DEVELOPMENT OF THE FRAMEWORK

The framework is built in three stages. The first stage identifies the strategic purpose of the system, its intended market, the type of system and the level of system requirements. Together these form the "strategic context" for development or acquisition. Working through this part of the framework locates the proposed system within a strategic "space".

Associated with any specific location are a number of issues. These include anticipated strategic outcomes, the nature of the costs, benefits and risks, potential organisational advantages and disadvantages, options for requirements management and for managing the systems effort within the organisation. The framework thus has two dimensions, strategic purpose and issues, which we show as a matrix. Each cell in the matrix is, from the manager's point of view, a potential issue to be resolved.

The next step in the argument is to set out the contemporary options with respect to the systems development and acquisition process. Following Fine [4] the guiding principle here is that for optimal results, development and acquisition choices must be made together - as Fine has it "3-D concurrent design". We are arguing additionally here that 3-D concurrent design should be done within a strategic location which affects the logic of the choices, and extend the matrix appropriately.

The third part of the framework emphasises the new thinking in supply chain management and organisational design. This special focus is interesting because there is now enough experience with outsourcing and COTS, and with the new ways of doing business, to address questions of inter-organisational relationships, contract design and supply network composition in the context of software engineering management.

The last step is to glue the matrices together. The final matrix serves two purposes. First it is a map for managers to address the options and issues systematically, whilst maintaining sight of the whole "terrain". Second, it is a research framework which we will ourselves use to research and develop guidelines for managers to help them address the issues.

4. THE NEW SYSTEMS DEVELOPMENT AND ACQUISITION CONTEXT

Corporate Strategy

In terms of strategic purpose there are fundamentally three positions which the system(s) under consideration will be required to address. These positions are defined with respect to the current technological frontier.

First, the organisation may be lagging behind its competitors. The purpose of the proposed system(s) is to catch up.

Second, the organisation may be technologically in contention, but want to shift the balance of resources. For example as the larger, well-established firms move into e-commerce, they come to strike a strategic balance between all the diverse channels open to them.

Third, the organisation may be attempting to change the rules of the game using new technology. Nowadays that usually means information and communications technology, but it could be a different way of organising, as in the new supply chain management. The questions are on two levels: first, how can we use systems to help achieve a competitive breakthrough and second, can better software management of itself produce a radical breakthrough - reducing costs and time to market.

The first question is therefore is about how a proposed system chimes with strategic purpose. This will affect the view taken on the costs of the system, the range of benefits expected and the attitude to risk. Strategic purpose will affect decisions as to what can be safely outsourced and what may have to stay in-house. Requirements too will be affected. One can speculate, for example, that the requirements of a system intended to support a strategic breakthrough will be volatile - because they are "new". This would imply a significant meta-requirement for flexibility. The first three rows of Table 1 summarise these considerations.

Table 1

Policy Implications: corporate strategic purpose								
	Economic			Organisational		Strategic		Technical
	Costs	Benefit	Risks	Advantages	Disadvantages	Competitive advantage	Co-operative advantage	Requirements Management
Catch-up								
Re-positioning								
Break through								

System type

The extensive use of software systems notably includes different genres of system, for example embedded systems, information systems, distributed systems and groups of systems in product families. Individual systems have for many years been set in the context of a systems strategy, but for product lines, as Clements and Northrop [1] point out, "... the evolution of a single product must be considered within a broader context i.e. the evolution of a product line as a whole" (my emphasis). It is the "broader context" which requires new strategic thinking.

Market Structure

The stereotypical textbook system of the early 1990s was a single, standalone system built for a single, known customer. Many systems still are. But system today may also be built for:

- a very small number of known customers
- for a mass market, for example COTS or web sites
- or, in the case of "future legacy" systems as it were, like banking systems, for future unknown customers and technological conditions.

Table 2

Policy Implications: System Type, Market Structure, Level of Requirement								
	Economic			Organisational		Strategic		Technical
	Costs	Benefits	Risks	Advantages	Disadvantages	Competitive advantage	Co-operative advantage	Requirements Management
Single, standalone system								
Product family								
Re-use								
Single, known customer								
Very small no. of known customers								
Mass market								
Unknown, future customers								
Institutional, high level								
Strategic level								
System level								

The level of requirement
Not all requirements arise from the fact of the system. Some are strategic requirements of the organisation. Others may come from standard industry institutions or legal regulation. Industries operate in different ways and these ways will be reflected in the requirements for any system serving the industry. King [8], for example, is researching "high-level requirements". These are requirements that flow from the organisational and institutional settings of a system, rather than directly from the application of the system itself.

Corporate strategic requirements may also be outside the immediate functional demands of the system, adding strategic value, rather than functional value. The tracking systems whereby customers can follow the progress of a parcel, for example, are not essential to the progress of the parcel itself, but essential to the competitiveness of the organisation. The distinction is important because attempting to change a high-level or strategic requirement will have severe implications for costs and risks. One can suggest therefore a division of requirements into

- Industry, institutional, or "high level", requirements
- Strategic requirements
- System level requirements.

Table 2 summarises the system type, market structure and requirement level considerations.

5. PROCESS MODELS

There are a variety of models for system development. We mention these without expansion as they will be familiar to software engineers: waterfall, spiral, prototyping, extreme programming and "synch. and stabilise" [2]. As with the other options, they raise financial, organisational and possibly strategic issues and the corresponding matrix is shown in Table 3

Table 3

Policy Implications: process models								
	Economic			Organisational		Strategic		Technical
	Costs	Benefits	Risks	Advantages	Disadvantages	Competitive advantage	Co-operative advantage	Requirements Management
Waterfall								
Spiral								
Prototyping								
Extreme programming								
"Synch. and stabilise"								

6. POLICY OPTIONS FOR SOFTWARE ACQUISITION

The 1990s saw a variety of developments in the procurement possibilities for software.

Outsourcing
The outsourcing movement was, and is, a complex one. Outsourcing has not always been successful and is still evolving [10]. Nevertheless, it represents a distinct option for contemporary software engineering managers and, as its most critical feature for this discussion, means co-operation between two organisations, partly embodied in a formal contract.

COTS
The concept of purchasing readymade software has also become commonplace with the development of commercial off the shelf system (COTS). COTS are significant in software acquisition because they offer a different relationship between vendor and purchaser from that of an outsourcing contract. At the extreme, a shrink-wrapped package for example, the relationship is a spot contract. A typical outsourcing contract is relational, extending over a period and involving some form of partnering. From the point of view of the purchasing organisation, moving to COTS implies a different form of internal organisation, moving from "being a developer and producer of systems, to being a consumer and integrator instead" [15].

ASPs
A second, related development in that it utilises pre-developed software is the advent of software Application Service Providers (ASPs). Significant issues that arise with ASPs for the consumer organisation, however, are those to do with how the systems are financed: from the capital or the revenue budget. The timing of expenditure is different too. ASPs are essentially rented systems. The cost is spread out over the life of the system in use, not largely up-front as it is with other types of acquisition [5].

Open Source
With the success of Linux a third development is perhaps at hand, namely the serious, industry-strength use of "free", open-source software, exemplified also by Netscape Navigator [6].

In the face of these choices the software engineering manager needs to construct a strategy for software acquisition. Table 4 summarises the various types of acquisition. In practice each row is not "pure". There is a variety of outsourcing types, just as there are various degrees of "off the shelf". Moreover, the reality is likely to include a combination of all these methods and the combination is itself a management issue. But the matrix

as tabled will at least act as a starting point for further detailed discussion in the organisation.

Table 4

Policy Implications: acquisition methods								
	Economic			Organisational			Strategic	Technical
	Costs	Benefits	Risks	Advantages	Disadvantages	Competitive advantage	Co-operative advantage	Requirements Management
In-house development								
Outsourced development								
COTS								
ASPs								
Open source								
Legacy upgrade								
Combination								

As before the issues form the columns of the Table. However, the major implication of the new methods of acquisition is that all of them entail relationships beyond the user/ consumer organisation. There are thus very significant issues relating to the nature of these relationships, their conduct and value. Two key groups of issues concern relationship management, both formal and informal, and questions of organisational learning in networks. These are taken up in the next section.

7. INTER-ORGANIZATIONAL RELATIONSHIPS

Inter-organisational systems, and hence inter-organisational relationships, have been on the Information Systems research agenda since the advent of EDI in the 1980s [13]. Outsourcing gave an added urgency and impetus to the research, particularly perhaps because many organisations found the outsourcing relationship difficult to manage, with the vendor organisation often able to hold the whip hand [11] as quoted in Lacity and Hirschheim [10].

We surmise that the management aspects of COTS, evaluation and the relationships with suppliers will provide an added twist. Open source acquisition will also throw the spotlight on relationships, if only because one side, the suppliers, are not providing the software for money, but for some other form of reward.

But Information Systems outsourcing is only a part of the story. In recent years there has been considerable development of new arrangements and forms including joint ventures, alliances, imaginative partnering arrangements and supply chain management [9, 12]. In partnerships, of what ever form, there is a range of choices as to how work is organised across the constituent organisations.

With the commoditisation of software systems in the shape of COTS, and the continuing move to re-use via components and kernels, software engineers do not necessarily develop large software systems themselves. Instead they assemble, compose and glue components together. The software manager has, in principle, the same kinds of choices as his counterparts in other areas of the organisation. There is a supply chain, or a supply net, in software development, albeit it is likely to be a short one. The next step in the paper therefore, is to spell out the choices.

Types of contract

A basic distinction is discussed by Kay [7] when he expands on the distinction between spot contracts, classical contracts and relational contracts. Spot contracts, like those for some COTS, or as Kay remarks a "lettuce from a green grocer", are short term, based on standard terms and take place at market prices.

Classical and relational contracts are different from spot contracts in that they are longer term. Classical contracts in Kay's definition are explicit. Kay gives the example of a property lease. They are formal, legal and binding. Outsourcing contracts are based on legal contracts.

Relational contracts are implicit. They may have a partial basis in law, but the dominant element is trust. Marriage is the example given by Kay. Early outsourcing contracts were often relational, although as Lacity and Hirschheim point out [10], this did not always lead to happy results for the consumer organisation.

Supply network composition

A principal finding in previous research on the supply chain is that "*In a fast clockspeed world, advantage arises from the concurrent design of products, processes and capabilities*." [4, italics in the original], [3]. "Clockspeed" is Fine's term for the rate of evolution of products, processes and organisations. He is particularly interested in fast clockspeed industries, those that evolve very quickly and in which products or processes have short lives.

We have looked in the course of the paper at product and process. Outsourcing can usefully be seen as a redistribution of capabilities [14]. COTS could be seen as embedded capabilities. In either case, but more particularly in outsourcing within a network of suppliers, there is a question as to where the capabilities lie and how to make optimal use of them. However, because capability takes time to develop, it may be that it is not necessarily an independent choice - independent that is of the choice of a supply network. With respect to capability we suggest not a new row, but a new emphasis on capabilities,

organisational learning and indeed knowledge management, within the column on co-operative advantage. Table 5 summarises these ideas.

Table 5

Policy Implications: Inter-organisational relationships								
	Economic			Organisational		Strategic		Technical
	Costs	Benefits	Risks	Advantages	Disadvantages	Competitive advantage	Co-operative advantage: *organisational learning*	Requirements Management
Supply network composition								
Contract type								
Distribution of expertise/ capability								

8. GLUING IT ALL TOGETHER

We believe that the idea that product, process and capability must all be designed together can now, and should now, be applied to software engineering. One way to do this is to swallow the matrix whole, working through it systematically, and it surely has to be iteratively, forming a holistic picture of product, process and capability across the possible networks and designing them accordingly. Table 6 therefore shows a summary version of the options, for ease of navigation

Table 6

Context, options and issues - summary								
ISSUES	Economic			Organisational			Strategic	Technical
	Costs	Benefits	Risks	Advantages	Disadvantages	Competitive advantage	Co-operative advantage: *organisational learning*	Requirements Management
CONTEXT								
Strategic purpose								
System type								
Market structure								
Level at which requirements are specified								
OPTIONS								
Process models								
Acquisition options								
Contract type								
Supply network composition								
Distribution of expertise								

9. CONCLUSION

This paper has attempted to define the terrain in which the software engineering manager makes choices about the way software is developed. It has sketched the context, the policy options and the areas which will be at issue as the choices are made. These are brought together in tabular form, providing a framework for decisions. The paper has also highlighted areas of the terrain, the idea of a software supply network, which will be unfamiliar because the choices are relatively new.

The framework makes a start in assisting managers to do concurrent design by drawing together the relevant issues and allowing them to be addressed systematically. However, we have as yet no practical experience to report. That comes next.

REFERENCES:

1. Clements P., and Northrop L.M., 1999) A framework for software product line practice, *SEI Interactive*, **2**, 3, On-line at <http://interactive.sei.cmu.edu/Features/1999/September/Background/Background.sep99.htm>
2. Cusumano, M.A. and Selby, R.W. How Microsoft Builds Software, *Communications of the ACM*, **40**, 6, (1997) 53-61
3. Farbey, B. and Finkelstein, A. Exploiting supply chain business architecture", Position paper, Edser-1 (1999). On-line at <http://www.cs.virginia.edu/~sullivan/edser1/>
4. Fine, C.H. *Clockspeed: winning industry control in the age of temporary advantage.* (1998) Perseus Books, Reading, Mass.
5. Gillan, C., Graham, S., Levitt, M., McArthur, J., Murray, S., Turner, V., Villars, R. and Whanlen, M.M. *The ASPs' impact on the IT Industry: an IDC wide opinion*. International Data Corporation. 20323 (permission needed for quote) (1999) On-line at <http://www.idc.com>
6. Hecker, F. Setting Up Shop: The Business of Open-Source Software, *IEEE Software,* 16, 1, (January/February 1999)
7. Kay, J. *Foundations of Corporate Success: how business strategies add value,* (1993) Oxford University Press, Oxford, Chapter 4
8. King, J.L. (on-line at current site to June 2000) <http://www.ics.uci.edu/~king/research.html>
9. Lorange P. and Roos J. *Strategic Alliances: Formation, Implementation and Evolution*, .(1993)Blackwell, Oxford
10. Lacity, M. and Hirschheim, R. Information Technology Outsourcing, in *Rethinking Information Systems,* Currie, W. and Galliers, R. eds., (1999) Oxford University Press, Oxford, Chapter 14
11. Lacity, M. and Willcocks. L. An empirical Investigation of Information Systems Outsourcing: Findings from Experience, Oxford University Working Paper (1996)
12. Lamming, R. *Beyond partnership: strategies for innovation and lean supply*, (1993) London: Prentice Hall
13. Reekers, N and Smithson, S. The Impact of Electronic Data Interchange on Inter-organisational Relationships: Integrating Theoretical Perspectives, HICSS-28 Minitrack, "Measuring the Effectiveness/ Impact of Emerging Technologies", Jan 3-6, 1995
14. Scarbrough, H) The External Acquisition of Information Systems Knowledge in Willcocks L.P. and Lacity M. (eds.) *Strategic Sourcing of Information Systems: Perspectives and Practices*, (1998) Wiley, Chichester, Chapter 4
15. SEI (1998), Carnegie Mellon University, COTS BASED SYSTEMS initiative. On-line at <http://www.sei.cmu.edu/cbs/practices.html>