Patterns in Application Development Sourcing in the Financial Industry

Anders Martensson

Stockholm School of Economics, anders.martensson@hhs.se

Follow this and additional works at: http://aisel.aisnet.org/ecis2004

Recommended Citation
http://aisel.aisnet.org/ecis2004/149
PATTERNS IN APPLICATION DEVELOPMENT SOURCING IN THE FINANCIAL INDUSTRY

Mårtensson, Anders, Stockholm School of Economics, P.O. Box 6501, SE-113 83 Stockholm, Sweden, anders.martensson@hhs.se

Abstract

Sourcing, or the question of what a company should carry out in-house and what should be acquired elsewhere, has been discussed for quite some time. This paper addresses the sourcing of application development efforts in the financial industry. Based on a framework for selective sourcing, four cases are investigated and sourcing choices are characterized.

Patterns consistent over time can in fact be found in the companies and several factors explaining such patterns are discussed. Load balancing and competence issues are identified as factors implying that company size is an important determinant of companies’ degrees of freedom when making sourcing choices for proprietary solutions. Legacy issues both in terms of traditional IT legacy and softer issues such as people’s experiences are also important.

Concerning the choice between standard packages and proprietary solutions, companies tend to apply the same decision rule but evaluate the result differently. One proposed explanation to such differences builds on the distinction between application instances and application roles. Similar applications can play very different roles in different companies from for example business strategy perspectives. Such differing roles can explain different choices for seemingly similar applications.

Keywords: sourcing, application development, financial industry
1 INTRODUCTION

Sourcing, or the question of what a company should carry out in-house and what should be acquired elsewhere, has been discussed for quite some time. Coase (1937) asked two fundamental questions: Why are firms created at all? If it is such a good idea, why isn’t there one firm doing everything? In Coase’s terms there is a balance, or an equilibrium firm size, where any increased (decreased) efficiency by increasing (decreasing) the size of the firm is exactly balanced by increased (decreased) coordination costs and decreased (increased) price mechanism costs. Building on these ideas, Williamson later developed the transaction cost theory (Williamson 1975), which is commonly used as an underlying theory for outsourcing discussions (e.g. Lacity & Hirschheim 1993, Scarbrough, 1998) and make-or-buy decision (e.g. Walker & Weber 1984). There are many different theoretical foundations for outsourcing which is illustrated by Cheon, Grover and Teng (1995) who provide a theoretical framework where they combine transaction cost theory with resource-based theory, resource dependence theory, and agent cost theory.

Outsourcing can be addressed from a multitude of perspectives (see Kakabadse & Kakabadse 2000, for an overview). This specific paper addresses the sourcing of application development efforts in the financial industry. By applying an existing framework for selective sourcing (Mårtensson 2003), patterns in companies’ application development sourcing activities are identified and discussed. The aim is to further our understanding of various outsourcing choices and thus potentially help companies structure their outsourcing discussion. The aim is not, however, to provide a normative framework suggesting to companies what they should outsource and what should be kept in-house. Before the framework is presented, existing outsourcing theories are discussed. Then, the research study is described and the cases are presented. Finally, after a discussion there are some concluding remarks.

2 SOURCING

2.1 Defining Outsourcing

Outsourcing is not a new concept. In 1963 Electronic Data Systems took care of data processing services for other companies (Lacity & Hirschheim 1993). Since Eastman Kodak’s decision in 1989 to outsource its IT activities the drivers for outsourcing have primarily been cost-effectiveness, avoiding building in-house skills and access to special functional capabilities (McFarlan & Nolan 1995). Since then the outsourcing industry has grown and now encompasses all types of IT services. Production cost advantages offered by vendors also remain a very important impetus for outsourcing (Ang & Straub 1998).

In most definitions of outsourcing the directional aspect is central (e.g. Lacity & Hirschheim 1993, Grover, Teng & Cheon 1998). Outsourcing is interpreted to mean that some activity, which used to be carried out within the organization, is handed over to an outside party. In fact, the transitional stage of actually handing over the activity often lies at the core of these outsourcing definitions. The reversed process, i.e. bringing some activity into the company that was previously carried out by an external party, is sometimes discussed in the literature, albeit without the same coherence in vocabulary.

2.2 What to Outsource

By taking into account that the world is not black or white, Lacity, Willcocks and Feeny (1996) introduce the concept of selecteive sourcing in their categorization of sourcing decisions.
- **Total outsourcing**: At least 80% of the IT budget is outsourced.
- **Total insourcing**: At least 80% of the IT budget is insourced.
Selective outsourcing: Some IT functions are outsourced and some are insourced but neither stands for 80% of the budget.

De facto insourcing: Internal IT department is used without evaluating outsourcing alternatives. It is noteworthy that in this case selective sourcing refers only to the share of the IT budget that is outsourced. A different approach is presented by Lacity and Hirschheim (1993) who present three different types of outsourcing.

Body shop means that short-term demand is met by hiring contract programmers managed by company personnel.

Project management is the outsourcing of a specific project or portion of work. The difference from the body shop is that the vendor is responsible for completing the work.

Total outsourcing is used rather as a quantitative measure as it relates to cases where a significant piece of IS work is outsourced.

In this case, there is a mix where the first two alternatives (body shop and project management) refers to what types of activities are being outsourced while the last refers to how much is outsourced.

Ang (1994) also subscribes to the directional notion of outsourcing meaning moving something across the boundaries of the organization. She does however distinguish between three different criteria for determining these boundaries: geographical, legal ownership, and control boundary. Geographical occurs when activity is performed away (spatially) from the organization. Legal occurs when the focal organization does not have legal ownership of physical assets used or employs the people involved. Control occurs when the focal organization relinquishes behavioural control.

A competence perspective is commonly applied to outsourcing issues, especially in studies focusing on the decision process of whether to outsource or not, and if so what to outsource. Scarbrough (1998) actually provides one approach to extending Williamson’s transaction cost theory with a knowledge perspective. Quinn and Hilmer (1994, p. 43) focus on core competencies, which should be kept in-house, and other activities, “for which the firm has neither a critical strategic need nor special capabilities”.

The sourcing question is also addressed from a make-or-buy perspective where it is typically framed as whether “a firm should provide its own software development resources, or use external sources” (Rands 1993, p. 279). External sources in this context mean, “suppliers may market specialized software packages for the area concerned […] or undertake bespoke applications projects” (Rands 1993, p. 280). It is important to note that no distinction is made between software packages and bespoke application projects as external resources, i.e. the focus is on the location of the resource rather than its type.

### 2.3 A Framework for Selective Sourcing

By using Lacity and Hirschheim (1993) distinction between body shop and project management outsourcing and extending Ang’s (1994) notion of the possible outsourcing of control from the original organizational aspect to the control of the application, a framework for selective sourcing has been developed (Mårtensson 2003), see Table 1.

<table>
<thead>
<tr>
<th>Type of Solution</th>
<th>Code Development</th>
<th>Project Management</th>
<th>Control and Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-house solution</td>
<td>In-house</td>
<td>In-house</td>
<td>In-house</td>
</tr>
<tr>
<td>Outside developers</td>
<td>External</td>
<td>In-house</td>
<td>In-house</td>
</tr>
<tr>
<td>Turn-key solution</td>
<td>External</td>
<td>External</td>
<td>In-house</td>
</tr>
<tr>
<td>Standard package</td>
<td>External</td>
<td>External</td>
<td>External</td>
</tr>
</tbody>
</table>

Table 1. Framework for Selective Sourcing.
In-house solution means that everything is done in-house with no use of external resources. Outside developers means that the company makes use of outside developers but retains the project management and responsibility for the project, i.e. resources and not results are acquired. Turn-key solution means that the company acquires a tailor-made application from an external party, which delivers a result (the application) rather than resources (e.g. man hours). Standard package means that the company buys a standard package from a software provider that retains the control and responsibility for the application.

The important distinction between rows three and four (turn-key solution and standard package) is that a company buying a turn-key solution from e.g. a major consultancy company still owns and controls the solution. For example, they have the right to sell it to others or turn to others to maintain the application. In the standard package case the company buys a license for the package and controls its use of the application but not the application itself. Normally, the buyer in such relationships has no right to sell the application to a third party.

We now turn to the empirical part of the paper, which starts out with a short description of the methodology applied.

3 METHODOLOGY

This paper builds on a multiple case study of four companies in the financial industry. Generally, case studies are well suited for studying complex phenomena that are little understood and exist in specific settings (Benbasat, Goldstein & Mead 1987). The data was collected mainly by interviews. A total of forty interviews were carried out varying from five to fifteen per company. In each company the interviewees included representatives from the IT department, the business organization and top management. Each interview lasted between 45 minutes and two hours and 45 minutes. The interviews were characterized by a limited degree of structure imposed by the interviewer, mostly open questions and a starting point in real life stories, or specific situations, rather than abstractions and general opinions (King 1994).

The cases are all from the financial industry and represent three brokerage firms and one commercial bank. Collecting all cases from the same industry increases the possibility of performing relevant inter-case analysis. Furthermore, in the financial industry, IT is tightly intertwined with the business, and IT support is extremely important for the business. Business development efforts are usually dependent on adequate IT support in terms of some degree of application development. The financial industry has been dependent on IT for a long time (Garbade & Silber 1978) and IT has been shown to serve as a source for competitive advantage (McKenney, Copeland & Mason 1997).

There are however some drawbacks with the financial industry as an empirical area. The type of business carried out differs from most industries, which could reduce the relevance of the study for other industries. However, even though few industries have the same conditions as the financial industry today, conditions in other industries are changing, and furthermore, they are changing in the direction of the conditions existing in the financial industry today (cf. Wilhelm & Downing 2001).

4 CASES

In line with Eisenhardt’s (1989) reasoning, the case companies were chosen so that the variation will help the understanding of the cases. In this sense, the aim is to cover different ends of the spectrum in some important dimensions. The two dimensions chosen were company complexity and targeted markets segment. Company complexity was used as a composite measure consisting of the two dimensions, sheer size and scope of services offered (e.g. analysis, asset management, corporate finance, trading) resulting in Figure 1.
4.1 Gamma

Gamma is a full service retail bank that consists of a number of business areas and has several thousand employees. A number of years ago Gamma Markets was created when Gamma gathered stockbrokers, bond traders, derivatives trading, foreign exchange, etc into one organizational unit. There are two main back-office applications at Gamma Markets. One is a mainframe application for order entry and trade execution management and the other is a mainframe system for brokerage account management. Both were developed in-house using Cobol and originally stem from the late 1970s and early 1980s.

Years ago the business side was free to invest in IT on its own; this coincided with an explosion of financial systems. As a consequence, Markets’ IT portfolio is a jumble of different systems. During a long period of time a lot of different systems were bought. The fast-moving Markets perceived the rest of the organization to be too slow and chose to buy what they needed. The users were very heavily prioritised, which caused the application portfolio to grow quite randomly. People at Markets believe that other people’s view of Markets’ application portfolio can to some extent be attributed to a lack of understanding for the hectic business environment the traders live in.

Gamma has a rather ambivalent, emerging approach to sourcing. The tradition within Gamma is to build applications in-house. Over the years, Gamma has been, and still is, large enough to be able to keep sufficient resources within their IT department. The in-house scale simply is large enough to reach critical mass (Lacity et al 1996) and any temporary shortages are covered by external consultants. Even though the core applications discussed above are developed in-house, Gamma’s IT department is currently striving to use more standard applications and to reduce their in-house development in order to cut costs.

The history at Gamma Markets is slightly different, though. Markets has had a long-standing tradition of being slightly apart from the rest of the organization and to evaluate IT investments on a case-by-case basis. As long as a trader, broker or asset manager felt a need for a new standard package and somehow could show the profitability of buying it, the acquisition was made. The short lead times involved when acquiring standard application packages were also an important factor. As business opportunities were perceived to vanish quickly, rapid implementation was essential.

Gamma’s web site for stock trading is an interesting deviation from this standard application package pattern. When it was initiated, no standard package was acquired (or available for that matter). Instead, Markets used external consultants to develop the application in-house. The brand new technology, which Markets did not think the IT department mastered, along with the importance of a short time-to-market, ruled out Gamma’s IT department as a viable option in Markets’ opinion.
Over time this has led to a rather wild and complex portfolio of different standard packages more or less well integrated with each other and often with over-lapping functionality. Gamma’s mixed approach using both in-house solutions (Gamma) and standard packages (Markets) is illustrated in Table 2.

<table>
<thead>
<tr>
<th>Type of Solution</th>
<th>Code Development</th>
<th>Project Management</th>
<th>Control and Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-house solution</td>
<td>In-house</td>
<td>In-house</td>
<td>In-house</td>
</tr>
<tr>
<td>Standard package</td>
<td>External</td>
<td>External</td>
<td>External</td>
</tr>
</tbody>
</table>

Table 2. Characterizing Gamma’s Sourcing Strategies

From an IT departmental view it could be argued that, apparently, Markets has already gone where the IT department wants to go, albeit for all the wrong reasons. This is not completely true, as Markets portfolio is not as well groomed as Gamma’s overall portfolio.

4.2 Delta

Delta is a full-service investment bank with a few hundred employees. The main areas of the business are Equity Sales and Trading, Equity Research, Asset Management and Corporate Finance. Delta’s focus is institutional investors and the brokers are generally quite experienced. Delta also has a few traders, i.e. people who are trading in Delta’s own books and take on positions on behalf of the company.

There are a few major applications in Delta’s IT portfolio. The front-office system and the back-office system are the major parts from an Equity Sales and Trading perspective. There are also several complementing information systems of different kinds.

The front-office application is a proprietary solution, which is in line with Rands’ (1993) suggestion to “make” strategically important applications where company skills are relatively high. A recurring assumption at Delta is that the higher costs associated with having a proprietary application instead of a standard application in this area is worth the extra cost from a business perspective.

The back-office system is owned by Delta and some other market participants. It is also sold to other companies. Joining forces with other market participants is quite reasonable from a sourcing perspective, as Lacity et al (1996) suggest in-sourcing for applications where managerial practices are leading, but in-house scale is sub-critical. It can be argued that the co-operation with others is a way of increasing scale in order to reach critical mass in terms of application usage, and thus become cost-effective. Delta has started the work of finding a successor to the back-office system and one important requirement is to make the application more modular. Furthermore, there is no ambition to develop the whole application, instead it is preferred to buy components and only develop in-house when there are no external components available.

<table>
<thead>
<tr>
<th>Type of Solution</th>
<th>Code Development</th>
<th>Project Management</th>
<th>Control and Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outside developers</td>
<td>External</td>
<td>In-house</td>
<td>In-house</td>
</tr>
</tbody>
</table>

Table 3. Characterizing Delta’s Sourcing Strategy

A clear pattern in Delta’s behaviour is to develop proprietary applications using external developers, see Table 3. The traditional approach at Delta is to outsource code development but to keep project management in-house. Typically, outside consultants work on Delta’s premises in a quite straightforward body shop approach (Lacity & Hirschheim 1993). The main argument for not having code developers employed is knowledge and competence issues. Delta prefers to acquire the service from people with the right set of skills, rather than have to keep a group of in-house developers up to speed on a variety of
technologies. This is in line with Domberger’s (1998) notion that competence does not have to be kept in-house if it can be acquired under reasonably competitive conditions.

4.3 Lambda

Lambda was founded in the mid 1990s by three entrepreneurs all still working with the company. Currently Lambda has 13 employees. From the very beginning a fourth person joined Lambda who took charge of developing the trading system, EdbTrade. Buying the system was not a viable alternative, mainly due to the lack of existing standard application packages. Available components were discarded partly for reasons of secrecy. Porting the green field aspects over to the IT applications also had the advantage of not bringing in any IT legacy from a standard solution.

The IT portfolio of Lambda is dominated by EdbTrade, which is physically separated from the other applications. Apart from EdbTrade, the portfolio consists of standard application packages. There are informational systems (e.g. Reuters, Bloombergs), marketplace software (client applications to connect to different electronic exchanges) and ordinary office applications (e.g. Microsoft Office products). The back office is outsourced.

The head of IT development worked with specifications of EdbTrade, but the actual development and project management was left to the external company. Further development of EdbTrade is still done by the same external consultancy, which has turned into a preferred contractor (cf. Lacity et al 1996). In a case such as Lambda’s with low company skills (or rather meagre resources) and a strategically important application, Rands (1993) suggests a strategic alliance. In a way, Lambda achieved this by forming a strong relationship with a consultancy company.

Lambda chose very consciously to outsource neither the control of, nor the responsibility for, EdbTrade. Instead, not giving up the control of the application was a very important issue to Lambda and is reflected in their tight control of what changes to make to EdbTrade. In this respect, it is not important that the changes are made by the external consultancy. This is in line with Quinn and Hilmer (1994), who point to loss of control as a main strategic concern when outsourcing.

Concerning back-office solutions, there were more alternatives available since this is carried out by every brokerage firm and there were several standard applications available on the market. Lambda opted for outsourcing the business function rather than the application. Even so, Lambda has retained in-house competence in order to, for instance, answer customer questions. People at Lambda are aware that customers turn to Lambda to get a certain service and expect Lambda, rather than an outsourcing provider, to provide that service. The literature on sourcing issues adopts a strongly internal perspective, discussing

<table>
<thead>
<tr>
<th>Type of Solution</th>
<th>Code Development</th>
<th>Project Management</th>
<th>Control and Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turnkey solution</td>
<td>External</td>
<td>External</td>
<td>In-house</td>
</tr>
</tbody>
</table>

Table 4. Characterizing Lambda’s Sourcing Strategy

Given the framework for selective sourcing developed above Lambda’s solution can be characterized as a turnkey solution, see Table 4. People at Lambda feel comfortable being in control of their fate and not outsourcing that to some third party vendor. They are very well aware of the responsibility they have taken on and live by the phrase “know thy systems”. Adhering to Quinn and Hilmer’s (1994) advice to focus on core competencies and outsource activities “for which the firm has neither a critical strategic need nor special capabilities” (p. 43), Lambda has chosen not to build internal capabilities for code development and project management. Decisions on enhancements to EdbTrade, however, are kept in-house as a core competence.
outsourcing (or not) from an efficiency perspective (e.g. Domberger 1998). The literature does not explicitly address how outsourcing affect customers, especially those expecting one-stop shopping.

4.4 Tau

Tau is an Internet brokerage employing some 60 people. The company strives to provide customers with a simple and cheap way of buying and selling stock. To Tau’s CEO IT is a way of reducing costs. He wants commonly available competence in-house but prefers to buy leading-edge knowledge. Tau’s strategy is to avoid the first hits by being maybe number four or five in introducing new technologies.

Tau has its clear low-cost business strategy, which translates quite explicitly into a strategy to buy standard packages whenever possible. The head of the IT department, Otis Thornton, neatly sums up the overall strategy using three key questions: Is it already available in-house? If not, can it be bought? If not, can we develop it ourselves? Thus, the main sourcing strategy is to acquire standard packages with the backup being to acquire turn-key solutions, as described in Table 5.

<table>
<thead>
<tr>
<th>Type of Solution</th>
<th>Code Development</th>
<th>Project Management</th>
<th>Control and Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turnkey solution</td>
<td>External</td>
<td>External</td>
<td>In-house</td>
</tr>
<tr>
<td>Standard package</td>
<td>External</td>
<td>External</td>
<td>External</td>
</tr>
</tbody>
</table>

Table 5. Characterizing Tau’s Sourcing Strategies

Any deviation from the buy strategy is well founded and is never done in an ad-hoc manner. One deviation is the web site where there was no standard solution available. Tau currently uses two separate consultancy companies for their web site. The first one, which is responsible for databases and application logic, provides an interface for the second one, which builds the user interfaces. In both cases a long-term relationship is sought, and the purchasing focus is on results (cf. Lacity et al 1996).

When tailor-made applications are deemed necessary, in-house development is rarely an alternative. Economies of scale, or rather lack thereof, are often proposed as a reason to outsource (Lacity et al 1996). For Tau with its limited size and budget, this is an important factor in outsourcing application development. Keeping development resources in-house is just not feasible.

However, lack of size has sometimes turned out to be a restricting factor. On some occasions, Tau has sent out requests for proposals to different vendors for a tailor-made application, only to have the vendors decline because the project was too small. Thus, Tau basically was too small to be interesting to the type of contractors interesting to Tau. It seems like Tau, looking to develop mission-critical IT with demanding requirements, felt obliged to approach larger consultancies to whom their projects were not large enough to be interesting. Presumably, smaller projects in most cases are managed by smaller consultancies.

5 Discussion

5.1 Identifying Patterns in Application Development Sourcing

<table>
<thead>
<tr>
<th>Type of Solution</th>
<th>Gamma</th>
<th>Delta</th>
<th>Lambda</th>
<th>Tau</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-house solution</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outside developers</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turn-key solution</td>
<td></td>
<td>Yes</td>
<td></td>
<td>Yes (Partly)</td>
</tr>
<tr>
<td>Standard package</td>
<td>Yes (Markets)</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 6. Summarizing Approaches to Application Development Sourcing
Simply by summarizing Table 2 through Table 5 in Table 6 patterns emerge describing how the different companies acquire mission-critical IT. Gamma combines in-house development and standard packages (Markets) into an emerging dual strategy. The strategy is emerging (cf. Mintzberg & Waters 1985) since there was no intended strategy at Markets to acquire standard packages. Instead it was the result of many, more or less independent, decisions to acquire such packages. Delta consistently keeps project management in-house and has several full-time project leaders. Code development on the other hand is outsourced to contracted programmers. Lambda relies on outside project management since the company only requests new functionality from the consultancy managing EdbTrade. Tau, finally, strives to standard package solutions in line with its overall low-cost strategy. When no such solutions are available choosing turn-key solutions come naturally since there is a lack of in-house resources for managing and carrying out development projects.

There are two aspects of the patterns identified in Table 6 that are important to bear in mind. Firstly, in real life few projects are as clear-cut as they may seem when looking at the table. When standard packages are acquired, there is often some need for adaptations either to the package or to surrounding applications. Similarly, in e.g. in-house solution projects there may well be some external programmers involved, too. Secondly, the focus is on mission-critical IT, which means that for instance various support applications are not considered. Obviously, all companies use standard packages as office productivity tools etc.

5.2 The Choices Revisited

When a proprietary solution is chosen, there is still a choice to be made as to how the resources needed should be organized, as illustrated in Figure 2.

![Figure 2. Choices in Application Sourcing](image)

As discussed above, this is a simplification since choosing a standard package more often than not requires some adaptations. The proprietary adaptations then has to be sourced forcing yet a choice to be made, and this time the options typically are restricted to the proprietary alternatives.

Parts of the IT sourcing literature emphasize the make-or-buy distinction (e.g. Rands 1993) and downplay the distinction between buying a standard package and buying a turnkey solution from a consultancy. Instead the distinction between resources that are provided in-house and acquired elsewhere is stressed. This is important, but for Delta, Lambda, Tau and Gamma it is not the only important distinction. The issue of code ownership and control is also central. In many cases it is far more important whether you control (and is responsible for) the application (“code”) or not then whether you produced it in-house or paid someone else to do it.
In terms of Figure 2 a proposed divider thus is between the turn-key solution and the standard package using the control and responsibility as key. This divider does not mean to replace but to complement the approach that puts the divider between in-house solutions and outside developers.

5.3 Explaining Patterns in Application Development Sourcing

Given these varying patterns in application development sourcing a natural question is to ask “how come?” As shown above, the case companies generally outsource code development, except for some interface programming needed for applications to interact. In fact, it is only Gamma that consistently develops code in-house, which is not unrelated to the fact that Gamma is larger than the second largest company, Delta, by the order of magnitudes. There are several espoused reasons among the case companies for not keeping development competence and resources in-house, where the two most prominent are:

- **Load balancing.** Development efforts call for a lot of resources when developing the application and significantly less when maintaining it.
- **Competence issues.** Keeping the staff up to date on every new technology is a daunting task.

Load balancing is generally not framed in this way or explicitly discussed. However, it can be read into the general discussions on cost reduction (e.g. Willcocks & Lacity 1998). Staffing to meet the development needs would mean over-staffing for the periods of time when no major development effort is underway and thus incurring excessive costs. Gamma moderates this to some extent by temporarily hiring external developers when the need for development resources peaks.

Competence issues are fairly well treated in the literature (e.g. Quinn & Hilmer 1994). When analysing IT sourcing, the concept of core competencies (Prahalad & Hamel 1990) can be applied to code development and project management. Outsourcing code development could then be seen as a consequence of whether it is a core competence or not. For example, project management would be a core competence of Delta but not Lambda.

Closely related to the competence issue is the issue of habit. If, as at Delta, project management is usually kept in-house while programming is outsourced, the organization acquires and retains the competence to manage project. When a new project comes along it is quite natural to do what one usually does, in this case project management. There is in a sense a soft path dependency, which is one explanation for the emergence of patterns.

Path dependency is also important in terms of legacy systems. Clean slate projects are few and far between. Normally application development projects are either additions to existing applications or new applications extremely tightly intertwined with existing solutions. In such cases, it is also natural that prior sourcing choices have a significant impact on new choices.

When discussing sourcing choices, even if primarily for mission-critical IT, it is important to keep in mind that these application play different roles in the companies. The application instance is different from the application role (Mårtensson 2003). A front-end application for example, is to Tau a means of entering customer orders into the marketplace system of the exchange. For Delta on the other hand it is a tool for the brokers to provide a superior service to their clients and for the traders to make good trades on behalf of the company.

This distinction is important when considering the traditional adage to outsource commodities and keep strategic applications in-house. The reasoning behind this adage is that strategic application is a source of competitive advantage and it will be possible to make a difference by having a better application. The reasoning in the case companies rather seems to be that you dare not renounce control over strategic applications, even though someone else may be better suited to develop the application.
Summing up three separate factors can be identified that explain both the differences between the companies and the emergence of patterns in their sourcing of application development.

**Company Size.** The company size has consequences for the degrees of freedom. A larger company (Gamma) may achieve economies of scale on the development level. A smaller but still sizeable company (Delta) may achieve it on project management level. The smaller companies (Tau and Lambda) have a harder time in this respect.

**Legacy.** Hardly surprising, an existing legacy portfolio seems to force some development, either on the application level due to the functional constraints of existing applications or on the interface level due to the amount of interaction between applications.

**Role vs. Instance.** In some cases it may seem that the same type of application is bought by one company and built by another for no particular reason. To explain this, it is important to take the role of the application into account. For example, Delta’s front-end application plays an entirely different role than, say, Tau’s. For Delta and its traders it can be a true source of competitive advantage, and the front-end application qualifies as a strategic application, while to Tau it is merely a supporting application.

6 CONCLUDING REMARKS

In specific situations all companies usually strive to use standard application packages rather than proprietary solutions. When no standard application package is available to provide the functionality deemed necessary, or the code ownership is important to the company, proprietary solutions are chosen. Despite the fact that the same decision rule is used across the companies (“buy standard package if possible, otherwise buy”) they draw different conclusions. A possible explanation to this is that the evaluation of the answer differs. Is it really possible to buy a standard package? This boils down to an evaluation of the expected benefits of, for example, a proprietary front-end compared to the expected cost. The difference in this evaluation can be, at least partly explained by the distinction between application roles and instances. Sourcing decisions for similar applications vary, partly due to the fact that the importance of the application to the company varies. The same application plays different roles in different companies, thus spurring different choices of standard packages vs. proprietary solutions.

A common belief among the companies is that the level at which standard application packages are used are rising. Standard applications packages are used when possible, and any competitive advantage comes rather from being able to integrate the applications than from the applications themselves.

To explain the patterns in sourcing of proprietary solutions both the legacy factor and the company size is important. Legacy in this respect refers both to traditional IT legacy and softer issues such as people’s experiences. Company size determines the degrees of freedom of the company. Is it large enough to achieve economies of scale on project management level? On code development level? As discussed above this refers both to competence and load balancing issues.

To conclude, companies seem to approach the standard package vs. proprietary solution similarly, but they draw quite different conclusions. Once a proprietary solution is chosen, companies seem to be quite consistent in the way they source the application development effort.

References


