ABSTRACT
An increasing number of senior executives are discovering that reaching their strategic objectives requires large investments in high-risk efforts to rebuild the information systems that support their core businesses. Few firms have addressed this problem to their satisfaction. Failures, restarts, and non-starts are common. While some of the traditional factors in failures of information systems development contribute to this unsatisfactory record, the principal difficulty is that core architectural reconfigurations are very different from ordinary application development programs. Conventional evaluation techniques, management approaches, and systems development methodologies can point a firm in the wrong direction for addressing architecture issues.

INTRODUCTION
Competitive applications of information technology (IT) are now a strategic necessity in financial services, retailing, and distribution [5, 6, 11]. As companies push to integrate information technology with their business strategies, senior managers soon find themselves facing tough questions posed by antiquated systems architectures. The speed with which new applications can be implemented and the functionality these applications can support are limited by data structures and applications architectures that often date back to the early 1970s.

Many senior executives have found that to reach their strategic objectives, they must invest substantial resources in programs aimed at rebuilding the information systems that support their core businesses. Such programs are high-risk because they:

- Require huge investments, frequently tens or even hundreds of millions of dollars.
- Provide the operational foundation for competitive differentiation (whether through unique customer offerings, an advantageous cost structure, or better management information for targeting resources) and for acquisitions.

- Lock the company into particular ways of doing business for a decade. Despite the increased flexibility of new technology tools, every system has design assumptions embedded within it that mirror business assumptions about how the firm manages itself, the products and services it offers, the market segments it pursues, and how it competes in the marketplace. Since these assumptions are often "hard-wired" into systems, the business is largely locked into that way of doing business for the lifetime of the system.

Examples of these strategic programs in financial services abound. New York Life, for example, is midway through a $150 million program to establish a central customer file and improve customer service [10], and Chase Manhattan recently completed development of a huge central customer information database [2, 12]. Manufacturing companies such as Chrysler are also investing heavily in integrated CAD/CAM systems, sophisticated MRP II packages, and customer-activated order-entry and service applications [13].

These programs can be particularly dangerous, because most of these companies—and their CEOs—have never before engaged in systems efforts of similar scale or strategic importance. Senior management thus has little experience to guide them as they make these fundamental "bet-the-business" (and my career) decisions. The results can be catastrophic. After spending $20 million on systems for its institutional trust business, for instance, Bank of America recently fired its top information systems executive and the head of private banking and trust because the system failed. The bank has set up a $60 million reserve and exited the processing business [3, 7].

Deferring the decision is hardly less risky. In an industry such as financial services, where information systems represent the "factory" of the company, pushing architecture problems aside can lead to a competitive disadvantage. The choice of whether, and how, to proceed with a major architectural reconfiguration is, therefore, particularly important and difficult.

A Complex Problem
During the past three years, we have worked with eight firms facing decisions about architectural reconfigurations, and have held discussions with managers at a dozen oth-
ers. Few of these companies seem to have addressed architectural issues to their satisfaction. Failures, restarts, and—most frequently—non-starts have been the rule rather than the exception.

The most notable pattern is the difficulty with which firms seem to build consensus around embarking on a new direction. While some firms—such as Chase Manhattan and Bank of America—have forged ahead with large efforts, many more seem paralyzed. One bank executive we spoke with has talked about architecture problems for at least three years, yet has gotten no closer to deciding how to proceed. Another common occurrence is repeated “re-starts,” as in the case of one financial services firm that decided several years ago to rewrite its core accounts processing systems. The project is now in its third incarnation of requirements specification; even as everyone agrees that something must be done, the project becomes more and more a source of cynical jokes among many IS staff. Of course, any development project can be subject to this sort of “analysis paralysis.” What is interesting is that this particular class of effort—strategic architecture reconfiguration—seems especially susceptible.

Few executives want to “bet the business” on a systems project. While there will always be some risk in architectural reconfiguration, good management can reduce this. The first step is to understand some of the unique challenges that differentiate architecture efforts from other large systems development projects.

**UNIQUE CHARACTERISTICS OF ARCHITECTURE EFFORTS**

To some degree the inability of firms to address their architecture concerns satisfactorily can be traced to some of the same factors that plague other large systems development efforts. The rules of good project management that apply to conventional development projects [1, 8] also apply to architectural reconfigurations. Some of the more spectacular disasters (for example, the failures of New Jersey’s vehicle registration system in 1986 [9] and Bank of America’s MasterTrust system [3, 7]) can probably be attributed to violations of these rules rather than to factors uniquely associated with their “bet-the-business” character.

Yet violations of conventional project management guidelines do not explain why architectural efforts are so often marred by false starts or simply never start at all. Nor do they help in understanding how to evaluate and design a core system reconfiguration. In several important respects, reconfiguring an architecture differs significantly from conventional applications development (see Table 1).

The issues raised by these contrasts make themselves felt early in the project life cycle, beginning with initial evaluation of the need for reconfiguration and continuing through specification and design. The typical result of a firm’s inability to deal with one or more of these issues is paralysis or, at best, extended analysis and redesign or rechartering of the effort. While this seldom results in public embarrassment, it can leave the firm at a competitive disadvantage.

---

**TABLE 1**

Contrasts between Conventional Application Development and Architecture Reconfigurations

<table>
<thead>
<tr>
<th></th>
<th>Conventional Projects</th>
<th>Core Systems Reconfigurations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Champion</strong></td>
<td>Business unit or functional manager.</td>
<td>Senior IS executive or above</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Deliver functionality.</td>
<td>Deliver a platform for the future.</td>
</tr>
<tr>
<td><strong>Scope</strong></td>
<td>Single product or function.</td>
<td>Multi-product multi-function.</td>
</tr>
<tr>
<td><strong>Design requirements</strong></td>
<td>Build on existing architecture.</td>
<td>Redefine the architecture.</td>
</tr>
<tr>
<td><strong>Management process requirements</strong></td>
<td>Eliminate uncertainty.</td>
<td>Acknowledge, accommodate uncertainty.</td>
</tr>
</tbody>
</table>

**Champion**

Today’s conventional wisdom says that every major IS project must be championed by a senior business unit manager. This is plausible for product-focused development efforts. Without adequate user sponsorship, a project can be starved for funds and management resources. But the situation is somewhat different for major architectural rewrites.

Consider an analogy. A conventional applications development project is similar to adding a new production line in a factory or modifying an existing line. Such a change is driven by the business unit; it happens only when the business says, “This product is necessary.” But reconfiguring a systems architecture is much more akin to rebuilding the entire factory. While the need for such an investment might have stemmed from changes in the product line or in other dimensions of the service/product configuration, the initiative comes from either the head of manufacturing or the CEO. These are the individuals responsible for the firm’s manufacturing strategy.

In a similar manner, the impetus for rebuilding a firm’s core systems architecture—the IS factory—must come from the senior IS executive or above. These are the executives who must set the information systems strategy of the firm. Where the IS executive is unable or unwilling to play this role of champion—as we’ve observed in this bank, where the businesses refuse to acknowledge the legitimacy of IS-initiated efforts—projects aimed at defining an architecture for the future will languish.

**Purpose**

Most application development projects are oriented toward delivering functionality. Application specifications focus on the particular screens, functions, and data structures the user needs. Development methodologies focus on techniques for determining what the user wants and translating these into concrete specifications. In contrast, architectural reconfiguration is oriented toward building a platform for the future. There is less concern with the screens, functions, and data structures of today than there is with those that might be required tomorrow.
This is a less concrete task than writing traditional specifications. The team needs a sophisticated understanding of the firm’s business direction and the ability to anticipate needs and options; its project manager must justify decisions in terms of the ease with which the new architecture will support such options. These are difficult decisions, with much room for second-guessing. The second-guessing contributes to the slow pace we have seen for architecture efforts at most firms. It can also lead to their stillbirth as project managers become frustrated at gaining agreement on any significant reconfiguration.

**Scope**

A second difference lies in the scope of an architecture project. Most application development supports a single product or function, while architecture reconfiguration touches many products and many functions—possibly all of them. One consequence is that the user support for architecture projects requires substantial cross-organizational cooperation, since most user organizations are aligned around products or functions. This increases the potential for conflict and indecision in the process.

One retail bank with which we worked found it virtually impossible to settle on a new architecture for its credit card applications. The problem was that the system supported two business units that served very different customers, and no one understood or represented the needs of both groups. The strategies of these two units were so different that key elements of the rewrite were deferred when they could not agree on requirements or priorities.

**Design Requirements**

Reconfiguring the architecture is principally aimed at removing the constraints that restrict the strategic options available to a firm. This makes the design task considerably more complex. Conventional application development efforts must conform to the restrictions embodied in the existing architecture. This may be frustrating, but it also removes several degrees of freedom from the design task, allowing the project team to concentrate its efforts on only those elements it can affect. Because the object of architectural reconfiguration is to remove restrictions from the architecture, the project team must address a problem with many more dimensions than usual. The breadth of such a charter can overwhelm the effort.

Furthermore, the applications development organization may well feel threatened when the project team proposes solutions that effectively undo some of their most “elegant” programming efforts (those designed around the constraints). Similar opposition emerges when new approaches introduce a technology (such as a relational database) that requires new skills.

It is often possible to achieve cost savings or service improvements during an architecture rewrite by reconfiguring the end-to-end chain of functions that support the marketing, manufacturing, and distribution of a product. Such improvements come not by automating existing processes but by changing the way things are done. While the net effect of the change may be positive, there can be winners and losers within the firm. At one major financial services firm, some of the biggest arguments over reconfiguration stemmed from the uneven distribution of the staff cuts that could potentially result from the effort. Not until senior management declared these changes to be part of the project’s objectives was the effort able to proceed.

**Management Process Requirements**

Standard application development methodologies try to eliminate uncertainty as part of the specifications process. But architecture efforts must embrace uncertainty. Some things cannot be known in advance, although they can be bounded. A large part of the project manager’s task in rewriting an architecture lies in determining the dimensions around which uncertainty will be tolerated—that is, determining how flexible systems should be and how to implement that flexibility.

Although this might seem like a trivial point, it can paralyze a project. When the project team has no way to identify the unknowns about which it must make assumptions and around which it will invest in building flexibility, work comes to a standstill. An insurance company with which we worked faced this difficulty in rewriting its core claims systems. The program was mired in conflict until the project team recognized which elements of the firm’s product mix had to be regarded as the “building blocks” of the overall set of products. Once that was done, flexibility in how products were supported (e.g., claims processed) could be accommodated.

**Other Obstacles**

Several additional factors raise barriers in the path of major reconfiguration projects:

- The benefits of a rewrite seem vague and their value is difficult to assess. Cost improvement is rarely enough—and is, itself, difficult to assess.
- Few firms have experience with architecture projects, which increases the perceived risk and uncertainty about the need.
- The perceived poor track record of applications development in many firms, which has resulted in increased emphasis on responsiveness to the user.
- The accounting treatment of software development, which expenses rather than capitalizes it as an investment, making it difficult to gain funding for an effort that can be as big as all other development projects combined.

For the most part, these factors are common to all application development projects. They are particularly relevant here because they have their greatest impact early in the project, when the firm is deciding whether, when, and how to proceed. Thus, these factors tend to make it more difficult for a firm to step confidently forward and address the architectural issues it faces.
KEY QUESTIONS

Our research continues both on the nature of the obstacles to architecture projects and on methodologies for overcoming these barriers. Many of the techniques commonly used to evaluate, design, and manage development efforts fall short of a firm's needs when management is deciding whether to bet the business. What this problem requires is a set of approaches that allow management to address the three key questions of a reconfiguration decision:

- Is there really a need to reconfigure the core information systems architecture? Reconfiguration can bring cost savings (e.g., in future development), greater applications effectiveness, and the ability to respond quickly to strategic opportunities. The evaluation methodology should accommodate uncertain benefits and the value of opening strategic options for the firm.

- How can flexibility be built into the new architecture? Systems flexible enough to meet all potential requirements would be prohibitively expensive. The most difficult task facing the project manager of an architecture reconfiguration is deciding on the strategic dimensions around which the new architecture will be built.

- What approach to implementation—evolutionary or revolutionary—should be attempted? Some firms have chosen to evolve toward a vision rather than attempt a multi-year program to rewrite all or part of the core systems. Others have engaged in comprehensive development programs. Both approaches have their share of successes and disappointments. What is needed is a methodology for controlling the implementation of architecture projects in a way that limits the risks such efforts present while avoiding the trap of analysis paralysis.

SUMMARY AND CONCLUSIONS

The information system architectures of the 1970s, enhanced by extensions and maintenance efforts for over a decade, are now becoming a serious constraint for firms for whom information technology is playing an increasingly broad strategic role. Whether the goal is strategic necessity or competitive advantage, senior management is finding that it often cannot reach it.

Although a growing number of firms are taking a critical look at their core systems architectures, few have addressed their problems to their satisfaction. This poor track record seems to be associated with unique characteristics of architecture efforts that lead to such projects bogging down in the evaluation and design stages. To date, few firms have satisfactorily resolved the problems raised by antiquated core systems architectures. Reconfiguration efforts seem unusually prone to bogging down in the early stages of evaluation and design. This is due in large part to the unique challenges associated with architecture rewrites.

A firm reconfigures its core systems to build a platform for future applications. Such an effort is only partially driven by current functional needs, cuts across multiple products and functions, must incorporate a much greater degree of freedom in the design, and accommodates uncertainty rather than eliminating it. Conventional evaluation techniques and systems development methodologies can point a firm in the wrong direction for addressing these issues.

Reconfiguration efforts depend on matching strategic assumptions with systems potential, a task that resists conventional methods for translating user requirements into screens, functions, and data structures. Conventional wisdom holds that applications development efforts must be driven by users. This also appears incorrect for core systems efforts; IS must take the lead in this area. Yet, knowledge of the firm’s business is even more critical to the success of reconfiguration efforts than conventional application development projects, for the effort must be tightly tied to the firm’s business strategy.

These characteristics of architecture projects should be accommodated in the charter of a reconfiguration effort and selection of project staff. This is the first step in reducing the risk associated with reconfiguration and ensuring timely implementation.

ACKNOWLEDGMENTS

Our thanks to Eric Clemons of the Wharton School and to James Cash of the Harvard Business School for their insightful suggestions; to Steven Wayne, a student at the Wharton School, who provided invaluable assistance with the literature review; and to Michael Row of the Wharton School and Janice Brodman of the Harvard Institute for International Development for their comments on this paper.

REFERENCES


