

The **Paris guide** to IT architecture

Jürgen Laartz, Ernst Sonderegger, and Johan Vinckier

City planners try to preserve viable old assets, to replace outmoded assets, and to add new assets—all in the context of an infrastructure linking them coherently. IT developers have a good deal to learn from that approach.

Companies that want to gain a competitive edge, whether by being the first into a market with new products or by launching an electronic-commerce channel, know how much they depend on information technology architectures to achieve their aims. Usually, though, the architecture is a costly and aging maze of applications, hardware systems, and networks. Far from making it possible to achieve strategic goals, it can make a mockery



of them. But by looking at the evolution of another complicated set of systems—those that make up a modern city—senior managers can begin to understand more fully how the controlled and rational development of an IT architecture can enhance the ability to compete.

Stories about companies that stumbled because their IT architectures couldn't accommodate rapid and drastic change are legion. Fast-growing companies are liable to hit a wall when their architectures fail to expand quickly enough to serve new customers cost-efficiently. Long-established companies can lose market share if their architectures lack the flexibility to move products to market as quickly as their competitors do.

A leading international bank discovered this problem the hard way. Its strategy involved launching an Internet channel that included an on-line securities brokerage, but its deadlines were so tight that there was no time to integrate the customer interface with the back-end systems where transactions were to be processed. The bank's back-office staff therefore had to input each one by hand, an error-prone process feasible only in the early days of e-brokerage, when volumes were tiny.

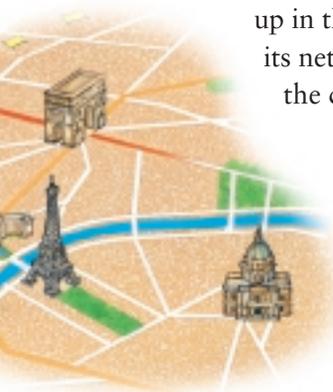
How could the bank's IT managers have allowed the systems they supervised to undermine rather than advance a strategic goal? The design decisions that had shaped the systems landscape over the years were, individually, sound. The people responsible had focused on delivering the required functionality on time and within budget. But no one had kept an eye on the big picture. The overall systems landscape thus became too complicated, extremely costly, hard to manage, and incapable of responding flexibly to the bank's business needs.

Jürgen Laartz is a principal in McKinsey's Frankfurt office; **Ernst Sonderegger** is a principal in the Zurich office; **Johan Vinckier** is a principal in the London office. Copyright © 2000 McKinsey & Company. All rights reserved.

This article can be found on our Web site at www.mckinseyquarterly.com/infotech/pagu00.asp.

Whether chief executives wish to undertake new e-commerce or customer-relationship-management (CRM) initiatives or to replace existing IT systems, they can learn from the way modern cities adapt to the needs of residents and businesses. For, in the same way, a properly conceived and administered IT architecture adapts itself to the diverse and changing needs of the company that deploys it.

We could use several cities to make the analogy between city planning and IT architectures, but Paris seems particularly apt. When you walk around Paris, you see a wonderful variety of buildings dating from many centuries; a closer look reveals that a majority of these structures went up in the past hundred years. The infrastructure of Paris, such as its network of roads and bridges, unites these buildings, defining the cityscape and setting the terms in which it has evolved.



When Baron Georges-Eugène Haussmann was commissioned by Napoléon III to create a new plan for parts of Paris, in the 1850s, he cut boulevards through the existing pattern of streets to facilitate commerce and the movement of troops. Since then, Paris has carefully planned the redevelopment of sections of the city, so that neighborhoods and districts have distinctive social or economic roles. Most office towers, for example, are located outside the historic borders. As a result, the Paris of today, with its Beaubourg and La Défense, is a modern and highly efficient but unmistakable patchwork of many eras. City planners try to preserve and renovate those old assets that are still viable while replacing others and adding new ones in a coherent way. Finally, a city like Paris makes sure that a good infrastructure binds all these assets together for the long term.

This city-planning analogy can help established companies avoid IT architecture problems. A company might believe that major chunks of its applications are so unfit for current needs that they might as well have come from the Middle Ages. At the same time, IT departments are repeatedly asked to add new functions and to integrate business units and allied companies. This ceaseless growth in automation has made it less and less feasible to replace all systems completely. The challenge is to stay abreast of the different life cycles of a given IT system's components so that the architecture as a whole doesn't become obsolete (*see* sidebar, "Case study: A leading international bank," on the next spread).

The planning of cities like Paris has taught us four lessons that can help companies manage the evolution of their IT architectures to further their strategic initiatives.

1. Define a long-term plan

Most cities have zoning codes that designate some areas as residential, some as industrial, and some as parks. While mixed use (shops with apartments upstairs, for example) is permitted in some districts, certain functions are incompatible with each other: a responsible planning board would not, for example, permit a smelting plant to be located next to a hospital.

Similarly, good IT architectures break down complicated applications landscapes into coherent, manageable parts often known as domains. Typically, each domain performs a discrete function. An insurance company, for instance, would have domains such as product systems for life or automobile insurance and channel systems for call centers or physical branches.

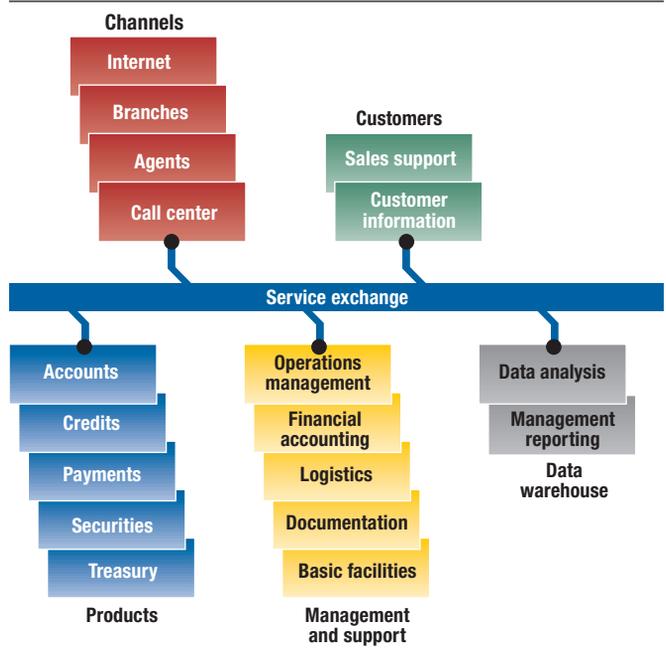
This scheme permits IT functions to be built in one domain and then made available to other domains that need them. A product-pricing algorithm, for instance, could be built in the product domain and then made available to all channels. But a call center and an e-commerce channel that each built its own product-pricing algorithm would be needlessly duplicating each other's IT effort. There would also be a serious risk of inconsistencies—conflicting price quotations, for example—even if two channels sold the same product under the same brand.

IT architectures that are opaque and tangled should be broken down into clearer domains. Within a financial institution, this effort might involve disaggregating a system providing both channel functions and product functions. By contrast, IT functions that are identical but dispersed across an

architecture—databases containing different bits of information about the same customers, for example—should be grouped into a single domain. Exhibit 1 depicts one bank's vision of what its IT architecture ought to be.

EXHIBIT 1

Plan the city block by block: A bank's IT architecture



2. Build a stable interface infrastructure

To offer neighborhoods standard services such as power and water, a city needs a stable infrastructure. The infrastructure must serve prospective as well as existing needs and have uniform interfaces—including the same

Case study: A leading international bank

Senior managers of a large international bank developed a strategy to get products to market faster, offer more customized services, and reach larger numbers of customers through new channels. To execute the strategy, the bank had to redesign its information technology architecture, and it decided to do so with an approach incorporating lessons about how cities grow. Three possibilities were considered: a greenfield approach, with each division building a new platform tailored to its requirements; migration to a banking software package using a modern IT architecture; and the transformation of existing systems according to an IT architectural master plan—something akin to a long-term city plan. The third approach would have to permit the immediate addition of new kinds of functionality, such as electronic commerce, in parallel with a thoroughgoing renovation (or even replacement) of some core systems.

The first option was rejected because the costs of the new system and the time required to build it would have been excessive. The second option was tempting in theory, but banking packages capable of handling the bank's product breadth and high volumes were not available. The third option had three important advantages. It could be introduced gradually, starting with the existing application portfolio. It would continue to use existing applications as they evolved, thus reducing the payback period on the investment to a year. Finally, it would spare the bank the

need to spend at least five years developing and maintaining the old and new architectures in parallel.

1. Define a long-term plan

The bank's IT architecture left much to be desired. The customer database held payment information—an arrangement resulting in unnecessary interactions between software engineers from the bank's customer side and software engineers from the product side. The deposits system was so closely woven into the securities and customer information system that several attempts to replace it with a more modern version had failed. And the Internet channel systems offered access to the back-office systems only through an aging interface, a problem that became a performance bottleneck delaying new software releases and jeopardizing levels of service.

To resolve these issues, management designed an IT architecture: the long-term city plan. It laid out the components—that is, the "city's" zones (the securities domain, the customer information domain, the credit domain, and so on), the data they should contain, and the way they were to communicate with other systems. This set of rules was intended to transform the systems landscape into a more modular environment where particular systems could be altered or exchanged to minimize the impact on other systems.

types of outlets, plugs, and voltages—so that business can be carried out not just among neighborhoods but across an entire country.

Similarly, the interface infrastructure of an IT architecture makes it possible for domains to exchange information and instructions—what IT specialists

2. Build a stable interface infrastructure

Just as streets usually have a longer life than do individual buildings, so too the communications infrastructure and interfaces between systems tend to remain in place longer than the systems themselves. One challenge in making the bank's IT architecture work was to design and implement communications and data exchanges between the separate domains. To do so, the IT department had to master new technology and application design skills. Most difficult, groups of software developers pursuing the city-planning approach could no longer make private arrangements among themselves about how to integrate their applications.

3. Appoint a zoning board

Both to ensure that the systems developers adhered to the rules of the IT architecture and to get a long-term plan adopted, the bank's chief information officer created and chaired a steering committee. Every deviation from the rules had to be reported to the committee, which granted approval, at best, on a temporary basis—and then only on condition that there was a clear plan to make the system architecturally compliant.

4. Make the most of what you have

It was clear from the outset that the bank had no practical chance of eliminating its legacy systems in the near future and that, in any case, doing so would be uneconomical. On the posi-

tive side, so many functions had been built into the systems over the years that the bank could offer a wide array of customized services (such as tax statements for customers in different countries) and an extremely broad set of products. Even Russian government bonds from the pre-Communist era could be processed in securities accounts.

The bank thus began renovating its legacy systems to reduce their complexity while maintaining their broad functionality. To permit the almost round-the-clock use demanded by e-commerce applications, the bank overhauled applications monopolized at night by accounting batch runs. This approach extended the life of existing systems and in this way lowered the amount of new investment needed, while the systems' reduced complexity brought down IT operations costs.

Overall, the bank found that its evolutionary approach addressed its changing IT needs in an economical and low-risk way that enabled it to respond properly to urgent business priorities, such as e-commerce, while resolving fundamental problems in its systems. The bank expects the effectiveness of its IT architecture to improve so much that it will recoup the substantial investment it made—an average of 20 percent of its annual IT development budget for three years—within a year of finishing the project.

refer to as “services”—with middleware functioning as the transporting technology. A customer information system, for example, should provide a standard service (name, address, customer number, birth date, and family members) regardless of whether it is requested by a salesperson’s application or the company’s Web site.

Ideally, all interfaces between domains pass through a limited set of stable and standard services. If they should lack stability, a change to any one of them would require all domains using it to go through a maintenance cycle. When a database of customers is migrated from an older mainframe to, say, a newer UNIX server, channel applications need not be affected as long as the customer’s profile service hasn’t changed.

Services also drastically limit the number of access routes to a domain. Once upon a time, all developers might have been able to blaze their own trails in search of the information they needed in other domains, but services force all developers to access domains in a standard way. Just as each city resident uses the same type of plug to draw a standard type of voltage, a channel system or a customer-billing system should retrieve a customer’s profile with a standard service. Some financial institutions have set themselves the goal of gradually replacing the tens of thousands of differently tailored interfaces among their domains with a set of 500 or so services.

3. Appoint a zoning board

Many cities have created zoning boards to draw up and carry out long-term plans that both reflect and help shape evolving patterns of public use and the general preferences of elected officials. Similarly, a company needs its own zoning board, usually a separate group within the IT organization, to manage the evolution of an IT architecture.

The board’s role is to define the discrete domains and the functionality belonging to each. Domains can be defined according to which kinds of functionality and data belong together, the de facto standards that software packages set for them, and which part of the business should manage them. To achieve IT efficiencies, the board should also identify the technologies and software development tools to be used in the domains. Finally, the board must ensure that the company builds and manages a strong and highly stable service infrastructure.

A bank’s zoning board might, for example, decide that certain kinds of software belong in the call-center domain: the software that generates the scripts telephone operators read when customers call, the software that integrates the computers of those operators and their telephones, and the software that

generates reports on call-center performance indicators. The board might also adopt UNIX as the system's platform and require all interfaces with software in domains other than the call center to pass through the standard services.

Occasionally, managers seek an exception to the rules—in city-planning jargon, a “variance.” If, say, company X offered a delayed-payment function, which allows a bill to be paid on a specific date at the customer's direction, the managers of company Y, a competitor, might ask to have a similar function built into their own call-center application if the payments domain couldn't accommodate the new feature immediately.

Normally, deviations of this nature aren't allowed, since they deny other channels easy access to the function. But in specific cases, the zoning board, adopting an overall business perspective, will choose to weigh trade-offs: waiting until the payment domain can deliver the feature, allowing the call-center domain to deliver it temporarily, or delivering a quick fix by writing an add-on program in the payment domain. The last option is generally the preferable one, since the people in charge of the payment domain would then be responsible for turning an improvised solution into a well-engineered feature of that domain.

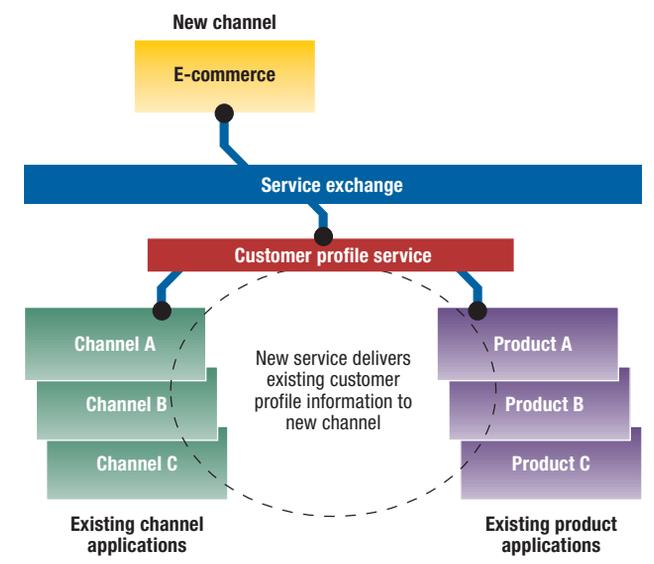
4. Make the most of what you have

Before making new investments, good municipal leaders ensure they have made the most of existing assets. Cities, for example, would generally try to add bandwidth to existing copper wire before submitting themselves to the disruptions of laying fiber-optic lines.

Likewise, companies that contemplate building new systems should first try to get what they can out of their current applications, however messy. Such companies might, for instance, be able to build adapters on to existing applications—a practice known as

EXHIBIT 2

First exploit existing systems



“wrapping”—so that they can be hooked up to a standardized service infrastructure. Wrapping permits a company to deliver a clean and stable customer profile service, for example, to an important new e-commerce domain even when the data are drawn from a variety of existing systems (Exhibit 2, on the previous page). A data warehouse that serves the purpose of supplying background for a CRM initiative could be added in the same way. Under this approach, new applications and domains such as e-commerce can be insulated from the problems of an existing IT architecture even while it is being renovated or replaced.

Companies reviewing their architectures will probably discover a multitude of past “city-planning” mistakes—the IT equivalent of putting a smelting plant next to a hospital. In general, companies can’t justify correcting all of these mistakes, but new business initiatives can provide the occasion for doing so.

When an IT architecture embodies decades of ad hoc decisions and short-term projects, it can be no less disorderly than Bangkok or Mexico City. But even planners in those cities look for evolutionary ways to exploit important assets. Tearing them down and beginning again or building a brand-new city next door to the old one is rarely a serious possibility.



Instead, cities adopt change programs. They typically start by mapping the roles of existing zones and the capacity of the transportation infrastructure. Then they calculate how much growth to expect in particular areas within a given time frame and how much infrastructure must be added or restored to support that growth. Finally, public undertakings are put out to tender, and private proposals that are not “as of right” are submitted to the approval process. Often, public-private partnerships—such as New York City’s Lincoln Center, built in the 1960s on the site of slums—have a transforming effect on the surrounding neighborhood and beyond.

Similarly, an IT program starts by determining the extent of problems such as duplication. If the company’s goal is to become a multiproduct, multichannel financial institution, for example, it must organize domains around those key areas of functionality. A first step toward implementing this architecture will usually be the construction of services that wrap current applications; these services can then offer the existing capabilities of legacy systems to new applications in a stable, standardized way. Once a service infrastructure of

this type is established, applications can coexist with new kinds of functionality, such as e-commerce. Gradually, the IT architecture becomes an evolving asset, with applications in each domain—all bound together by a stable service infrastructure—added, enhanced, renovated, and replaced along different time lines.

Our four lessons from city planners have helped companies begin redesigning their IT architectures in order to get products and services to market more quickly and to improve their returns on IT assets. Even chief executives who believe that their companies' IT architectures resemble Mumbai rather than Paris are likely to find that an approach taking into account the existing fabric while laying the groundwork for growth will turn their IT assets into virtual cities of light. *MQ*