McKinsey&Company

MARCH 2010

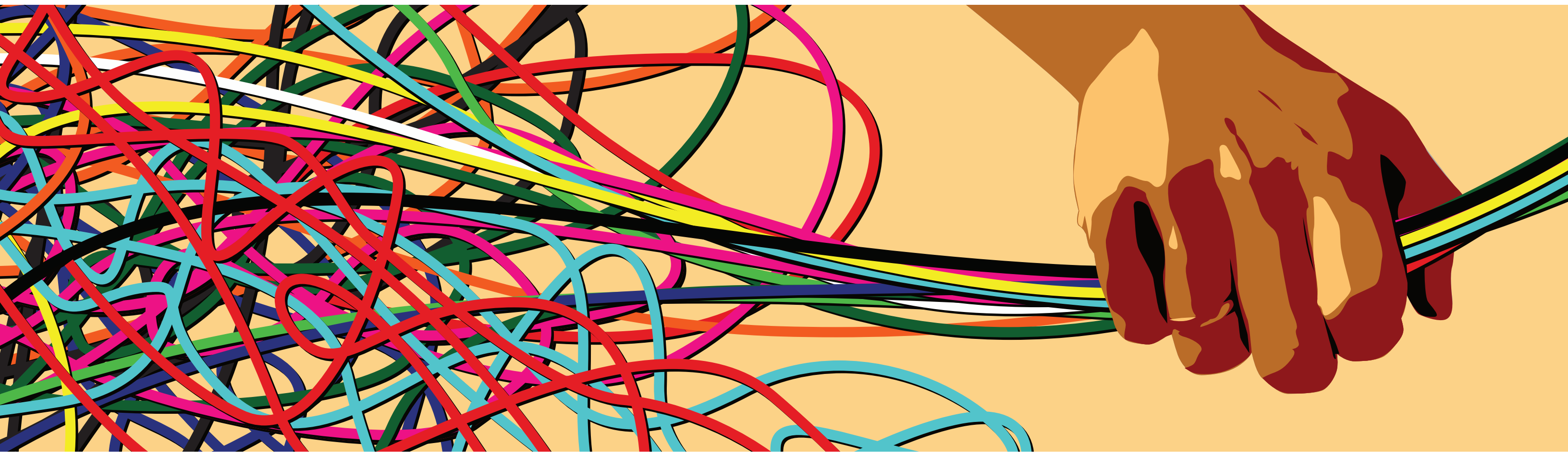
McKinsey Quarterly

BUSINESS TECHNOLOGY OFFICE

Tackling IT complexity in product design

As more products are loaded with technology, tangled IT designs can undermine product strategies. Product managers and technical specialists need a better game plan.

Marcus Schaper



Today, it seems that just about every product contains some sort of embedded computing technology. Cars, phones, even washing machines boast interactive features that would not have been imaginable, much less affordable, a decade ago.

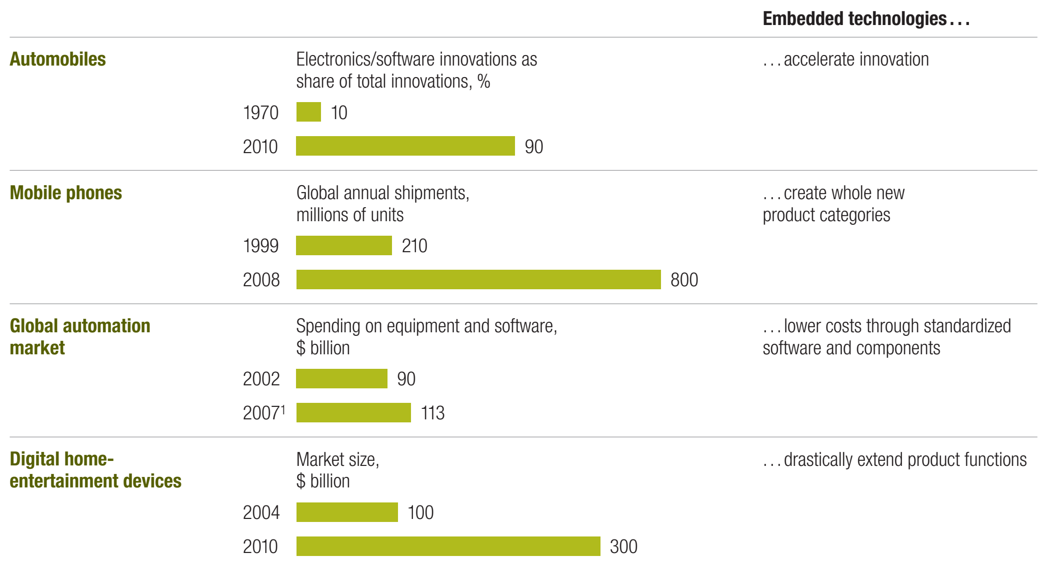
That such products have entered the mainstream is easy to understand (Exhibit 1). Smart phones, electronic navigational equipment, and Wi-Fi-enabled TVs offer convenience, portability, and personalization at reasonable prices. But the price of such progress is growing IT architecture[1] and design complexity.

Consider the fact that in the past five years, the average number of tech-enabled engine control units in each new vehicle produced by the automotive industry has risen to 80, from 20. In the mobile-phone sector, the number of IT-based updates per year is about 40, more than twice the level back in 2000. And despite challenging economic times, the number of avionics features—from cockpit VoIP and wind sensors to the electronic flight manuals introduced in newer aircraft—has doubled in recent years. The growing demand for electronics-based product enhancements finds companies across industries struggling to keep costs in line amid fast-changing technologies and the constant pressure for product upgrades.

[1]Architecture refers to the technical and business model used to plan and govern the design, functionality, and integration of software, hardware, and IT.

Exhibit 1

A growing demand



Source: ARC Advisory; global industry analysts (for industry automation); IBM; IDC; Mercer; McKinsey analysis

The problem of technical complexity

The pace of product development is creating enormous engineering and technical-development challenges. Traditional product development was driven largely by hardware considerations. When you pressed a button on an analog telephone, the button dialed one number, and that was the limit of its functionality. Today's tech-enabled products depend on the successful integration of multiple hardware *and* software components—a multidimensional process. Now, a single button on a new smartphone may connect, in different ways, to a dozen distinct applications.

Software is by nature more abstract—strings of program code are pieced together in interconnected layers. The IT architecture underlying new product designs is thus far more intricate than the specifications of traditional products. In the predigital world, for instance, one vacuum cleaner operated more or less like any other. But throw in a silicon chip, and today you might have a Roomba: an intelligent robot vacuum cleaner guided by sensors and processors.

Poor product architecture

Development teams, beset by demands for customized features, may fail to realize that poor product architecture decisions upfront can have costly downstream consequences. Those teams, often led by electrical engineers, sometimes lack the specialized software-engineering skills needed to anticipate potential programming, upgrade, or reuse issues. This problem can create a vicious cycle, where poor design decisions and architecture lead to unmanageable code and greater complexity. In the auto industry, for example, a recall resulting from electronics issues cost a manufacturer close to €300 million. These design mistakes can tarnish a company's reputation—as a high-end automaker learned after introducing new user interfaces that proved overly challenging to operate and broke down as a result of software problems.

Juggling a range of technical requirements for any single product can hinder a company's ability to think more broadly about how certain features and functions might be leveraged across its product portfolio. This failing can force companies into a series of "one-off" applications. The Apple engineers who designed the iPod touch successfully combined new hardware (a touch screen) with better software logic (an intuitive, user-friendly interface), but others struggle to create flexible, integrated architectures. A home-electronics supplier, for instance, had to design a new user interface each time it upgraded a product—which put it at a disadvantage to competitors that took a more modular (or plug-and-play) approach to product development.

Weak linkage with business priorities

For some organizations, the bigger issue is that the technical challenges of tech-enabled products often conflict with business strategy.

Many companies have decades of experience developing mass-produced hardware-oriented products, such as televisions, radios, and home appliances, with relatively long shelf lives. Design and R&D processes have therefore been optimized around volumes and efficiency. The new era of tech-enabled products, with their niche features and faster pace of product change, requires a different set of processes and skill sets, as well as a longer learning period while organizations reshape traditional ways of doing business. The result is a significant risk of escalating costs, cycle times, and mission creep. Product managers seeking to eclipse the competition may push for next-generation design changes that stretch the limits of current technology. IT developers keen to leverage an application's reach may overengineer a group of features.

For product strategies to be successful and sustainable, business considerations must drive technical ones. In an effort to deliver a consistently high-quality gaming experience, for instance, Nintendo deliberately kept the architecture underlying its Wii system simple, limiting the feature set in favor of rigid quality and control standards. Those traits delighted consumers and made the platform a best seller. Other companies may overlook the importance of ensuring that business audiences understand the far-from-intuitive choices that go into a product's underlying architecture.

The abstract nature of many embedded IT systems makes functionality hard to describe. Nontechnical managers in the business units can struggle to determine which set of features and options is suitable from the standpoint of cost, ease of use, and process time. A mobile-phone company, for example, learned this lesson when unclear lines of authority led its product-development and engineering teams to argue over which of them was responsible for the features, costs, and timetables of a certain product. The confusion resulted in long lead times in completing it and a failure to offer technology matching that of the company's rivals.

Cost management is also far more complex in the tech-enabled-product environment, where life cycles for software and hardware components often don't align, making architectural integration difficult. The present tough economic environment exacerbates this problem. Managers are under intense pressure to bring new products to market quickly while also saving costs. In the absence of a clear development framework that brings both technological and business considerations to bear, development teams too often resort to quick fixes or "strokes of genius" rather than more sustainable solutions. To meet a launch date, for example, engineers at one company built a device with readily available, function-rich, but expensive third-party software. That helped the company meet its release target but exposed it to higher-than-expected costs.

Capturing greater value

With consumer demand for tech-enabled products continuing to grow across industries, some manufacturers are addressing these issues and optimizing electronics architectures.

Align business and engineering goals

Underlying the success of some companies in industries such as automotive and high tech is an integrated approach to designing the architectures of electronics products. In practice, that means aligning the product vision with design, the road maps of individual products with the broader electronics platform strategy, and, ultimately, the business side of product management with the engineering side. Only when companies set clear goals that demand such an alignment will customer and commercial considerations remain squarely in the center, grounding the development process and minimizing unnecessary complexity. The difference between a well-run and a poorly run development unit can vary overall productivity by a factor of ten.

A good architecture has a number of important characteristics. It is modular, allowing sections to be tagged, stored, and applied in different products. It is built on standards, providing for easier integration. It is configurable, letting one system serve many customer requirements. And it is updatable, allowing new features to be implemented without any need to discard large parts of older releases.

This list may seem straightforward. But it's one thing to recognize a good architecture, another to build one and keep it in good shape. Setting the right goals also requires clarity about the dimensions of the task. Consider what goes into the body of a typical tech-enabled product. Hardware, plastics, resins, nuts, and bolts make up the skeletal frame. Transistors, microcircuitry, and other kinds of electronics manage the flow of information, much as tissues and veins do in living things, and software provides the neurological connections that direct and control operations. The layered architecture defines that anatomy, specifying everything from the user interface to the way the product functions to its interactions with other systems and components.

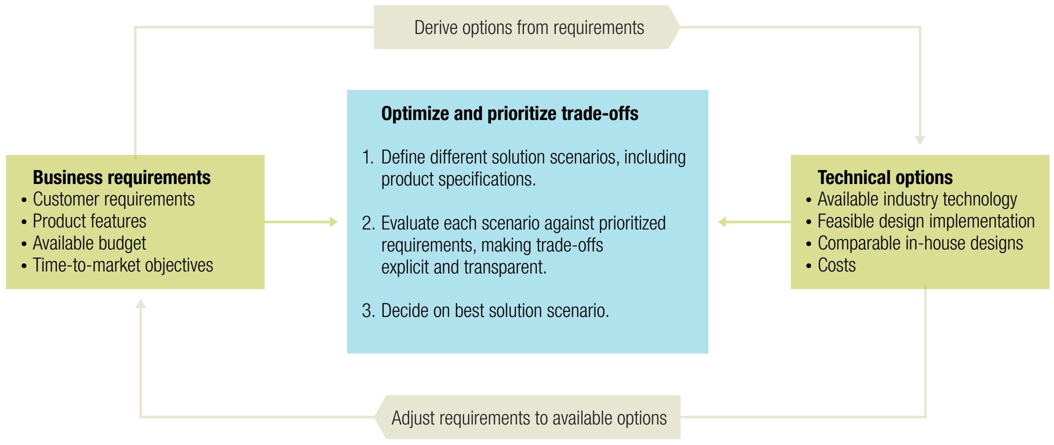
Adopt a transparent process

Companies must adopt a management process that optimizes the way these layers work together. The process involves a new form of collaboration among engineering, marketing, product design, and other product functions. Interaction helps IT- and engineering-development teams balance what the market demands in a feature set against what the business requires in costs and cycle times—all, of course, within the realm of what is possible technically. As Exhibit 2 illustrates, the best solution usually involves balancing multiple trade-offs.

Along the lines of this framework, teams from both business and IT begin by mapping their product requirements and addressing such questions as, "Who is the buying audience?" "Where is the greatest opportunity?" and "What are the right features?" Factoring in cost, budget, and other constraints, teams emphasize features likely to have the greatest impact on customer satisfaction. Those features are translated into a range of architectural components, each with its own strengths and weaknesses from a cost and

Exhibit 2

A balancing act


Derive options from requirements
Optimize and prioritize trade-offs
1. Define different solution scenarios, including product specifications.
2. Evaluate each scenario against prioritized requirements, making trade-offs explicit and transparent.
3. Decide on best solution scenario.
Business requirements
• Customer requirements
• Product features
• Available budget
• Time-to-market objectives
Technical options
• Available industry technology
• Feasible design implementation
• Comparable in-house designs
• Costs
Adjust requirements to available options

performance perspective. As teams toggle through which component options make the most business and technical sense, the answers define the underlying product architecture.

With the business requirements sorted out, the teams examine their architectural design options. They may find that what they need is not yet commercially viable or that it requires too much customization for mass production. Such constraints oblige the teams to toggle between what they want and the real-world options in order to find the most suitable architecture. This iterative process forces business and IT perspectives to fuse, creating an alignment between the product's overall strategic objective and the appropriate tech-enabled architecture.

Focus on a subset of possible architectures

One mobile-phone maker needed to create a low-cost handset for sale in emerging markets. Business requirements dictated a rugged design and a limited feature set. After a series of iterations, the engineering team presented a number of design options that emphasized electronic and mechanical components that could withstand harsh physical conditions, coupled with a small and relatively inexpensive processor. Another mobile-phone maker, with plans to compete for the iPhone demographic, had a different set of business requirements, which favored a more complex, memory-intensive architecture supporting a variety of features, despite the higher costs.

The ability to balance different design options in accordance with business strategies is the basis of an optimized tech-enabled architecture. Yet different facets of an architectural framework support different aspects of product performance—for example, those that govern customer-facing processes, define how software and hardware relate, or guide the interplay among applications (Exhibit 3). We have found that teams can simplify the process by focusing on a few specific architectural facets, weighing their relative importance to the overall set of business objectives and product capabilities.

Consider what happened at an automotive supplier struggling to improve its fuel injection system. In the company's original siloed world, engineers built custom IT components from scratch for each product upgrade. This approach bogged down the production timetable and prevented the company from keeping pace with other market leaders.

With improvements in time to market as the main business goal, the company established a cross-functional product team that sat down to work through the iterative process described here. To stop building everything from scratch, the team agreed to create an internal library of resources to make better use of existing technologies. These embedded software systems and widely used electronic components were governed by the architecture's capabilities (domain) layer. To streamline development and reduce component costs, the team sifted through the list of required fuel injection hardware and identified basic components that could be standardized across multiple engines. To speed up development time, engineers and product managers decided to take advantage of modular software and electronics elements. These could be plugged into a variety of different applications that made the fuel injection system work.

The resulting electronics architecture brought products to market twice as quickly as the older, more labored one. By optimizing the interior electronics, thus replacing a complex architecture with a simplified one, the company saved a few euros for every car it built. This savings added up to more than €10 million over the entire series. In addition, the

Exhibit 3

A layered approach

Architecture layer	**Function**
Use case	Defines how user will interact with product—eg, through graphical user interface
Capability (domain)	Identifies groups of product capabilities required to operate user processes, as well as environmental constraints (based on intersection of hardware and software capabilities—eg, ability of mobile phone to access network)
Application/integration	Maps applications to domains and defines interface between applications and software environment in which they work—eg, how car's engine controller sends information on revolutions per minute to instrument cluster
Software technology	Describes software technology stack from top to bottom—eg, which operating system is used to run applications and how data are stored
Hardware abstraction	Defines how applications are distributed across product infrastructure (electronics and hardware) and how they can be made independent from individual hardware elements—eg, which application runs on which control unit in a car with many networked electronic control units
Hardware technology	Identifies list of hardware technologies to be included—eg, which processor and storage types are used

platform, initially designed for four-cylinder engines, can now be used in eight-cylinder engines as well, saving the company additional costs and development time.

In another example, a power-equipment supplier was eager to get its new windmill product line up and running. But the advanced control facility for these windmills faced a sizable hurdle. The company needed to find a cost-effective way to monitor how much power the windmills were supplying to the grid. The business requirements for this feature made it clear that the best solution would be a cheap control device with an architecture that allowed it to be scaled up easily across a network of windmills.

The technical team decided that the domain-based architecture layer was the place to concentrate efforts to meet the product requirements. It met with the team from the business unit and presented three options for managing the system's processes: a general-purpose processor, a microcontroller, and a digital signal processor. All three would measure power output, but each solution had its own costs and benefits. The technical team needed to make sure that the product managers understood the various trade-offs so that together the technical and business sides could make the optimal choice for the new tech-enabled architecture.

Weighing the three alternatives, the team found that the general-purpose processor had the advantage of being easy to install and upgrade, but the hardware supporting the processor had to be purchased separately, making it too costly. The digital signal processor offered a stripped-down operating-system architecture that was cheap to develop and could monitor basic power use, but it could not provide needed billing and reporting features. This left the microcontroller as the best option. It cost more but gave the product team the flexibility of a complete off-the-shelf solution, since all the needed hardware and software was built right in.

Building a better product-development organization

Integrated development of tech-enabled products requires an equally integrated management approach. One successful company began by creating a project steering group led by the overall product group manager and the chief technology officer, who together outlined the product platform strategy and directives. Working below this leadership level, business and engineering teams mapped out the consumer and technical requirements and then came together to discuss and prioritize the elements of the final approach. The teams shared the resulting architecture with the product group manager and the CTO to confirm that the solution would meet the company's consumer, cost, and market delivery objectives. They also established a series of performance metrics to quantify cost and productivity gains and to further refine their ideas about where improvements could be made. This holistic procedure helped ensure that both the business and the technical team focused on the features that mattered most—those tied to the product's overall strategic objective.

Related thinking

"IT Architecture: Cutting Costs and Complexity"

"Better Software into Manufactured Products"

Establishing the right architecture for tech-enabled products is not a one-time effort. It's an ongoing process that is especially important to support the product life cycle. When the search for the right architecture is elevated to a discipline followed by both engineers and the business side, companies with tech-enabled products can experience gains in productivity, quality, and costs. ○

Marcus Schaper is a principal in McKinsey's Frankfurt office.