strategy&

Formerly Booz & Company

Preventing complexity drift

&

A capabilities-driven approach to IT program success

pwc

Contacts

Chicago

Mike Connolly
Senior Partner
+1-312-578-4580
mike.connolly
@strategyand.pwc.com

Düsseldorf

Jens Niebuhr
Partner
+49-211-3890-195
jens.niebuhr
@strategyand.pwc.com

Kuala Lumpur/Sydney

David Hovenden
Partner
+60-3-2095-3188
david.hovenden
@strategyand.pwc.com

London

Richard Bhanap
Partner
+44-20-7393-3519
richard.bhanap
@strategyand.pwc.com

Eduard Gracia
Principal
+44-20-7393-3200
eduard.gracia
@strategyand.pwc.com

São Paulo

Ivan de Souza
Senior Partner
+55-11-5501-6368
ivan.de.souza
@strategyand.pwc.com

Shanghai

Sarah Butler
Partner
+86-21-2327-9800
sarah.butler
@strategyand.pwc.com

Zurich

Alex Koster
Partner
+41-43-268-2133
alex.koster
@strategyand.pwc.com

About the authors

Jens Niebuhr is a partner with Strategy& based in Düsseldorf. He focuses on IT strategy and transformation for large enterprises and leads the firm's European IT activities in the communications sector.

Eduard Gracia is a principal with Strategy& in London and a member of the firm's energy, chemicals, and utilities practice. He focuses on business and IT transformation and program management, primarily in the energy industry.

Abhishek Pathak is an associate with Strategy& based in London. A member of the firm's digital business and technology practice, he focuses on IT strategy and ERP-driven business transformation across industries.

This report was originally published by Booz & Company in 2014.

Executive summary

Every large-scale IT program begins with the business and IT teams agreeing to contain complexity by adhering to strict standards in order to minimize the inherent risk. That's when the problems begin. In time, business units demand that the system include specific requirements reflecting how they operate, and users ask for functions tailored to how they like to work, driving a program off-standard. The ensuing complexity slows the project down, increases its cost, and promises even greater maintenance and upgrade challenges in the future.

The only way to avoid this drift is to limit such departures (or "customizations"). We offer a pragmatic framework for analyzing each customization's contribution to the core capabilities that truly differentiate a company's products and services from its market competitors. Customizations that contribute should be allowed, but simplified as much as possible; those that don't should be decided on the basis of a strong business case considering their full life-cycle cost, or not allowed at all.

Putting this into consistent practice isn't easy. IT departments need to have the authority and stature to stringently police the project's scope, work closely with the business to set standards, and sell users on the value of maintaining those standards. Otherwise, complexity drift will be an ever constant problem, and large programs will continue to be prone to failure.

Good intentions

The scenario is all too familiar to IT professionals in large corporations: The company is setting out on a complex, large-scale system transformation project costing tens or even hundreds of millions of dollars. IT and the business have been working together for months to lay out specifications for the project; settle on business goals, scope, costs, and time line; and choose a systems integrator to design and implement the project. Both IT and the business understand that complexity is the major driver of cost and risk threatening the program; hence, both parties are determined to "stick to the standard." So the team works out ways to incorporate into the new system business processes that are intended to reduce the complexity of the overall design.

Over time, however, as the specifications become more tangible and people begin to rethink their initial assumptions and requirements, the appetite for changes emerges. New demands pop up as users grow more concerned about new ways of working and hope to revert to their old ways of doing things, and further complications arise as they think up additional functions the system might support.

It doesn't matter whether such changes are essential or just nice to have; they begin to stack up, quickly increasing the complexity of both the project effort and the new system. As a result, the project's schedule has to be extended, the budget grows significantly, and both IT and the business get increasingly frustrated, given all the unmet expectations. Ultimately, the project fails to deliver against the good intention to keep it simple.

The history of corporate IT is riddled with such projects, and yet the situation never seems to improve. For CIOs leading such efforts, the consequences can be dire: More than a third lose their jobs as a result of failed projects — the second most common cause, according to a recent Strategy& study analyzing the role of the CIO (see "Memo to the CEO: Are You Getting the Best Out of Your CIO?" by Richard Bhanap, Rainer Bernnat, Martin Roets, and Nicolai Bieber, Strategy&, 2013).

The root cause is the ever-increasing complexity that bedevils them as a result of seemingly endless customizations and modifications, and the inability of IT and the business to contain such requests. The solution, however, does not lie in the typical recommendations that CIOs have heard for years: how companies should go about managing these projects, or the various tools designed to aid in their management efforts.

In fact, the solution lies deeper, in a clear understanding of the business strategies and goals the new system is intended to support, and what is required to make a company win in the market. That in turn will give IT leaders the information needed to scope the project properly, and to determine the core and noncore capabilities the new system needs to support.

Making complexity visible

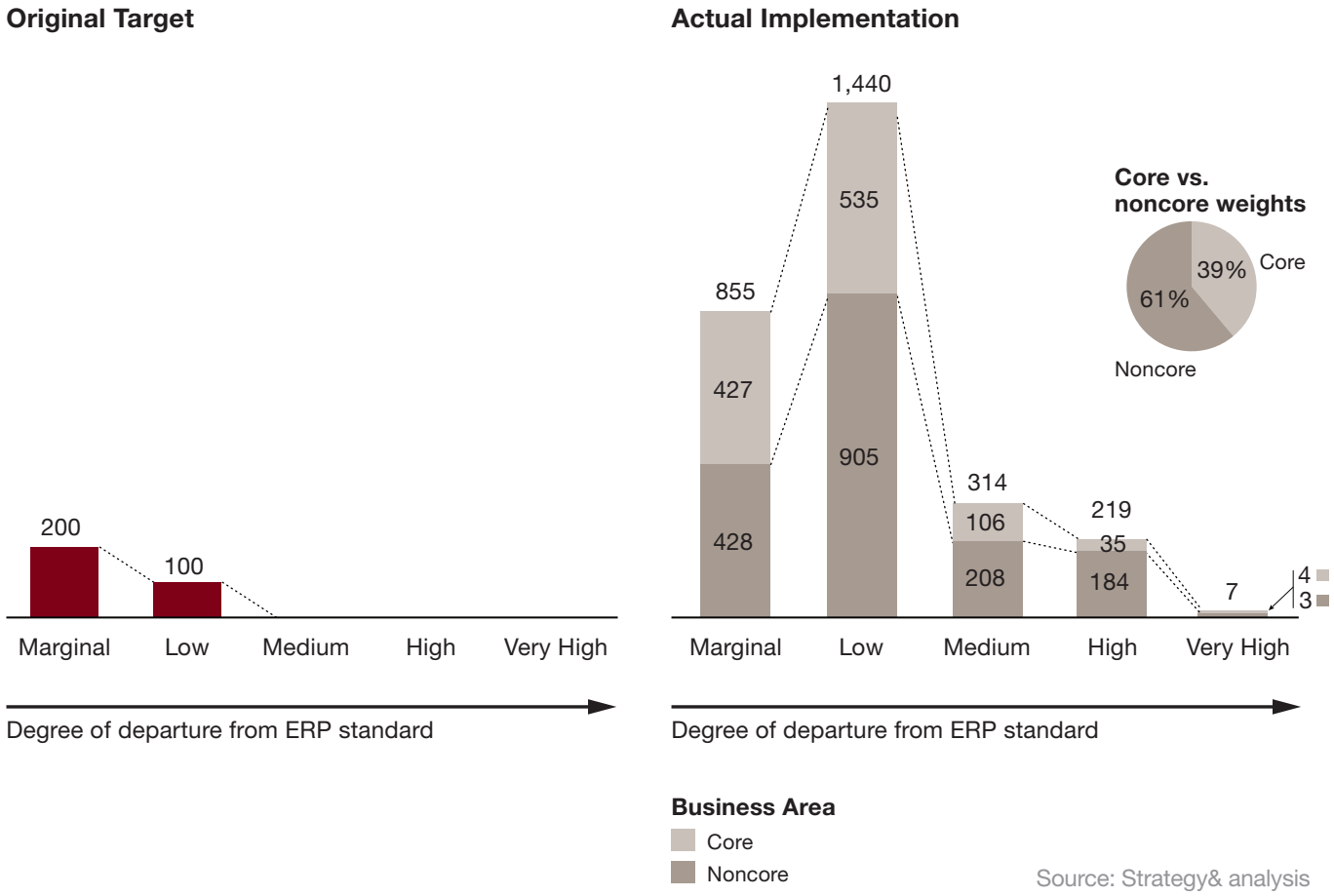
Academics and IT professionals have offered many different, and not always helpful, definitions of IT complexity. To actually manage the problem, a very pragmatic way to understand complexity in any large-scale enterprise-wide transformation program is to count the sheer number of modifications and additions that depart from the new system's standard processes and functionalities. We call such off-standard changes customizations, as distinct from the usual configurations that all systems require and that do not demand a departure from the standard. All customizations should be assessed in the context of the associated business capabilities they are intended to support, to determine whether they are justified, since every customization invariably boosts the design and implementation effort, triggers additional testing complexity, and creates more deployment risk. Moreover, the cost of implementing them is multiplied over the life cycle of the system, as a result of the extra efforts required to maintain the system, release upgrades, and the like.

A case in point: When a large integrated oil company decided to carry out a wall-to-wall ERP system implementation, the IT department was determined to limit customization instances to the bare minimum, which it defined as fewer than 300 customization instances with marginal or low departure from the standard (such as custom input forms or specialized reports and workflows). But demands from business users, compounded by weak governance, raised the number to almost 3,000, including more than 200 high and very high departures from the standard (*see Exhibit 1, next page*). The customizations included things like complex inbound interfaces from external systems and modifications to the underlying code. As a result, the project took more than twice as long as initially planned and cost significantly more.

Worse still, a closer analysis revealed that few distinctions had been made between customization requests related to core business capabilities (those on which the business's competitive advantage rests) and noncore capabilities (largely commoditized functions that bring no particular competitive advantage). In total, nearly two-thirds of the customization instances were devoted to modifying noncore functions.

Exhibit 1

The number of customizations in an ERP implementation grew from a target of 300 to nearly 3,000; the majority were not core to the company's strategy



Source: Strategy& analysis

Causes of complexity drift

The complexity drift that leads to problems with so many enterprise system transformations can be traced to four primary shortcomings, each of which must be understood and dealt with if companies are to generate higher rates of success:

Enduring legacy processes: At companies with long-standing legacy IT systems, end-users have developed deeply embedded ways of working, including the handling exceptional cases. No off-the-shelf system can mimic all the peculiarities of those processes; as a result, re-creating old processes in the new system invariably results in the need for extensive customization and an unnecessarily complex solution.

So although the new system will likely deliver considerably greater value to the business, it forces users to relearn how they do their jobs. If IT can't successfully sell the new system to business users, opposition will mount and grow more passionate, and the initial commitment to the project on the part of senior business managers will fade.

Overstretched program scope: All too often, IT and the business embark on a big transformation program hoping to solve all problems for all people. As a result, locally powerful users demand all kinds of disparate requirements that will fulfill their particular needs. Soon, the temptation to develop custom solutions to keep all these different stakeholders happy becomes too strong to resist, and the program's ensuing complexity overwhelms it.

Weak design governance and lack of IT power: To be sure, every transformation program must ultimately be led by the business. But on its own, the business is rarely aware of the impact of extensive customization and growing complexity on the new system's total cost of ownership. As the business works with IT to determine the program's requirements, and learns more about its proposed future state, those requirements often change and grow. Few companies, however, have sufficiently strong design authorities in place to govern the specification requirements and ensure the maximum conformity to standards from an architectural perspective. And few IT functions have the organizational stature needed to counter the growing number of requirements.

The integrator's dilemma: Companies frequently underestimate the conflict of interest on the part of the systems integrators that are hired to put together and potentially run large new systems. Naturally, integrators try to deliver programs that meet their customers' requirements. But their incentive structure is often determined by whether they deliver all the benefits the business is looking for — not the reduction in complexity that IT wants. As a consequence, budget and time objectives suffer, as the business struggles to manage the integration process.

A capabilities-centric lens

Ultimately, the only way to avoid these pitfalls and successfully manage the customization demands that lead to excess complexity is to develop a systematic way of determining systematically what matters and what doesn't. Many companies enter into large-scale IT implementations without clearly laying out the various capabilities and functions the system needs to support — and then fail to distinguish between the core capabilities that are critical to the company's ability to carry out its business strategy and the capabilities and functions that aren't. Without this understanding, no company can make good decisions about how much customization should be allowed.

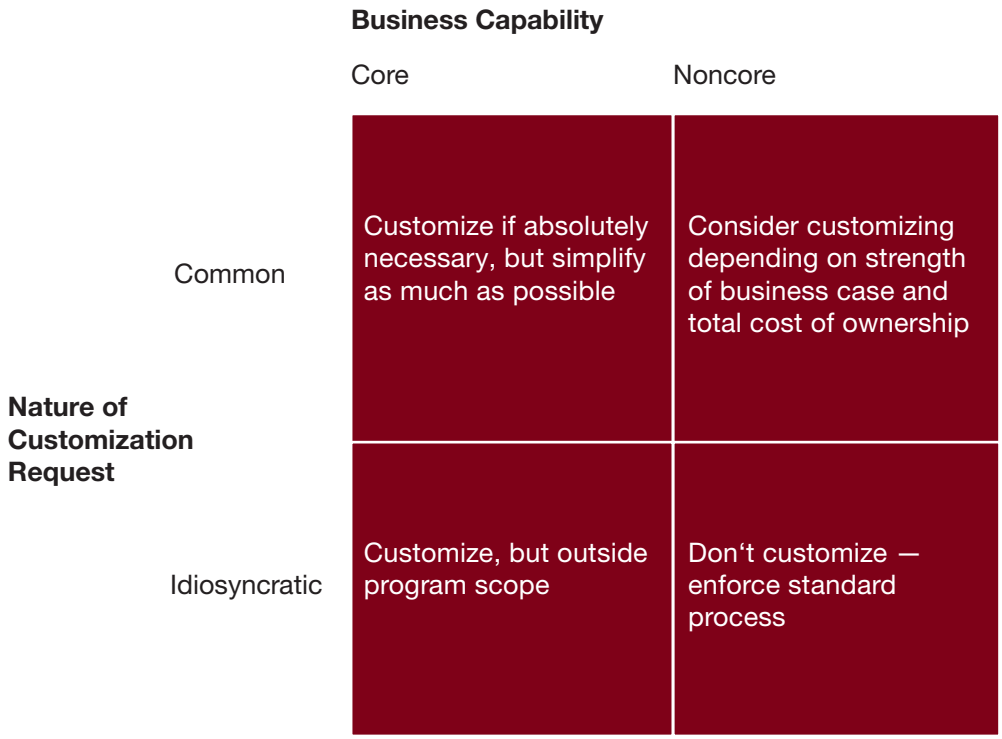
Moreover, certain capabilities and functions may need to be supported by IT but aren't widely used throughout the company, and therefore do not require an enterprise-wide deployment; an example is special requirements for particular geographic markets. So, as a further step, companies must also determine which of these "idiosyncratic" capabilities and functions lie within the scope of the project, and which should be designed and implemented separately (*see Exhibit 2, next page*).

Common core capabilities: Fully supporting this category of processes — such as an e-channel system for a large retailer — is the most critical, given their importance in ensuring the company's competitive advantage. Accordingly, these capabilities merit a higher degree of freedom to customize, in order to make sure the system fully meets the business goals. (However, the key stakeholders should first analyze how to keep any divergence from standard as simple as possible.)

Common noncore capabilities: For processes supporting noncore business capabilities and other functions, such as standard back-office operations like financial accounting and human resources, all customization requests should be accompanied by a strong business case, backed up with realistic numbers. Then, each case should be considered in the light

It is important to institute strong governance procedures to validate each claim.

Exhibit 2
A framework for distinguishing between core capabilities and noncore capabilities



Source: Strategy& analysis

of how it affects the system's total cost of ownership. For such a mechanism to work, however, the business must be disciplined in quickly generating the business cases needed to assess each requests.

Idiosyncratic core capabilities: Every company will need to develop technology to support specialized core capabilities that will be used only by small segments of the business. These could be "must have" capabilities that are required only in a given region, or country-specific but mandatory legal requirements. It is often wise to develop IT support for such capabilities independent of the main system, in order to avoid the considerable risk inherent in trying to build a system that will be "all things to all people." As such, any requests for customization in these instances should also take place outside the program's scope.

These requests can be designated as a low priority, if possible, and dealt with separately from the program scope, with a separate budget and implementation process.

Idiosyncratic noncore capabilities: Nice-to-have processes that are neither business drivers nor globally required — such as a complex report that only one local office wants — should be kept standard, and customization should not be allowed. A large proportion of custom requests are likely to fall into this quadrant, but by strictly adhering to the standard, companies can significantly reduce the system's complexity, and thus its total cost of ownership.

Of course, every business leader requesting a change will claim that the capabilities his or her department supports are core to the business. That's precisely why it is so important to institute strong governance procedures to validate each claim — and effectively challenge it, if necessary — even under the enormous time constraints typical of all enterprise system implementations.

Fighting complexity

This capabilities-driven framework for analyzing customization requests in large-scale IT projects is only a first step in neutralizing complexity drift. To ensure success, CIOs must also follow up in several key areas: Boost the authority of IT, convince business users of the importance of standard processes, maintain the scope of their projects, and manage their systems integrators effectively.

Boost the authority of IT: A strong role for IT in the initial scoping of large projects is critical in providing a system-centric view of all the business processes to be incorporated in the system. This will help ensure that noncore, non-differentiating processes are made to fit into the system standard, rather than enabling the business to modify the standard to fit existing processes. Moreover, IT, in its design capacity, must be consulted whenever the business proposes customizations in either core or noncore processes.

Convince business users: Getting buy-in from business users is imperative if processes are to remain as close to standard as possible. End-users have an understandable predilection for established processes, and it is key for IT to explain the business benefits of standardization. To do so, IT must work closely with the business — the better it understands business needs, the more effectively it can define and deploy workable standards. Working sessions need to be set up with business users to understand the nature of core processes and the reason for requests for off-standard customizations, and to identify opportunities for simplification. Here, prototypes of standard workflows can be an effective tool in guiding the discussion. For noncore capability areas, a much more resolute stand should be adopted to drive home the value of standardization to all end-users.

Maintain the scope: To better maintain scope and manage the requirements process, IT must put into effect measurable targets for both standardization and customization, along with business rules (including funding and costs

charged back to the business) that reflect those targets and discourage requests for out-of-scope processes and customizations. Business unit heads who demand the most customizations should be identified and managed accordingly, and this information should be incorporated into performance reviews.

Manage systems integrators: Many companies find it difficult to effectively control systems integrators in large projects, since they typically have much more subject-matter expertise and challenging them can be hard. It is critical to make sure the integrator understands the capabilities-driven principles being used to design and manage the project, and the need for it to abide by those principles when contemplating any proposed custom development, especially in a noncore capability area. Doing so will arm the company with a powerful lever for containing program complexity and hence present and future costs.

Conclusion

Complexity drift is an old devil that has defeated many determined efforts to tame it in the past. But we know from experience that it can be and has been managed successfully in engagements that use the capabilities framework, scope the project carefully, get end-users on board, and manage systems integrators accordingly. Ultimately, the key is to understand unambiguously where the company's real differentiating capabilities are (those that truly set it apart in the marketplace), build them into the system's processes with as little customization as possible, and treat the remaining, non-differentiating capabilities as commodity processes that must be adapted fully to the project's standards.

Strategy& is a global team of practical strategists committed to helping you seize essential advantage.

We do that by working alongside you to solve your toughest problems and helping you capture your greatest opportunities.

These are complex and high-stakes undertakings —often game-changing transformations. We bring 100 years of strategy consulting experience and the unrivaled industry and functional capabilities of the PwC network to the task. Whether you're charting your corporate strategy, transforming a function or business unit, or building critical capabilities, we'll help you create the value you're looking for with speed, confidence, and impact.

We are a member of the PwC network of firms in 157 countries with more than 184,000 people committed to delivering quality in assurance, tax, and advisory services. Tell us what matters to you and find out more by visiting us at strategyand.pwc.com.

This report was originally published by Booz & Company in 2014.

www.strategyand.pwc.com