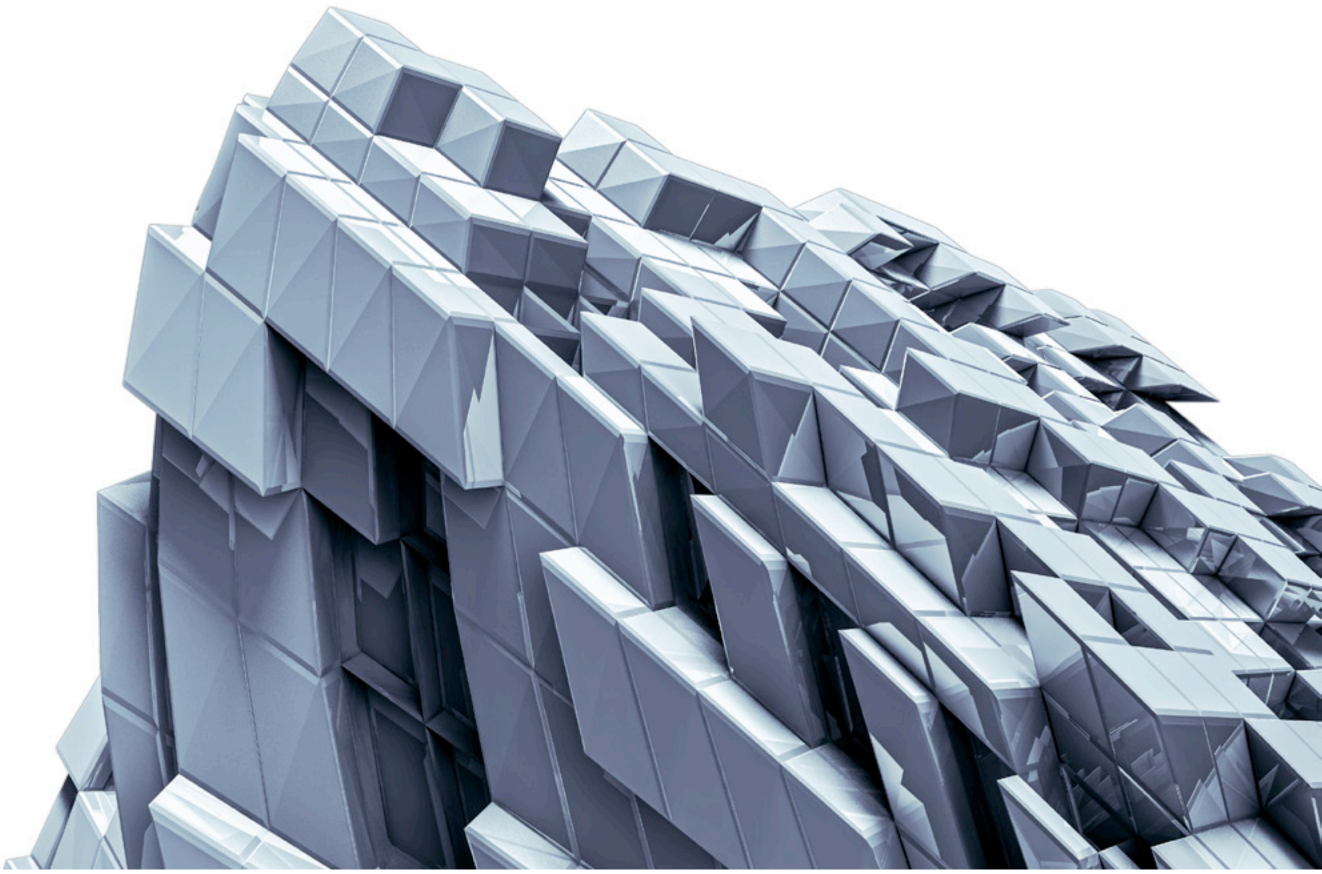
McKinsey Digital

Managing large technology programs in the digital era

To successfully implement large technology systems, first accept the complexity, and then take these six actions.

by Katya Defossez, Mark McMillan, and Hrishika Vuppala



November 2020

Every IT executive wrestles with implementing large technology programs.[1] In fact, two out of three large programs regularly exceed initial budgets, miss schedule estimates, and underdeliver against business objectives and benefits, often by significant margins.[2]

Our past research has found that 25 to 40 percent of programs exceed their budget or schedules by more than 50 percent. This failure rate is especially debilitating for the business because large programs are typically of critical importance—for example, for consolidating multiple financial systems to enable better operational insights or for implementing health-insurance enrollment systems.

That failure rate does not have to be the norm. A number of new digital practices and technologies that have emerged in the past few years can drastically improve large program implementations. When combined with disciplined managerial and talent practices to effectively deal with the vast complexity of large technology programs, success rates can be as high as 90 percent or more.

What drives failure rates?

So what drives success (or failure)? If you ask 100 technology leaders, you are likely to get almost as many different answers, from unclear objectives to ineffective change management, poor team capabilities, or vendor deficiencies. Others would say excessive customization, the wrong platform decision, or ineffective decision making. In many ways, they are all right, and this reflects an overriding reality of large technology-implementation programs: they are extremely complex. While that shouldn't come as a big surprise, technology leaders continually underestimate the extent and depth of that complexity. For this reason, there is a natural tendency among IT leaders to think (wishfully, perhaps) that by employing a handful of simple fixes or by finding the right systems integrator, the vast majority of their large-program problems can be solved.

Unfortunately, success can occur only when tech leaders sufficiently acknowledge the complexity. In practice, that means driving superior execution across ten domains (Exhibit 1). Each of these domains is a significant topic unto itself, requiring cross-functional skills and capabilities for effective execution.

But the main consideration is how to balance the tremendous complexity of the program against the practical need to make progress. In our experience, hitting that balance successfully requires organizations to prioritize five to ten success factors for each of the ten domains and to develop large-program management capabilities accordingly. Traditional project management is simply not up to the complexities of managing a large number of interdependent workstreams, the need for technical mastery across many domains, and the importance of adjusting many dependent variables during the inevitable setbacks and challenges of a program at this scale.

For example, traditional project-manager training teaches managers to develop a sequential and precise timeline of actions from project start to finish and then to manage rigorously against those actions. For large, complex programs, this quickly becomes an impossible task, and the amount of work and rework needed to create this level of detail is not worth the effort. While big programs still need integrated schedules, they should not pretend to have it all worked out up front. Because of the interplay and dependence across domains (the architecture, for example, depends on the sourcing strategy, and the sourcing strategy depends on the architecture), the complexity can be effectively managed only by working through the domains iteratively and in parallel, not by laying out every possible step in advance.

[1] While there is no standard definition of a large technology program, more than $25 million in one-time investment can serve as a useful threshold.

[2] Michael Bloch, Sven Blumberg, and Jürgen Laartz, "Delivering large-scale IT projects on time, on budget, and on value," October 2012, McKinsey.com.

Exhibit 1

Effective implementation of large programs requires superior capabilities across ten domains.

Strategy

1. Vision and strategy
Align stakeholders on prioritized objectives and a clear future-state visualization

2. Value drivers and cost
Quantify value and cost of use cases and capabilities directly linked to the business case

3. Product road map
Sequence and execute the product and architecture road maps

Technology

4. Customer experience and design inputs
Define user journeys and requirements based on human-centered design

5. Platform and data architecture
Design and implement a modern architecture, starting with a minimum viable product (MVP)

6. Vendor strategy and management
Design and execute sourcing strategy and vendor relationship management

Governance

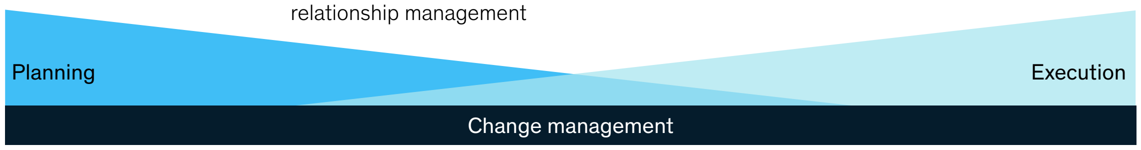
7. Program management and governance
Establish agile control tower to oversee and enable efficient delivery

8. Talent planning and capability building
Assemble or build needed skills to execute the program and maximize value realization

Adoption

9. Operations and management
Transition to future-state operating model, supported by changes driven by the transformation (eg, new business processes)

10. Culture and change management
Incorporate change management across the organization, and build a culture of continuous learning



Six actions that make a big difference

In our experience working on more than 500 large technology-implementation programs, the chances for successfully executing against these ten domains significantly increase when tech leaders take six specific actions. Four of them take advantage of new digital capabilities, while two others are proven, long-standing approaches but are often neglected.

Digital approaches

1. *Use select agile methods.* Even enterprises committed to agile development often are resistant to using agile for large programs. There have been, however, significant successes in programs that have embraced select agile methods: clear product ownership, prioritized product backlog and road map, small cross-functional teams, iterative releases with time-boxed sprints, modular architecture, objectives and key results (OKRs)[3] to manage value capture, and a commitment to a minimal viable product (MVP) and iterative releases. Agile mindsets can also be powerful in supporting a willingness to respond to change, test and learn, and collaborate.

One large organization, for example, started a claims-system modernization program by using the standard waterfall method. After spending the entire $200 million budget, the program was only a quarter of the way through development—and

[3] An approach to defining and tracking desired outcomes (versus activities).

what had been developed was a frustrating user experience and full of defects. The organization made the difficult decision to start over using an agile approach focused on small cross-functional teams working in sprints through active test-and-learn cycles with a stable of smaller vendor partners. The results represented a stunning turnaround for the program, improving delivery velocity and productivity more than threefold, with a massively improved user experience and a first release in months rather than years.

2. *Ground the work in design thinking.* Many large programs may meet requirements but not user needs. Successful large programs use design thinking—a method of problem solving anchored in end users' needs—to address this issue. The practice helps deliver products and services that users want and need and are therefore more likely to use. Another benefit is savings, since teams develop only those features that are needed.

For large programs, design thinking starts with uncovering user needs at the outset, typically through a blend of survey-based quantitative and field-based qualitative research. These efforts derive a clear picture of how people use the service or product, signature moments, and unmet needs. Regular and immersive user engagement throughout the program delivery—for example, in prototyping and user testing—then ensures the program maintains alignment with user needs over time.

One leading automotive company decided to modernize its product life-cycle management (PLM) systems. Instead of the traditional process of collecting requirements from R&D, production, sales, and after-sales, it applied design-thinking principles in cross-functional workshops and interviews to collect current pain points and requirements. Using the "digital twin" approach, it essentially created a digital simulation of a PLM system for the modernization team to work with. This enabled the team to identify clear issues, such as complex collaboration processes. Based on this effort, the company created a "data exchange" for suppliers, providers, and developers (among others) to drive better collaboration around product design specs and order management. This approach significantly improved collaboration among teams and accelerated the release of features.

3. *Use cloud-based services.* Most enterprise leaders still tend to reduce the benefits of cloud to efficiencies around infrastructure management. The capabilities, components, and services that many cloud service providers (CSPs) offer, however, allow companies managing large-program implementations to innovate much more quickly and get to market faster through rapid environment provisioning and simpler ways of prototyping or exploring novel solutions.

By migrating to the cloud while also rapidly scaling cloud-native features for analytics, database management, and content management, for instance, a state-government agency was able to consolidate and modernize three disparate legacy systems across millions of residents and become 30 percent more efficient in terms of operating costs. Running reports, which had been resource intensive and slow, happened much more quickly on the cloud. The agency also took advantage of the CSP's call-center-management application, which greatly

So what drives success (or failure)? If you ask 100 technology leaders, you are likely to get almost as many different answers.

simplified a system that had previously relied on multiple providers.

For large programs, leaders need to systematically evaluate how best to take advantage of the cloud. Selecting a software-as-a-service (SaaS) solution, for example, can avoid the effort of a custom build and can result in a best-in-class solution that is easier to maintain. Or leveraging a platform-as-a-service (PaaS) solution can enable greater developer productivity and access to an ecosystem of thousands of innovative services.

4. *Use modular architecture to increase flexibility and vendor competition.* Many organizations are moving to more modular, flexible architectures, such as microservices. This move not only creates longer-lasting, more "future-proof" applications but also allows companies to use a multi-vendor sourcing strategy and thus solve one of the longest-standing challenges with large-program delivery: keeping vendor incentives aligned with your own. With single-vendor solutions, it's nearly impossible to apply steady cost pressure, as often a significant risk premium is worked into a fixed-price contract and change orders are common. Alternatively, time-and-materials contracts create incentives for vendors to extend and expand programs and thus grow their revenue stream.

Instead, modular architectures allow companies to work with multiple vendors who can be replaced as needed, leading to significantly better outcomes. For example, one public-sector organization awarded a development master-services contract to four development vendors. For each phase of the program, the vendors either competed or were directly awarded small packets of work, such as front-end design services or development and testing services for each component. Over time, the strongest-performing vendors—those bringing their A team at reasonable cost—won more of the work, leading to superior outcomes.

Proven approaches

5. *Get people with large-program (ideally comparable) experience.* Given the complexity of large program implementations, it is crucial to have people who have already done them, or something comparable. There is just no substitute for that kind of experience and "pattern recognition." Without it, failure is far more likely. As might be expected, these people are hard to find, especially since these sorts of large programs happen infrequently for most organizations.

IT leaders naturally try to address this issue by bringing together a team of the best people they can find. But ensuring this team addresses their talent gaps requires an honest assessment of the team's existing talent and a willingness to bring in the right people, either by hiring them or contracting with vendor partners. This can be time consuming, but it is necessary. Hiring a systems integrator to fill the holes often won't work, since it has different incentives—scope creep or delays increase its revenue—and is focused on delivering against the contract rather than ensuring you are doing everything needed to manage the program effectively.

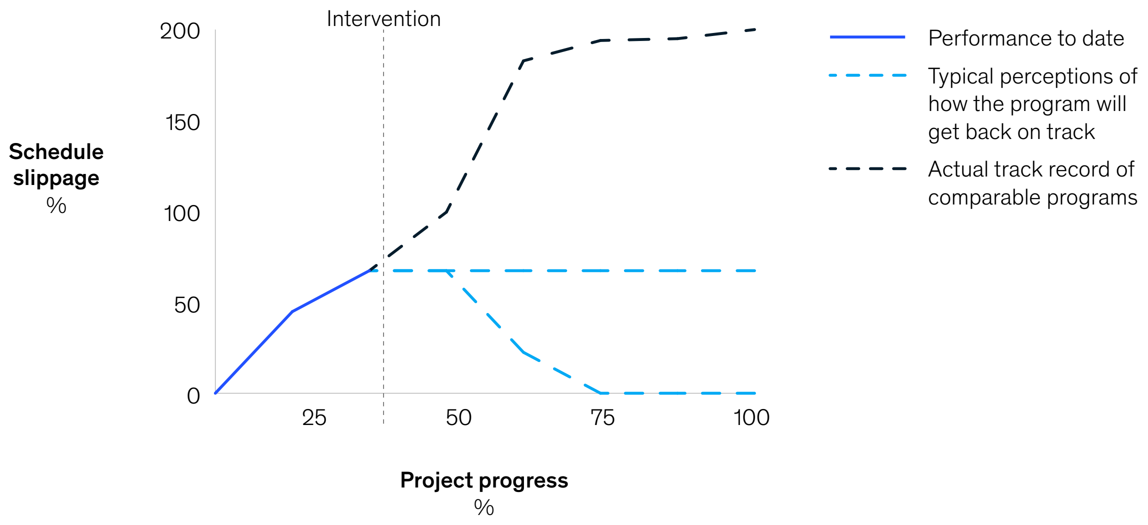
6. *Be aggressive about necessary course corrections.* Any program of this scale is going to run up against issues. When that occurs, CIOs and the leadership team analyze the problem and recommend a correction. But these interventions are often not aggressive enough to get to the root cause of the problem. That's understandable, since these programs are so complex. Their multiple interdependent systems can make it difficult both to pinpoint the source of the issues and to muster the often significant effort needed to course correct.

However, an unwillingness to admit—or the inability to realize—that the issues are more complex and require more work than anticipated means that problems continue and often get worse. The research is quite clear on this point. Early cost and schedule overruns end up, on average, much worse in most programs, often costing twice as much as anticipated—and that's despite the interventions of program leaders (see Exhibit 2 for an example analysis).

Fortunately, there are many examples of successful interventions. One public-sector organization, having invested $60 million of its $200 million budget for a tax-processing

Exhibit 2

Despite interventions, issues typically get worse over a project's life cycle.



modernization program that was way behind schedule, decided to forfeit the initial investment and start over by making some aggressive changes. It first hired a new systems integrator and software vendor. It then developed a new business case as a "north star" to guide the relaunched program. The results: a successful project for less than $125 million—less than its original budget, even accounting for the initial sunk investment.

These and other examples show that organizations can be successful with their most important technology investments—if they master a broad array of success factors, take advantage of new digital techniques to de-risk delivery, and ensure they have the right capabilities from the start.

Katya Defossez is a partner in McKinsey's Houston office, **Mark McMillan** is a partner in the Washington, DC, office, and **Hrishika Vuppala** is a partner in the San Francisco office.