Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Technology Services | the way we see it

IT Architecture in Large Scale Programs

Critical Success Factors

Contents

Management Summary

Recently, we have seen an increasing number of large-scale transformation programs, and if the current merger and acquisition activity gives any indication, this is likely to continue in the near future. This may come as a surprise, considering the fact that the industry track record in delivering large-scale information technology projects has not been impressive, with many programs spending far too much time in development, being too expensive or not delivering the expected benefits. Considering the growing need of large global organizations to achieve economies of scale and standardization of business processes, large IT organizations and service providers need to develop an industrialized capability to deliver this type of program.

In general, there are two underlying reasons for poor results of programs with a significant Information Technology (IT) element: Firstly, we have not been very good at connecting business with IT. Secondly, we seem to have a difficulty maintaining direction of large-scale programs in the face of organizational volatility, technical and social complexity, and delivery issues related to scale.

The approach our industry has taken to address these issues thus far has revolved around two disciplines: program management and IT architecture, which made the architecture community in particular rather excited about the value it can add to business. Yet one of the learnings from this experience is that merely assigning architects to a program or developing architecture on its own does not guarantee successful delivery. The main reason of this is complexity, uncertainty and volatility of large transformation programs that often prevents the use of traditional methods and approaches effectively. The approach we normally use, based on phased reduction of complexity through analytical methods, has a limited impact when the scale of complexity defies effective analytical treatment of the problem, and where volatility of the external environment soon invalidates the results of such analysis.

To improve the success of large-scale programs, we therefore need to develop specific approaches to help us pragmatically operate under the triple challenge of complexity, uncertainty and volatility. Following a review of specific architecture-related practices that worked under such conditions, this paper presents five critical success factors:

Architecture leadership is perhaps the most important factor that pre-determines the extent to which architects can play a role in ensuring the success of large programs. The architecture capability needs to be headed by a strong lead architect backed by a competent and versatile core architecture team—the leader must have the authority and possess necessary functional and technical competency, while team members must have highly developed communication and consulting skills.

Apart from individual skills, **collaboration** is the single most important factor that can help manage volatility, improve quality and accelerate the delivery process.

There are several pre-requisites to effective collaboration, with the most relevant to large programs being right institutional incentives. Building on these, programs need to develop and cultivate collaborative culture, leveraging skills and shared experiences of program teams.

The next key factor is **integration of the architecture with the program delivery approach**. Large programs need to adopt specific roadmaps in integrating architecture front-end with the remainder of the program delivery process to ensure smooth flow of work between concurrently run requirements, architecture and delivery activities. Secondly, such engagements need to have an active role in delivery to drive program results.

The last factor is **architecture governance—**a system of controls enforcing checks and balances over delivery. Efficient governance in large programs needs to build on individual accountability and be based on clearly defined roles and responsibilities aligned with program structures.

Large Program Challenges

Despite frequent publicly voiced reluctance to undertake large-scale engagements, we have seen a number of large transformation programs with a significant information technology element. And if the current mergers & acquisition activity gives any indication, this is likely to continue for some time to come. Such being the case, it makes sense to prepare and tool ourselves with the understanding of what is different about large-scale programs and how to make them successful.

To do this, we first need to address an interesting paradox. On one hand there is a need to achieve economies of scale through industrialization, while on the other, the complexity of large-scale engagements resulting from such an effort prevents us from effectively using many traditional architecture tools and techniques.

Challenges of Uncertainty and Complexity

You may have seen the statistics such as (1) that show that up to twenty percent of projects are cancelled and some fifty percent of projects run over budget or exceed their timescales. Even the conservative estimates of overrun size show that (2) average cost overruns are around thirty to forty percent.

Because these statistics are often derived for average sized projects, the picture for large multi-year programs spanning multiple organizations with team sizes running into hundreds of staff is probably even more discouraging. Spectacular failures of Sainsbury's supply chain system project, the FBI's Virtual Case File project or newpaper headlines on troubles of the UK Connected for Health program are few case points demonstrating the magnitude of the challenge faced by large programs. As personal experience of individuals involved in such engagements confirms, delivering successful programs with over 300 staff or at a cost over £100 million is at a completely different scale of difficulty in comparison to a more traditional projects.

A quick look at the headlines shows that we have not learnt much from our past and that we fruitlessly repeat our old mistakes. However, what statistics do not reveal is the increasing complexity of the task such programs set out to achieve. Consider an example of a typical business transformation program, such as a merger of two large organizations, during which essentially the whole way they run their businesses, including their processes, culture, staff and IT landscape are changed; whilst the merging organizations still need to carry on with their business-as-usual activities. Other transformation programs may have different objectives, such as standardization of operating models across the company's international operations or introduction of global sourcing models, but they all have traits that were much rarer a decade ago, including:

- The need for significant, wide-ranging and dramatic change in many aspects of organization operation

1 D. Hartman, "Interview: Jim Johnson of the Standish Group", *InfoQ*, August 2006, http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS
2 M. Jørgensen and K. Moløkken, "How Large Are Software Cost Overruns?", *Information Systems and Software Technology*, 48(4), pp. 297-301, (Simula Research Laboratory, April 2006), http://www.simula.no/research/engineering/publications/Jorgensen.2006.4

- Frequent requirement for multi-organization funding and cross-organizational delivery
- Technical solutions that need to support increasingly broader variety of business requirements
- Need to cope with a more volatile external business environment, that requires faster response
- Increasing demands on integration of a growing mix of legacy and new systems.

Putting aside growing complexity and volatility, which we cannot control, we discover that we do have tools to influence other causes of poor project performance. Amongst the key factors that are critical for program success which we often can influence are board-level support, visionary program direction, strong program management team, good working relationships with customer representatives and an effective program architecture capability. Even though practitioners of these disciplines often try to sell their specialty as the "silver bullet", in reality these represent complementary tools that allow large programs to deal with uncertainty, complexity and volatility of the environment they operate in. Considering the main theme underlying this paper is how to pragmatically operate under such circumstances, it may be useful to compare how the most popular methods that traditionally operate at a program or enterprise-level address this problem.

Figure 1: Practices Reducing Unpredictability & Complexity

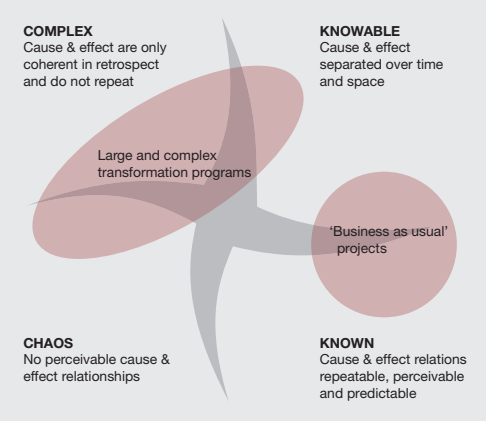
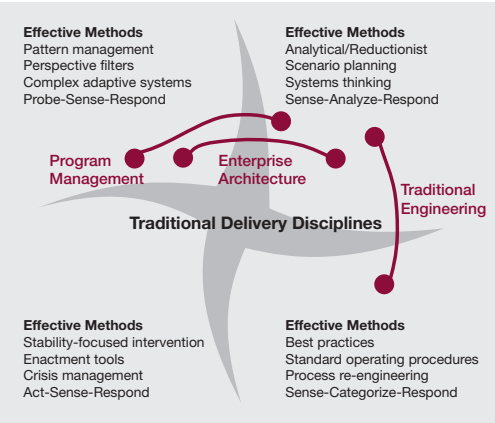


Figure 2: Appropriateness of sense-making methods



Based on Kurtz and Snowden, 2003

Positioning the key tools, including program management, enterprise architecture, traditional systems and software engineering in the sense-making framework presented in (3), it becomes clear that these methods often work on the principle of reducing the complexity of the business process and its environment; while program management and enterprise architecture do that for the "complex domain", engineering and project management apply this technique on problems at a smaller scale and order of complexity.

Architects Response to Complexity

Architects in particular became rather enthusiastic about adding value to the programs through the application of traditional methods to larger, more complex and "softer" challenges through the enterprise architecture discipline. It is ironic that in many cases this effort left a lot to be desired.

This, in our opinion, essentially stemmed not that much from deficiencies in methods, as much as the lack of practical approaches that architects adopt in dealing with "messiness" of complex transformation environments. The way we organize programs and the semi-linear approach to architecture development we normally take does not well address some of the challenges of large and complex engagements, such as:

- Disparate business customers with contradicting business needs and goals, often lacking a single point of management, which results in the need for consensus-based decision making
- Requirements expressed primarily as vision statements, leading to the evolution of the requirements in response to changes in the environment, user expectations or technology
- Competition for resources and alternative solutions resulting from the system crossing organizational boundaries
- Commercial and contractual commitment to deliver against pre-defined milestones despite the high degree of uncertainty and incomplete capability to control all aspects of the work.

Practitioners sometimes suggest that perhaps more experienced senior management or more time for analysis and planning could help resolve these issues. What is true is that our industry has a vast array of good practices that deserve to be studied and applied on engagements regardless of their size; however, we need to acknowledge the degree to which these can help us "resolve" complexity and volatility of the environment. This is particularly highlighted when we juxtapose different methods to deal with complexity across various types of engagements.

In summary, we can neither completely abandon our existing methods, nor hope that their more rigorous application will give us the "silver bullet" to resolve systemic issues. The challenge architects are thus posed with is to maintain their traditional analytical and problem solving role, but also to develop new individual and institutional capabilities to help cope with complexity and volatility of large transformation programs, including:

- Architecture leadership and personal skills
- Effective collaboration and communication
- Integration of architecture with program delivery and
- Architecture governance.

This paper explains the rationale behind these factors, showing underlying practices that have proven to work well on large, complex, transformation programs and which have provided significant contribution to the success of these programs.

3 C. Kurtz & D. Snowden, "The New Dynamics of Strategy: Sense Making in a Complex and Complicated World", *IBM Systems Journal*, Volume 42, Number 3, pp. 462-483; http://www.research.ibm.com/journal/sj/423/kurtz.pdf

Architecture Leadership

"Never doubt that a small group of thoughtful, committed people can change the world. Indeed, it is the only thing that ever has."

- Margaret Mead, anthropologist

What Architects Lead On

Let us firstly focus on the "architecture" part, looking at the way in which architecture leadership is different from other instances of team or project effort.

Architects are the key program enablers that shape and safeguard the internal consistency and alignment of the solution with customer requirements, which are aspects established by product development research (4)—the key for success of the solution.

The most important task architects are responsible for is the alignment of the solution with goals, strategies and needs of customers and other stakeholders. This responsibility requires architects to deal with a number of stakeholder groups:

- First and foremost, a solution needs to support the goals and needs of business users and new or changed business capabilities and processes. The architect's job is not only to elicit what the key goals are but also assist with their translation into solution requirements. This role often starts with facilitating agreement of various parts of the business user community on common goals, objectives and vision;
- Secondly, the solution needs to be in line with broader enterprise strategies, and work done within other programs. To manage this type of alignment, architects need to become dependency managers and advisors to the decision making on allocation of funds;
- Thirdly, the solution must be internally consistent and deliverable within the budget and time constraints. This is traditionally the most important part of architects' job, which requires them to become design managers;
- Lastly, all programs face the issue of ensuring the solution is fit for operations. Architects address this by explicit identification of operational performance objectives and ensuring technical and non-technical arrangements are set up to meet the same.

Balancing these tasks is not easy and you can almost compare architects to someone permanently juggling the balls of requirements, internal consistency, deliverability, value, timescales and budget.

Where Architects Add Value

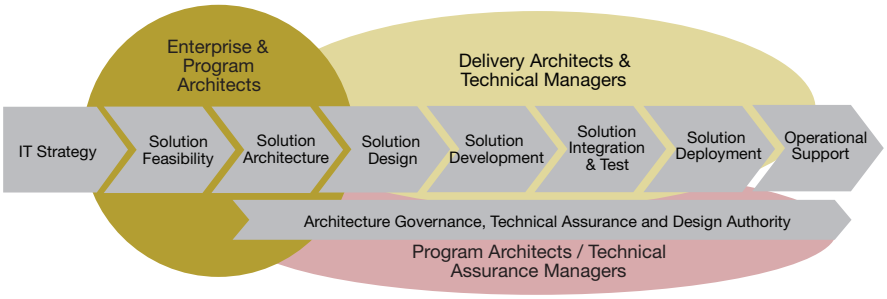
Frequently, architects are involved during the front-end of the solution delivery process. Less visible, but ever so more prevalent is their involvement in delivery, that they can become the driving force to success. This echoes observations of Kim Clark's research into successful product development practices (4): *"... companies whose products consistently succeed accomplish two things without fail. They focus the whole development organization on customer satisfaction. And they devise processes (both formal and informal) for creating powerful product concept and infusing them into the details of production and design"*. Once an experienced program manager was asked about his views on the role of architects in the program, to

4 K. B. Clark and T. Fujimoto, "The Power of Product Integrity", *Harvard Business Review*, pp. 107-118, (Nov-Dec 1990)

which he humorously responded that to ensure success of the project, he normally likes to chain architects to their desk until it is delivered!

Humorous notes aside, this leads to a question: What do the architects do in the solution lifecycle? The answer to this question to an extent depends on their role and specialization:

Figure 3: Architects' responsibilities throughout the lifecycle



In large organizations, architects are likely to have a specialized role; with the most frequent ones being enterprise, program or domain and delivery architects.

Where enterprise architects exist, they typically

- Comprehend the full scope of the solution including people, process and technical aspects
- Identify/target a high level solution that provides the intended business benefit
- Build a shared vision of the future and the transformation roadmap.

Program architects:

- Scope out technical solution, including its application, data, infrastructural, security and operational components
- Identify scope of individual projects and initiate development of the project architecture
- Define program-wide solution delivery roadmap and
- Manage architecture governance, program design authority and technical assurance processes.

Delivery architects typically:

- Identify design principles
- Lead the identification of key component-specific requirements
- Identify architecture for a component or groups of components delivered within a scope of a single project and
- Oversee delivery of the solution together with technical managers.

From this it can be seen that architects' role is multi-faceted and in many respects can go well beyond the development of architecture artifacts and documentation.

Strong Architecture Lead

One of the key enemies of conceptual integrity is fragmentation of architecture vision. As the existence of the term "architecture by committee" and its derogatory undertones suggests, our industry has a wealth of experience with such approaches.

Because architecture leadership is the key to counter this threat and its potential impacts on program success, most experienced practitioners agree on the need for a strong solution lead, who is in IT organizations called "Lead Architect," who will

- Own the solution vision responding to the needs of its customers
- Act as the ultimate solution decision maker
- Serve as the guardian of the solution vision from conception until delivery—infusing the solution concept into the solution design and enforcing compliance of engineering efforts with the vision.

In order to be effective in this role, the lead architect needs to have a significant level of authority allowing him or her to

- Make significant solution-related decisions
- Influence allocation of funds and resources
- Shape program strategy and plans, program organization and timescales.

This level of authority not only requires sponsorship at the level of the program board, but also a very strong informal authority and relationships with key stakeholders. To once again quote Clark's views on the role of a product manager that is broadly equivalent to the IT lead architect (4), "As concept guardians, heavyweight product managers, draw on both personal credibility and expertise and organizational clout that comes with the job. Themselves engineers by training, heavyweight product managers have a broad knowledge of the product and process engineering required to develop the end product. Years of experience with their companies give their words weight and increase their influence with people over whom they have no formal authority."

Core Team

Every large program will have its core group of architects who will move the program forward through its various stages and who could be considered as formal or informal leaders in their area. This core group is likely to span across the formal team boundaries, typically comprising the key members of both central and delivery teams, and in some cases crossing the customer-supplier boundary.

Considering leaders always have disproportionate impact on the culture and results of their teams, it is essential to be careful while selecting the key staff during the onset of the program. Selection need not take into account only individual behaviors and competencies, but also team dynamics in order to achieve the right blended team. Ed Yourdon, one of the influential writers on the subject of software developments, recommends two approaches to building an architecture team for challenging projects (5): *"Insist on a well-honed, mission-impossible team that has worked together before; or choose mere mortals, but make sure they know, what they're getting in for"*.

A pool of resources available in most professional solution delivery organizations should provide a reasonable opportunity to follow one of these approaches. Still, it is worth maintaining a healthy caution as to the skill set of staff offered to avoid the approach described as, *"Take whoever you're given"*, which according to

4 K. B. Clark and T. Fujimoto, "The Power of Product Integrity", *Harvard Business Review*, pp. 107-118, (Nov-Dec 1990)
5 E. Yourdon, "People in Death March Projects", in *Death March: The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects*, pp. 99-128, (Prentice-Hall, 1999)

Yourdon: *"... is to be avoided at all cost. If the project turns out to be dumping ground for personnel that no other project wants, then it is almost certainly a suicide project".*

In addition to considering individuals on their own, it is essential to chose individuals in such a way that the core team has the right blend of skills, knowledge and behaviors:

- Deep understanding of complementary aspects of the business and solution domain
- Complementary personality profiles, e.g. politician, system thinker, 'just let me get it done', facilitator, co-ordinator or contrarian
- Relationships with all key stakeholder groups, including customer, program management, delivery and operations teams and other programs.

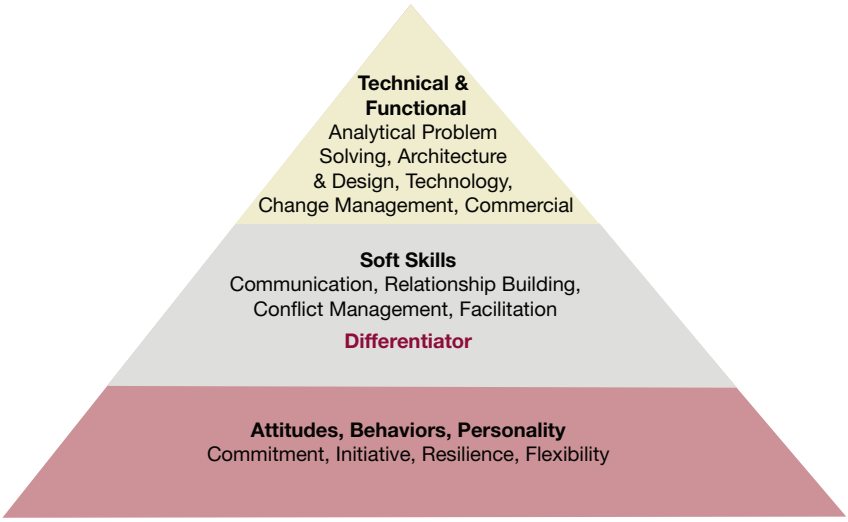
The end result of getting the right blend of people is that the whole team becomes much bigger than the sum of its parts, leading to an exponential increase in the effectiveness of both the team and the program.

Personal Characteristics

Discussion of architects' qualifications often gets simplified to a question of professional skills, which in most cases become equivalent to "technical skills" or "subject matter knowledge". It is true that it is impossible to deliver IT projects without adequate subject matter expertise. Yet, when we review approaches appropriate for dealing with volatility and complexity of large programs presented in the first section, it becomes apparent that we need to re-evaluate some of our assumptions on the required competency profile of key architects.

What are the criteria that qualify individual architects as suitable for a large-scale engagement? Using a competency framework (6) and reflecting on architects interviewed personally by our research team, we narrow down three such broad qualifications: technical and functional competency, soft skills and attitude, behaviors and personality.

Figure 4: Architects Competencies Relevant to Large & Complex Programs



6 R. J. Wieringa, ed., *Competencies of the ICT Architect*, (in Dutch), Nederlands Architectuur Forum, November 2006; http://www.naf.nl/content/bestanden/competenties_van_de_ict-architect.pdf

Firstly, it is not wise to hire an architect without core technical and functional competencies. Problem domain expertise, competence at problem solving, decision making, understanding of technology, architecture and design methods, and awareness of change management and commercial aspects of solution delivery are the foundations for a successful architect. It is rare that a single individual will have all of these and so it is a good idea to approach selection of staff from a viewpoint of the team having a combination of such skills . It is worth noting that complementary skills do not imply specialization of individual architects. Although specialist architects will be required for many specialist tasks (i.e. security or data), an ideal profile of the core architecture team member is someone who is "versatilist", meaning someone who can *"apply depth of skill to a progressively widening scope of situations and experiences, gaining new competencies, building relationships, and assuming new roles"* (7). Hiring a specialized architect for every new task is not a viable option for large programs because of the several months of lead time it takes for a new core team member to acquire contextual knowledge and become truly effective in a large and complex environment.

Attitudes, behaviors and personality pre-determine whether a person with the right skills can be effective in a program, and they also can, to an extent, compensate for shortcomings in skills. Architects that thrive in a complex, volatile and "messy" environment of large transformation programs have a specific personality profile. Because cognitive methods appropriate for such environments are more experimental in nature, successful architects will be more action-oriented, willing to take the initiative, yet still retaining their analytical and idea generation skills. To remain effective throughout the whole lifecycle requires a degree of mental and behavioral flexibility to cope with the changing nature of the work. An architect on a multi-year engagement may start as a big-picture strategist and facilitative business analyst, then during later stages become an analytical solution architect and further on directive and detail-oriented problem solver, technical or assurance manager during development and deployment stages. All these need to be underpinned by a high degree of resilience, which core team members need to posses in order to cope with significant amounts of pressure large-scale programs put on key individuals.

Soft skills is the key factor that distinguishes architects who are merely good from those who are truly excellent. One of the key value adds of architects is their ability to help balance individual requirements and development of the shared vision for disparate customer communities. The larger the program and the greater and more varied the stakeholder community, the more important this brokering of agreement becomes. To do it successfully, architects need first class communication and influencing skills, including the capability to manage relationships with senior customers, decent facilitation skills and a prowess at conflict resolution. Because architects need to deal with audiences that are culturally very different, they need to be excellent communicators, capable of explaining complex problems in a simple manner whilst maintaining the information value and avoiding over simplification. Consulting and communication skills are the biggest differentiators for individuals looking to work on large-scale programs. Incidentally, they seem to be some of the hardest to learn and rare to find.

7 D. Morello, "Unlocking the Business Value of People: Building Versatility", Gartner Research Paper, May 2003

Summary

The most important task architects are responsible for is the alignment of the solution with goals, strategies and needs of customers and other stakeholders. In order to achieve it, programs need a strong lead architect to own the vision, act as the ultimate solution decision maker and serve as the guardian of the vision throughout the delivery. On large-scale engagements, the lead architect will be supported by the core team of architects. As the quality of staff of the core team has a disproportionate influence over success or failure of the whole program, personal selection of the team members needs to apply strict criteria. Firstly, the core team needs to have the right mix of individuals with complementary skills, knowledge and working styles. Secondly, to be successful in complex and volatile environments of large and complex transformation programs, such individuals need not only have relevant technical competency, right attitudes and behaviors, but also highly developed soft skills to allow them to deal with variety of audiences in volatile and challenging situations.

Collaboration

"If I could solve all the problems myself, I would"

- Thomas Edison, when asked why he had a team of twenty one assistants

Getting different teams to speak effectively to one another and to work together within large programs seems to be a subject that has transfixed attention of many. Some of the classical books, such as Brooks' "The Mythical Man Month" deal almost exclusively with this topic, and review of the root causes of project issues reveal that the most significant ones can be traced either to problems in communication or the lack of engagement from one of the stakeholders. Clear, frequent, and two-way communication helps understanding objectives of respective stakeholders as well as, their constraints, and helps keep everyone informed about the challenges and implications of solutions once they are in development. However, before the information exchange can take place and have the desired effects, the parties involved need to have a basic level of personal relationship and mutual trust. Without it, program processes are like an engine driven by a blind-folded driver.

This is important for architects for specifically two reasons: Firstly, architects often play the role of the catalyst working with multiple teams, so effective collaboration is critical for the success of their job. Secondly, to fulfil their role of the trusted advisor to the program board, they need to be aware of this critical success factor and the underlying practices.

Even though collaboration is one of the "soft" factors that seemingly "emerge" from program interactions, there are some specific levers that can be consciously used to create a productive collaborative environment and culture. These are enablers that make it easy for people to collaborate within an organization that leads to a collaborative working style, practices to build and sustain the collaborative culture and techniques to facilitate effective communication.

Enablers

There are several factors, that on their own do not make everyone behave in a collaborative way, but create an environment in which collaboration is easy. A number of them apply to programs regardless of their size:

- Appropriate communication skills giving individuals basic tools, allowing them to engage in collaboration
- "Slack" in the teams' capacity to allow staff to talk to one another rather than just focus on their own, narrowly defined, deliverables
- Appropriate physical space to support formal and informal interactions between customer and delivery teams, including co-location and collaborative working spaces for both daily problem solving workshops as well as less frequent larger stakeholder gatherings (town hall meetings)
- Technology enabling knowledge sharing, such as document repositories; tools to support basic disciplined collaborative working, such as version and bug tracking systems; tools to facilitate work done by geographically distributed teams; or the new generation of collaboration tools such wikis, weblogs or teamrooms.

But the enablers probably of the highest relevance to large programs are institutional incentives and measures stimulating right kinds of collaborative behaviors on both the customer and supplier sides. Without the right incentives, business stakeholders will not be motivated to participate and suppliers are likely to spend more effort jockeying for increased share in the program or negotiating about the scope of work, than delivering. Successful approaches build on shared risk, rewards and structures to encourage initiative, flexibility and pragmatism, focusing on measuring delivered value.

Organizing for Collaboration

Organization of work and work teams provides a powerful tool fostering alignment between various stakeholder groups. To build collaboration that will work regardless of the program organization structure requires the creation of blended permanent or virtual cross-functional teams with involvement of different stakeholders. How exactly this can be achieved varies across different contexts:

Architecture Activities

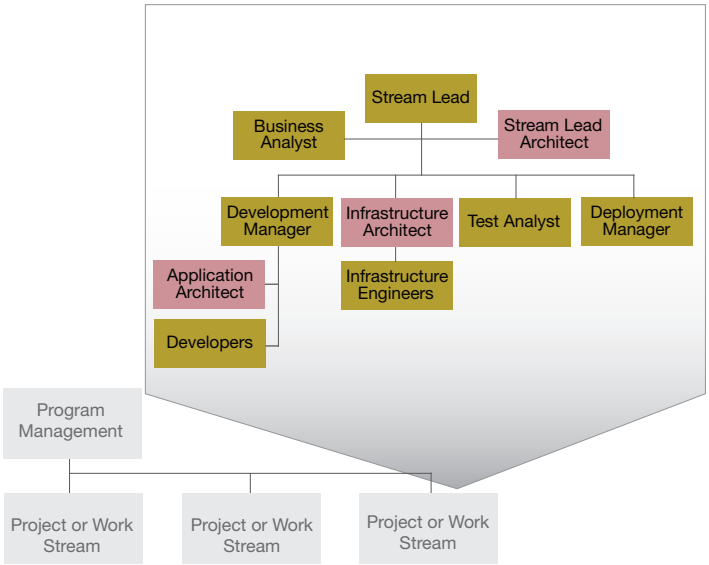
Large program architecture teams are likely to can organize themselves in different ways, depending on the stage of the program in the lifecycle. During the earlier program phases, the most sensible thing to do is to organize architects by business stakeholder community with problem domain divided into business segments, responsibility for which is allocated to individual enterprise architects within the team. When the program moves further on in the lifecycle, business and enterprise architects will be complemented by solution architects responsible for individual projects or project portfolios.

This paper advocates generalist architects to the specialists; however, at a certain point during the program, a team staffed with generalists will need to be complemented by specialist architects focused on tasks that require high degree of specialist skill, such as security or data architecture.

Delivery Activities

One way of creating a blended team that can be particularly effective during delivery stages is to set up a permanent project team with various discipline

Figure 5: Permanent Blended Delivery Team



representatives. Despite the fact that such a structure provides strong support for team building between various disciplines and is probably most conducive to collaboration, it may not necessarily fit all projects. On such projects, it may be replaced by a matrix organization structure, where an activity owner has to request appropriate staff from the functional manager from the area they belong to.

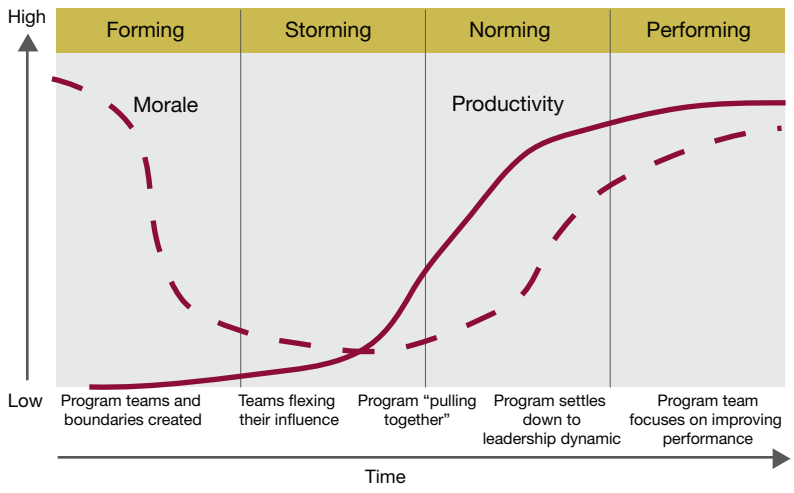
Building Collaborative Culture

Organization arrangements provide the structure that makes it easier to collaborate, but on their own they still cannot ensure that individuals do indeed collaborate. That requires the program to apply a conscious effort to develop and cultivate the culture and style of collaborative working.

Developing the right culture starts at the top as the core program team is likely to have a disproportionate impact on individual and team behavior.

Because collaborative culture develops over the course of personal interactions, both within and without project teams, it is something that cannot be mandated in a command and control fashion. However, it is something that to an extent can be cultivated by careful interventions into the social dynamics of the program team. A widely used team development model (8), on which such interventions are typically based, identifies a number of stages in team dynamics, which teams typically go through before reaching high performance. Although the model was originally developed for small teams, it can be also applied to inter-team dynamics within the large program.

Figure 6: Development of program Team Dynamics



Architects' relationships with other stakeholders, including the customer, program managers and delivery teams, needs to develop through these stages, to achieve high levels of collaboration.

Although a collaborative working style develops through program interactions, improving collaboration is not solely a function of elapsed time. This means that enlightened program and architecture managers can and should seek specific opportunities to accelerate this development. This can be done in two different ways. Firstly it is by using the hiring process to select people who have broadly similar shared values. This is then typically built upon by identification of activities that have a common tangible goal requiring various teams to come

8 B. W. Tuckman, "Developmental Sequence in Small Groups", *Group Facilitation: A Research and Applications Journal*, Number 3, (Spring 2001), http://dennislearningcenter.osu.edu/references/GROUP%20DEV%20ARTICLE.doc

together and completing the program timescales that cannot be achieved without breaking down the commonplace inter-team barriers.

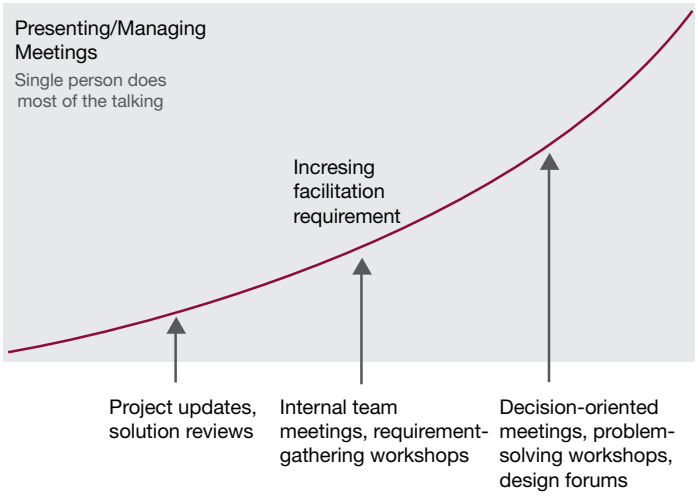
Depending on the stage in the program lifecycle, such a task can take the form of physical architecture development, delivery roadmap or implementation of a trial system. Stretch objectives of the activity create an environment full of pressure but also creative tension that results in a strong emotional experience and personal bonding, helping program teams to move from being a collection of disparate teams to performing as a highly cohesive program team with a strong collaborative ethos.

Conversations, Meetings & Workshops

Most of the architecture work is done through formal or informal conversations. Even though informal communication is important, a majority of communication in large programs takes the form of workshops and regular meetings, such as meetings of the business design authority, architecture board, program management or design steering group.

Considering how much time can be wasted in non-productive meetings and workshops, it is important for architects to be both aware of as well as utilize basic techniques for running effective meetings:

Figure 7: Facilitation of Different Types of Meetings



1. Decide on the right form for the meeting depending on the required degree of control over the meeting
2. Before the meeting, prepare clearly defined objectives and agenda and organize the relevant attendees. Ensure staff expected to contribute prepare content
3. Identify roles for the meeting to ensure it is kept on track. Workshops, for instance, should use a facilitated process to ensure they stay on track and promote participation of individual attendees. Use the process to ensure contribution of attendees is broadly in line with the meeting objectives
4. After the meeting, debrief and review the results and concerns. Capture key decisions and actions resulting from the workshop or meeting and distribute in the form of formal minutes of meeting.

To ensure steady progress of problem-solving workshops, the architect team should either have team members trained in facilitation or alternatively use services of professional facilitators.

Summary

Effective collaboration is the secret ingredient that allows large programs to handle complexity and accelerate progress whilst dealing with non-linear and turbulent delivery processes.

To reap the benefits of collaboration and to overcome challenges related to size and heterogeneity of stakeholders, programs need to adopt the right institutional incentives, such as risk and rewards structures, and invest a conscious effort in development and cultivation of a collaborative culture using individual communication skills and building on shared experience of involved teams.

Architecture Process

"Companies that consistently develop successful products —products with integrity—are themselves coherent and integrated. Moreover, this coherence is distinguishable not just at the level of structure and strategy but also, and more important, at the level of day to day work and individual understanding"

–Kim B Clark: The Power of Product Integrity

Good organization of the delivery process underpins both quality of the delivered solution as well as the efficient use of program resources. This applies both to architects and the rest of the program team. On large programs, contribution of architects to the success of the whole engagement depends, to a large extent, on how well they integrate architecture processes with the program and delivery management.

To ensure successful delivery, process integration needs to stretch throughout the program, but arguably the need to integrate is most visible during the program initiation phase. One of the biggest program challenges, and the source of conflict during this stage, comes from the need to balance various conflicting objectives, business goals and requirements, robustness of program architecture and the need for timely delivery against the pre-defined budget. Whereas the architecture team's prime objective is identification of key requirements, scoping out solutions and optimizing them across various projects, delivery teams are focused on meeting delivery dates of their respective projects. Such division of responsibilities for achieving objectives works fine as long as the architecture and delivery activities are done in series, but it creates significant challenges when the two are carried out in parallel.

Yet it is exactly this situation that is created by large programs requiring rapid ramp-up of hundreds of delivery staff. Ideally, program mobilization should be slow, starting with the careful selection of key staff, but often pressures from the client will prove too strong and the lead time to hire the necessary staff too long, overruling the conservative approach. There can also be other factors leading towards the need for concurrent delivery and architecture activities, such as inter-dependencies with external programs or significant requirements discovered only late in the lifecycle (such as during trials or pilots). Such instances require architects to focus on one specialized aspect, leapfrogging several stages of the lifecycle, whilst keeping in mind (and sometimes working on) other parts of the program that are at a completely different stage in the lifecycle.

To be effective under such conditions, architects on large programs need to develop a capability to deal with non-linearity in the program delivery process, and ability to work at different levels of abstraction with different stakeholder groups at the same time. Apart from skills, doing this successfully requires a distinctive set of processes.

Program Delivery Style

Before turning the attention to the architecture process, let us investigate the forces that drive the overall program approach as they exert probably the single biggest influence over the scope and type of architecture activities.

At the most basic level, large programs delivery can be organized either as a project or as a portfolio of projects:

- Project-style delivery focuses on a solution that may consist of a number of business and technical components, where these cannot be used and delivered

in isolation. An example of such a program could be a business transformation based on the implementation of an Enterprise Resource Planning software package.

- Portfolio-style delivery is based on a larger number of individual projects that deliver solutions or business capabilities which can be mostly used and delivered on their own right. An example of such a program can be a large infrastructure technology transformation, or some types of custom development. This also seems to be the organization model used by projects whose scale goes beyond some four hundred staff.

The style which should be used will be a conscious decision of program managers; however, it will be greatly impacted by the external environment of the program. There seem to be three factors that have a particular influence over program delivery style:

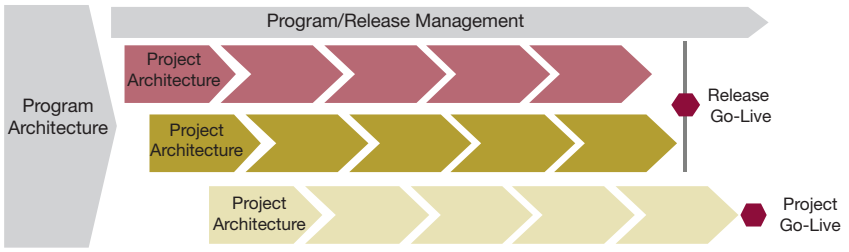
1. An existing or target operating model of an IT organization (9). For instance, delivering a solution for a diversified multinational via a number of IT suppliers will naturally lead more to a portfolio style. At the opposite end of the spectrum, where the IT organization works as a single entity, or is in a transition towards this model, projects will tend to be much more centralized.
2. Scope and complexity of the problem and solution domain. A more integrated nature of a packaged application, for instance, lends itself better to a project organization style than distributed desktop service implementation.
3. Number of staff to manage at a single time: the bigger the scope of work that needs to be done within a limited amount of time, the larger the number of delivery staff that needs to be used, and the bigger the challenges of maintaining a centralized organization.

As the scale of the program grows, the traits of the project portfolio approach become more pronounced; therefore, the remainder of this paper focuses primarily on this delivery style.

Engagement Roadmap

The project portfolio style program is likely to adopt a specific engagement roadmap suitable to its circumstances. This roadmap, which is essentially based on the Evolutionary Delivery Model (10), reflects the fact that the program scope is split up into a set of projects, which deliver increasing amount of business value, either in the form of business, functional or technical capabilities.

Figure 8: Project Portfolio Engagement Roadmap



9 J. W. Ross, "Forget Strategy: Focus IT on Your Operating Model", *CISR Working Paper 359*, Massachusetts Institute of Technology, (April 2006), http://mitsloan.mit.edu/cisr/papers.php 10 T. Gilb, "The Evolutionary Delivery Method", *The Principles of Software Engineering Management*, pp. 83-113, (Addison-Wesley, 1988)

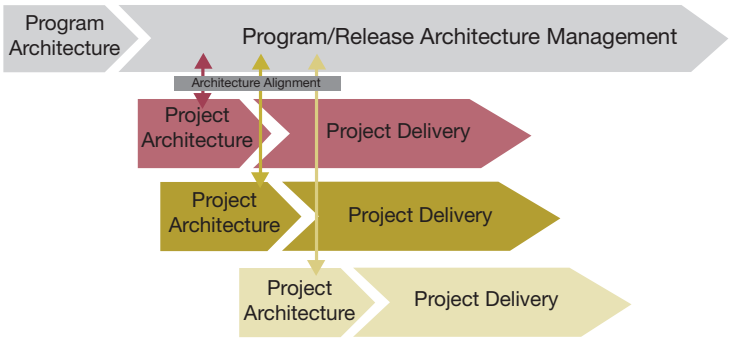
Using this approach, the program is mobilized against a high-level scope identified by the program architecture, which sets the key objectives, expected outcomes and capabilities, and provides a rough scope for the solutions to be delivered. This effort is likely to result in a relatively clearly defined core of "must-do" projects, and peripheral "should-do" or "could-do" projects. Based on business-driven priorities, validated from a technical delivery perspective, the program initiates must-do projects with a dedicated lifecycle. Each project is delivered by a dedicated team.

Rather than a single project, such a large program resembles a smaller version of a dedicated IT organization delivering a range of projects in a factory style delivery system.

Architecture Approach

In order to fulfil its critical role in the program, architecture development and governance activities need to adjust to and contribute to the shaping of the program delivery style. In order to prevent architecture activities from becoming a bottleneck in the program delivery system, architecture processes need to adopt the piecemeal, iterative approach, compatible with the overarching portfolio delivery style.

Figure 9: Architecture Roadmap



Program architects first engage with the customer to identify key requirements and scope of key solution components, and develop an initial version of the program architecture. This needs to be done through a relatively rapid iteration from conceptual to physical architecture.

In the next step, program architects, working with the customer, need to take into account program deployment and rollout approaches. This will lead to identification of the essential core capabilities, solution aspects that will be subject to localization, and capabilities that are nice-to-have add-ons.

Further steps require the program to identify the biggest "hard" dependencies between solution components to allow correct identification and scoping of delivery projects. As this activity lies at the boundaries of physical architecture, functional specification and high level design, it demands a strong collaboration of architects and delivery teams.

Once the key projects are identified, the core architecture team working with program managers needs to agree on the correct sequencing and scheduling of delivery activities, taking into account both dependencies, implementation approach, business appetite for change and program delivery capacity.

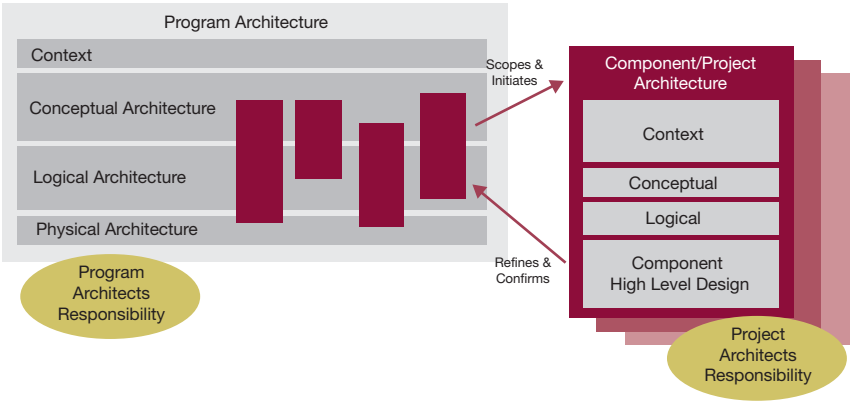
One of the benefits of this approach is that it quickly identifies the minimal scope that is "safe" to deliver, allowing quick mobilization of delivery. However, making this process work requires robust collaboration between various program teams and frequent feedback and communication between project teams and program architects. Once the architecture for an individual project is sufficiently developed, it needs to be checked for alignment against the program architecture. This is likely to lead to refinement of logical and physical levels of the program architecture and some changes in the component design to ensure their integration with other projects.

Architecture Artifacts

As indicated earlier, architecture in a portfolio delivery style program is a developed via a collaboration of program and project architects. Considering this effort spans boundaries of established teams and sometimes organizations, it is worthwhile and necessary to identify type and content of architecture-related artifacts delivered in this process as well as responsibility for their development. The key distinction is between program-wide and project-specific deliverables:

- Program architecture artifacts includes mostly conceptual architecture framework addressing all program aspects, and key logical architecture artifacts required to scope out solutions to be developed. Physical architecture artifacts will be included in the program architecture selectively—only where these cannot be developed efficiently by a project (e.g. security standards). Program delivery roadmap will be a combined deliverable produced by architects and the program management team
- Project architecture consists of full conceptual to physical architecture, but only for use cases and components in scope of a particular project. Whereas conceptual and logical elements will be relatively concise, the core of project architecture will be formed by physical architecture artifacts. As the project architecture is more narrow in scope, it can be extended by including a closely related functional specification and high-level designs artifacts.

Figure 10: Project & program artifacts



Standardization Of Architecture Process

Harmonization of architecture methods and key artifacts is one of the factors that can significantly assist or hamper efficiency of integration between architecture and program delivery, especially on multi-sourced programs.

In an ideal case, delivery will be carried out by a single organization, which should ensure common and integrated architecture and delivery standards, such as in the case of Capgemini Integrated Architecture Framework (IAF) and widely used Rational Unified Process (RUP). The situation grows harder with the increasing number of organizations involved. Standardization of lifecycle process, methods and key artifacts sounds like a sensible approach; however, frequently it will be beyond the means of a program to drive through required changes in delivery processes on tight timescales.

In situations like this, it is best to accept the existing delivery methods and life-cycle as long as they are in line with good industry practice and are conceptually compatible. Spare capacity then can be put to work on ensuring that architecture and design artifacts have the right content that originates from the right engagement with stakeholder groups.

Summary

Effectiveness of architecture activities on a large program revolves around integration of architecture efforts with program delivery processes, especially during the initial stages of the program. To deal with the challenges of mobilization, large programs often adopt a project portfolio approach to delivery, which becomes one of the main factors influencing architecture approach. Using this approach, program architecture is developed upfront followed by a more detailed elaboration of individual project architectures developed on case-by-case, as needed basis.

Delivery Considerations

Almost anybody with the intelligence of the average businessman can produce [exciting new ideas], given a halfway decent environment and stimulus. The scarce people are those who have the know-how, energy, daring, and staying power to implement ideas."

– Theodore Levitt, business academic

There are two factors that make architects' involvement highly desirable not only during the initial program stages, but throughout the whole program lifecycle.

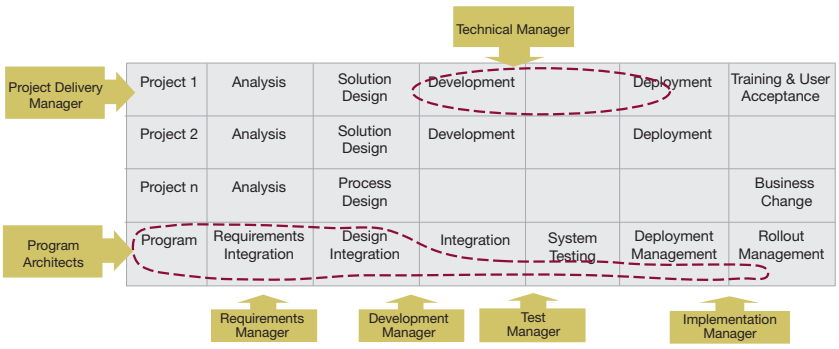
Firstly, it is the need to enthuse the engineered solution with the concept, which is something that is difficult to do when the contact between architects and delivery teams is through the pile of architecture documentation. Related to this is the need for empirical validation of the architecture in the face of day-to-day reality of delivery and usage of the resulting solution. Considering the gap that often arises between the solution in the architecture stage and delivered results, successful delivery is probably one of the few ways to confirm correctness of the architecture. Involvement of architects in the later stages of the lifecycle then becomes a question of upholding their accountability.

Second reason is the need for large-scale engagements to manage complexity. Whereas qualitative aspects of regular-sized projects allow complexity to be managed directly by program managers, in large-scale transformation engagements—with their vague problem domain definition, diverse groups of stakeholders and their numerous chained and circular dependencies—this task becomes much more difficult. Yet it is exactly the type of activity that plays to architects' traditional strengths at problem solving, ability to connect business and technology worlds, backed up by their general systems approach.

Role of Architects in Delivery

Large programs have specific staffing requirements driven by the need to manage their internal complexity during delivery stages. To illustrate this, the diagram below shows typical program and project level activities in a portfolio-based program:

Figure 11:



Consider the typical roles in a matrix-managed program: program managers will focus mostly on time, budget and scope management. Functional specialists, such as test or implementation managers, will focus on managing activities dedicated to

their specific stages in the lifecycle. At the project level, delivery project managers will focus on logistics of delivery and budgets, and technical managers on management of technical teams and resolution of technical and process issues.

What is missing in this picture for large programs is a program-level solution integration capability spanning all stages of the lifecycle. Because functional skills required for this role—requirements elicitation, solution and process design, and technical dependencies management—are a close match to the traditional architect skill set, architects are well equipped to take on this role and transcend their traditional delivery roles as design authority or technical managers. To do this well, however, requires a deeper understanding of challenges surrounding delivery disciplines on large programs.

Requirements Identification

To a large extent, program success starts with professional and pragmatic business design and requirements identification. Although nominally owned by the Business Design or Requirements Authority working through various business forums, it is one of the areas that solution and enterprise architects are inevitably drawn into.

There are two lessons learnt regarding the approach to the identification of requirements that are specific to large-scale programs:

Firstly, for large engagements, it is critical to allow for early trials and proof of concept activities. Solution experiments are one of the best ways for businesses to understand their needs better, while technical experiments help understand limits of the technology early enough to inform the requirements and delivery planning.

Secondly, virtually all large-scale solutions follow the core platform/localization approach to deployment and rollout. This approach dictates the need for the program to decide on the requirements constituting core functionality as well as migration principles during the early stages of the program:

- The program needs to identify the core set of requirements addressing the essential needs of key stakeholder communities
- During design stages, it needs to ensure the solution design is sufficiently flexible to support variations of business requirements through configuration, without the need for extensive testing and dedicated technology deployment activities
- Where it is not possible to migrate users on the core platform and the user group is large, extend the core design of the platform to support the requirements of the new user group, where possible, generalizing the requirements rather than developing a specific variation.

Design

For many, solution design is the key stage during which programs define both the exact solution functionality and how it will be realized in a technical scenario. Although the uncertainty of the solution in this stage is dramatically decreasing, it is still not uncommon for programs to fail or have significant overruns, which have its root causes in this activity. The biggest reason for this are overlooked inter-stream dependencies.

It is likely that only most visible dependencies are addressed during the architecture stage, because of the system behavioral complexity and time constraints. Design stage is therefore the first point during which dependencies can be investigated at a greater depth.

Issues caused by overlooked dependencies are in turn often caused by less experienced development teams that have sometimes a tendency to focus on more obvious solution aspects such as functionality of solution components first, forgetting external dependencies. When these are discovered later in the solution lifecycle, the design of the implemented system often needs to be changed at higher monetary and timescales cost.

For these reasons, the earlier the in-depth dependency mapping is carried out, the more flexibility the project has to manage its scope, budgets and timescales. Architects in large programs are the key group driving the mapping of project dependencies and components comprising the program.

Analysis and mapping of design dependencies requires a combination of top-down and bottom-up efforts. It needs a top-down perspective to identify potentially inter-dependent projects and teams, but since only the designers responsible for individual components have the detailed knowledge of their functionality, technical dependencies can be identified fully only through bottom-up efforts. This is one of the occasions where the program needs to exercise its collaborative capabilities with designers from various parts of the organization coming together. Dependency mapping is best achieved through a series of facilitated dependency mapping workshops in which the teams will utilize use cases, scenario analysis and interaction diagrams to identify scope of components and their dependencies. Often this activity will lead to identification of gaps and misalignment, giving rise to the need to modify the solutions.

Because of the multiple ways of achieving the same technical and business capability, this analysis is likely to take few iterations before it is complete. Once the initial analysis is finished, the impact of alternative solutions on time and cost need to be investigated. As the next step, the impact of various alternative solutions on both project and program outcomes needs to be identified and alternatives prioritized until the program reaches a consensus on the optimum solution.

Testing

Architects' role during testing is normally limited, but they do need to understand this stage even if only to ensure its effectiveness, understand its implications on other stages of the lifecycle and its limits.

Most testing activities on very large and complex programs will not be dissimilar to the ones carried out on regular programs, except perhaps for their scale and degree of automation. Some testing aspects, such as performance testing, are likely to be more important because of the size of the user population, but the approach taken will not fundamentally change.

An activity whose nature will be different on large and complex systems—mostly because of logistical complications of assembling a solution developed by a large number of individuals—is integration testing. Because of the number of dependencies in a large-scale solution, integration testing needs to focus not only on individual interfaces and point-to-point interactions, but also on correct functioning of interactions spanning multiple components and projects.

Apart from logistical complexities, it is important to be aware of the limits to the effectiveness of testing in large programs. In a perfect world, the biggest dependencies should have been discovered and addressed during the design activities, and testing should be about fine-tuning the implementation. However, because it can be difficult to fully analyze the behavior of complex large-scale systems with their chained and circular dependencies, it is likely that testing activities will discover misalignment in the specification or design. A rule of

thumb for well designed systems on a smaller scale is that issues discovered should not require the project to return more than one stage back in the lifecycle. However, due to the reasons explained, this has limited applicability to large-scale complex systems, which in turn has several implications for large program delivery approach:

- Firstly, the program needs to develop more sophisticated methods of assessing quality of deliverables handed over between individual stages
- Secondly, programs need to use early and continuous testing to identifying gaps in specification, design as well as implementation
- Thirdly, programs need a plan to have spare capacity and timescale contingency to resolve potential front-end lifecycle issues discovered during testing.

Deployment and Rollout

It is not only engineering activities that take time. Often, a geographically diverse rollout affecting a large number of users can take up to several years. The length and criticality of rollout activities then affect both architecture and program management:

Firstly, in addition to a good practice of "designing for operations" the solution needs to be designed for rollout. This requires a development of technical means that reduce the need for manual intervention within deployment activities such as software and hardware deployment, technical migration, rollout planning, end-user communication or training.

Secondly, large programs need to develop a set of industrialized deployment processes that are fine-tuned for efficiency using the automation technology, and synchronized with the rest of the program. The objective is that the deployment activities, especially those related to localization of the core solution, create a "pull-type" demand for engineering activities.

Thirdly, complex programs need to have a strong capability for handling deployment issues. Despite assurance activities, the risk of deploying large and complex solutions remains relatively large, and because of the system complexity, diversity of deployment contexts and logistical challenges, there are practical limits to how much this risk can be reduced by additional testing or analysis. Rather than focusing solely on risk prevention, large programs therefore need to adopt an intelligent, risk-conscious deployment approach with robust issue containment and rollback mechanisms, underlined by a very strong issue handling capability. This capability needs to be headed by a senior delivery manager with a program level mandate that:

- Has immediate access to the program director and program board, allowing quick escalation of crisis issues to the highest level
- Gets active involvement of all key managers responsible from all streams and lifecycles
- Is supported by a problem solving, cross-functional team, consisting of the strongest program delivery staff, supported by a virtual team of subject matter experts.

The experience shows that in large programs, this capability needs to be maintained throughout the rollout period, until the solution is stabilized and all major issues resolved.

Summary

Standard good industry delivery practices do need to be applied to large transformation programs; however, we need to be aware of limitations of such practices to address difficult to analyze complexity and unpredictability of large and complex systems. To handle the resulting issues during the delivery lifecycle, programs need to develop a solution integration capability operating throughout the solution lifecycle. In particular:

- Capability to drive identification and prioritization of new requirements throughout the duration of the program
- Development capability responsive to the changes in requirements and specification be driven by architecture, testing and deployment activities
- Pragmatic testing capable of dealing with the scale and complex inter-dependencies
- Industrialized deployment management processes and tools, with a very strong program-wide issue handling capability.

Program architects are likely to either take the lead on some of these activities or provide significant support to the management of dependencies and integration of output of individual project teams.

Architecture Governance

The purpose of organization is to achieve the kind and degree of order and conformity necessary to do a particular job. The organization exists to restrict and channel the range of individual actions and behavior into a predictable and knowable routine. Without organization there would be chaos and decay."

-Theodore Levitt, business academic

In the same way that people are critical to moving delivery forward and processes give a framework within which this effort takes place, the program also needs formal structures that prevent mistakes. This is achieved through architecture governance, which according to The Open Group [11], is about control and monitoring of architectures throughout the IT organization value chain, building on individual accountability and implemented via specific organizational arrangements.

Accountability Principle

Good governance starts with accountability. Establishment of clear accountability and separation of roles is critical as it:

- Allocates decision making rights between various groups of stakeholders
- Sets core mechanisms, ensuring balance of the three key delivery objectives—cost, time & quality
- Forms the basis for distribution of work between involved stakeholder groups and
- Simplifies conflict resolution process by reducing the number of involved parties.

What sounds simple in principle can be harder to achieve in a context of a large program organization with multiple stakeholders. Often a large number of people who the program interacts with come from different business units, sometimes from different organizations, which easily leads to gaps, overlaps and disagreements about the scope and responsibilities. To ensure clarity of decisions under such circumstances, decision making over the key program tasks needs to be delegated to the three authorities representing a community of stakeholders:

- Architecture authority, accountable for the solution;
- Business authority, responsible for developing the target business operating model; identification and prioritization of business requirements; representing interests of business stakeholders; working through various business forums; and
- Live support authority, responsible for identification of support requirements and representing interests of the operational community.

In their role of playing the ultimate decision maker for their respective responsibilities, these authorities use monitoring and control mechanisms to ensure the program outcomes are fit to the objectives, deliverables and are sustainable.

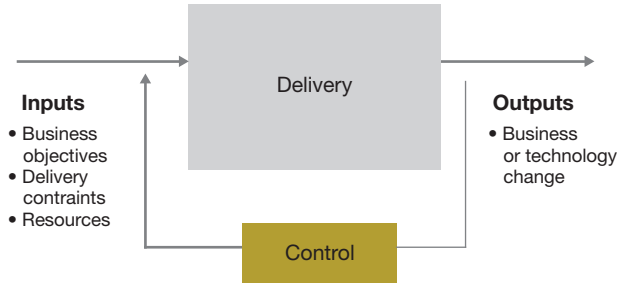
Monitoring & Control Mechanisms

Building on accountability is "a system of controls over the creation and monitoring of all architectural components and activities, to ensure the effective introduction, implementation, and evolution of architectures within the organization." (11)

11 The Open Group, *The Open Group Architecture Framework 8.1.1*, Chapter 26, (The Open Group, 2006), http://www.opengroup.org/architecture/togaf8-doc/arch/chap26.html

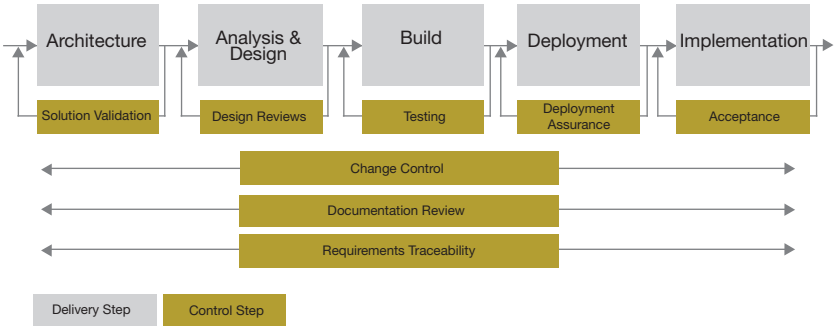
According to the control theory, control and monitoring of delivery process should be simple: the customer specifies requirements and the project team identifies and delivers the solution. Once that is done, the program assurance function compares the results against the original specification, as described in the diagram below.

Figure 12: Simple Model of Architecture Control and Monitoring



Whereas such an approach may work for a mid-size "business as usual" project, variance between expectations and results at the end of the delivery can easily become too great for large transformation programs. Considering the allocated budget at this point will have been spent, there would be few alternatives to contain the issues. A greater transparency of interim program outcomes coupled with stringent control and monitoring throughout all stages in the lifecycle can avoid this risk.

Figure 13: Lifecycle Architecture Control And Monitoring



From a practical perspective, most elements of effective architecture control and monitoring activities during delivery can be tied to quality assurance and program processes:

The diagram illustrates one of the interesting aspects of architecture governance—even though the proper execution of monitoring and control are essential to effective architecture governance, these activities are not a typical architect's responsibility. To ensure monitoring and control activities are carried out sufficiently, the architecture function needs to work with program managers and the program management office to allocate responsibilities for architecture governance tasks:

Control	Control objective	Control activity responsibility
Solution Validation	Proposed business solution in line with corporate operating models Proposed solution supports stakeholder's objectives & requirements and is within the budget Non-functional and functional requirements are identified. Proposed solution architecture supports the requirements Proposed solution meets the agreed enterprise architecture principles and standards	Enterprise or program architects Finance team
Design Reviews	Solution components compliant with the architecture and principles Design is based on best practices and supports non-functional requirements	Project architects – peer review "Deep-dive", in-depth, review of high design documents for high risk components by program architects
Testing	Developed solution is defect-free and meets functional and non-functional requirements	Test manager
Deployment	Deployment of the solution is defect-free. Technical solution behaves according to the design.	Technical assurance team
Acceptance	Technical and non-technical aspects of the solution meet business and operational requirements and are fit for use. Technical solutions are supportable. Proposed solution meets the agreed implementation standards	Implementation manager, Transition manager
Change Control	Ensure the changes to the requirements and solution are introduced in a controlled manner, in line with the vision and strategy	Program management office
Documentation Review	Architecture, design and implementation documents and plans are of acceptable quality	Program management office
Requirement Traceability	"Scoped", designed, developed and implemented solutions meet the functional and non-functional requirements	Program management office

Architecture Functions in a program

Before considering specific organization arrangements used to enforce architecture governance, let us turn our attention to the key program architecture functions including architecture board, program architecture team, delivery teams and design steering group.

Architecture Board

The architecture board is the ultimate decision maker on solution-related questions. Chaired by the program Lead Architect, it owns the vision for the solution and advises the program director and program board on solution-related aspects. To ensure participation and accountability, it should involve representatives of the key stakeholder communities:

- Business community should be represented by a nominee of the business requirements authority. In most cases this will be either a business "super-user" or the business design authority manager.
- Service support community will typically be represented by a receiving manager or service manager responsible for the ongoing delivery of live support for the solutions in scope.

Even though it can lead to communication challenges and potential clash of cultures, it is a good idea to involve these stakeholders from the start in order to ensure proper communication, participation and conflict resolution from the beginning, especially before the start of development and engineering phases.

Architects Group

Architects group is a virtual or permanent team that supports the architecture board and program management team. It develops the future vision responding to the needs of various stakeholder communities and identifies projects to realize the vision. In particular, it:

- Identifies the scope of solutions and key components
- Identifies relevant design principles and guidelines
- Identifies key architecture-relevant functional and non-functional requirements
- Takes the lead on making changes to the vision and shaping solutions to significant new requirements.

Depending on the program setup, some of the architects can carry out a number of additional value-added activities, including:

- Scope and requirements management
- Product selection
- Interviewing and vetting lead designers and developers
- Contribute primarily to the identification and resolution of program risk & issues
- Provide technical assurance including review of key project architectures and designs, or management of code inspections and deployment assurance activities.

Delivery Teams

Responsibilities and organization of delivery teams will vary depending on the type of the project, ranging across teams specialized in development of functional requirements, business change management, design, development, testing or deployment. As mentioned in the previous sections, depending on its purpose, type and scale, the program can make a conscious decisions to include architects in delivery teams as a means of fostering collaboration and alignment between various groups.

Design Steering Group

Large programs, especially where they span organizational boundaries, often have to develop tightly coupled solutions that are delivered by different projects or delivery streams. This creates a need for active management of such dependencies. The design steering group addresses then need for pro-active inter-stream design alignment and management of detailed designs dependencies, supporting solution integration activities described in the previous section.

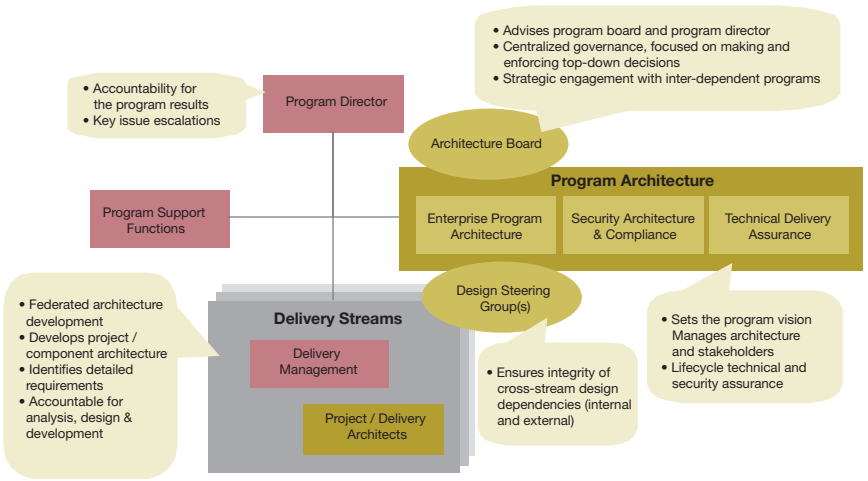
Integration with program organization

Eventually, the architecture governance responsibilities need to be reflected in the organization structure of the program. Placement of the architecture function in the organization's structure will be primarily driven by a program delivery style:

- Project style program organization will depend on a centralized architecture team
- Portfolio style organization will consist of a larger number of individual projects and is likely going to have architects both in the program management and project delivery teams.

An illustration of a portfolio program organization structure with a specific focus on architecture-related functions is provided below.

Figure 14: Architecture Functions in program organization



The example above assumes an operating model based on a single point of management with a single budget that leads to a decision to centralize program architecture authority. But due to the complexity and scale of the solution domain, a significant amount of architecture in this model is carried out and managed by project delivery teams.

If the scope of the program was for instance narrower, architecture organization could be more centralized, with architects only in the central team. On the other hand, if the program was to be delivered by multiple organizations with independent budgets and multiple points of ownership, it would retain the organization's structure illustrated above, but the operation of the governance functions would be much more oriented on negotiation and co-ordination with its partners.

Summary

Architecture governance establishes checks and balances necessary to avoid mistakes in program delivery activities. The objective of program architecture governance is to ensure compliance of solutions with the program strategy and objectives, and external standards while ensuring efficient implementation of architectures.

At its core, architecture governance is implemented through several control and monitoring activities. To ensure individual accountability in often highly federated environment of large organizations, it is critical to clearly define architecture roles and responsibilities and to align these with an overall program organization model.

Governance on its own is however no silver bullet; to achieve significant contribution to the program, it needs to be underpinned by soft factors as well as delivery process integration.

Conclusions

Large-scale engagements are indeed different and often difficult, yet that does not mean that we cannot learn how to improve our success rate. To quote the Royal Academy of Engineering (12), *"... there is a widespread perception that success rates for the delivery of IT projects are unacceptably low. However, good practice is routinely implemented in a number of sectors and some of the achievements in large-scale IT projects are truly remarkable."*

For more than a decade we have heard about the gap between business and IT and it has been suggested that architecture methods can help us both leap over this gap. Perhaps we have mislead ourselves by the language we are using. The metaphor often used to describe program efforts is not leaping, but herding cats. Whereas leaping implies a one-off activity (e.g. developing architecture), herding cats as well as delivering large programs, requires ongoing patient and focused efforts tackling complexity and changing nature of the problem bit by bit.

Technical, social and delivery complexity of large-scale programs will not go away and is likely to continue to pose challenges to the approaches that use mostly traditional analytical methods. What we need to learn is that architecture methods are only the first step. For a lasting contribution of architects to the successful delivery of large-scale programs, we need to build on these methods and implement ongoing architecture management practices specific to external circumstances, integrated with program processes, and backed by personal leadership of the core team working collaboratively with the rest of the program. It is their efficient interplay in the challenging environment supported by their personal efforts and expertise that makes large-scale engagements successful.

12 The Royal Academy of Engineering, *The Challenges of Complex IT Projects*, (The Royal Academy of Engineering, 2004), http://www.raeng.org.uk/news/publications/list/reports/Complex_IT_Projects.pdf

Program Profiles

Apart from lessons drawn from available literature on IT architecture, software engineering and product development, this paper is based on experience and lessons learnt from a number of large programs Capgemini has managed and delivered.

The paper draws extensively on the researcher's three years of experience as the core architecture team member on a successful technology transformation program providing a standardized user environment for users following the merger of two large organizations. At a cost comparable with industry metrics, the program achieved rapid delivery with a consistent peak throughput of 1,500 desktops a day, and improved service levels allowing next day provisioning of software services.

In addition to this, parts of the paper also draw on the researcher's involvement in a business transformation program currently delivered by Capgemini at one of the Global 300 manufacturers with more than 100,000 staff, aiming at global standardization of processes in all aspects of their business—from engineering, to manufacturing, distribution, sales and service support. The solution side of the transformation involves a consolidation of a globally distributed application landscape consisting of thousands of applications, onto a core ERP platform.

The section on collaboration communication draws on the researcher's experience contributing to an SAP implementation enabled transformation of finance, HR and procurement processes of an organization with over 100,000 employees.

In addition, case material produced by other programs and teams have been used as well in this paper, including:

IT Breakthrough—An application transformation program to reduce the complexity and total cost of ownership of the ageing and highly complex IT landscape of a global process manufacturing organization with more than 15,000 users. The program consisted of the introduction of enterprise architecture and a transformation program based on SAP implementation.

Business Development & Innovation—A dedicated architects team within the largest UK bespoke custom development IT organization, responsible for front-end phases of program lifecycle, and identifying and defining innovative architectural solutions to business requirements.

Bibliography

Clark, K. B. and Fujimoto, T., "The Power of Product Integrity", *Harvard Business Review*, pp. 107-118, Nov-Dec 1990

Gilb, T., "The Evolutionary Delivery Method", *The Principles of Software Engineering Management*, pp. 83-113, Addison-Wesley, 1988

Hartman, D., "Interview: Jim Johnson of the Standish Group", *InfoQ*, August 2006, http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS

Jørgensen, M. and Moløkken, K., "How Large Are Software Cost Overruns?", *Information Systems and Software Technology*, 48(4), pp. 297-301, Simula Research Laboratory, April 2006, http://www.simula.no/research/engineering/publications/Jorgensen.2006.4

Kurtz, C. and Snowden, D., "The New Dynamics of Strategy: Sense Making in a Complex and Complicated World", *IBM Systems Journal*, Volume 42, Number 3, pp. 462-483, http://www.research.ibm.com/journal/sj/423/kurtz.pdf

Morello, D., "Unlocking the Business Value of People: Building Versatility", Gartner Research Paper, May 2003

The Open Group, *The Open Group Architecture Framework 8.1.1*, Chapter 26, (The Open Group, 2006), http://www.opengroup.org/architecture/togaf8-doc/arch/chap26.html

Ross, J. W., "Forget Strategy: Focus IT on Your Operating Model", *CISR Working Paper 359*, Massachusetts Institute of Technology, April 2006, http://mitsloan.mit.edu/cisr/papers.php

The Royal Academy of Engineering, *The Challenges of Complex IT Projects*, The Royal Academy of Engineering, 2004, http://www.raeng.org.uk/news/publications/list/reports/Complex_IT_Projects.pdf, (accessed 27 April 2007

Tuckman, B. W., "Developmental Sequence in Small Groups", *Group Facilitation: A Research and Applications Journal*, Number 3, Spring 2001, http://dennislearningcenter.osu.edu/references/GROUP%20DEV%20ARTICLE.doc

Wieringa, R. J., ed., *Competencies of the ICT Architect*, (in Dutch), Nederlands Architectuur Forum, November 2006, http://www.naf.nl/content/bestanden/competenties_van_de_ict-architect.pdf

Yourdon, E., "People in Death March Projects", in Death March: *The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects*, pp. 99-128, Prentice-Hall, 1999

With thanks to the colleagues who contributed ideas and provided feedback for this paper, including Pete Thompson, Gunnar Menzel, Gordon Blain and Julie Morling.

About Capgemini and the Collaborative Business Experience

Capgemini, one of the world's foremost providers of consulting, technology and outsourcing services, enables its clients to transform and perform through technologies. Capgemini provides its clients with insights and capabilities that boost their freedom to achieve superior results through a unique way of working —the Collaborative Business Experience —and through a global delivery model called Rightshore®, which aims to offer the right resources in the right location at competitive cost. Present in 36 countries, Capgemini reported 2007 global revenues of EUR 8.7 billion and employs over 83,000 people worldwide.

More information is available at **www.capgemini.com.**

For more information contact:

Jiri Ludvik
Senior Enterprise Architect
+44 (0)870 238 8856
jiri.ludvik@capgemini.com

20080422

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

www.capgemini.com

Australia
Level 4
50 Carrington Street
Sydney NSW 2000
Tel: +61 2 9293 4000

Belgium
Bessenveldstraat 19
B-1831 Diegem
Tel: +32 2 708 1111

China
Unit 1101-04, Azia Center
1233 Lu Jia Zui Ring Road
Shanghai 200120
Tel: +862 161 053 888

Denmark
Ørnegårdsvej 16
DK-2820 Gentofte
Tel: +45 70 11 22 00

Finland
Niittymäentie 9
02200 Espoo
Tel: +358 (9) 452 651

France
Tour Europlaza
20 ave. André Prothin
92927 La Défense Cedex
Tel: +33 (0)1 49 00 40 00

Germany
Hamborner Strasse 55
D-40472 Düsseldorf
Tel: +49 (0) 211 470 680

India
Piroshanagar, Vikhroli
SEP2 B3 Godrej Industries Complex
400 079 Mumbai
Tel: +91(22) 5555 7000

Italy
Via M. Nizzoli, 6
20147 Milano
Tel: +39 02 41493 1

Middle East
P.O. Box 502 420
Dubai
UAE
Tel: +971 50 884 77 64

Netherlands
Papendorpseweg 100
3528 BJ Utrecht
Postbus 2575
3500 GN Utrecht
Tel: +31 30 689 0000

Norway
Hoffsveien 1D,
0275 Oslo
Tel: +47 24 12 80 00

Poland
Al Jana Pawla II 12
00-124 Warsaw
Tel: +48 (22) 850 9200

Portugal
Edifício Torre de Monsanto
Lugar de Romeiras
Miraflores
1495-046 Algés
Tel: +351 21 412 22 00

Spain
Edificio Cedro
Calle Anabel Segura, 14
28100 Madrid
Tel: +34 91 675 7000

Sweden
Gustavlundsvägen 131
PO Box 825
161 24 Bromma
Tel: +46 8 5368 5000

United Kingdom
76 Wardour Street
W1F 0UU London
Tel: +44 20 7734 5700

United States
623 Fifth Avenue
33rd Floor
10022 New York
Tel: +1 212 314 8000