12 Scaling Agile to an Enterprise Level

AS A RESULT OF THE widespread adoption of agile practices, large corporations are beginning to apply agile at an enterprise level. This has introduced some new challenges of how to scale agile principles and practices to an enterprise level and what to do about many of the existing project management office (PMO) practices and other higher-level business management practices that are typical in large enterprises for managing portfolios of projects and products.

> Our experience is that 'core' Agile methods such as Scrum work wonderfully for small project teams addressing straightforward problems in which there is little risk or consequence of failure. However, 'out of the box,' these methods do not give adequate consideration to the risks associated with delivering solutions on larger enterprise projects, and as a result we're seeing organizations investing a lot of effort creating hybrid methodologies combining techniques from many sources.[1]

Dean Leffingwell identifies two primary challenges involved with scaling agile to the enterprise level in his book, *Scaling Software Agility—Best Practices for Large Enterprises*:[2]

1. The first challenge is overcoming the challenges inherent in the methodology. Scaling an agile methodology to an enterprise level requires reinterpreting the values and principles behind the methodology in a much larger context, and typically also requires some adjustments in agile practices to adjust to that context.
2. The second challenge is overcoming the limitations imposed by the enterprise that will otherwise prevent the successful application of new methods. As I've mentioned earlier, implementing an agile approach at an enterprise level can be like plugging an appliance that

[1] Scott Ambler and Mark Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise* (IBM Press) (Upper Saddle River, NJ: Pearson Education, 2012; Kindle Edition), pp. 349–353.
[2] Dean C. Leffingwell, *Scaling Software Agility* (Reading, MA: Addison-Wesley, 2007), p. 87.

requires DC current into an AC outlet; there's a fundamental incompatibility in many cases that requires some adaptation to get an agile approach to work inside of an organization that wasn't designed around being agile.

The first challenge will be discussed in this chapter, and the second challenge will be discussed in the Chapter 13, "Adapting an Agile Approach to Fit a Business."

ENTERPRISE-LEVEL AGILE CHALLENGES

Beyond the factors previously discussed to adapt an agile approach to a company's business environment, there are a number of challenges associated with scaling agile practices to an enterprise level.

Differences in practices

Applying agile development practices at the enterprise level typically involves some adjustments to those practices, as shown in Table 12.1.[3]

TABLE 12.1 Adaptations to Agile Practices at the Enterprise Level

Typical Small Agile Project	Typical Enterprise-Level Implementation
Customer Participation The customer is integral to the team.	The customer may be remote or may not have the skills or time available to participate directly in the agile team. The customer may also consist of a number of different users and stakeholders. A business analyst (BA) on the team many times plays the role of a proxy for the customer, but that BA should ideally be empowered to act on behalf of the customer.
Development Team Organization Developers, product owners, and testers are co-located and not separated by time zones and language barriers.	It is likely that many team members may be in different geographic locations and different time zones and perhaps even speak different languages.
Application Architecture In a small-scale agile project, the application architecture is expected to emerge as the project progresses.	With larger-scale systems, the costs and difficulty of refactoring the design as the project progresses to accommodate changes in the architecture make it essential in many cases to do more upfront architectural planning and design in the project. Large-scale system designs typically require breaking up the design into components, and without having sufficient architecture defined, it becomes impossible to allocate the work to teams.[4]

[3] Ibid., pp. 88–89.
[4] Ibid., p. 204.

TABLE 12.1 (*Continued*)

Typical Small Agile Project	Typical Enterprise-Level Implementation
Requirements Management The agile development effort can take place one story at a time, and the design incrementally evolves over the duration of the project.	In large enterprise-level implementations, this approach doesn't work very well. A more integrated approach may be required to coordinate the development of the stories to ensure that they all really work together to produce releasable functionality that fulfills the business need.
Project Portfolio Management Typical agile projects do not provide a mechanism for higher-level integration to fulfill typical corporate needs for portfolio management of a large set of agile projects.	It can be very difficult to integrate a number of agile projects into a typical enterprise-level project portfolio management approach; however, some level of integration and management is necessary to make portfolio management decisions. This will many times require adopting a hybrid approach to provide the necessary balance of predictability, control, and agility.
Team Organization Ideally, agile teams consist of peer-level developers who take responsibility for their own actions with a minimum of direction. It is intended to be a team of equals with no formally designated technical team leader.	For large-scale development teams, this can be very difficult, if not impossible, to achieve. Many times it is necessary to build teams of more junior-level developers led by a more senior-level tech lead who can provide some level of guidance and direction to the rest of the team.

Reinterpreting agile manifesto values and principles

There is also a need to reinterpret the Agile Manifesto values and principles in a different context at the enterprise level, as shown in Table 12.2.

TABLE 12.2 Agile Manifesto and Enterprise-Level Implementation

Typical Small Agile Project Values	Typical Enterprise-Level Implementation
Tools: "Individuals and interactions over processes and tools"	There is a greater need for tools at an enterprise-level: ■ There is more of a need for a defined process to coordinate and synchronize the work of large projects requiring multiple teams as well as coordinating other activities outside the teams. ■ Tools can become more important at an enterprise level as the scope and complexity of the effort grows.

(*continued*)

TABLE 12.2 (*Continued*)

Typical Small Agile Project Values	Typical Enterprise-Level Implementation
Documentation: "Working software over comprehensive documentation"	■ At an enterprise level, solutions tend to be much more complex, and software is only one part of the overall solution. As a result, some form of additional overall coordination is needed to integrate all the components of the overall solution. ■ At an enterprise level, a solution might also include training, business process changes, a support plan, a marketing and rollout plan, and many other requirements beyond just developing software. All of these may increase the need for some kind of documentation.
Collaboration: "Customer collaboration over contract negotiation"	■ At an enterprise level, there is typically a much broader range of customers and stakeholders to consider and managing expectations can be a lot more challenging. ■ Some form of project charter document may be worthwhile to help manage expectations but it could be defined at a fairly high-level.
Planning: "Responding to change over following a plan"	As the scope and complexity of an effort at an enterprise-level increases, there is typically a need for more planning to: ■ Coordinate the efforts of large projects requiring multiple teams. ■ Synchronize the efforts of development teams with other activities outside the scope of the development effort. ■ Adapt the development effort into higher-level management processes that may be more plan-driven.
Change Control: "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage."	■ Change never comes without consequences, and change control can be valuable for configuration management and validating that any new changes are consistent with other previously developed requirements and assumptions. ■ Done properly, it does not equate to stifling or *preventing* change. It means ensuring that unnecessary change (as ultimately defined by the sponsor) is rejected, but that necessary change is brought into the project with the full awareness of all concerned and that necessary adjustments to designs, plans, timescales, tests, contracts (etc.) are made with a minimum of wider disruption.
Communications: "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."	■ At an enterprise level, because of the focus on the overall solution, rather than just software, the team is typically broader than simply the people who are developing the software. ■ At an enterprise level, the communications strategy must include a broader set of people such as production operations and support that are important stakeholders in the implementation of the solution.

TABLE 12.2 (*Continued*)

Typical Small Agile Project Values	Typical Enterprise-Level Implementation
Progress Measurement: "Working software is the primary measure of progress."	■ At an enterprise level, success or failure is measured more in terms of delivering *real business value* to the users over simply developing functional software. ■ The primary measure of progress should be whatever the Sponsor defines as value. (Think about user training or process change, for example)

ENTERPRISE-LEVEL OBSTACLES TO OVERCOME

In addition to the challenges previously mentioned, there are a number of obstacles that are commonly associated with an enterprise-level agile implementation.

Collaborative and cross-functional approach

One key challenge is developing the collaborative, cross-functional approach that is required for agile. There are two aspects of this challenge:

1. Agile requires breaking down some organizational barriers that may exist and organizing people into dedicated, collaborative teams. Instead of having a QA department that is separate from the development organization and provided direct oversight of the testing process, you might have a QA organization that provided general functional guidance, but the QA testers would normally work as part of a dedicated team and would not be managed by the QA department on a day-to-day basis.
2. Another obstacle is associated with developing a collaborative partnership relationship between the business organization and the development organization. There is a need to develop a level of trust between these two organizations and a spirit of collaborative partnership rather than an arms-length contracting approach and that can be very difficult to achieve in many organizations that have not been accustomed to working that way.

The challenge to the agile project manager is providing strong leadership to help break down these organizational barriers and implement a more cross-functional management approach.

Organizational commitment

In many companies, agile cannot be implemented without some level of cultural change. Implementing it as a development process only and ignoring the need for organizational change will often result in a very limited level of effectiveness. It might be done that way as a first step to show results quickly; but ultimately, achieving the full benefits of agile will probably require some level of organizational change.

Many people make the mistake of assuming that you have to force the whole company to be more agile in order to implement an agile development process. That is not necessarily the case—a company needs to build its culture around whatever makes sense for the primary business that the company is in. Although becoming more agile is a good thing in most companies, it has to be positioned in the context of the company's overall business strategy. *Just becoming agile should not necessarily become an end in itself*.

The challenge an agile project manager may face is helping to plan how an agile development process should be integrated with the company's primary business environment and then helping to lead whatever enterprise-level transformation is needed. Developing the appropriate strategy can be a very challenging role and might involve making compromises between trying to transform the company to become more agile and adapting an agile development project management approach to fit with the company's existing business environment and culture.

Once the overall strategy is determined, implementing that strategy can also be very challenging because it can involve a significant amount of change management for whatever organizational and cultural changes may be required.

Risk and regulatory constraints

There may also be factors in the company's business environment that impose constraints on how far you can go with an agile implementation that need to be taken into consideration. For example, if a company operates in an environment that requires some level of risk and/or regulatory control, it may be necessary to adapt the agile approach to fit that environment but it's not impossible to do that with the right approach and tools. Also, requirements traceability and design control combined with an effective testing approach are usually very important criteria for developing an acceptable approach for meeting regulatory requirements. An agile project management tool can provide a way of satisfying those constraints—the tool can help demonstrate that the process does indeed provide an acceptable level control in those areas. The challenge for an agile project manager is in determining how to blend the right level of control and agility to provide the right balance to the company.

ENTERPRISE-LEVEL IMPLEMENTATION CONSIDERATIONS

There are also a number of enterprise-level implementation considerations that impact how agile projects are implemented at an enterprise level.

Architectural planning and direction

Enterprise-level solutions are typically more complex than a small standalone software application and upfront planning of the design architecture is typically needed for a number of reasons:

- The solution often will require multiple teams, and some sort of architectural direction is needed to define how the work among the teams should be organized and coordinated.

- Because the solutions are typically much larger and more complex, there is much greater risk associated with having to redesign and/or refactor the solution after the design is in progress because the level of effort required may be much larger.
- The solution will also need to integrate with other enterprise-level software and conform to whatever standards the organization uses to ensure that it is interoperable with other applications. That will typically require some planning and design reviews early in the project.

Dean Leffingwell has very accurately identified the need for what he calls *intentional architecture* at the enterprise level:

> For small, Agile teams who can define, develop, and deliver a product or application that does not require much coordination with other components, products, or systems, the basic Agile methods produce excellent results. But what happens when those teams must coordinate their activities as their components integrate into subsystems, which in turn integrate into larger systems? Moreover, re-factoring of these larger scale systems may not be an option because many hundreds of person-years have been invested and the system is already deployed to tens of thousands of users.
>
> "For these systems, the Agile component teams must operate within the context of an intentional architecture, which typically has two characteristics: (1) it is component-based, each component of which can be developed as independently as possible and yet conform to a set of purposeful, systematically defined interfaces; (2) it aligns with the team's core competencies, physical locations, and distribution (if this is not the case, it is likely that the teams will realign themselves thereto!).[5]

Of course, this doesn't necessarily mean that a big upfront design approach is needed for every project—common sense should be used to determine the level of depth that needs to go into upfront architectural planning to reduce the risks and uncertainties involved. For example, if there is significant uncertainty associated with the architecture that would have a high potential risk on the project if the architectural direction is not addressed and resolved early on, it may then necessitate including a special iteration (sometimes called a *spike*) to investigate and resolve that uncertainty. One method that is frequently used is to define and develop a prototype or *slice* of how the ultimate system will be implemented as a proof of concept. That prototype, or proof of concept, can then be used as a reference model by the teams designing and implementing the rest of the system.

Enterprise-level requirements definition and management

In some cases, there is also a need for more upfront planning of requirements at an enterprise level:

- Architecture and requirements are intimately related, and it's impossible to define architecture without some idea of what the requirements are. If there is a need to define the architecture prior

[5] Ibid., p. 190.

to development to reduce the risk, it will probably be essential to define more of the requirements up front in a typical enterprise-level agile project.

- The requirements can be a lot more complex, and more upfront analysis of the requirements may be needed to determine the most appropriate solution and the optimum architecture as well as understanding any interdependencies and interrelationships among the requirements. A technique called *functional decomposition* is often used to break down requirements into a logical organization. Functional decomposition is also a useful way of understanding how the requirements are aligned with supporting the business objectives of the system.
- There are typically a larger number of stakeholders involved in the development of the requirements. For example, a support group will many times have a key role in determining supportability requirements.

There are some significant challenges associated with planning and managing requirements for solutions at an enterprise level:

> Agile's practice of working on a few stories at a time is a wonderful focusing mechanism for the team. But in larger systems, what drives these stories into existence? Who says these are the right stories? Will the summation of all these stories (now in the thousands) actually meet our customer's end-to-end use-case needs? Does our team's Product Owner have clear visibility into stories others are building? Are they likely to affect us? If so, when? And when developing solution sets (large sets of products that must be deployed together and support end-to-end use cases for the user or customers), how do we know that the stories on the table will actually work together to achieve the final objective? Can building an enterprise application, one story at a time, possibly work? Well, perhaps not exactly that way.[6]

However, Michael Hurst, corporate PMO for Harvard Pilgrim Health Care, suggests one clear benefit:

> The most Agile teams use their ability to code quickly and efficiently as a requirements discovery process and avoid the overhead of formal specifications. This practice can work effectively because a small team can write and rewrite code at a rate faster than many organizations can attempt to determine and codify their customer requirements anyway![7]

The key thing to consider is that this is not an all-or-nothing decision of having no requirements at all or having large and unwieldy requirements documents. Good common sense should always be

[6] Leffingwell, p. 190.
[7] Dr. Michael Hurst, personal e-mail, April 20, 2014.

used to determine how much upfront planning and what level of detailed definition should go into the requirements for any project. If the requirements planning and definition effort is done using electronic tools rather than traditional Word documents, it can significantly accelerate the development effort once the development is started. For example, defining a structure to the requirements and organizing them as epics and user stories in an electronic tool makes it much easier to plan, organize, and track development and testing tasks against those requirements.

Release to production

The process for releasing mission-critical applications to production is another important factor that increases the complexity of large, complex enterprise-level applications. Dr. Hurst provided some comments on that from his experience with Harvard Pilgrim Health Care, which is one of the major case studies used in this book:[8]

- We have found in the management of large programs that once the team build passes the acceptance criteria test cases, it is really just the first step in releasing to a complex production environment.
- Ok, my component works locally on the development environment (functional testing), and it works on the nearest neighbor environment (contiguous testing), but does it work in the end-to-end production environment for full User Acceptance Testing UAT)?
- We have found that Kanban process works better than Scrum by the Release Management team since they really need teams to go through a sequence of environments and connections on their way to the full production environment. These stages are highly susceptible to staffing and other resource constraints (test data, test environment updating) that allows only so many things to be in queue at once, but items in queue can be replaced by other things as ones pass through successfully. Perfect production line Kanban.

The whole process of releasing applications to production in an enterprise environment can require a lot of coordination outside of the agile team, which is a key area of value-added that a project manager can provide. The challenges associated with this are typically referred to as DevOps:[9]

[8] Ibid.
[9] Blog, "DevOps: What It Is, Why It Exists and Why It's Indispensable," posted by Luke Kanies, August 23, 2011, http://readwrite.com/2011/08/23/devops-what-it-is-why-it-exist#awesm=~oBYtsF1U5Vw3bm.

In my experience in operations there's always been a difference in perspective between Dev and Ops, but it's always been more of an impediment than a benefit. The common goal should be getting apps deployed as quickly, safely, and efficiently as possible, but each group instead has a more short-term priority not necessarily related to the results the business is looking for. Lee Thompson (formerly of E*TRADE, now of DTO Solutions) coined the term wall of confusion to describe the apparent inability for development and operations teams to communicate around a common goal, and this wall of confusion is a critical barrier to effective teamwork . . .

Ideally, companies work out a system in which whoever makes the mistake pays the price. The reason Ops is so often scared of Dev deploying is that Dev doesn't really care how secure their apps are, how hard they are to deploy, how hard they are to keep running or how many times you have to restart it, because Ops pays the price for those mistakes, not Dev. In most organizations the mandate of a developer is merely to produce a piece of software that worked on a workstation—if it worked on your workstation and you can't make it work in production, it's Operations' fault if they can't get that to thousands of machines all around the world . . .

Google is a great example in switching up that process. When they deploy new applications, the developers carry the pagers until the stop going off—only when they stop getting outage alerts does operations take over the operational running of a system.

ENTERPRISE-LEVEL MANAGEMENT PRACTICES

There are different levels of management that typically come into play in large, complex enterprise-level projects. Some of these levels of management are shown in Figure 12.1.

There are three primary challenges here:

1. Integrating the efforts of multiple teams from a development perspective
2. Aligning the efforts of all teams with the business objectives of the organization
3. Coordinating with other related efforts outside of the project team and providing tracking and reporting to management

All three of those are very significant challenges, and there is much too learn in all three of those areas. Many aspects of the knowledge of how to make agile work at a team level is relatively mature; however, the knowledge of what needs to be done to scale agile to an enterprise level is far less mature. This is also an area where it becomes essential to figure out how to integrate agile principles and practices with traditional plan-driven principles and practices in the right proportions to fit the situation.

Scrum-of-scrums approach

At a minimum, for large projects that require more than one team, some kind of mechanism is needed to coordinate and synchronize the work among individual teams. The *Scrum-of-Scrums* approach is

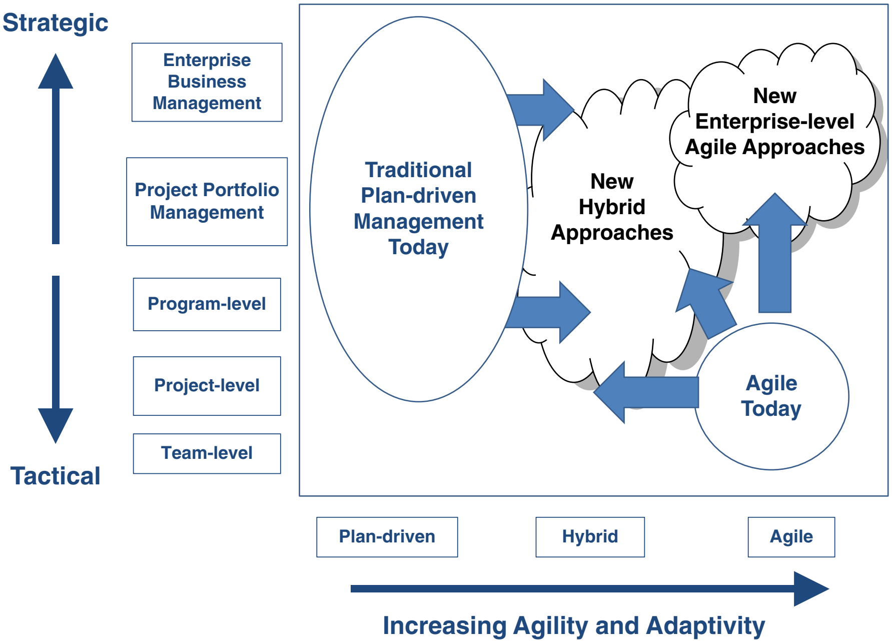
Strategic
Tactical
Enterprise Business Management
Project Portfolio Management
Program-level
Project-level
Team-level
Traditional Plan-driven Management Today
New Hybrid Approaches
New Enterprise-level Agile Approaches
Agile Today
Plan-driven
Hybrid
Agile
Increasing Agility and Adaptivity

FIGURE 12.1 Typical enterprise levels of management

one way to fill that need. When multiple Scrum teams are engaged in a project, each team does its normal, individual, daily standup meeting to discuss items within the scope of that team's own work, and each team sends a representative(s) to the Scrum-of-Scrum meetings to provide a mechanism for coordination and collaboration across different teams.

> The scrum-of-scrums meeting is an important technique in scaling Scrum to large project teams. These meetings allow clusters of teams to discuss their work, focusing especially on areas of overlap and integration . . . Each team would then designate one person to also attend a scrum-of-scrums meeting. The decision of who to send should belong to the team . . .
>
> Being chosen to attend the scrum-of-scrums meeting is not a life sentence. The attendees should change over the course of a typical project. The team should choose its representative based on who will be in the best position to understand and comment on the issues most likely to arise at that time during a project.[10]

The frequency of the meetings should be determined by the teams depending on the nature of the project and the amount of cross-team communication and coordination required; however, the

[10] Mike Cohn, "Advice on Conducting the Scrum of Scrums Meeting," Scrum Alliance, May 7, 2007, http://www.scrumalliance.org/articles/46-advice-on-conducting-the-scrum-of-scrums-meeting.

meetings may not need to be daily. The organization of a Scrum-of-Scrums meeting follows the same general format as a daily Scrum meeting for one of the individual teams. It should be short (typically no more than 15 minutes), and it is focused on the same types of questions as the daily stand-up meeting for individual teams, as explained by Mike Cohn, president of Mountain Goat Software:

> Because the scrum-of-scrums meetings may not be daily and because one person is there representing his or her entire team, these three questions need to be rephrased a bit. I also find it beneficial to add a fourth question:
>
> 1. What has your team done since we last met?
> 2. What will your team do before we meet again?
> 3. Is anything slowing your team down or getting in their way?
> 4. Are you about to put something in another team's way?[11]

The Scrum-of-Scrums approach is a good mechanism for coordinating the work of multiple teams in a project, but it clearly has its limitations:

- It is highly dependent on the maturity level of the individual teams to be totally self-organizing, not only within each individual team but across all of the teams in the project.
- It is typically limited to coordinating the activities of development teams and is not typically used for coordinating other activities outside the scope of the development teams. For that reason, it is not really intended to be an overall project management approach. Overall, project management is more the domain of the product owner(s) than it is the domain of the Scrum team and the people who may participate in the Scrum-of-Scrums meetings.

Figure 12.2 shows how the Scrum-of-Scrums approach works.

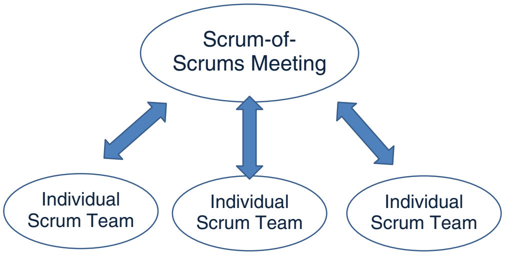

FIGURE 12.2 Scrum-of-Scrums meeting approach

[11] Ibid.

A representative of each individual Scrum team participates in the Scrum-of-Scrums meetings to represent the interests of each Scrum team and coordinate activities:

- The representative may or may not be the Scrum Master.
- The role may be rotated among different team members depending on the need.

Project/program management approach

At an enterprise level, a number of factors contribute to the need for some level of project/ program management in an agile project:

Multiple Distributed Teams

Because of the size of enterprise-level projects, there will frequently be a need for multiple teams. In many cases, those teams may be distributed in different locations and cannot be easily collocated. That will require some level of coordination and communications among and across those teams to ensure that those efforts are well synchronized and consistent with producing the intended business results. The Scrum-of-Scrums approach may be a partial solution to that need, but it is not really intended to provide to be a substitute for project management.

Integration with Higher-level Business Goals

A major source of value-added that a project manager can provide at the enterprise-level is integration with the company's business objectives. At the team level, that role is typically provided by a product owner. However, at an enterprise level, the workload associated with that function might easily justify a project manager. In some cases, the project manager might facilitate a group of product owners or other business stakeholders who act as a steering group for the overall project/program.

Coordination with Other Organizations and Stakeholders

Because large, enterprise-level solutions can have a very broad impact, there is typically a significant need for coordination with other organizations and stakeholders who might be outside of the direct day-to-day project team.

Management of Other Related Activities

At an enterprise level, a number of project-level activities might be outside of the individual project team, such as planning and managing release to production, business process changes, user training, and support requirements.

The role of a project management office (PMO)

A Project Management Office (PMO) in a company typically has several major roles.

Project/Product Portfolio Management

The role of a PMO is primarily to act on behalf of the appropriate business managers to manage the implementation of the company's project/product portfolio management strategy. As a result, the role of the PMO will vary, depending on the level of rigor and control behind the project/product portfolio management approach:

- In a more traditional project/product portfolio management approach, investment decisions to invest in projects will typically require at least a high-level estimate of the costs and schedules of those projects so that they can be intelligently evaluated against other alternatives.
- In a more agile environment, a much more dynamic approach with less financial rigor behind it might be used.

Progress Tracking and Reporting

Once a project has been initiated, at least some minimal form of tracking of progress is needed to determine if the projects are, in fact, actually producing the desired business results and return on investment. The actual level of tracking and control should also be commensurate with the level of rigor in the overall project/product portfolio management approach to determine if the projects/products are really fulfilling their goals:

- In a more traditional environment, there is likely to be a much more rigorous and detailed approach to tracking progress against specific goals and milestones, and the PMO will likely play a significant role in consolidating and validating the reporting information.
- In a more agile approach, the approach may be less rigorous and might put more responsibility on the individual project teams for reporting their own progress. The PMO might play more of a facilitative role and less of a controlling role.

Project Methodology

Another typical role of a PMO is to provide a focal point for sponsoring and managing project methodologies and standards used by the organization:

- In a traditional PMO, the methodologies might be fairly rigidly defined and the PMO might have a responsibility for ensuring compliance with the methodology process requirements, such as completion of required documentation and phase-gate reviews.
- In a more agile PMO, there is likely to be a much higher level of flexibility and adaptivity delegated to the project teams in how the methodology is implemented, and the PMO may play more of a consultative and supporting role to provide training to the project teams as needed to help make them successful.

Project/product portfolio management

Many companies have some form of project/portfolio management approach that is designed to manage the return on investment from their projects. They may also have a PMO structure in place to provide overall management of those projects and to do some form of resource and capacity planning. There's a misconception that an agile development approach is totally inconsistent and incompatible with that kind of PMO management scheme—I don't believe that is the case. There are a range of different approaches that can be adapted to the company's business strategy.

Traditional Financial Portfolio Management Approach

One factor in determining the project/product portfolio management strategy is the level of financial rigor required in making project/product portfolio management decisions. If it is expected that product/product portfolio management decisions will be made on the basis of quantifiable data such as ROI or IRR, it will likely slant this toward a more traditional project/product portfolio management approach. It would be difficult, if not impossible, to do in a true agile approach because there typically just isn't that much information available to support that kind of analysis upfront. In an environment with very high levels of uncertainty, it might be impractical to try to take that kind of approach.

Agile Portfolio Management Approach

A more agile approach is described in the Valpak case study later in this book. Valpak created high-level epics for each of their major business initiatives and used a high-level team of senior executives to plan and prioritize these high-level initiatives similar to the way you might plan and prioritize product backlog items at a project level. The advantages of this approach are that it is very dynamic and can be shifted easily to adapt to different business conditions and priorities.

Lean Startup Approach

An even more agile approach is the *lean startup* approach that was described in the previous section. The lean startup approach is really well-suited for companies that have a very high level of uncertainty associated with their business initiatives and want to take more of an incremental approach to trying out initiatives to see how they work before making a major commitment to any of them.

The key thing to recognize is that it is also not an all or nothing decision to have no management at all or totally oppressive over control with rigorous financial analysis of projects. The alternatives already described provide different levels of control versus agility, depending on the level of uncertainty in the company's business, the company's culture, and other factors.

SUMMARY OF KEY POINTS

Implementation of an agile approach at an enterprise level requires reinterpreting some of the agile values and principles in a very different context. It may also require adapting some of the agile practices to work in a somewhat different environment. The following is a summary of some of the key differences:

1. **Enterprise-Level Agile Challenges**

   There is a need to reinterpret the agile values and principles at an enterprise level in a different context:

   - Implementing agile values and principles at an enterprise level requires a focus on overall *solutions,* not just *software,* and implementation of those solutions might typically require coordination with other activities outside of the direct realm of software development. There must be a plan for how to achieve that coordination that goes outside of the boundaries of the immediate software development team.
   - Many times at an enterprise level there are a number of different stakeholders who have input into the development of a solution. That will require an approach for ensuring that the inputs of those stakeholders are effectively integrated into the project as it progresses.

2. **Enterprise-Level Obstacles to Overcome**

   There are a number of potential obstacles to overcome at an enterprise-level to get an agile approach to work. One of the biggest challenges is to develop a collaborative and cross-functional approach, and that is very critical to successfully achieving enterprise-level agility.

3. **Enterprise-Level Implementation Considerations**

   At an enterprise level,

   - Architectural considerations of how an application interacts with other parts of the architecture can become very significant and may require more upfront planning to ensure that any interdependencies and constraints that must be observed in the architecture are understood and incorporated into the design of the software.
   - The requirements can be much more complex, and more upfront analysis of the requirements may be needed to determine the most appropriate solution and the optimum architecture.
   - The process for releasing projects to production at an enterprise level will typically impose some constraints that must be considered.

4. **Enterprise-Level Management Practices**

   As projects are scaled to an enterprise level, a number of higher-level management considerations beyond the level of individual teams come into play.

There are three primary challenges here:

1. Integrating the efforts of multiple teams from a development perspective
2. Aligning the efforts of all teams with the business objectives of the organization
3. Coordinating with other related efforts outside of the project team and providing tracking and reporting to management

DISCUSSION TOPICS

Enterprise-Level Agile Challenges and Obstacles

1. What do you think is the most significant difference you will encounter in a typical enterprise-level agile project? Why?
2. What do you think is the most important obstacle to overcome in implementing an enterprise-level agile transformation? Why?

Enterprise-Level Management Practices

3. What are the limitations in a Scrum-of-Scrum approach?
4. How is the role of a Project Management Office (PMO) different in an agile environment?
5. What is the lean startup approach? What value does it provide? Where would it be most useful?