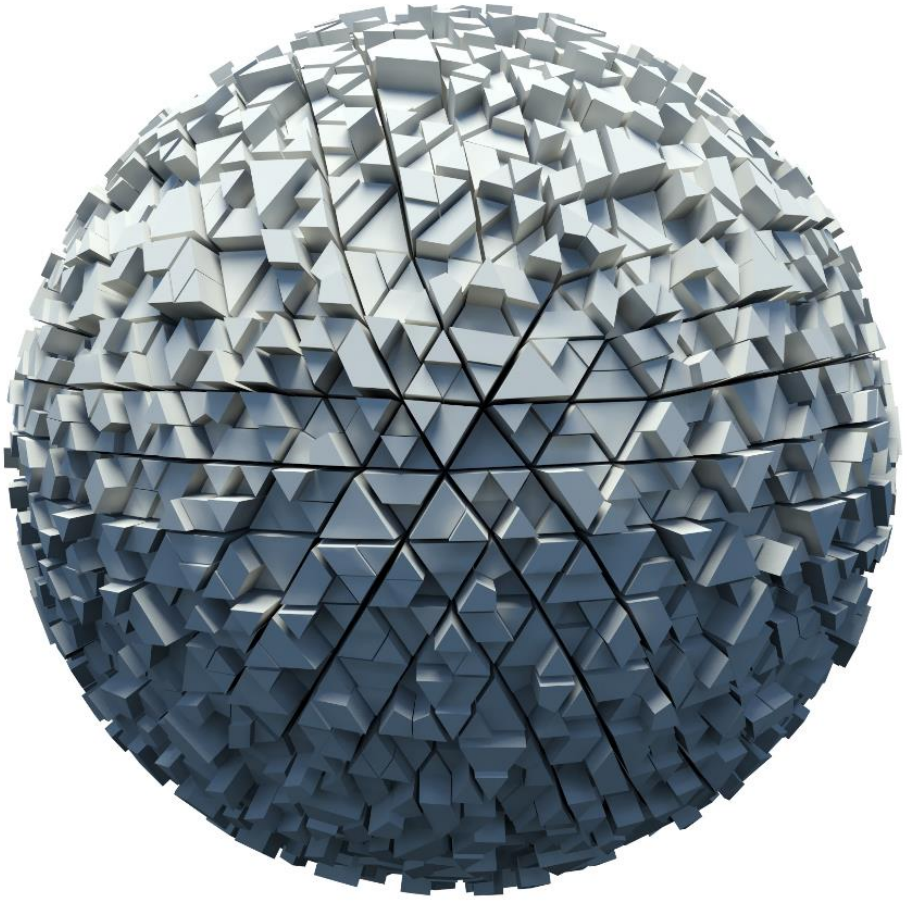
Deloitte.



Fast and frequent value delivery

How financial institutions can harness the power of Agile, DevOps, and cloud to focus on progress, not perfection

Contents

Taking the leap

Agile, DevOps, and cloud are on everyone's mind. But are they just some new buzzwords, or are they here to stay for a while? Financial services seem to be taking the leap toward these new ways of working, but are they really delivering fast and frequent value? More importantly, how do organizations even measure this value?

Small experiments with Agile often deliver promising results. But the same is not always true of broader, organization-wide transformations. These transformations can fail when such considerations as scaling, globally distributed teams, third-party service providers, and legacy infrastructure aren't properly addressed.

The journey so far

The financial services industry is at a tipping point—either disrupt or be disrupted. As Big Tech looks to expand its influence beyond cloud, social, and e-commerce, financial institutions aren't battling Silicon Valley for just talent, but also for customers. A number of financial technology firms (fintechs) are winning over a new breed of clientele—those that demand customized and targeted experiences. One Fortune 500 CEO famously described the situation as an elephant being chased by a few greyhounds. So, how does the elephant retain its power and still achieve the agility of a greyhound?

This is where the Agile manifesto comes in. Released in 2001, the manifesto outlined a set of values and principles for better developing software. Agile methodologies like Scrum have now become quintessential to software development. In financial services, their popularity has exploded in the past couple years. Agile coaches, Scrum masters, and product owners are in demand like never before. Teams are running sprints and conducting ceremonies. But how much of this is delivering fast and frequent value to customers? Unfortunately, not a lot.

"It's not about doing things differently. But actually doing different things."

Agile values and principles are often easy to understand, so what exactly is going wrong? Marketing Agile as "doing things differently" eases the job of advertising it to clients, but ultimately it needs to be effective—and the reality is, with some notable exceptions, Agile adoption is often a mere exercise in retitling people, running sprints, and conducting ceremonies rather than a way of transforming processes. That's why we see so many organizations continue to face the same challenges as with traditional Waterfall approaches.

Overemphasis on adopting tools, practices, and methodologies often comes at the price of real, quantifiable business outcomes.	Force-fitting a scaling framework that has worked elsewhere or blind rollout of a radical structure that doesn't align with the organization's DNA	Adopting Agile in a silo, without wholehearted support of business partners, results in doing the same old things, just in a different way.	Trying to sustain adopted Agile practices without engineering investment (e.g., test automation, CI/CD, monitoring)

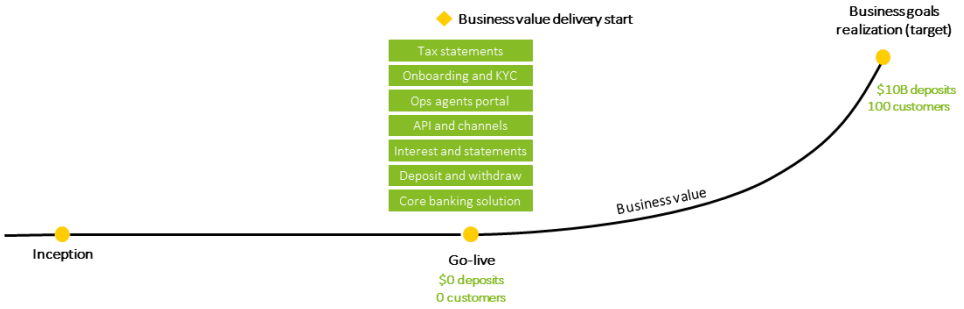
Many teams that have succeeded in their Agile adoption have taken a value-driven approach. They encourage cross-functional collaboration, keep progress transparent, make decision-making efficient, and continuously inspect in order to improve. They make investments in tooling and infrastructure to reduce capital costs, avoid reinventing the wheel, and minimize overall delivery risks.

Agile is about incrementally delivering value, not code. It's about reducing delivery risks. It's about maximizing the work not done.

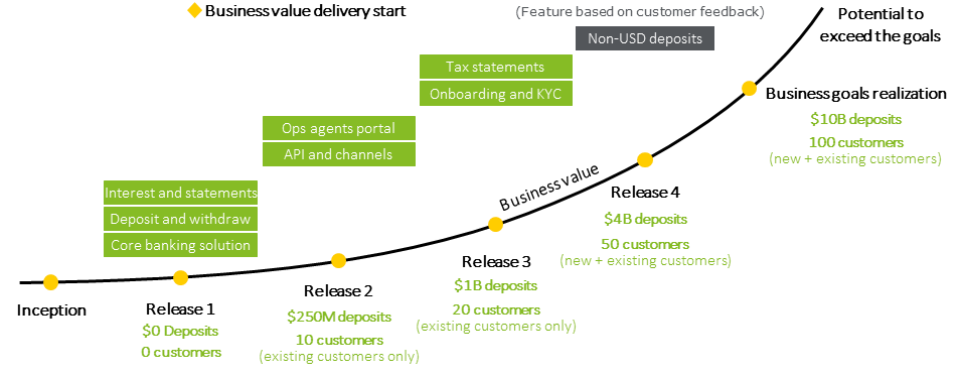
Iterative value delivery: An illustration

Consider a bank that decides to offer a new type of account to its business customers. Conventionally, it would spend a long time gathering requirements, building and testing the solution, and getting operations ready for the go-live. The bank would then start selling the product to prospective customers. It can only hope that its initial assumptions remain true and that it can achieve its original business goals.

Conventional go-live approach



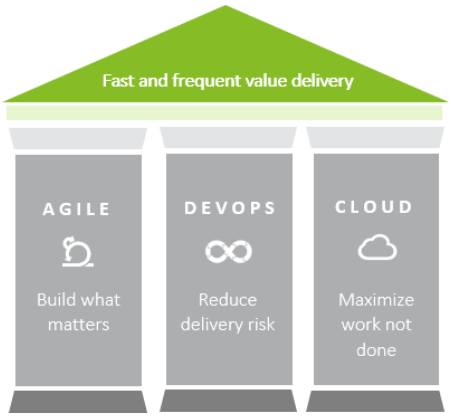
Conventional go-live approach



If the bank were to follow Agile, its first release could be a minimal set of features required to onboard just 10 customers. As an example, these customers could be supported manually, without having to make sophisticated enhancements to the customer support portal. Furthermore, customers could be selected from the existing customer base, which would eliminate the need to build onboarding and "know your customer" (KYC) processes. Such a minimum viable product (MVP) could be released within a few months, allowing the bank to realize business outcomes from the outset. Continuous measurement along the way and feedback from real, paying customers would help to correct the course, as necessary.

Fast and frequent value delivery

Agile practices alone cannot sustain fast and frequent value delivery. They should be accompanied by robust DevOps tooling, which helps reduce delivery risks. When experimentation and innovation are the priority, on-demand infrastructure and services from cloud are often indispensable.

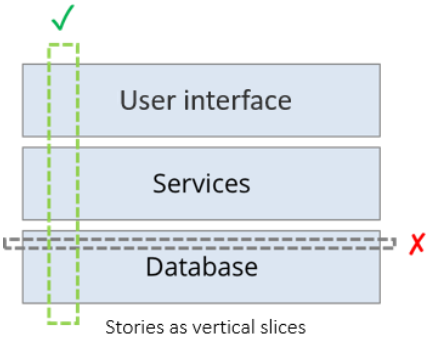


Building what matters with Agile

Frequent and meaningful feedback is at the core of any successful Agile delivery. This can allow the team to react to the changes, learn from experimentation, and eventually build what really matters to the customer.

Batch size reduction: Waterfall consists of a single batch that moves serially across each of the five stages of development—requirements, design, coding, testing, and deployment. This can delay feedback and lead to higher variability. On the other hand, Agile breaks down work into multiple, smaller pieces, which move rapidly through the same five stages, often within a few days. These smaller batches allow frequent feedback and keep variability low.

Writing good stories: For feedback to be meaningful, the incrementally delivered code has to be usable. Therefore, each story must be written from the user's perspective. A good story is a vertical slice of the system, meaning it cuts across user interface, services, and database layers. This is the biggest change for both product owners and engineers. Techniques like story mapping can help.



Self-organizing teams: Finally, individual teams should be largely self-sufficient: They should have all the required skills and freedom to make certain decisions about how to accomplish their work. When teams are dependent on others for tasks such as architecture decisions, code reviews, and production deployments, it can adversely affect the team's throughput.

Reducing delivery risks with DevOps practices

DevOps tooling and practices can provide greater stability, quality, and security to Lean and Agile technology delivery, thereby reducing the delivery risks that come with acceleration. The most important DevOps practices include continuous integration (CI), release strategies, monitoring, and telemetry.

Continuous integration: Because Agile teams write and change code continually, it is impossible to manually verify that such changes don't have any unintended consequences. A robust, continuous integration infrastructure combines automated build and testing to ensure that the codebase is in a releasable state after every check-in. Therefore, continuous integration plays a key role in mitigating risks associated with the flow acceleration, from development to deployment.

Continuous integration practices for fast and frequent value

- Everything is in version control
- Multiple small-sized code commits
- Every check-in triggers build and testing
- Automated tests validate release-ability
- Reject commits that fail testing

Release strategies: Early nonproduction feedback is always helpful. But nothing compares to the feedback received from real users in production, and this is especially important for financial institutions that are building new products. Proposing to go live without months of testing is naturally going to raise some red flags. That's where release strategies like blue-green deployment and canary releases play a critical role. Blue-green introduces redundancy, which allows instant switchbacks in case of failures. Canary releases allow the rollout of new features to a small sample without affecting the broader, more sensitive user base.

Monitoring: Advanced monitoring tools help Agile teams identify problems as they occur rather than waiting for users to report them. As the velocity of production changes increases, capabilities like log aggregation, alert monitoring, environment health monitoring, and statistical abnormality detection are useful. Continuous measurement of business metrics can also provide valuable input into prioritization of features.

Leveraging cloud to maximize the work not done

Infrastructure provisioning and configuration is a common bottleneck for many software development teams. They often spend a lot of time configuring and maintaining multiple environments, including performing tasks like deployments, database administration, patching, backups, logging, and monitoring.

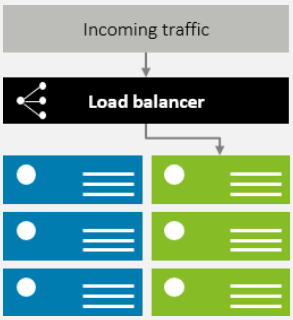
Automation: By leveraging a public, cloud-based infrastructure, teams can start delivering working software right from the first sprint. Furthermore, they don't have to identify all their infrastructure needs up front—rather, they can scale up or down as necessary. Teams can also leverage out-of-the-box automation tools for otherwise time-consuming administrative tasks.

Microservices: Agile teams don't build monoliths: They build smaller, independent, modular services, which collectively deliver the desired functionality. Leveraging public cloud capabilities like serverless compute and container-as-a-service can accelerate development and deployment of microservices and eliminate worries about underlying runtime configurations.

Software as a service: A number of software vendors offer their products as services through public cloud marketplaces, allowing consumption on a pay-as-you-go basis. These offerings can help accelerate time to market, improve scalability and availability, and provide automatic access to the most recent versions of the software, including security patches.

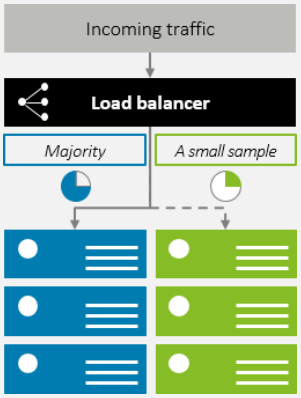
Financial institutions wishing to rapidly deliver new, innovative products should look to a combination of Agile practices, DevOps tooling, and cloud infrastructure.

Blue-green deployment



Green servers receive the new code, while blue remain on the previous stable version. A load balancer then routes all the traffic to green. In case of any issues, all traffic is routed back to blue servers.

Canary release



Only a subset of servers (green) receive the new code. A load balancer then routes the preconfigured sample of users to green. All other users continue to use previous, stable version on blue.

How cloud can enable fast and frequent value

- Low up-front investment in infrastructure and software
- Rejection of commits that fail testing
- Significant reduction in time and cost to implement
- Opportunities to experiment due to reduction of sunk costs
- Capacity freed up to focus on development
- Unlimited disposable environments for testing

Making it real

Financial services institutions often operate in an environment that is fundamentally different from that of technology product companies. Successful adoption of Agile in financial services is contingent upon addressing important differences in the areas listed below.

- **Scaling Agile beyond a team**
  Often, program goals are too complex to be achieved by a single Scrum team, requiring adoption of scaling frameworks to deliver value.
- **Managing globally distributed teams**
  Due to cost and talent constraints, it's not uncommon to have teams split across locations, posing certain challenges to Agile adoption.
- **Working with third-party service providers**
  Many critical applications are built by integrating customized versions of third-party products, resulting in architectural complexity and continued reliance on service providers.
- **Collaborating with legacy, Waterfall teams**
  Interdependent systems often result in Agile product teams having upstream or downstream dependencies on some legacy teams, who may not be as responsive to change or as flexible with testing.
- **Measuring success**
  Monitoring tools, as well as Agile project management software, can track a lot of data points, but the key is to select a few that can reliably tell the tale and allow immediate course-correction.

Scaling Agile beyond a team

While most financial institutions are now implementing Agile at a team level, we occasionally see them struggling when it comes to applying it to larger programs.

Scrum is designed around a small, self-organizing team. But a team of six to 10 individuals isn't always sufficient. In financial services, the projects are often large and require a lot more people to come together in order to achieve the objectives. As the team size grows, however, so does the complexity of communication and the need to manage interdependencies.

Scaling refers to the application of Agile principles to such large teams. However, scaling is complex and should be avoided whenever possible. Instead, ask yourself the following questions:

- Can the program objectives be met by a team of eight to 10 high-performing engineers?
- Can we create a single, collocated team representing all the required skills?
- Can we manage the interdependencies through a low-touch communication between multiple product owners and Scrum masters?

If the answer to any of these is no, you should then consider scaling. Leading Agile scaling frameworks provide a useful set of practices, tools, and patterns that help in solving typical problems. But remember that these are not step-by-step recipes. High-performing organizations customize these frameworks to suit their own organizational context, often with the help of trained coaches and change agents. Blind adoption will do more harm than good.

Case study: Scaling Agile for a 300-person portfolio

Our client, a leading financial institution, wanted to build a new, strategic revenue stream by rapidly introducing next-generation offerings in the market. The initial market research identified an elevated customer experience and real-time analytics to be the key differentiators compared with the clunky and outdated offerings of the incumbents. It was crucial to experiment, fail fast, and fail cheap in order to win in the marketplace. The client therefore decided to build a cloud-native solution; invest in DevOps capabilities; and leverage an Agile operating model across product, engineering, and operations.

Implementation of a scaled Agile model enabled rapid ramp-up of headcount, established top-down traceability from portfolio objectives to individual stories, improved metrics-driven transparency, and enabled new product releases every one to two months.

Organizational structure

We started with engineering teams aligned to different technology platforms, but managing dependencies proved challenging. As the product vision solidified, three main product areas emerged. Product, App Dev engineers, and operations were aligned across these areas to improve efficiency and collaboration. The pods managing the cloud infrastructure were retained as a shared service due to the potential for economies of scale.

Operating model

We adopted Scrum for all the App Dev pods, which operated on a two-week cadence. We also leveraged Lean flow practices to ensure sustainable value delivery. Cloud platform pods couldn't fit their work into two-week time boxes, so they adopted Kanban to focus on flow. They prioritized activities based on App Dev demands marked "due by a sprint."

The client's total commitment led to significant investments, including hiring more than 20 Scrum masters and product owners, essential ingredients for a successful implementation.

Product strategy and OKRs

The product strategy set the objectives for each of the strategic initiatives and identified key results that could be used to measure success at the end of each program increment (PI). These included concrete business outcomes, such as number of customers and average daily volume or value of transactions. The objectives and key results (OKRs) were prioritized by leadership and then used to adjust funding across product areas on a quarterly basis. Following each release, the actual results were measured, root-cause analysis was performed for any deviations, and adjustments were made for the following PI.

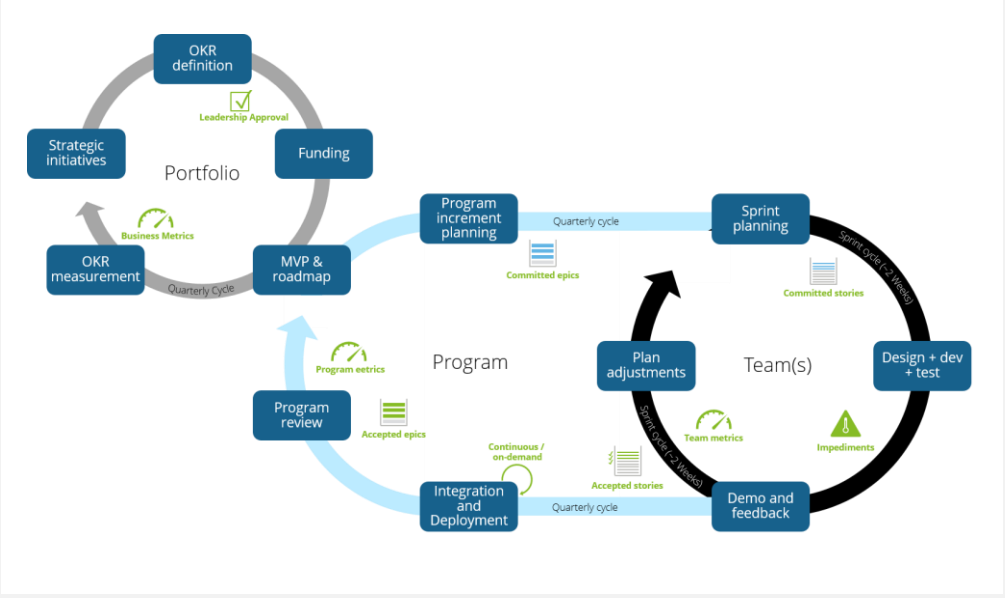
PI planning

We started with a program iteration length of three sprints, but eventually adopted a quarterly cycle made up of six sprints. Conducting PI planning for a globally distributed organization is not easy. Based on the experience of first few PIs, the following adjustments were made to better accommodate the organizational context:

- OKRs and desired features were communicated at least two weeks in advance.
- Architect(s) and tech managers came to an agreement on the high-level solution approach.
- Offline team-specific breakout(s) were facilitated by Scrum masters with their product owners to break down features into stories, which were then sized and sequenced into the sprints.
- On the day of PI planning, the program plan was prepared based on each team's plan, and adjustments were made to factor in business priorities, capacity trade-offs, and dependencies.

Coordination and metrics

Scrum masters facilitated their pod ceremonies, and teams in each product area conducted a scrum of scrums to track dependencies and impediments. At the end of each sprint, the teams provided an integrated system demo to get feedback from senior stakeholders. They also reported on key metrics (for example, percentage of delivery against commitment, average cycle time, and technical debt), which were used to provide coaching feedback.



Engineering excellence

Tracking performance against the flow and quality metrics helped the client focus on the following objectives:

- At least 90% coverage for automated unit testing
- Reserving up to 10% capacity each sprint for addressing outstanding technical debt
- Automation of functional test cases across user interfaces, as well as application programming interfaces (APIs)
- Version-controlled scripts for any infrastructure provisioning and configuration changes

Anti-patterns to watch out for

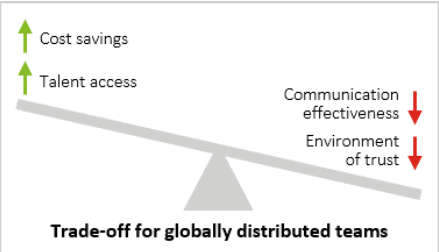
- ☒ Product roadmap dictates deadlines for product features instead of target business outcomes
- ☒ Conventional structures like steering committees and working groups influence decision-making
- ☒ Manipulated red, amber, green (RAG) statuses, rather than real flow metrics, are used to communicate progress
- ☒ Teams have no freedom to choose alternate ways of achieving committed business outcomes
- ☒ Scrum masters spend most of their time managing interdependencies between the teams
- ☒ Engineers feel that meetings are being thrown at every problem
- ☒ Teams struggle to showcase a meaningful integrated demo at the end of each sprint
- ☒ Teams don't dedicate capacity to pay down their technical debt
- ☒ Production deployment is considered an achievement and is often a big event

Managing globally distributed teams

Most Agile delivery models prove effective when collocated, but globally distributed teams are a reality in the financial services industry.

Colocation is ideal because it helps to enable efficient and open communication while creating an environment of trust and mutual respect. However, cost and talent constraints have made globally distributed teams a common phenomenon in the financial services industry, where business units are often split across multiple locations. These teams may have members working on different floors in the same building, in different buildings within the same city, in different cities in the same country, or in different cities around the globe. Such teams have to rely on email or phone as their primary mode of communication rather than face-to-face conversations. Lack of in-person interaction can make distributed teams prone to assumptions and misinterpretations, leading to an environment of uncertainty and doubt.

Globally distributed teams are often split across multiple time zones. This means all key meetings need to be squeezed into common working times, which in certain cases can be less than two to four hours. The limited window can affect the productivity and usefulness of these meetings, as members are often struggling with competing priorities or are forced to stay late or come in early. Furthermore, there is often a psychological mismatch, since people are often more vigilant and alert in the morning and more thoughtful and emotional in the evening.



Trade-off for globally distributed teams

Nevertheless, there are ways to address the challenges posed by distance through strategies such as videoconferencing or quarterly travel for face-to-face meetings. These strategies vary depending on the degree of team distribution, but provide a possible solution to some of the inherent communication and coordination risks (see table).

Investments in communication channels and travel should not be perceived as a burden. Every misunderstanding can mean a little less value delivered.

Strategies for managing globally distributed teams

	Team split within the same city	Teams with more than four hours of time zone overlap	Teams with less than four hours of time zone overlap
Structure the team right	• Collocate all members split across multiple buildings within the same city.	• Structure pods into tightly coupled teams with a common product backlog.	• Structure pods into self-sufficient teams with minimal inter-location dependencies.
Invest in right tools	• Make face-to-face conversations the primary mode of communication.	• Leverage videoconference capabilities for all key meetings. • Rely on phone calls and instant messaging rather than emails whenever possible.	• Conduct all day-to-day Scrum meetings in-person (within a location). • Leverage videoconference capabilities for cross-location meetings. • Rely on phone calls and instant messaging rather than emails whenever possible.
Travel as required	• *Not applicable* – All team members are already collocated.	• Encourage team members to travel to other locations on a monthly or quarterly basis for key meetings and sessions. • Employ rotation-based staffing across locations to build relationships within the team, as well as with business partners.	• Require leads and representatives to attend quarterly PI planning and big room planning sessions in person.

Anti-patterns to watch out for:

- ☒ Teams conduct stand-ups or retrospectives via email or spreadsheets
- ☒ Teams send pictures of whiteboards to each other rather than brainstorming together
- ☒ Scrum master is located in another country
- ☒ Daily Scrum call has been set up for just 45 minutes to accommodate everyone
- ☒ Teams send product screenshots for sprint reviews and demos
- ☒ There is a dedicated sprint for code merging, integration, and bug-fixing
- ☒ PI planning sessions take place behind closed doors at all locations
- ☒ Team members have never met and don't appreciate one another's individual styles or differences

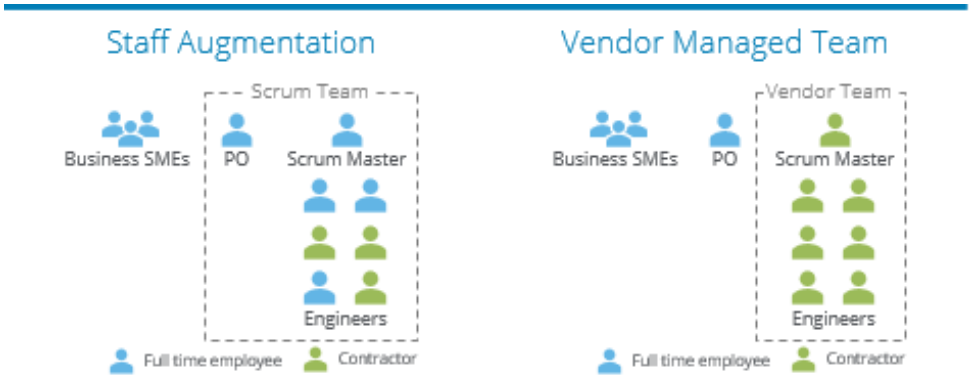
Working with third-party service providers

With Agile, customer collaboration is more important than contract negotiation. But for financial services organizations, which often rely heavily on the product or service vendors, this can be easier said than done.

Build versus buy versus partner has long been a topic of debate. Without exception, application portfolios in financial services companies are a combination of all three. Core banking, payment clearing, and collateral management are common instances in which a vendor product is customized to a client's needs, then integrated with other systems, either by the same or a different vendor. However, Agile teams often find it challenging to work with vendor personnel, who are trained to follow a contractually mandated plan rather than being open to change.

Outsourcing could provide talent and/or cost advantages. Use dedicated vendor-managed teams when cost is a sole driver. But, if building a strategic product, integrate contractors into internally managed teams. Also, choose vendors that have a vested interest in having a long-term partnership. That way, the vendor contracts don't have to be legal fortresses, but can provide the necessary leeway to truly build trust and encourage innovation.

There's no foolproof way of writing an Agile contract, but a little trust and some guardrails make all the difference.



Selection and contracting	• Source all contractors from a single vendor, if possible. • Let the Scrum team interview and select any new members. • Time and materials contracts work best.	• Select a vendor with demonstrated Agile delivery experience. • A contract should specify expected outcomes (not features), a fixed capacity (price) per sprint, and responsibilities of each party.
Team structure	• Treat contractors as equal members of the self-organizing team. • Contractors can play any role within the team, including Scrum master. • Contractors aren't excluded from any training needed by the team.	• A full-time equivalent (FTE) product owner prioritizes work based on the team's capacity. • Management and reporting overhead within the vendor team is kept below 10%. • Vendor is responsible for the right mix of skills based on the PI (~12 weeks) priorities.
Transparency and collaboration	• Use Kanban boards for maintaining status transparency, and collaborate with product owners and stakeholders, like a typical Scrum team.	• All progress during the sprint is transparent to the product owner. • Vendor team's process and cadence is aligned with other teams.

Continuous improvement and innovation	• Analyze reports and metrics to improve team performance. • Recognize accomplishments and efforts; don't penalize people for failures during experimentation.	• Encourage frequent, two-way feedback between PO and the entire team. • Incentivize vendors to be innovative in solving business problems and improving productivity and quality.

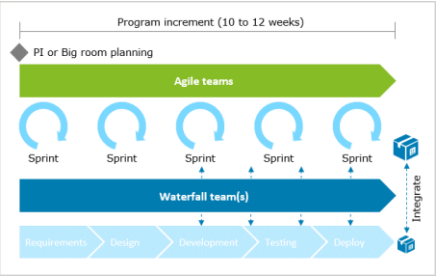
Anti-patterns to watch out for:

- ☒ There is no two-way transparency between vendor and internal teams in terms of status and prioritization
- ☒ Vendor team follows a different sprint cadence than other internal teams it is working with
- ☒ Vendors are forced into a fixed-price contract that mandates delivery of a set of features
- ☒ Contractor personnel are 100% dedicated to preplanned work, leaving no slack

Collaborating with legacy, Waterfall teams

It's hard to find a 100% Agile organization. Most Agile teams have to work with other, less Agile teams, which could lead to conflicts.

Teams adopting Agile, even those building new products, have some dependency on Waterfall teams, which manage systems like customer data, legal, compliance, and books and records. Many times, conflicts arise between Agile and Waterfall teams due to differences in how they plan, budget, deliver, and report. Waterfall teams need advanced commitments and are often resistant to change, partly because they are measured based on their delivery against the plan. Agile teams, on other hand, don't have this hard concept of commitment because they are open to the changes that occur throughout the life cycle of a project. Nevertheless, there are ways to improve the working relationship between these two types of teams.



Up-front planning: A sprint of two weeks is typically too short for Waterfall teams to react. But they can move to a PI schedule, which has a horizon of 10 to 12 weeks. Waterfall teams can join their Agile counterparts for the PI planning and provide input as the plan for next 10 to 12 weeks takes shape. Ultimately, Waterfall and Agile teams can both commit to the PI objectives.

Cadence and alignment: Waterfall teams' PMs and Agile teams' Scrum masters should connect regularly so that they can keep track of dependencies and impediments. Teams should also agree on the cadence for deployments, utilization of shared resources, status reporting, and other important issues. For example, the reporting cycle shouldn't be out of sync with the sprint schedule.

Frequent integration and testing: Code developed by the teams should be integrated as many times as possible, including in the preproduction environments. Agile teams test continuously, and Waterfall teams—which don't typically dedicate testing resources throughout the entire project—can benefit from this feedback mechanism. Furthermore, whenever possible, teams should provide test stubs and mock test data for dependent components that may be under development. Agile teams can also minimize impact of their deployments by adopting techniques like feature toggles or flags that allow modification of the code without modifying the system behavior.

Anti-patterns to watch out for:

- ☒ Scrum teams' output is falling short of their commitment due to unmet expectations from other teams
- ☒ Outputs of Agile and Waterfall teams are not being integrated until the end of the PI
- ☒ Agile teams are being asked to submit mid-sprint status reports

Measuring success

Organizations invest a lot of resources in adopting Agile. Their leaders rightfully expect a way to measure the value added by these practices.

Knowing that we are doing something is not enough. It is equally important to understand if we are doing it right or wrong. Agile cannot thrive without a constant feedback loop, and there is no better feedback than quantifiable business and operational metrics. They provide the necessary assurance that things are on the right track and an immediate alert when they are not. Agile teams should track two broad categories of metrics: business metrics, for measuring the "what," and operational metrics, for measuring the "how."

Business metrics: Business metrics measure the business value delivered by the Agile team. Delivering an on-time, high-quality software release is itself not a goal. It's a means of achieving specific business outcomes like new revenue, reduced expenses, and improved customer satisfaction. Agile teams use metrics like these to measure their ultimate success. But these are lagging indicators—they are available too late to allow a course correction. Therefore, it is important to identify corresponding leading indicators that can be used to predict success or failure. For instance, increasing revenue is the ultimate goal, but it is a lagging indicator. A corresponding leading indicator could be a high count of daily active users, which could in turn predict the revenue. If the count of daily active users were to fall, a corresponding decline in revenue would likely follow. These metrics can't always be readily captured, so it may be necessary to have some stories in the backlog to build metric-monitoring capabilities. It's imperative that product owners prioritize such stories.

Lagging indicators	Leading indicators
These metrics are output-oriented and easy to identify and measure, but they become available too late to predict outcomes or influence improvements. *Examples: Revenue, profit, growth, customer satisfaction*	These metrics are input-oriented, and sometimes hard to identify and measure, but they provide immediate feedback, which helps predict outcomes and bring about desired improvements. *Examples: Daily active users, volume of complaints, referral volume*

Operational metrics: Operational metrics focus on the "how," namely, the flow, throughput, efficiency, and quality of the delivery. These metrics may not be of direct interest to business leadership, but they provide immediate and direct feedback to the team at the end of each sprint, thereby enabling continuous improvement. In fact, these are often leading metrics that can predict the business value much earlier. Most Agile project management tools capture these metrics readily.

Metric	Definition	Potential benefit
Cycle time	Average elapsed time for developing stories in a given interval (e.g., sprint)	Shorter cycle times indicate smaller stories and less multitasking, which leads to higher throughput
Commitment	Actual story points delivered against the sprint planning commitment	Reduce underestimation of dependencies and overcommitment, leading to higher throughput
Technical debt	Estimate of long-term fixes needed to replace tactical implementations	Determine if current technical debt is sustainable or specific investments are needed to address it
Deployment frequency	Frequency of production deployments in a given interval (e.g., sprint)	Decouple deployments from sprints and promote multiple smaller changes to reduce delivery risks
Flow	Review of cumulative flow diagram for a given interval (e.g., sprint)	Identify trending differences in work arrival versus departure rate, queuing, and bottlenecks

Anti-patterns to watch out for:

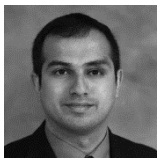
- ☒ Metrics are used to reward and penalize teams rather than to drive improvement
- ☒ Teams are compared based on their operational metrics
- ☒ Leadership is not trained in how to use flow metrics and what questions to ask
- ☒ False metrics like velocity, lines of code, and preproduction defects are used to measure productivity
- ☒ Teams don’t keep their boards up-to-date
- ☒ Kanban teams are forced to report Scrum-specific metrics
- ☒ All teams are required to use the same metrics and target ranges, regardless of their maturity

Concluding the journey

Agile, when paired with DevOps and a cloud infrastructure, could be a game changer for financial services organizations. It has the potential to not only transform business delivery, but also improve productivity and employee satisfaction. However, the journey to Agile adoption can be painful, since old habits die hard. What's more, financial services institutions can't simply "plug and play" the same Agile practices used by technology companies. Rather, they will need to adapt these practices to their unique industry context. But with the right approach, financial services organizations will be able to harness the power of Agile to achieve both fast and frequent value delivery.

Authors

Tushar Daru
Managing director
Deloitte Consulting LLP
tdaru@deloitte.com



Vaibhav Sathe
Senior Manager
Deloitte Consulting LLP
vasathe@deloitte.com

Geetika Aggarwal
Manager
Deloitte Consulting India Pvt. Ltd.
gaggarwal@deloitte.com

We thank **Samarth Shah**, **Bruce Park, and Shea Driver** for their valuable contributions.

Deloitte.