WHITE PAPER

The Agile Boundary

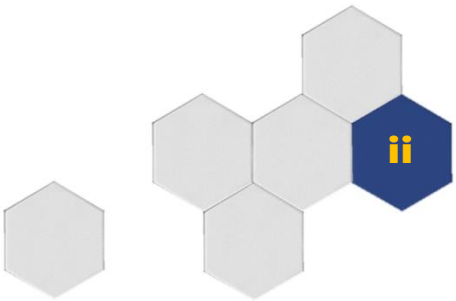
Version 1, August 2021

The Agile boundary demarcates the parts of an organization that are using Agile practices from the parts that are not. Understanding the boundary, its placement, and how it can be strategically evolved over time is essential to effective Agile transformation and improved organizational performance.

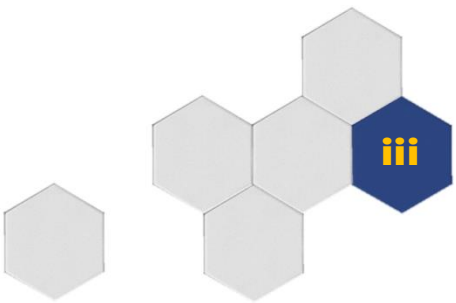
Construx

Copyright

Construx Software Builders, Inc.
10900 NE 8th Street, Suite 1300
Bellevue, WA 98004
U.S.A.



Contents



Introduction

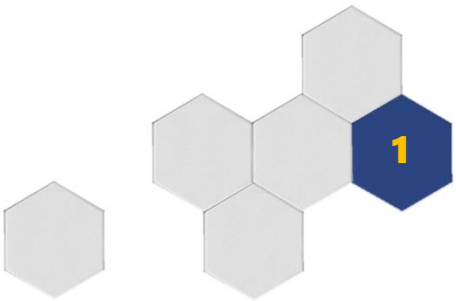
Organizations naturally contain boundaries of different types, such as team membership, functional organization, and physical sites. These boundaries can evolve over time in response to events and actions, such as reorganizations, mergers and acquisitions, and changes to business strategy and practices.

We have found that organizations struggle to connect disparate groups and work effectively across the boundaries among them. "Silos" are frequently criticized, but even silos offer benefits as well as drawbacks.

This white paper addresses one specific organizational boundary that is related to Agile adoption. *The Agile boundary* divides the parts of an organization that are using Agile approaches from the parts of the organization that are not. The Agile boundary is dynamic, and the percentage of the organization inside the Agile boundary typically expands over time as more people and groups adapt to working in Agile ways. However, this expansion is often haphazard and relatively unplanned, which can lead to uneven progress, frustration, lack of perceived benefits, and efforts that "die on the vine."

To help organizations avoid such problems, this white paper addresses the following topics:

- Defining the Agile boundary
- Creating a strategy and roadmap for Agile boundary expansion
- Working across the Agile boundary



Defining the Agile Boundary

Hybrid Development Is Ubiquitous

One school of thought over the last two decades is that an organization must be 100% Agile to thrive. The alternative to Agile is often described as a specious scenario consisting of a 100% sequential, or "waterfall" approach.

In reality, the 100% Agile and 100% sequential approaches are both unrealistic. More than a half-century ago, Winston Royce wrote about the nonviability of a purely sequential approach, saying it is "risky and invites failure."[1] Even in 1970, Royce recognized the need for flow back and forth between adjacent activities, as well as combinations of activities such as system test and requirements that tend to be further apart.

Organizations that attempt to be 100% Agile struggle to work coherently at longer time scales, meet regulatory or safety requirements, or manage hardware and software co-development. Business requires both predictability *and* the ability to respond to change.

A business must be able to match the rate of change in its environment, based on factors such as new technology, changing customer needs, increasing complexity, actions of competitors, etc. These factors vary in intensity across business domains, so the necessary responses by organizations will also vary. If an organization cannot keep pace with change, it inevitably trends towards irrelevance and loses to competitors who are able to keep pace with change.

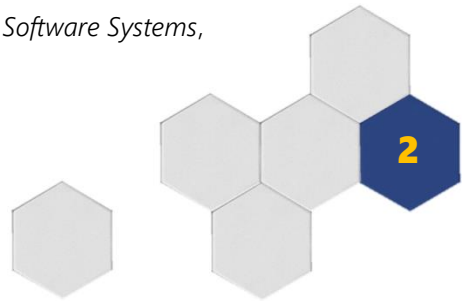
Agile is not an all-or-nothing proposition. A balance between Agile and sequential approaches is often the best solution.

There Is *Always* an Agile Boundary

Once an organization has begun to use Agile, there is *always* an Agile boundary. The Agile boundary is most often contained within the organization itself. But even when an entire organization has adopted Agile, it is very likely that a few of its customers, suppliers, or partners within its ecosystem will not have. So, the questions become where exactly to locate the Agile boundary, how to work across it effectively, and how to evolve it over time.

---

[1] Winston Royce, *Managing the Development of Large Software Systems*, IEEE Wescon, August 1970.



Agile Beginnings

At the start of Agile adoption, the Agile boundary will encompass just a small portion of an organization. Because Agile practices have been motivated and driven primarily by software development over the past 20 years, the first foray into Agile practice will probably be within the software development portion of the organization. At this early stage, it is uncommon to see other business functions inside the Agile boundary.

Agile often enters an organization through a small group of highly motivated champions. In this sense, Agile is an *innovation* within the organization and its champions are from the Innovator and Early Adopter populations described within the discipline of *Diffusion of Innovations*.[2] These Innovators and Early Adopters are dedicated, daring, influential, and intrinsically motivated to see the new Agile practices succeed. The Agile boundary appears soon after they begin their work, defined by their interpersonal and organizational networks, the new vocabulary they use, and the novel concepts and practices they espouse. Innovations are, by nature, different from what existed before. It is these differences that characterize the boundary and the populations on either side.

At the beginning, the Agile boundary tends to serve a protective function, creating a space in which new practices can be learned and practiced in relative isolation from the rest of the organization. This seems sensible, but it results in immediate friction between the software development team inside the Agile boundary and business units and existing business practices that are outside the Agile boundary. Numerous mismatches are possible when:

- Senior management perceives a lack of predictability and transparency
- Product management continues to use large batch sizes for new work
- The organization makes premature promises to customers
- The organization cannot shorten its long release cycles even after software development has shortened development cycles
- Etc.

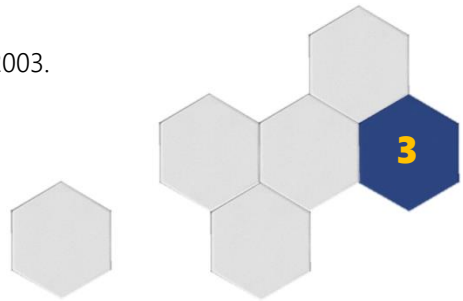
These conflicts are not sustainable, so it is paramount to define the Agile boundary in a way that avoids the collapse of the fledgling Agile effort.

If Agile adoption has already progressed beyond this scope in your organization, extend your search for the current boundary. Use these tactics to find it:

- Follow both formal (org chart) and informal networks, especially those used by influencers within your organization. Ask known Agile practitioners for help in locating the other practitioners in their networks.

---

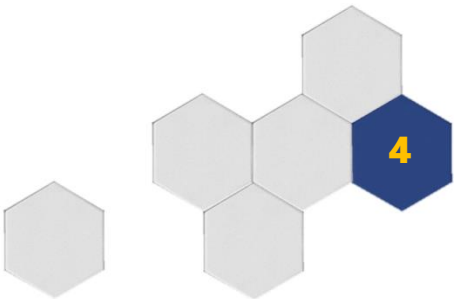
[2] Everett Rogers, *Diffusion of Innovations*, Free Press, 2003.



- Look for social networking channels devoted to Agile, brown-bag sessions, intranet websites, book clubs, and communities of practice.
- Publicize your efforts via push (email, newsletter, etc.) and pull (website) sources. Invite all Agile practitioners into a new forum or community.
- Analyze existing process descriptions, flow diagrams, and value stream maps for functions and groups inside and outside the boundary. These functions and groups may be suppliers, sources, recipients, or consumers.

These techniques and others will help you find the portions of the organization within or about to be within the Agile boundary.

Knowing the extent of the existing Agile boundary is key for creating a strategy for where and when to expand the boundary. Organic adoption is fine at the beginning, but delaying the establishment of consistent practices for Agile adoption is counterproductive.



Creating a Strategy and Roadmap for Agile Boundary Expansion

After determining the location of the current Agile boundary, expand it strategically, which has numerous benefits compared to a hands-off approach. However, stepwise, linear planning is not suitable to Agile boundary expansion.

Organizations Are Complex Adaptive Systems

Organizations and their people form a complex adaptive system that is not deterministic and fully analyzable no matter how much time and effort you devote to the task. Use an approach that seeks to understand what is happening now and what is feasible in the immediate future.

A three-year Gantt chart describing each step for becoming Agile will fail. This does not mean that all planning is useless—it means that different planning is required. Use Agile to create Agile. Agile boundary expansion is a combination of planned and opportunistic work. The key is determining the current state of the organization and feasible next steps.

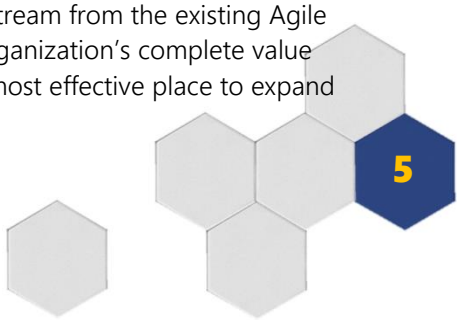
Because Agile usually first appears within the software development organization, the first step is broader Agile adoption in that organization. It then becomes necessary to move the Agile boundary beyond software development.

Software test and QA are an early expansion area if those functions were not part of the initial Agile adoption. Cross-functionality and team autonomy are greatly enhanced by test and QA helping to create releasable increments according to a robust Definition of Done (DoD).

Next, look upstream and downstream from software development. On the upstream side, include product management so that the inputs to the software development teams are sized and prioritized in a way to enable Agile development. On the downstream side, focus on Operations and DevOps to improve integration and release capabilities and accelerate business value delivery. This is a common pattern. What is right for your organization might be different.

Start with the Value Stream

Looking at the activities that are upstream and downstream from the existing Agile boundary is a feasible first step, but analyzing your organization's complete value stream is best for *strategic* boundary expansion. The most effective place to expand

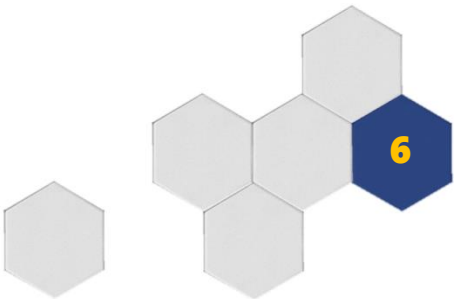


the Agile boundary might be distant from software development. Contiguous expansion outward from software development is not always the best approach.

Another approach is to expand according to product or product line rather than by organizational function: transforming an organization "vertically" rather than "horizontally." Horizontal transformation—function by function across the entire organization—can create islands of new practice within a sea of legacy practice, which can wash away the new islands before change can take hold.

Organizations can couple this vertical form of Agile boundary expansion with environment and architecture migration, especially cloud migration. All the necessary functions shift to an Agile approach at once but in numbers limited by the product(s) chosen for migration. The application environment and architecture shift at the same time.

Capture your strategy as a high-level roadmap to guide your expansion efforts, accepting that changes will be necessary and unforeseen opportunities will arise. Make the roadmap more detailed and specific in the near term (one quarter) and decreasingly detailed out to a year. Opportunities beyond a year are rarely stable enough, even as aspirational goals. The organization is complex, and every action you take will have both unintended and intended side effects. As one charismatic senior executive has put it, "The plan is the plan—until we change the plan."



Working across the Agile Boundary

As described earlier, the location of the Agile boundary will change with time but there will always be a boundary *somewhere*. You must be able to work across the Agile boundary, wherever it happens to be.

Boundaries can contain, exclude, and filter. To work across the Agile boundary, determine what is permitted to flow across it (in one or both directions), how often it flows, and by what channel. Also determine what is prevented from crossing the boundary.

For example, software architecture is often housed in a single organization that serves all the software development teams. The availability and turnaround time needed to support Scrum or another Agile approach might be difficult for the architecture team to support because they are working according to an older model with longer lead times and a sequential approach to architecture definition.

The boundary between architecture and software development requires careful definition to satisfy the needs of both groups. The situation requires a protocol for how to request service, a way to supply the necessary background information and data for the request, a protocol for how the architecture team prioritizes and resources requests, etc. Must the same architect service a particular project or team across time? On which side of the boundary does the architecture specification live? What working agreements are necessary, and what connections across the boundary do they imply?

If these elements are difficult to get right, that might indicate that the portion of the boundary in question is a candidate for being changed sooner rather than later. In this example, pulling some of the architecture team inside the Agile boundary might make the most sense.

Agile software development has several characteristics that help determine the nature of the Agile boundary and how to work across it effectively. These characteristics define what is different about an Agile approach in comparison to a more (but not completely) sequential approach.[3]

---

[3] Steve McConnell, *More Effective Agile: A Roadmap for Software Leaders*, Construx Press, 2019. See especially Chapter 2, *What's Really Different About Agile?* The table shown next is from that chapter.

Sequential Development	Agile Development
Long release cycles	Short release cycles
Most end-to-end development work performed in large batches across long release cycles	Most end-to-end development work performed in small batches within short release cycles
Detailed up-front planning	High-level up-front planning with just-in-time detailed planning
Detailed up-front requirements	High-level up-front requirements with just-in-time detailed requirements
Up-front design	Emergent design
Test at the end, often as separate activity	Continuous, automated testing, integrated into development
Infrequent structured collaboration	Frequent structured collaboration
Overall approach is idealistic, prearranged, and control-oriented	Overall approach is empirical, responsive, and improvement-oriented

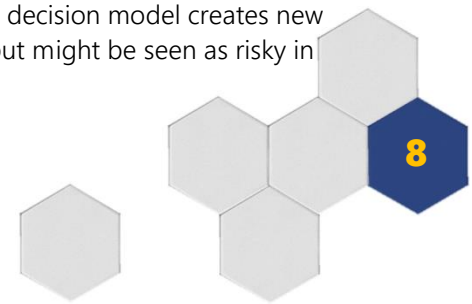
These characteristics have many implications for what information must flow across the Agile boundary, how often, and via what channels.

Short Release Cycles and Small Batches

Short release cycles and small batch sizes inside the Agile boundary mean that the inflow and outflows across the boundary will occur with greater frequency but with smaller scope than before. Existing working agreements should be adapted to the increased frequency. Also, check for a mismatch between the small, more frequent outflows and standing meetings and processes for reviewing progress. Downstream processes and practices for release can be substantially affected by more frequent code deliveries, whether or not those deliveries translate into releases.

High-Level Up-Front Planning with Just-in-Time Detailed Planning

Planning with reduced up-front detail challenges existing norms for investment decisions and milestone reviews. A more incremental decision model creates new opportunities for flexibility in product management but might be seen as risky in



some organizations. To counter these problems, relax or adapt milestone exit criteria and move to a more incremental funding model—there's no need to fully fund a program on fractional data. You can also create a parallel life cycle and funding mechanism for "Agile projects" vs. "traditional projects," but even better is tailoring the funding mechanism to the nature of the system being built.

Even though the "iron triangle" is unrealistic in any nontrivial setting, fixed-bid contracts that dictate scope, date, and investment constrain organizations. *Iron triangle thinking* can create serious issues even when an entire organization has adopted Agile internally. The best course of action is to change contract deliverables from outputs to outcomes and to switch from large monolithic contracts to smaller incremental contracts. Bringing customers inside the Agile boundary might prove challenging but can afford the greatest benefits of all.

High-Level Up-Front Requirements with Just-in-Time Detailed Requirements

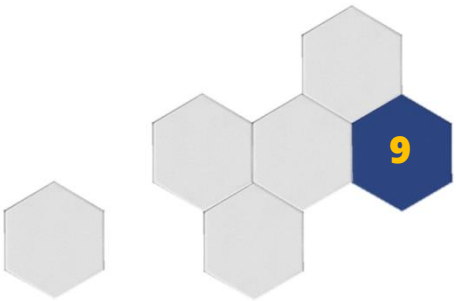
Many teams create too much requirements detail initially. Much of that detail is concentrated in areas where the knowledge is best—such detail is readily available and easy to write. However, capturing this known detail does little, if anything, to reduce project risk. If an organization correctly identifies which requirements can and should be written up front vs. left for later, it will improve time to market and reduce the waste of writing a lot of detail in the hope it will be useful someday.

On nearly every project, some requirements *must* be specified early and others *cannot* be known and specified early without incurring too much risk of later change. Agile reduces speculation, leaving specification of anything that can be postponed until near the time it will be used in construction.

Adapting life cycle milestone exit criteria is a part of addressing requirements work across the Agile boundary, but communication links across the boundary is also important. When product management is outside the Agile boundary, communication between the development team and product management must be adapted to increased frequency and smaller batch size. Without this capability, requirements validation will be slow to nonexistent— the project will suffer from delays, wrong requirements, missed requirements, incorrect assumptions, and substantial rework.

Emergent Design

Emphasis on emergent design rather than detailed, up-front design creates issues across the Agile boundary even though the design function is normally contained within the Agile development teams.



In organizations where a centralized architecture function exists outside the Agile boundary, architects operate in an episodic, service-provider fashion with teams, with a single architect assigned to service a given team. Agile teams will require numerous small interactions rather than a single large interaction at the beginning of the project. This creates resource loading and scheduling challenges for the architecture organization. Architecture organizations will benefit from adopting Kanban to clarify work queues, limit work in progress (WIP), and help prioritize their time. Kanban is also an effective way to communicate status and WIP across the Agile boundary.

A similar issue occurs when a User Experience (UX) team is centralized and located outside the Agile boundary. Practices in Lean UX are becoming more common, but many UX groups operate similar to centralized architecture groups, helping teams in a "service provider" fashion. Again, Kanban adoption makes sense, but in both cases, the question becomes why those functions should be centralized versus made part of the Agile development teams to increase development team autonomy.

A third design challenge occurs when the product management organization is outside the Agile boundary. If Agile development teams must work with product management to validate design decisions internally and with external stakeholders, once again scheduling and resourcing will be issues. Long feedback loops will delay decision making during development and lead to rework. Organizations can establish product councils, management review committees (MRCs), Customer Circles, and similar structures to expedite these reviews and prevent long feedback cycles. The groups can convene on a regular cadence or on demand.

Continuous, Automated Testing, Integrated into Development

Testing in a continuous, automated way that is integrated with the development environment is challenging if the test function is partially or completely located outside the Agile boundary. Communication of what is coming when prevents delays and ensures testing readiness. Agile development teams that reprioritize on a Sprint-by-Sprint basis using Scrum, or even more frequently when using Kanban, can create thrash for the test organization as it plans, resources, and instruments its tests and testing frameworks.

The same sort of transactional service model that challenges architecture and UX is even more challenging for test. For this reason, test is among the most important functions to pull in across the Agile boundary. If embedding test within development is not possible, organizations should increase communications across the boundary. Test personnel will listen in on planning activities and daily stand-ups

done by the development teams and review Kanban boards to see up-to-date information on the state of the work in progress.

Generally, a test organization outside the Agile boundary must step up to work on smaller batches with greater frequency—this is not a simple task because of the lead time that is endemic to test development. Automation helps, especially for regression testing. Shifting some testing to developers can also be effective, using, for example, Acceptance Test-Driven Development (ATDD). Also, organizations realize benefits by improving their use of DevOps and the tool chain to reduce delays in the interval from pull request to production.

Frequent Structured Collaboration and Empirical, Responsive, Improvement-Oriented Environment

The factors in these final two areas depend on cultural foundations—the organization's beliefs, values, and history influence whether issues across the Agile boundary exist in these areas and how large those issues are.

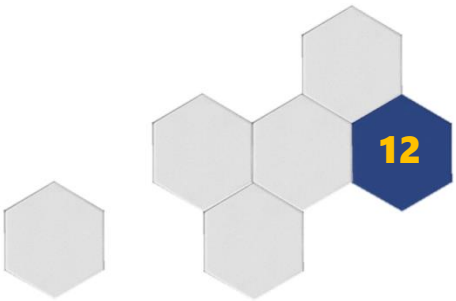
In an organization that values individual expertise and achievement, the Agile boundary provides one more reason for an individual to work in isolation. If the organization reinforces this behavior pattern in annual reviews by rewarding individual contributors over teams, no shift towards increased collaboration and responsiveness will occur.

Organizations that have learned to bury failure or simply blame other teams cannot expect success in transitioning to an improvement-oriented environment without addressing the root cultural issues. The Agile boundary provides a convenient enabler for such blaming and ignoring, because people on each side of the boundary think and act differently from one another based on different perceptions of what is correct and appropriate. Any boundary runs the risk of dividing a population into "them" and "us"—this is not peculiar to the Agile boundary. But this contrast illustrates what might be the most difficult, and most important, aspect of Agile transformation and Agile boundary expansion.[4]

[4] A topic so important that it is addressed in a separate Construx white paper titled *Organizational Transformation*.

Conclusions

- The Agile boundary appears as soon as an organization begins using Agile practices, whether that is driven by grass-roots adoption or by a planned Agile transformation.
- Once it exists, there is always an Agile boundary, even if the entire organization adopts Agile.
- Organizations benefit greatly from taking a strategic approach to expanding the area inside the Agile boundary over time.
- The nature of the differences between traditional sequential development and Agile development characterizes the populations inside and outside the Agile boundary.
- To increase overall effectiveness, organizations can use a combination of bringing more people and functions inside the Agile boundary and enabling the flow of information and work products across the boundary.
- The challenges presented by the Agile boundary relate to Agile's smaller batch sizes, more frequent release cycles, and a just-in-time emphasis in planning, requirements, and design. Deeper cultural elements also play an important role.



Construx

Construx Software is the market leader in software development best practices training and consulting. Construx was founded in 1996 by Steve McConnell, respected author and thought leader on software development best practices. Steve's books *Code Complete*, *Rapid Development*, *More Effective Agile,* and other titles are some of the most accessible books on software development with more than a million copies in print in 20 languages.

Steve's passion for advancing the art and science of software engineering is shared by Construx's team of seasoned consultants. Their depth of knowledge and expertise has helped hundreds of companies solve their software challenges by identifying and adopting practices that have been proven to produce high quality software—faster, and with greater predictability. For more information about Construx's support for software development best practices, contact us at consulting@construx.com, or call us at +1(866) 296-6300.