Low-Code and Agile 2: Like Peanut Butter and Jelly

Accelerating software development with low-code and advanced Agile.

Low-Code and Agile 2: Like Peanut Butter and Jelly

Accelerating software development with low-code and advanced Agile.

By Cliff Berg and Suvajit Gupta

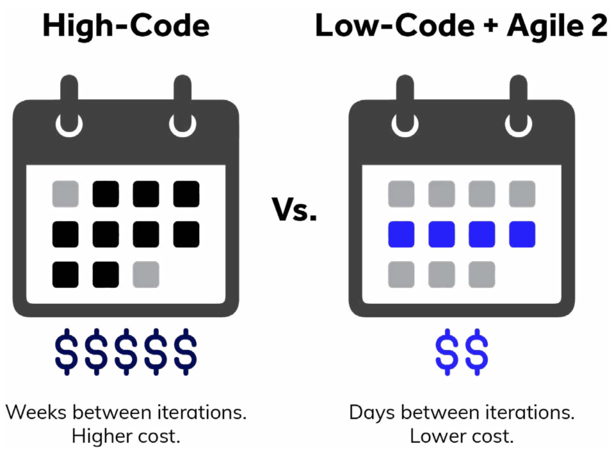
Imagine if a new product feature could be released right away, just days after you'd envisioned it.

For organizations offering digital products accessed over the internet, this dream is often far from reality because of the complex infrastructure requirements for taking working code to an available product. For example, deploying to a cloud such as AWS requires defining a network cluster, load balancers, and multiple "availability zones," often for multiple geographic regions. A single app with one service might have several dozen or more unique requirements, which could require the creation of a hundred infrastructure components. Now multiply this by the number of apps needed, exponentially increasing the level of complexity.

This is where low-code can help. The best enterprise low-code tools pre-define an application's infrastructure architecture, so you only have to deal with the workflows, data, and interfaces—the rest takes care of itself, no coding required. Low-code simplifies and speeds app development, letting you finish and deploy new features in just days. And low-code tools still allow you to use high-code if you need it—one does not preclude the other.

Low-code is a game-changer for business agility and complements enterprise-friendly Agile methods, such as Agile 2. Agile 2 provides a more nuanced, well rounded, and mature set of behavioral norms for agility at large organizations.

With an Agile approach using Scrum teams and programming languages, it commonly takes weeks, and sometimes months, to go from feature definition to feature implementation. But using low-code tools, you can go from prototype to a working version in days, and then to a complete production-ready version within a similar timeframe. This is because all of the underlying deployment complexity is taken care of by the low-code platform. This nicely supports Agile 2's preference for strongly linking product design and development.


High-Code
Low-Code + Agile 2
Vs.
$$$$$
$$
Weeks between iterations. Higher cost.
Days between iterations. Lower cost.

With low-code, teams can be truly agile: development teams can collaborate with product designers and try out features in days. They can release those features and tweak them based on immediate usage telemetry. Equally important, the right enterprise low-code platform gives development teams the agility to evolve applications in order to keep pace with evolutions in business, technology, and strategy. To take advantage of this, one has to think beyond legacy Agile approaches that do not include product design and that embed a multi-week turnaround cycle.

1. Yahoo!Finance, "JetBlue founder's new airline is a 'tech company that happens to fly airplanes,'" accessed September 1, 2021.

Is every company a software company?

What about companies that do not create digital products for the internet?

Most large organizations today conduct their business through a technology platform. That's why Jetblue and Breeze Airline's founder, David Neeleman, said that Breeze is "a tech company that happens to fly airplanes,"[1] and Capital One CEO Richard Fairbank called the company "a technology company that does banking."[2]

Even small businesses increasingly rely on technology, whether it be a restaurant relying on DoorDash for deliveries or a niche retailer getting business through Shopify.

This unavoidable reliance on technology platforms presents a dilemma, because these platforms often require quite a bit of technical expertise for customization—something that not all organizations have. Large organizations with complex products and services that operate together almost always need to create custom digital processes. The standard processes provided by Software-As-a-Service platforms are not customizable enough, and the specific way that a company's processes work is often an important differentiator.

Large banks, for example, typically have hundreds of software development teams and very complex and highly integrated digital services that handle all the different kinds of transactions users perform on a variety of devices. They also have digital integrations to connect with partners like retailers who co-brand their credit instruments and provide other credit and payment services through the banks.

Is this technical complexity sustainable? Can all these organizations operate effectively as tech companies? An opinion piece in USA Today says no, because of difficulty securing sensitive data.[3]

"The sad truth is that many modern banks don't much care about people's private information. The same apparently goes for companies that work with banks. On the same day the Capital One breach was reported, credit rating agency Equifax agreed to pay $700 million to settle a 2017 data breach."[3]

But regardless of industry, adopting technology is no longer a choice: rather, it is a question of survival. Tesla and Amazon have shown that if a company vertically develops its own technologies, it can outcompete others in the market. Both Tesla and Amazon are highly successful business technology platform companies. And "business technology platform" doesn't refer to the underlying infrastructure in this case: it refers to the functions that build or deliver value to customers.

To stay competitive, large companies must invest in their own business technology platform, which usually means investing in software development. They can create that platform using low-code or high-code, or a combination of both.

The question then is how to optimize that investment so that effort goes where it matters. This is particularly important for any product that contains software, because evidence shows that software development is difficult to do well, and there are just not enough skilled developers around.

An alternative is to look at ways to simplify software systems to reduce the number of resources that are required to develop and maintain them and the complexity of most components. This allows developers to focus on the most important differentiators for the company or the things that are mission critical.

2. The Motley Fool, "Capital One Financial Corp (COF) Q1 2019 Earnings Call Transcript," accessed September 1, 2021.
3. USA Today, "Capital One data breach shows why it shouldn't be a tech company that does banking," accessed September 1, 2021.

Enter low-code.

A particularly powerful way to simplify software development and maintenance is to use low-code tools. Low-code platforms enable developers to create a full-featured microservice or web application in very little time. Low-code ushers in the era of quickly crafted software to run almost every aspect of your business.

Low-code technology is visual—you express your intent by drawing and configuring instead of coding. It is declarative instead of imperative; that is, the process is about saying what you want to happen rather than coding all the details to make it happen. This allows business and technical people to communicate and collaborate much more effectively, since they can build things together and don't have to waste time in meetings translating back and forth. Low-code allows applications to be built 10 to 20 times faster than traditional development.[4]

It is important to note that not all low-code platforms are the same. They run the gamut from super simple for departmental use to enterprise-grade low-code. Enterprise low-code systems are highly secure, reliable, and scalable. Developers can focus on the features they are trying to create instead of all of the nuts and bolts, which are already built into low-code platforms. This provides developers more time to focus on innovation, while the low-code platform takes care of mechanics and upgrades, drastically reducing the cost of maintenance.

But does low-code work for everything? What are its "sweet spots"?

Low-code key characteristics.

Low-code tools empower product developers in many ways. Of course, there are vast differences between the various low-code tools that are available, but enterprise-grade low-code tools should offer some or all of the following benefits:

- **Give developers choice where it matters; remove complexity where it is not needed:** Allow developers to focus their efforts on building important functionality without being paralyzed by a million different ways of doing the same thing.
- **Improved developer experience:** An interface that removes technical complexity from the design process, enabling rapid prototyping.
- **Workflow automation:** Uses digital workers and technologies to execute a workflow with as little human intervention as possible. Workflow automation aims to improve efficiency and increase productivity using robotic process automation, artificial intelligence, business process management, decision rules, and case management.
- **Harness data:** Access, integrate, and leverage data across the organization from any source, including legacy systems, without expensive migration or requiring complex database programming. Should include integration and connectors to enterprise systems such as Salesforce, AWS, etc.
- **Enterprise-grade security:** The platform needs to be certified so you know it's hardened, then the application tools need to present the security of data and interfaces in a point-and-click way that business users can understand and collaborate on.
- **Extensibility through integration:** Enable applications to link across legacy enterprise systems and with third-party systems. This will allow the platform to support future growth.
- **Provide data governance and compliance:** Provide tools to enable organizations to monitor, audit, and control their data.
- **Mobile ready:** Applications should work across all platforms and devices, on and offline, with no additional development needed.
- **Integrated DevOps:** Fast and fluid continuous integration/continuous deployment, which supports development team collaboration, ease of testing, and real-time performance monitoring.

These benefits are compelling. You might wonder what must change in order to use low-code tools. For example, how does this impact your Agile teams?

4. Forrester, "The Total Economic Impact of Appian," June 2021: https://appian.com/resources/resource-center/analyst-reports/forrester-total-economic-impact-appian.html

What about Agile?

Most organizations that build software today use an Agile approach. The goal of Agile methods is to create and release new product features rapidly, enabling an organization to adjust course quickly if a feature is not well received or if new opportunities arise. In other words, to have business agility with respect to the digital business platform.

One problem is that today's software systems are extremely complex. For example, large banks might have thousands of microservices, each an independent program, and all interconnected and serving customers through a wide range of applications or inter-business connection points. These microservices provide great scalability, enabling online businesses to reach tens of millions of customers or more with high reliability and availability, allowing online business to be essentially nonstop, without maintenance outages.

That scalability and reliability comes at the cost of increased complexity, however, and increased complexity requires more development time. The time required to redesign and change something in development varies by increasing degree with its complexity. More time required to make a change means less agility. Today it is common for teams to envision a new feature in days, but it might take weeks for the feature to be developed using programming languages. And it might take even longer—perhaps months—to integrate those changes into the product suite, delaying when the feature can be released to users.

Another problem is that Agile does not always work in complex, real-world situations. It turns out that there is an "old" Agile and a "new" Agile. In the words of Jeff Patton, author of the best selling Agile book *User Story Mapping*, "When people say Agile today, they mean something different today than they did in 2001. . . it's come to mean something else."

Indeed, many Agile authors of the past ten years have been saying very different things from what the first crop of Agile authors were saying. The old Agile by and large did not answer questions such as how Agile ideas could work in a large and complex organization. It prescribed one-size-fits-all approaches, as well.

Today's more nuanced and enterprise-ready version of Agile is well described by Agile 2: a sophisticated set of behavioral norms that are needed for an organization to have agility.

New Agile thinking, as described by Agile 2, brings back nuance.

The original Agile ideas were very simplistic and extreme, yet extremes usually do not work well in a complex organization: what tends to work well is an intelligent blend of approaches. For example, early Agile had a blanket insistence on self-organizing teams. In contrast, Agile 2 treats self-organizing teams as a worthy goal, but introduces well-established leadership models, including transformational leadership. Effective leadership is needed to ensure that teams coordinate their work, and leadership is also needed to help teams progress toward being more self-organizing.

Early Agile approaches also overlooked many essential things. For example, Agile frameworks such as Scrum and eXtreme Programming were (and still are) light on product design. As a result, many in the product design community feel that the Agile movement displaced product design.[5] More recent interpretations of Agile, notably Agile 2,[6] re-emphasize the importance of product design and encourage organizations to empower product designers and product developers to co-create in an ongoing manner. For example, Agile 2 suggests considering a "dual-track" approach whereby designers and developers switch back and forth between collaborating and working independently.

Learn about Agile 2.

- https://futurecio.tech/podchats-for-futurecio-how-agile-2-re-aligns-agile-for-the-new-normal/
- https://www.agile2academy.com/about-agile-2
- https://agile2.net/more-resources/agile-2-in-a-nutshell/
- https://vimeo.com/625542974

5. https://www.youtube.com/watch?v=q2dRk3hokEw
6. https://agile2.net/

Some key characteristics of Agile 2.

Leadership: make it part of organization design.
Senior leadership ensures that the right kinds of leadership are in place when and where they are needed, as issues come and go; leadership style needs to be part of ongoing organization design.

Freedom of choice.
Individuals have choices about how they do their own work, teams have choices about how they work together, and product leaders have choices about how to build and deliver their product.

Early Agilists championed certain methods, such as Extreme Programming and sitting in a team room. Agile 2 promotes allowing people to decide—within reason—how they work best. This flexibility promotes a feeling of autonomy, which is important for motivation and also supports neurodiversity, as different people work best in different ways.

The right balance of collaboration and focus.
Both collaboration and the ability to focus without distraction are important. Ensure that people have opportunities for uninterrupted, focused work in addition to collaboration with their teammates.

Make data usable.
The Agile movement had the unintended consequence of removing data architecture from the product development cycle. The result has been that data lakes are often unusable for business intelligence and machine learning. Agile 2 brings data back, in an agile manner, and makes data a first-class stakeholder.

End-to-end focus.
Teams need to consider the entire product rather than just their own work. A product feature is not complete until it has been integration tested.

Integrate product design and product development.
Agile frameworks like XP and Scrum recommend a 2–4 week period during which development teams focus on a fixed set of features to implement. Agile 2 instead advocates having an ongoing collaboration between product design and development, using techniques such as "dual-track design."

In addition, Agile frameworks such as Scrum and XP treat product development as a "feature mill" driven by a single product owner, and do not even mention the important activity of product design. Agile 2 elevates product design to an essential activity.

How low-code supports Agile 2.

- Low-code approaches tend to reduce team size, relieving some of the pressure for uniformity in how everyone works.
- More advanced low-code tools offer DevOps collaboration features, which make it easy for dispersed development teams to effectively work together.
- Provide enterprise guidelines that recommend using low-code where it makes sense, but do not mandate low-code for everything. Discuss which cases low-code is best for.

Early Agile also displaced the role of the data architect. As a result, Agile teams often create software that sends data to a data lake, but that data is unusable by machine learning teams because it is not sufficiently defined or modeled. Agile 2 brings data back into the scope of Agile processes so you can make use of your data lake.

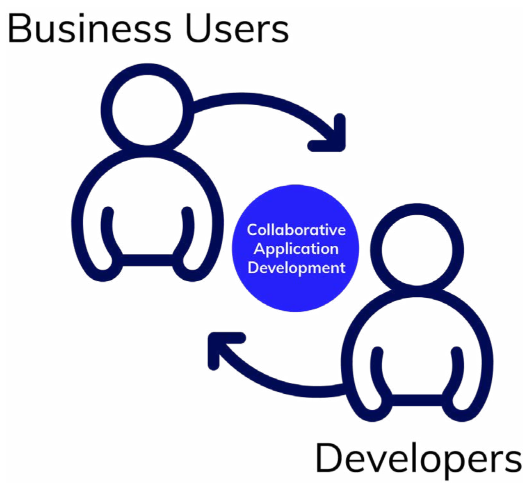
Shifting to Agile 2 is not a wholesale replacement of Agile. It is a matter of adding nuance and new ideas to the original Agile approaches. For example, Agile 2 tells us to consider having product design teams and link them to our Scrum or Kanban teams, if we have those. Yet, Agile 2 is not prescriptive: it is a set of ideas, not a set of rules, roles, or workflows. You can add Agile 2 ideas incrementally, when they are applicable to your situation. Also, Agile 2 is compatible with Agile process frameworks such as SAFe and DA—Agile 2 provides behavioral norms and approaches that enable things to work so much better across the board, no matter what your baseline processes are.

Low-code brings business, product design, and development together.

Agile 2 recommends that we tightly integrate product design and development, and suggests using dual-track teams to achieve that. But with low-code one can go further: low-code effectively matches the end-to-end development turnaround time to the time it takes to envision a new feature. That means that designers and developers can have a very collaborative envision and development workflow.

The visual design aspect of low-code also enables more robust collaboration between designers and developers, thereby speeding up the overall process of going from design to development. The same is true of business stakeholders: they can participate along with product designers and developers.

Since low-code takes care of much of the technical complexity of software development, programmers are working in a more creative way, and they need less "heads down" time focusing on details. That means that they have less need for quiet and isolation—working in an open and collaborative setting is more feasible.



Low-code helps save your data lake.

Agile 2 tells us to model data in an agile and ongoing manner. Low-code provides a powerful approach for doing that. Some low-code tools enable developers to define data in an declarative manner, which creates built-in documentation that is critical to leveraging historical data for machine learning and business intelligence.

A few low-code tools use a data modeling approach in which data object relationships are explicit. This approach is highly resilient, letting you make changes later, and also optimizes performance. Thus, instead of ending up with a jumbled pile of data objects in a data lake, one can trace the relationships between data objects and their meanings are documented in the low-code tools.

Some low-code tools also have built-in connectors to many kinds of data sources (including Salesforce), ensuring that all your enterprise data can be managed together.

Learn about low-code.

- Low-Code Guide
- Low-Code Buyers Guide
- https://appian.com

Case study: A bank uses low-code and advanced Agile practices to deliver rapid innovation.

Acme Bank is a pseudonym for a North American banking and financial services firm with +500 retail locations and between 5,000 to 10,000 employees.

According to their SVP of IT Solution Delivery, "Our organization deploys new functionality on either a monthly or sometimes weekly basis across multiple business lines. With low-code, small teams using Agile practices can rapidly deliver incremental improvements."

“We would rather fail in a small implementation that could be easily fixed in a few days than fail in a large one, which could have a significant impact on the business.”

— A banking SVP of IT Solution Delivery

Acme Bank attributes delivery success to three key attributes, all enabled by the use of a low-code platform and advanced agile practices:

Attribute #1: Agile teams.
“Our delivery teams are structured like a special forces team: they’re small (3–4 people, on average), self-contained, and cross-trained. They are built for speed, but they are also non-hierarchical: the most junior member of the team will review the coding of the most senior. This allows the team to learn, and deploy quickly.”

Attribute #2: An end-to-end perspective on capabilities.
“Our delivery teams are closely partnered with product owners, who report within business groups. They are responsible for understanding not just the existing systems supporting a business group, but the business capabilities required for that business group to deliver on its strategy. This context helps the teams proactively identify new capabilities needed. But also, because their teams understand the connection between systems and the underlying business capabilities they support, they are able to rapidly identify interdependencies between systems. This helps them quickly understand the full scope of new use cases and clarify user stories . . . giving them more time to develop solutions.”

Attribute #3: Data first.
“We used to start a project by focusing on understanding the desired workflow. That got us into trouble as later we would realize we missed a key reporting requirement or integration. Now we have a ‘data first’ approach: we start with understanding key data requirements necessary to support the desired business outcome.”

Impact.

Acme’s SVP of IT Solution Delivery continued, “The productivity of our low-code delivery teams came into sharp focus during the early stages of the pandemic, when both our low-code delivery teams and traditional delivery teams worked on similar projects under significant time pressure. In the end, our low-code delivery teams were able to deliver twice the functionality with half the total bodies.”

Summary.

Low-code makes Agile more effective and strongly enables evolutions in Agile, such as Agile 2’s emphasis on product design and including developers in product feature definition.

Using low-code strengthens collaboration between business and IT, which in turn accelerates innovation by shortening the design and implementation cycle, allowing for rapid iteration. This inevitably improves the product and enhances customer experience.

Because low-code simplifies the task of building applications, it makes technology more accessible to a broad range of organizations. This enables you to focus your most skilled technical resources on the activities that add the most value.

Low-code platforms are also a productivity booster for developers. And—coupled with Agile 2 approaches to data—low-code enhances data management, making data more accessible for business intelligence, machine learning, and other purposes.

Crucially, low-code platforms are now used to build complex business-critical applications—they are not just for simple things. That means that you can apply low-code approaches broadly, and not just for special cases.

Agile 2 approaches are much more “enterprise-ready” than the overly simplistic Agile ideas that predominated in the early days of the Agile movement. Low-code can supercharge agility by making everything simpler and faster and with more reliability built-in.

And that’s why low-code and Agile 2 go together—like peanut butter and jelly.

Authors.

Cliff Berg

Cliff is Managing Partner of Agile 2 Academy (agile2academy.com). He has helped with more than ten Agile or DevOps transformations. At Agile 2 Academy he has been applying organizational culture models to identify and remove cultural obstacles to the use of Agile and DevOps methods. Cliff assembled the team of 15 who founded Agile 2 (agile2.net), and is a co-author of Agile 2: The Next Iteration of Agile (Wiley, 2021). Other books of his include High-Assurance Design (2006), and Sun Microsystems' first book on "enterprise scale" Java (1998). Previously, Cliff was co-founder and CTO of a software startup that grew to 200 people. Cliff has degrees in operations research, nuclear engineering, and physics from Cornell University. He has been a nuclear engineer, an electrical engineer, written compilers, and created a variety of tools for software engineering.

Suvajit Gupta

Suvajit Gupta is the Executive Vice President of Engineering at Appian, where he oversees product management, development, and quality assurance of the Appian product. Suvajit has more than three decades of software development experience, and was Vice President of Development at Eloqua — a marketing SaaS provider that was acquired by Oracle. He received his master's degree in Computer and Systems Engineering from Rensselaer Polytechnic Institute and an undergraduate in Electronics and Communications Engineering from IIT Kharagpur in India.

Learn more about Appian at appian.com
Learn more about Agile 2 at agile2.net