mendix

Executive Brief:

How Low-Code Supports Agile Software Development

An overview of what to look for in low-code platforms, to facilitate an agile app development approach.

Agile development is a term used to describe an approach to building software solutions in a fluid and evolving way, rather than being fixed from the outset on delivering to a detailed, rigid specification. It's a project management methodology that focuses on short development cycles of one to four weeks, called 'sprints.' The idea is that at the end of each sprint, progress in the form of working software can be demonstrated to stakeholders for feedback. This helps to ensure that development work is aligned to requirements, and ultimately to business objectives. Small problems shouldn't become big problems. The risk of failure is much reduced and the probability of delivering good outcomes to both users and the business is greatly increased.

> Developing transformative applications at speed requires rapid experimentation, frequent iteration, and close collaboration between business and IT.

This 'Executive Brief' explains how organizations equipped with the right low-code platform can enhance the success of every agile development; whether a quick prototype or a complex project to build multi-channel experiences integrating with core back-office systems.

Agile: A Brief Overview

A guiding light on the agile methodology is the often-cited Manifesto for Agile Software Development, published in 2001. Seventeen leading software developers defined 12 principles for agile software development, declaring that from their own experiences in developing software, they had learned to value four overarching rules for success, which we might interpret as follows.

Individuals and interactions over processes and tools

Rules and practices need to be kept to a minimum, especially when compared to traditional waterfall-style development processes, and should be designed to be adaptable to all kinds of circumstances. The real focus must be on empowering developers of all kinds to collaborate, both with each other and with the business, to make decisions quickly and adapt to changing needs.

Working software over comprehensive documentation

While documentation of delivered software is certainly important, creating it should be seen as secondary and must not hinder the energy, the flow and focus of developers. In any case, Agile preaches an iterative and evolving approach, so getting the software to meet users evolving requirements is primary, rather than creating detailed documentation which will anyway require continual editing to marry with the continual evolution of the software.

Customer collaboration over contract negotiation

A contract may well set out the customer's (or user's) perceived requirement. But the key to delivering a successful development for the customer is not to continually reference the contracted requirement, but to instead encourage active participation and to harness the customer's domain knowledge to help ensure the final delivered solution delivers on its intended objectives. Effective collaboration requires continual communication, sharing of ideas and feedback.

Responding to change over following a plan

Change should be expected and every application development should support an iterative approach: continuous rather than one-off planning, continuous testing, and continuous integration. There needs to be a willingness to consider course alterations to meet the evolving objective and to know earlier of the need for change, rather than later, if time and resources are to have been deployed effectively.

How Low-Code Platforms can Support Agile Development

Most likely, your development teams practice Agile in some shape or form. But while Agile stresses collaboration and means for facilitating it, the methodology alone isn't sufficient.

Business-IT collaboration

Business stakeholders and their users do not usually understand application software code; a language only followed by programmers. The lack of a common language can limit cross-party collaboration, inhibit the ideation process and make it hard to exchange feedback. This can frustrate the ambitions of Agile and undermine results. But there's a solution. The right low-code platform can contribute significantly to outcomes by enabling participants to overcome communication barriers and collaborate more easily.

- Social collaboration

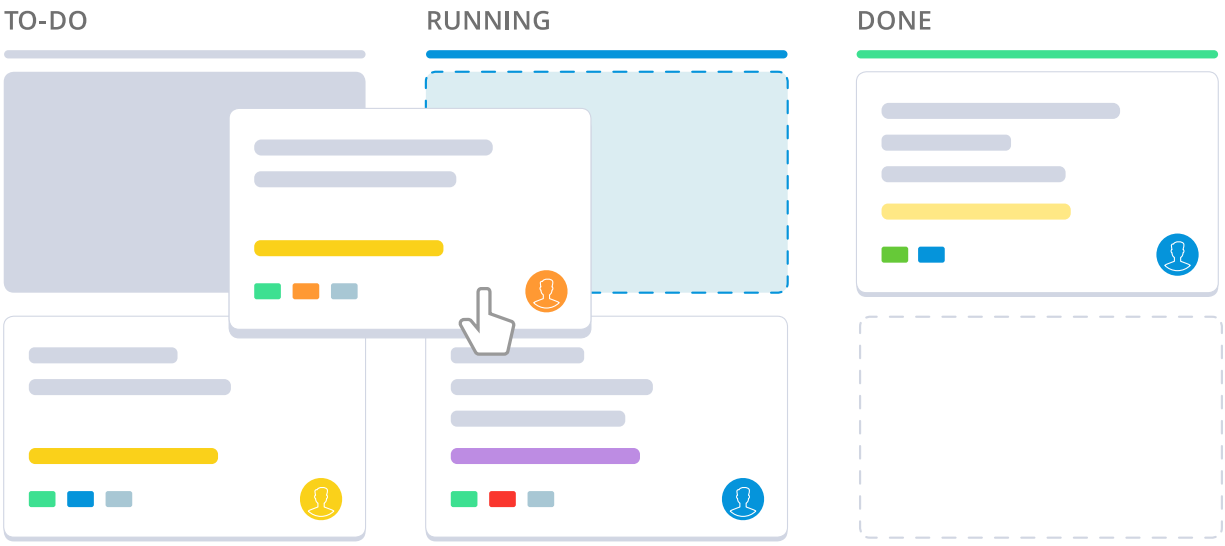
Social collaboration tools create a shared space for your entire team to communicate and stay up-to-date with the project. In the context of design thinking, the team can share and collaborate on artifacts (e.g. personas and journey maps), as well as solicit feedback through polls.

- Instant app sharing

Collaboration is further enhanced with the ability to preview and share live working apps instantly. This enables end users to see, and react to, the app early and often in the process, encouraging continuous feedback and rapid iteration.

Agile project management capabilities

It's advisable to look for a low-code platform with built-in agile project management capabilities. While it may be possible to integrate with third-party tools, developer-centric tools are often quite complex to use, limiting participation of business users.



- Project management portal

An easy-to-use project portal makes it easy for the entire team to create user stories, collaborate, and communicate throughout the agile development process. (Open platform API's support your existing project management tools, where preferred).

- Shared project space

Create a shared project space for your entire team and define individual roles (e.g. product owner, scrum master, developer, user).

- User stories

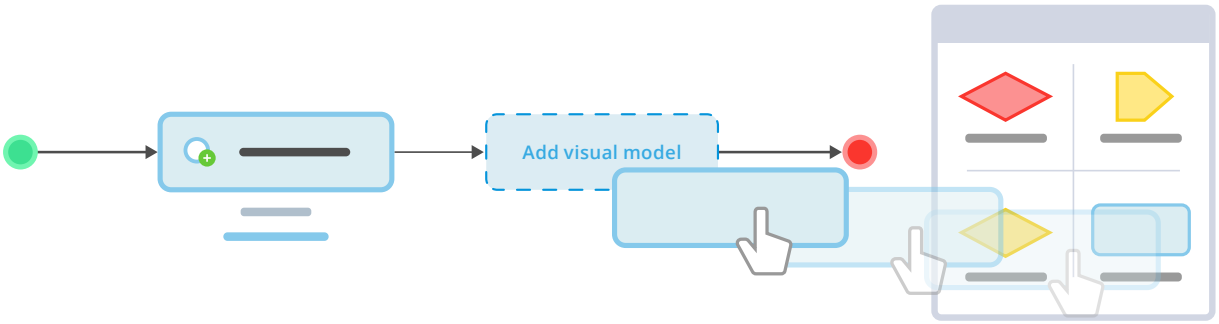
Designed for easy use by business users, these capabilities facilitate collaboration with business stakeholders to capture, refine and prioritize user requirements, plan sprints and future releases; and manage your backlog.

- Traceability through full app lifecycle

Having defined a user story, attach the story to specific parts of the development, note development progress, version controls, updates, revisions and test cases through to production.

Visual low-code development

Low-code platforms present a visual, rather than code-based development environment. Using visual models, development team members and business users can easily engage to create and review functionality, give feedback, validate assumptions, and identify improvements to evolve the application. People can exchange ideas easily, work creatively and experiment together.

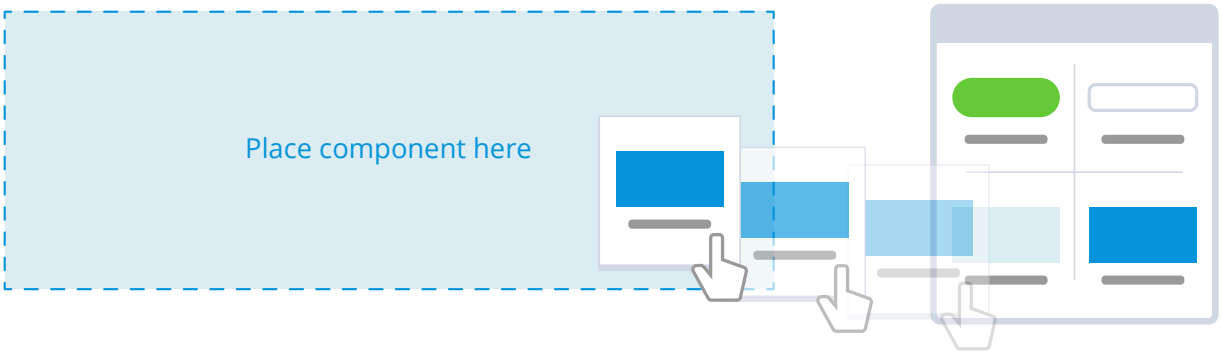


- Common visual language

Because they're easily understood by the entire team, visual models facilitate ongoing collaboration between developers and the business. At any point, even early in the process, they can sit together to discuss and review functionality, gather feedback, validate assumptions, and identify improvements.

- Rapid iteration

Because development is so fast, and apps can be previewed instantly, low-code platforms enable rapid iteration. Developers can make changes in real time based on user feedback, iterating continuously toward the desired results.



- Citizen developer friendly

Visual development 'drag and drop' environment allows business-oriented developers with limited or no coding skills to contribute their domain expertise to the app building process, while more technical developers deliver the more complex logic.

- Reusable components

The ability to capture app components as you create them, and save them to your own repository, provides a library of ready-to-use elements for speeding future application builds.

Automated testing

A low-code platform for agile environments should include an automated testing suite, offering a comprehensive set of tools for creating, refactoring and automating reusable end-to-end tests based upon the user stories of your apps.

- Tied to user stories

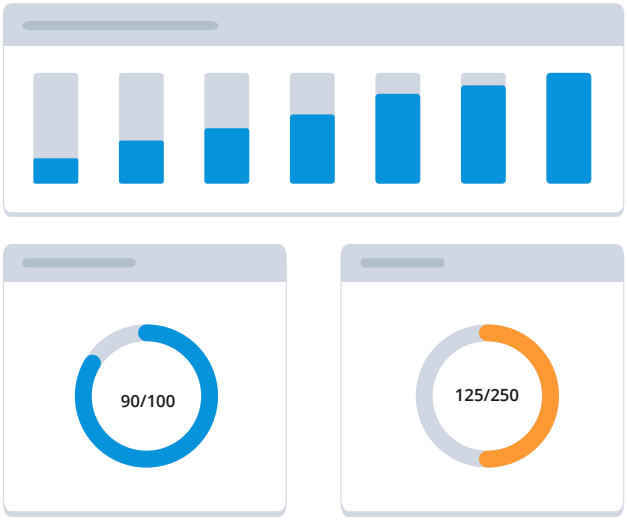
Allow for each story to be defined, coded and tested automatically as part of the development lifecycle.

- Range of tools

Easily integrated with 3rd party testing tools, including for UI testing.

- Continuous testing

It should be possible to use automated testing and quality monitoring at any stage of the application development process, rather than be fixed to specific stages.



Staged deployment

Today's IT organizations need to implement DevOps best practices to bring together development, testing, operations and line of business stakeholders to foster continuous integration, application monitoring and delivery of their app portfolios.

- Continuous integration

Integrate updates into a shared repository, automate build and test to find and resolve problems quickly.

- CI/CD tooling

Leverage your own continuous integration/ continuous development tools via an open API.

Cloud deployment

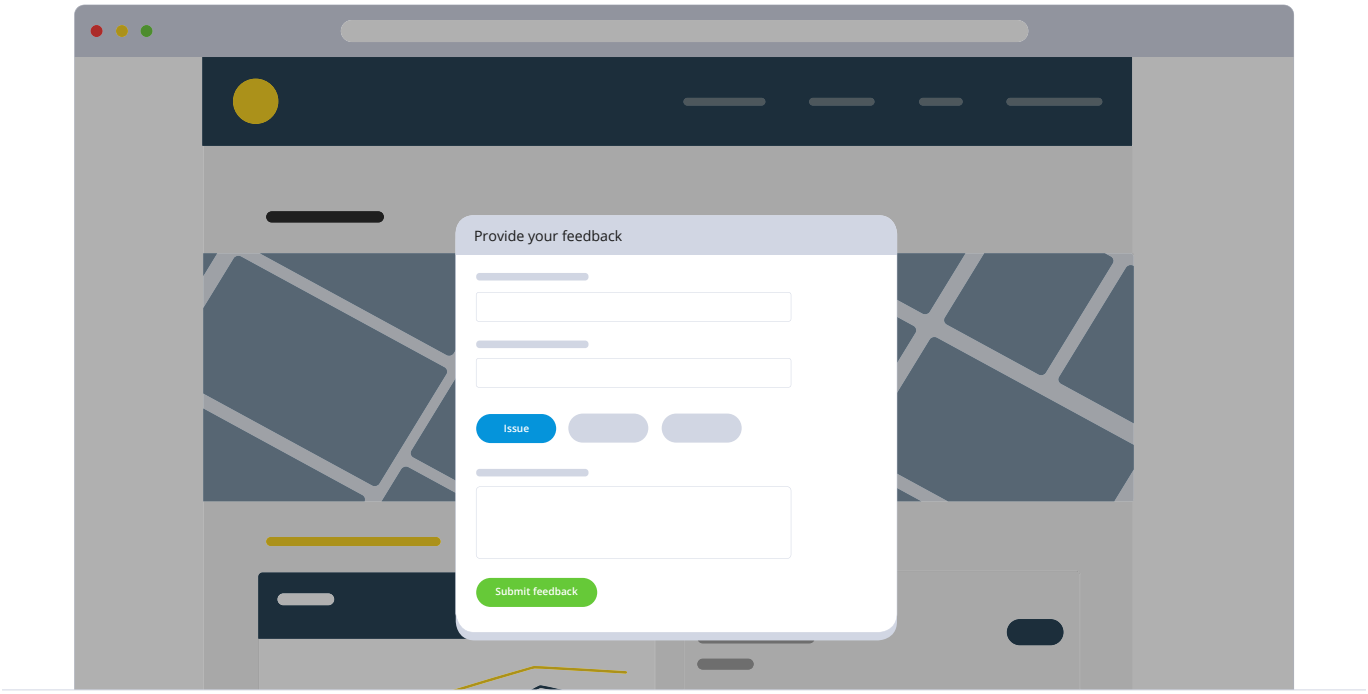
The ability to deploy into any Cloud of your choice is an essential element of a low-code platform for agile environments.

- Cloud-native environment

The platform should be Cloud-native, stateless and support containerization. Cloud portability is essential to enabling easy setup and automated deployment of apps to target environments for Test, Acceptance and Production (as part of the CI/CD process) and once in production, allows changes to be made rapidly.

User feedback loop

Taking input from business users as part of the dev cycle and feeding that back into the user story so forming an iterative loop.



- Restart the process

Having received user feedback to app elements, and defined which will contribute to story progressions and the next sprint, you can connect feedback precisely to the element of the app requiring further iteration.

- Embedded feedback

Widgets enable users to provide instant feedback directly within an application. Feedback is categorized as ideas, bugs, or enhancements, and includes commentary from the user along with metadata captured automatically by the platform.

Conclusion

It is widely accepted that if used correctly, agile development can contribute to positive outcomes in application software development projects. Indeed, research from the Standish Group shows that, particularly for medium and large projects, the failure rate for agile projects is half that of waterfall projects.

However, while collaboration between stakeholders, users and IT is at the core of Agile, business teams and developers tend to think and speak different languages. Low-code platforms overcome this barrier using visual models for much of the development work, enabling the parties involved to view work in progress, share ideas and feedback. The best platforms provide a comprehensive, integrated set of tools for the entire app lifecycle, from ideation and development through deployment and operation. Via low-code approaches, applications can be developed via an iterative process meeting users' evolving needs and organizational demands for digital innovation, to return higher levels of user acceptance and improved return on investment.

To learn more about high-productivity / low-code platforms and which is right for your organization, download the following analyst reports:

Gartner 2017 Magic Quadrant for High Productivity Application Platform as a Service

The Forrester Wave™: Low-Code Development Platforms For AD&D Pros, Q4 2017