How No-Code Enhances the Software Development Lifecycle

How no-code accelerates development, increases collaboration, reduces costs, and shortens the bridge between business challenge and digital solution.

unqork

unqork

Contents

The software development life cycle (SDLC) represents a complex, lengthy, and expensive series of challenges for the enterprise. It would be one thing if all the requisite investments and frustrations guaranteed a robust ROI. Unfortunately, that's not the case.

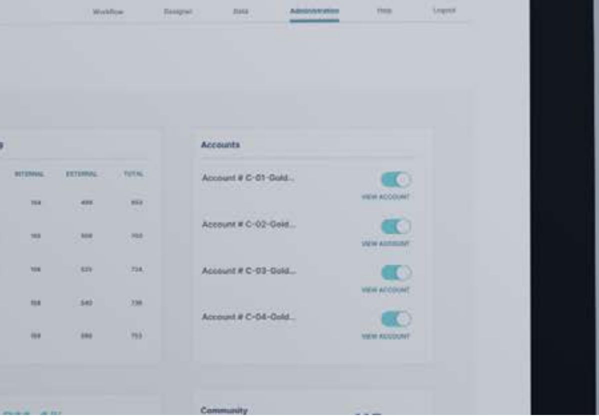
A 2020 report from the Project Management Institute, for example, found that among organizations with "highly mature" project management practices, 23% of IT projects failed to meet project goals, 33% of projects failed to remain in budget, 37% failed to be delivered on time, and 11% were considered complete failures. These numbers are even more dismal across the board for "low-maturity" organizations, which boast a frightening 21% rate of complete project failure.

Building and maintaining custom enterprise software is hard—and, it's getting harder: Following decades of steady improvements thanks to innovative development tools and methodologies, the past decade has seen a sharp uptick in average development times. Why? Traditional code-based development methods are simply not up to the task of keeping up with the complexity and expansiveness of today's modern enterprise software.

That's why a whole new approach is needed.

No-code represents a new development paradigm that accelerates every step in the process, while subsequently improving quality and reducing costs. Enterprise no-code isn't a new tool, it's a complete development platform that comes with all the tools needed to build and maintain enterprise software—and, as the name implies, it completely removes the need to write any code, so organizations can focus all their development efforts on business logic and creating a great user experience.

In this eBook, we will explore how no-code transforms the entire software development lifecycle, from initial planning to ongoing maintenance.



What Is No-Code?

Before we jump into the advantages of no-code in the development process, let's explore a little of what no-code is in finer detail.

No-code is a category of cloud-computing services that empower enterprises to develop, run, and manage applications on a single unified system. As the name implies, no-code also eliminates the need to write any code—indeed, it completely removes the presence of an editable codebase from the development process. That doesn't mean there's not any code anywhere in the system—no-code platforms simply provide an intuitive visual layer between code and creator. Let's take a deeper look:

HOW DOES IT WORK?

When you are building an application with code, what you're doing is reproducing a set of commands over and over again. The commands happen in different ways in different parts of your program, but they are the same commands. What a no-code platform does is repackage these commands in a graphical form, allowing you to configure and manipulate them visually. The platform then executes those commands as if they were written in code.

By stringing together such commands, you can build your program without having to see any of the code or write any of it yourself.

The application is configured visually from start to finish, and it runs entirely from the platform after it's deployed. Changes are made by simply logging in and reconfiguring the visual interface.

WHY IS IT HELPFUL?

Removing the need to write, edit, and debug lines of code speeds up the time to market, improves quality, and lowers the costs of initial builds and ongoing software maintenance.

WHO SHOULD USE IT?

Let's look at this one in a little more detail, and break it down by:

- **Company size:** No-code platforms can be used by companies of all sizes, but the companies that will benefit the most are the ones spending an outsized portion of their IT budgets on complex custom applications. The larger the company, the more it stands to gain from streamlining its application development process.
- **Industry:** Companies across all industries are using no-code platforms. Early adopters were primarily in industries like Financial Services and Insurance. But increasingly, organizations across all categories that spend a significant portion of their budgets on custom software projects are adopting the no-code approach.
- **Roles:** No-code platforms can be used by anyone that understands basic logic and conditional statements. This means everyone from classically trained engineers to business analysts, and anyone in between.

A Brief History of Software Development Productivity

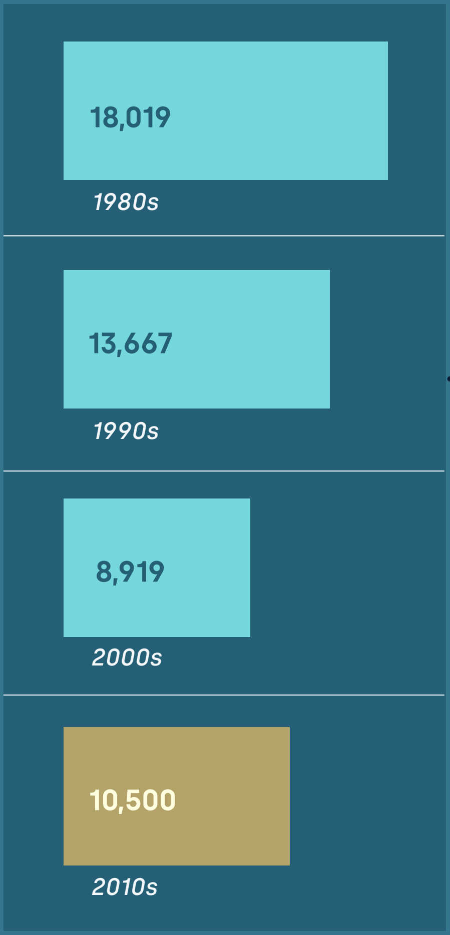
For decades, the standard in software development has been based on the idea of building faster.

This isn't a new challenge. In fact, many years ago, the problem was even worse, as many of the stopgap solutions we rely on today didn't even exist yet! Let's take a quick look at how building applications has evolved over the last several decades:

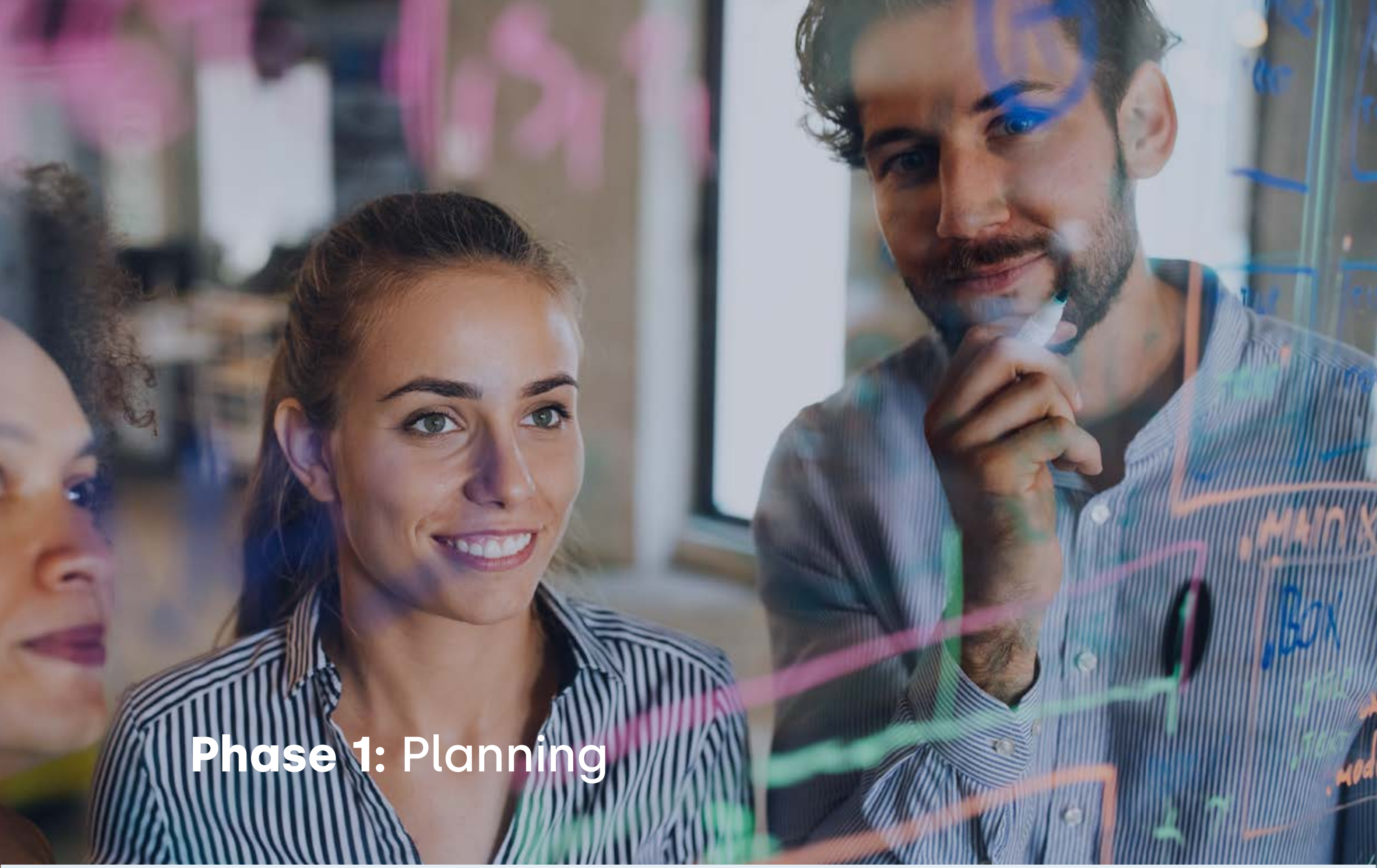
- **The 1980s** were a difficult period for software development. COBOL was the dominant language, but it was really difficult to use. Many projects failed to get off the ground or, worse, were released but malfunctioned. That's why many refer to this period as "The Software Crisis."
- **Then the 1990s** came along and things got a bit better. COBOL continued to dominate, but methodology innovations like Rapid Application Development (RAD) and higher-level (and more user-friendly) programming languages like Java started to gain traction in the enterprise and building software got more efficient and easier. Applications became more useful and the time spent creating them decreased.
- **Next came the 2000s**, which saw Java surpass COBOL as the dominant language. Innovations like frameworks (e.g., Spring) and Integrated Development Environments (IDEs) along with low-code platforms like Appian, Mendix, and Outsystems all helped developers become more productive.
- **Then the 2010s** came along, higher-level languages like Python started to gain adoption, and low-code platforms and frameworks became more advanced. Methodologies like Agile started to permeate enterprise development projects. And despite these advancements, the average time to complete a typical software project was **10,500 hours**, a 20% loss in productivity.

So, why this recent downtick in productivity? There's no doubt software got more complex in the 2010s, but it got more complex in previous periods as well. What's different is that the most recent increases in complexity have not—yet—been matched with a new set of development technologies that are sufficiently able to address these challenges, and as a result, budgets are exploding, backlogs are growing, and projects failing to meet requirements and timelines.

HOURS REQUIRED FOR TYPICAL ENTERPRISE APPLICATION



Source: QSM Software Development Database, 2019



Phase 1: Planning

THE NO-CODE DIFFERENCE

Accidental Misalignments → Purposeful Collaboration

The Planning phase involves studying competitors, aligning on the project's high-level specifications, and beginning the process of gathering requirements. This is also the stage where crucial misalignments on execution and expectations can take shape and make an impact deep into the process.

Even with a friendly and collaborative working environment, it's all-too-easy for miscommunications between the IT and business teams to occur.

Using traditional development methodologies, IT teams are responsible for "translating" the business team's vision into code, which leaves the door open to *mistranslations*.

Enterprises can no longer invest the months—or sometimes even years—into building a technology project, only to realize that it doesn't match up with what was needed.

No code-platforms enable business teams to efficiently and collaboratively create applications that are focused on the needs of the organization from start to finish.

Phase 2: Analysis

THE NO-CODE DIFFERENCE

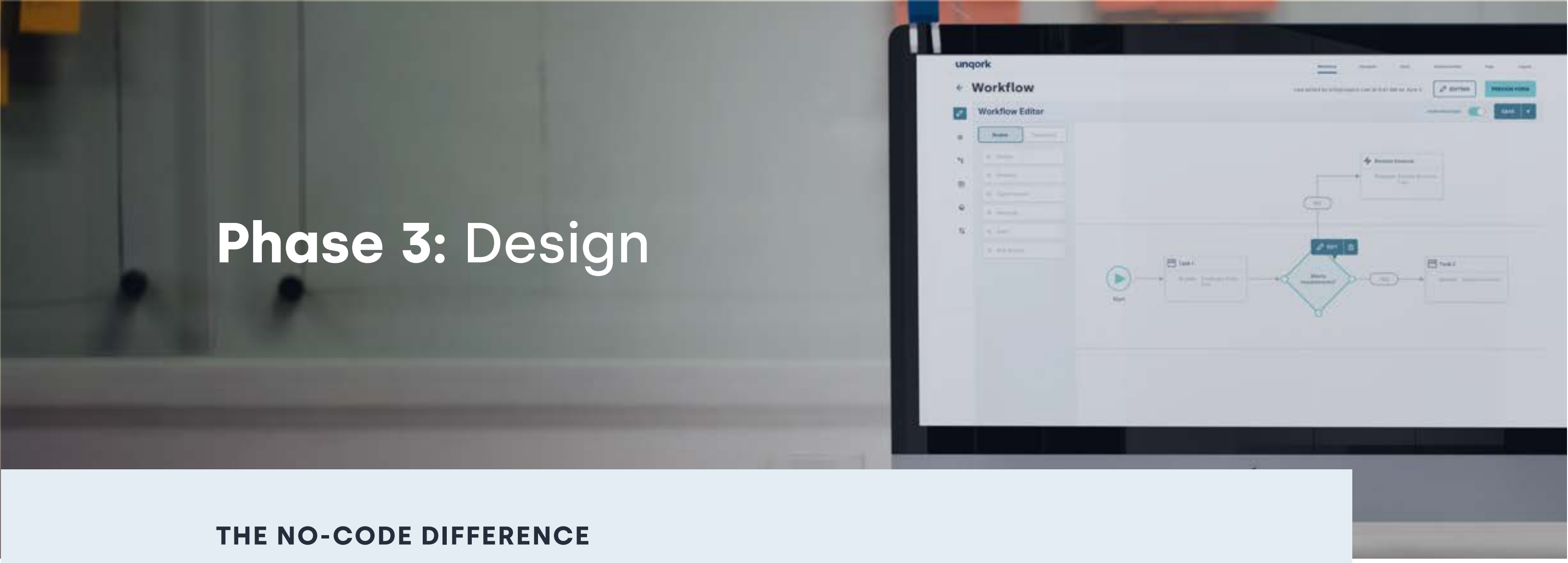
Ad-Hoc Decisioning → Ready-to-Go Infrastructure

In the Analysis stage of the SDLC, teams dive deeper into requirements-gathering and start anticipating potential technical obstacles that lie between them and the completed project. The resulting deliverable is usually a software requirement specification document, or “spec.”

The sheer number of decisions that need to be made just to set the groundwork for an application is never clearer than in this stage. It’s at this point that teams have to decide on the servers they’ll use, the languages they’ll code in, and the cloud provider they’ll work with, etc. These decisions also need to incorporate security concerns, which require enhancements and proper documentation throughout the build and implementation. There are always hidden costs in either time or money that come from an ad-hoc Analysis process.

No-code eliminates this initial decision-making step entirely by providing a fully built environment where the best technologies have already been selected. And while we’re at it, top-notch security features should already be built in for audits and compliance.

When teams have access to a robust off-the-shelf infrastructure, organizations can refocus from building an application to building business value.



Phase 3: Design

THE NO-CODE DIFFERENCE

A Game of Telephone → A Clear Mission

Between ideation and development, a product must pass through countless approval steps. Soliciting input on the project's optimal architecture and design approach can support organizational buy-in, but this is also the phase where many projects fall off the rails.

By the time it even gets to the project management phase, the application can morph so much that the needs of the client or end-user are completely lost. In other words, it's like an enterprise-wide game of telephone, where the original purpose of the project gets completely muddled along the way.

But that's not the only problem.

THE NO-CODE DIFFERENCE

Project Overruns → Accelerated Development

All the steps in the Design phase leave room for overruns. These delays make a big difference for enterprise tech projects—the longer a project is scheduled to last, the more likely it is that it will run behind schedule and over budget. In fact, every additional year spent on the project directly increases cost overruns by 15%. On average, large IT projects run 45% over budget and 7% over time, all while delivering 56% less value than expected.

No-code overcomes these challenges by enabling the formation of smaller teams, with fewer restrictions on the necessary skill sets of team members. A no-code development process brings key business leaders closer to the solutions being proposed, allowing enterprises to design applications that fit their original goals within the intended timeframe.

Phase 4: Implementation

THE NO-CODE DIFFERENCE

A Focus on Code & Syntax → A Focus on Business Logic & UX

At last, it’s time to actually build the project. Right?

Not quite. With traditional application development, there’s an incredible amount of back-end foundational work that needs to be accomplished before you even see the first logo or basic visual configuration on your screen.

Building applications from the ground using code up has proven time and again to be costly, labor-intensive, and hard to keep on schedule. Engineers are compelled to spend countless work-hours churning-out boilerplate code and thinking through syntax when they should be able to focus on addressing business challenges.

A no-code platform comes with securely configured back-end development, which means a huge, tedious chunk of the development process has already been completed. This frees developers to think through the logic of the project they’re building, rather than repeatedly writing out the same declarations and boilerplates.

THE NO-CODE DIFFERENCE

Approximations → Previews

Because there are so many steps in between the early stages of the SDLC and being able to view a close-to-finished product, there’s also significant room for error. Developers often can’t share tangible work until the project is almost complete, meaning there’s not a lot of time or room for substantive feedback as the process moves along. Giving feedback at this point is challenging and causes extensive (and expensive) delays.

Since no-code output is live and ready to use as soon as it’s built, development teams can share their work for iterative feedback at any time.

No-code platforms are a new, powerful tool in a developer’s repertoire, and that’s never clearer than in the implementation stage of the development process. With no-code, programmers can focus on more complex tasks, collaborate more effectively, and easily share progress with their colleagues.

Phase 5: Testing and Integration

THE NO-CODE DIFFERENCE

Endless Debugging → Accelerated Problem Solving

With various team members' contributions to the software complete, it's time to bring it all together and test for errors, hammer out integrations, and complete any other checks necessary to ensure that the project meets quality standards. This is when teams must be able to answer questions like:

- Does the deliverable meet the requirements and objectives we agreed to in the planning and analysis stages?
- Is it reliable?
- Are there any remaining bugs?

As you may know from personal experience, this process can drag on for much longer than expected because building software with code can be an open invitation to bugs and integration challenges.

Developers can easily spend hours sitting through technical meetings or scrolling through Stack Overflow looking for solutions to buggy code. All of these hours eat into time that could be spent on building new software that delivers value to the business. What's worse, once your developers have successfully debugged their way to an error-free project, the software can still fail to meet the business objectives that kicked off the whole project in the first place.

To radically improve the testing and integration step of the development process, teams need a functional back-end that's already set-up and ready to build on. This kind of platform would allow developers to dive into the front-end and start doing tangible, visible work from the get-go. With a good no-code platform, that front-end work is tested as it's configured, so any potential errors are caught before they can create severe problems downstream.

Phase 6: Maintenance

THE NO-CODE DIFFERENCE

Legacy Code → Efficient Upgrades

When the Testing phase is complete and the product is ready for deployment, it is time for its release into the marketplace. Sometimes this happens in stages or all at once, depending on the organization's business strategy (e.g., the product may only be released to current clients). Based on feedback, changes may be made before complete deployment occurs.

After the product's release, the team makes software improvements or change requests as needed. The ultimate goal of the Maintenance phase is to ensure that the product remains relevant and high quality. It involves ongoing evaluations of the system's performance.

If you've ever developed software, you already know the struggle of maintaining legacy code. Older systems or previous developers often leave behind code that's no longer compatible with your current system, and there's no quick way to work through it. Before diving into a new project, your developers have to find out how the existing system works, dissect the old code, and create new workarounds or rewrite the code entirely. By the time they finish this painfully time-consuming process, an important deadline has likely passed.

Legacy maintenance can be a huge drain on IT resources. Modifying or fixing an engineer's work requires diving deep into code to perform time-consuming reverse engineering. Over time, this means that enterprises are paying for highly expensive developers to simply maintain the status quo.

No matter the reason for legacy code headaches, there's always one common denominator—code. The moment new code is shipped, it becomes legacy.

With no-code development, not only can you stop producing the legacy code of the future, while maintaining existing legacy systems. Platforms like Unqork function alongside existing solutions, allowing developers to easily keep what works and change what doesn't. This results in faster builds with lessened maintenance costs, and more time for business and IT teams to focus on what really matters—solving problems for the business.

THE NO-CODE DIFFERENCE

Digitizing a Complex Insurance Process in 12 Months → 12 Weeks

The Accelerated SDLC in Action

THE CHALLENGE

The servicing operations at one of the world's largest insurers were overly reliant on manual processes. In order to make its business more efficient, flexible, and error-free, the company embarked on a company-wide digital transformation initiative.

To start, the company sought to digitize its complex (and extensively paper-based) electronic fund transfers (EFT) process. The initial build of this digital transformation project was anticipated to take anywhere between 12 and 18 months using traditional code-based methodologies. Frustrated with that time frame, the company looked for an alternative route and decided to tap into the power of Unqork's advanced no-code platform to deliver a comparable solution in a fraction of the time.

THE SOLUTION

Previously, the EFT process was a complex and convoluted one. Records would be faxed to the company's mailroom, where they would then be scanned by one employee who would then hand the file off to another employee who was responsible for uploading the file into an electronic system and cleaning/preparing the data.

Unqork worked with the company to build a fully digital AI-powered system. Now, when records are faxed in, they are scanned in and securely fed into a machine vision algorithm, which translates writing into machine-readable data points so that business rules can be automatically applied.

THE RESULTS

With traditional code-based development methodologies, this project would have taken the company the better part of a year to finish. With Unqork's configuration-based development platform, it took only 12 weeks.

With the help of Unqork's digitization solutions, the company saw an immediate 60% reduction in required labor, which meant employees could spend less time on repetitive tasks, and more time building value for the company in other areas.

Business impact realized by Unqork customers

FINANCIAL INSTITUTION

Digitization of home loan application process

Developed a single application automating end-to-end loan origination across borrower home loan application, underwriting, offer, and acceptance processes.

- **Speed to Market:** 6 weeks to go from ideation to production, with only 4 resources
- Increased revenue capture potential & improved broker productivity

A TOP 5 RETIREMENT SOLUTIONS PROVIDER

Digitization of plan sponsor onboarding

Developed an end-to-end, digital self-service solution automating sponsor, plan, servicing, pricing, advisor, and TPA data capture.

- **Speed to Market:** 16 weeks to go from ideation to production, with only 4 resources
- Accelerated client onboarding times from 4 weeks to 3 hours

GLOBAL P&C CARRIER

Digital front office and policy administration

Developed an end-to-end digital solution fully automating intake, quote, bind, issue for no-touch and underwriter referral workflows.

- **Speed to Market:** 12 weeks from inception to production, with only 5 resources.
- Reduced average time-to-quote by 90% (real-time quoting)

GLOBAL INSURANCE BROKERAGE

Digitized broker and carrier sales operations/marketplace

Developed an end-to-end digital marketplace concept that enables brokers and carriers to manage the lifecycle of the sales prospect.

- **Speed to Market:** launched marketplace concept in 8 weeks
- Improved response time to customers by more than 50% resulting in higher sales potential

A TOP 5 LIFE INSURER

Digitized invasive medical questionnaire

Developed a direct-to-consumer, customer authenticated, self-service application (mobile native) that digitized entire interview process and underwriting.

- **Speed to Market:** Launched app from ideation to production in 8 weeks
- Reduction in turn-around time, from 45 minutes to less than 10 minutes

A Better Way Forward

In today's digitally-driven marketplace, application development is a key differentiator of successful enterprises. However, the risks, costs, and delays baked into the traditional application development process have become increasingly unsustainable, especially as the size and scope of IT projects grow and become more critical to success.

We shouldn't settle when it comes to the platforms we rely on for application development, particularly when it costs teams so much money and projects so frequently fail. By rethinking the whole process and opting for a configuration-based, no-code method, we can simplify application development, speed up labor processes, and ultimately create an end product that is better aligned with business goals.

Advanced no-code platforms like Unqork can help your organization achieve adeptly engineer around challenges of any scale. Get in touch to see what we can do for you.

---

unqork

Enterprise application development, reimagined

Unqork is a no-code application platform that helps large enterprises build complex custom software faster, with higher quality, and lower costs than conventional approaches.

Request a Demo | Learn More