Qt Development

WHITEPAPER

Your Guide to Limitless Scalability

How to shorten development times, improve user experience, and get more done — without creating extra work or complexity for your development team.

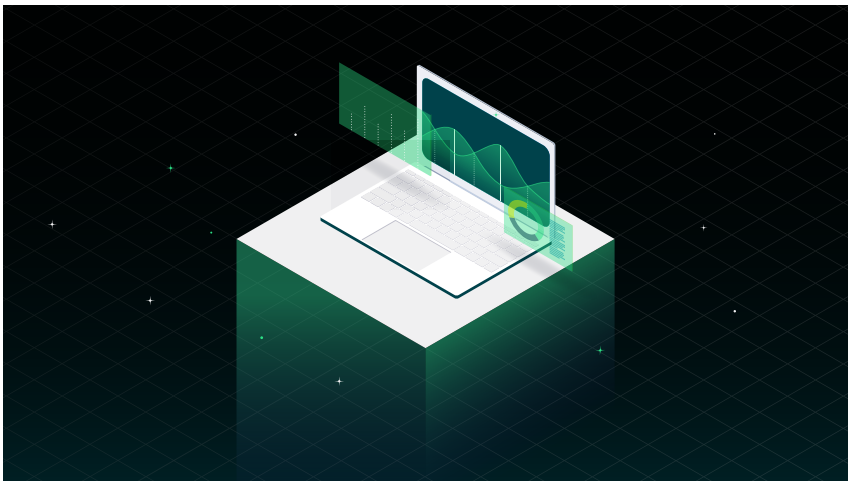
Contents

Your Guide to Limitless Scalability

Software development cycles used to be measured in years: now they're measured in months or even weeks.

Across industries, modern businesses are investing in proprietary software to provide more customer value and differentiate themselves from the competition. At the same time, new technologies and increasing customer demands have created a need for development cycles that are shorter than ever. In the past, it may have taken months or years for development teams to release new software updates. Now, developers may only have a matter of weeks to complete the same amount of work.

Trying to meet shorter development cycles with the same old manual processes is a recipe for failure and one reason burnout is an issue in the developer community. And when developers burn out, the customer experience suffers — which can further damage your business. In fact, a recent McKinsey survey found that users' top reason for dissatisfaction with a digital product or channel is poor UX.



But whether you're developing applications for mobile, desktop, embedded devices, or supercomputers, there is a way to grow your products, shorten development cycles, and keep costs in check all at the same time. We call it limitless scalability. This is how it works.

Scalability is the ability to adapt something to meet greater needs in the future.

The definition of scalability

Scalability is the ability to adapt something to meet higher demands in the future. In software development, this could mean supporting more users, features, or output, depending on the type of software you're creating. But the phrase "in the future" also implies that you need to be thinking about, and preparing for, future needs — even if you don't know exactly what those needs are. Scalability, then, can be seen as a quality of the lifetime value of software: how well it adapts to changing needs, how easy it is to grow, and so on.

There are many elements to scalability, including:

- **Performance:** how well the system works, especially when there's increased usage.
- **Response time:** how long it takes user requests to be processed.
- **Memory usage:** the amount of RAM space used for a certain task.
- **Throughput:** the number of requests that can be processed within a certain period of time.
- **CPU usage:** the amount of processing power required by the device running the application.

As your development projects grow, they usually need to scale in some direction. There are two kinds of scaling in software development:

Vertical scaling (scaling up or down) refers to improving performance, response time and throughput without significantly increasing costs or usage. But you can also scale things down, like time to market, resource costs, team size, and data complexity.

Horizontal scaling (scaling out) refers to expanding offerings: adding new features to your product, creating a companion mobile app for your product, or even expanding your product family.

When products scale up or out, the complexity of the product increases as well. That's why it's important to consider what you can scale down as your products grow. Otherwise, you are not really scaling your product: you're just growing it.

Agile is about being iterative. Limitless scalability helps you iterate faster.

So, what is limitless scalability?
In short, limitless scalability is being equipped with all the tools needed to adapt simply and easily to whatever the future holds — whether that's designing applications for microcontrollers (MCUs) or user interfaces for supercomputers — without creating extra work or complexity for the development team. It's the ability to, for example, create an iOS version of your Android app without having to start the development process from scratch. Or the ability to turn a designer's pixel-perfect mockups into working code with just a few clicks. Limitless scalability is how developers can work when all barriers to success are removed and people can work how they want to. In other words:

Limitless scalability is Agile.
There are 12 principles of the Agile Manifesto, and limitless scalability helps deliver on all of them, including:

- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.

By equipping your team with tools that help them create nearly anything you can imagine, you make the development team more adaptable to change. By breaking down barriers between departments (design and development, for example), you remove communication silos and speed up development.

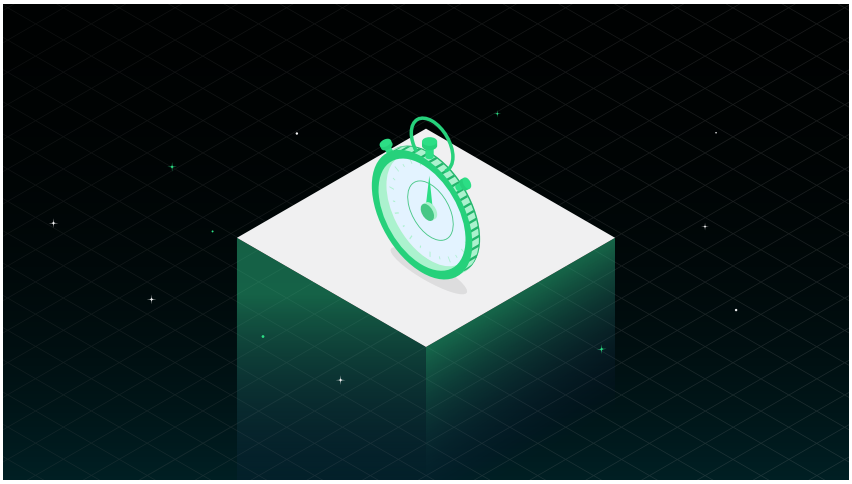
Limitless scalability is DevOps.
The main purpose of DevOps is to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and be more competitive in the market.

With access to advanced testing tools, and the latest developments in automation, your team can develop high-quality, functional products and applications faster than ever before — and release new features and updates faster too.

Lean software development is about reducing waste. Qt helps you find and remove waste earlier in the development process.

Limitless scalability is Lean.
Like lean manufacturing principles, lean software development relates defective codes or bugs to waste. This is because defects cause extra work in the form of rework. In general, expenditures on fixing a defect in the form of time, money and manpower can be significant in the absence of proper quality checks. Moreover, defects detected at an early stage of development are easier to fix (and not as harmful to company reputation and/or user experience) as those detected during the operation phase, or even later.

When you use Qt, you have access to over 50 software libraries of highly readable, easily maintainable, and reusable code with high runtime performance and small footprint. This cuts down on code waste and saves you time on code validation.



Achieving limitless scalability with Qt
Qt offers a variety of development tools to simplify developers' lives and help with not just coding but also tasks like building, compiling, localization, and more. Here are just a few of the ways that Qt can help you achieve limitless scalability:

Make better decisions about new features and updates
Choosing which features to prioritize isn't always an easy decision. And not every new feature is adopted the way we expect it to be. Spending time designing and developing new features that users don't want, need, or respond to isn't just time-consuming: it's demoralizing too.

But knowing what users want before you start creating is like trying to predict whether it will rain two weeks from now. All these additional features — if no-one is using them — are just cluttering up your product.

Qt Insight looks specifically at feature adoption and answers the age-old question, "are our users really using this feature?" by gathering data on your application's performance, usage, and user data that may not be otherwise attainable. With this information in hand, you can develop new features based on how users use your products — and remove features that they don't want or need.

To make better decisions, you need better — and more actionable — data.

Stop reinventing the wheel with every new project

When knowledge isn't shared between teams and projects, it creates knowledge silos in your organization and slows down development times as well. In other words, you end up trying to reinvent the wheel repeatedly.

But when you start from always-fresh code libraries and re-usable components — like those in the Qt development framework, your team doesn't need to start from scratch every time, and they can quickly achieve the functionality (and especially the quality) that you need from a proprietary application. The Qt code libraries save development time, freeing your team up to perform more value-adding tasks like optimizing user experience.

Imagine you're making a product line of washing machines: both low-end and high-end machines. These washing machines need a companion app so that users can remotely control wash cycles. This means you will likely need to make applications for both Android and iOS, as your users may have mobile devices with either of those operating systems.

Without Qt, you would have to design and develop each application separately, with a lot of testing and validating to make sure the user experience is the same on both devices. But with Qt, you can create both applications at the same time, with the same functionality, using the same code base.

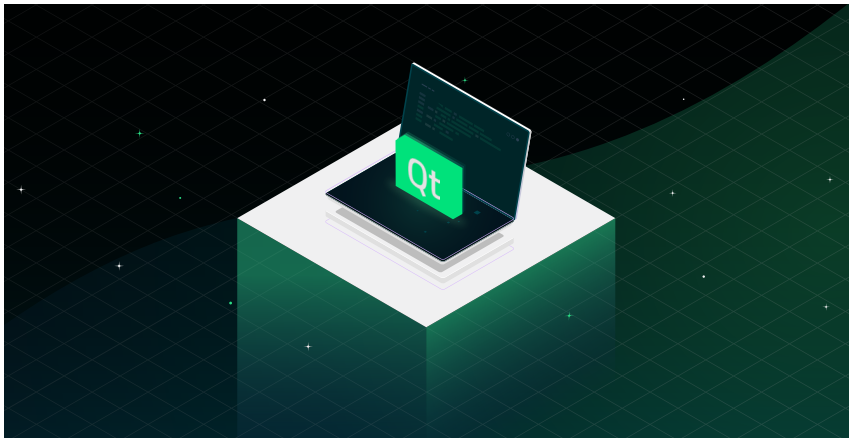
With Qt, everyone from the designer to the developer to the QA tester is working on the same code at the same time. You don't need an additional steep learning curve to move to a new platform operating system. In addition, you only need to support one code stack, meaning you don't need multiple teams with different skillsets to support different platforms. Upgrades, tests, and other changes need only be implemented once.

Qt's code libraries are constantly tested and verified by thousands of developers — aka our users.

Minimize defects and rework

Qt Quality Assurance tools allow you to perform automated cross-technology/ device GUI testing, code coverage analysis of your entire test framework, static code analysis, and check the compliance of your software architecture.

But you can also reduce defects by starting from a pre-built library of clean, bug-free code, like that found in the Qt development framework. When you start from code that is shared by thousands of other developers — and is being used in millions of devices — you can be confident that the code works as intended. You also avoid defects because somebody else has likely already found the errors and fixed them. Again: instead of reinventing the wheel, your team benefits from the knowledge and experience of thousands of other developers.



Conclusion

These days, development teams must deliver far more output in far less time than they ever had before. Trying to reinvent the wheel for every new project leads to low-quality code, dissatisfied users, and developer burnout. Instead of working harder and trying to do more with less, work smarter and enable limitless scalability with the Qt framework.

Qt's platform-agnostic framework enables more flexibility, significantly facilitating migrations to a different hardware and/or software platform. By leveraging libraries of ready-made components, your development team does not need to create things from scratch when developing new applications for desktop, mobile, embedded devices, or MCU.

With Qt you can code once and deploy everywhere — which is the true definition of limitless scalability.