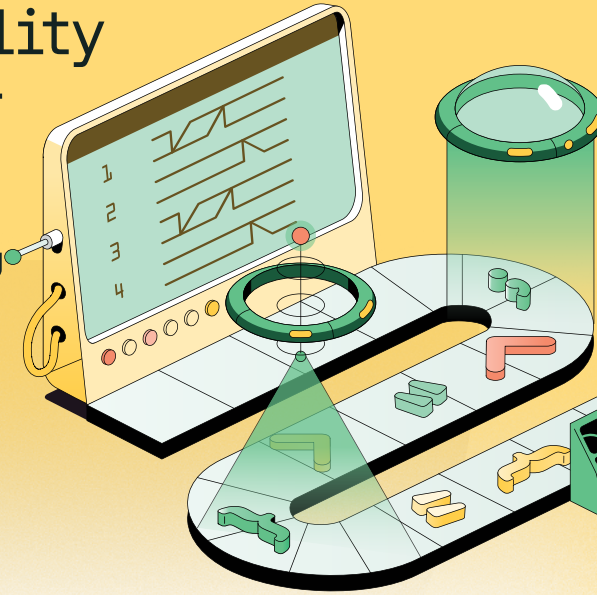
SauceLabs

The Continuous Quality Operating Model for Software Testing

A Strategic Framework For Improving Mobile App Quality

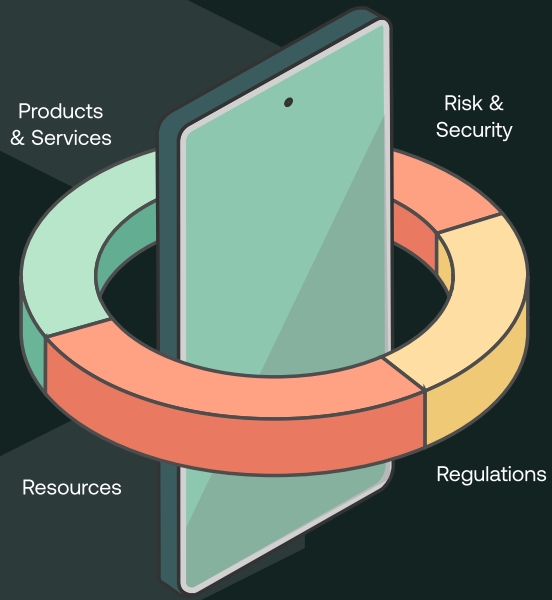


Digital transformation has reshaped how people interact with mobile and web applications.

Rather than wait in line at the bank branch or drive to the store to return items, customers now use a mobile app to complete these mundane but necessary tasks.

Merging technologies and features, such as AI-powered tools and new payment methods, means slow, manual testing processes are no longer viable options.

Continuous testing throughout the software development lifecycle allows brands and companies to keep pace with innovation and competitors while ensuring software quality at speed and scale. This helps their organization meet customer demands for new products while maintaining compliance.



While continuous quality is considered the best practice for ensuring mobile app quality, there is no one-size-fits-all approach.

The right continuous testing strategy depends on a number of factors, including:

- The types of products and services you offer
- Your organization's risk and security profile
- The structure and resources of your development teams
- Your specific regulatory requirements impacting software

This guide allows you to map your current testing strategy against a Continuous Quality Operating Model so you can better define where you are—and, more importantly, where you want your organization to be.

It will also provide actionable tips and adjustments so you can create a continuous testing strategy that meets your business requirements for scale, speed, performance, and security.

The Continuous Quality Operating Model

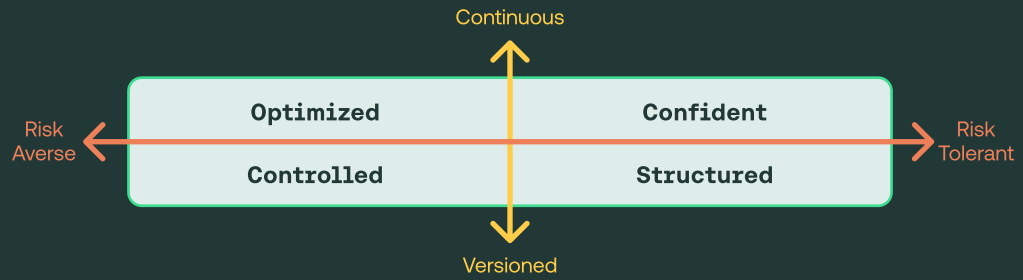
Based on those requirements, you can use the Continuous Quality Operating Model to plot your organization's current test strategy across the two most critical vectors for software development.

Release Velocity:

This axis measures how quickly you release updates, helping you meet your timelines without sacrificing quality or compliance. Being higher on the axis means you are releasing frequently instead of waiting for more traditional versioned software releases.

Risk Management:

This axis measures how you balance risk and coverage, helping you direct your resources where they matter most. The further right you are, the more risk-tolerant your testing strategy.



With this model, you can make more informed decisions about whether or not your current testing strategy places you in the category that meets your current and future business goals. It can also help you determine the changes and investments you need to make to ensure your testing strategy ultimately meets your business objectives while helping you deliver a high-quality mobile app experience.

Risk Management:

Risk Averse versus Risk Tolerant

Software testing and quality teams often fail to adequately integrate risk tolerance into their testing strategy, creating misalignment with their organization's overall risk approach.

Defining risk will look different depending on what sector the business operates in. Financial services, for example, focuses on identifying and mitigating financial loss, data breaches, and regulatory compliance issues.

Your risk management approach is determined by the balance of time and resources you invest in risk identification and mitigation. Ultimately, most risk management strategies strive to eliminate risk where possible and have a plan in place for responding to failures if and when they do happen.

Risk Averse ⟷ Risk Tolerant

X-AXIS Risk tolerance can be plotted between two ends of the spectrum:

Risk Averse

This end of the spectrum offers exhaustive coverage, testing every conceivable code path and user scenario. Unit tests are ideal for achieving high coverage, which provides a clear way to measure risk. However, it doesn't always account for how different services interact, how real users behave, or rare but potentially devastating issues that can't be identified by only evaluating code.

Risk Tolerant

Instead of trying to test everything, this end of the spectrum focuses on testing the critical areas of an app that impact the user experience, revenue, security, and regulatory requirements. Financial institutions that embrace risk-tolerant testing focus their testing strategy more on things like error reporting/production monitoring, canary releases, or selective testing that aligns with the importance of the changes and their impact on users.

Risk Management:

Risk Averse versus Risk Tolerant

Isn't the right answer to focus on risk identification and eliminate all potential risks?

Not necessarily.

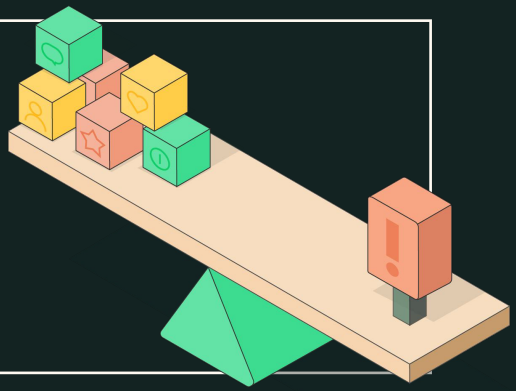
When it comes to your mobile app, you want to balance risk identification and risk management to be more strategic about how you test. If you are too focused on proactive risk identification, you may spend time and resources identifying issues with rarely used features, which takes time away from core functionalities.

Consider this example:

A team that decides to test every feature with equal rigor gives the same attention to a feature used by 0.5% of its users as it does to the most used features.

This team's zero-tolerance approach to risk is a mile wide and an inch deep, catching irrelevant issues while taking time away from identifying the risk that most impacts the user experience.

By focusing on broad coverage instead of strategic value, critical features receive less testing than they deserve, leaving them more exposed.



Risk Management:

Risk Averse versus Risk Tolerant

Your testing strategy goal should strike a balance between risk and test coverage to meet the strategic business goals so that you focus testing resources where they matter most.

With this balance, you can **safeguard the features and functionalities** that define your app's value, security, and trustworthiness in a high-stakes financial environment.

eMoney

Financial planning software developer **eMoney Advisors** leveraged a comprehensive automated testing strategy to improve its risk management by **increasing its real device coverage by 50% and virtual device coverage by 37%.**

Read the case study

Release Velocity:

Version versus Continuous

Release velocity plays a critical role in remaining competitive, innovative, and responsive. But this must balance the need to keep pace with customer expectations with the need to compromise compliance, security, or app reliability.

With the mobile app becoming the sole customer touchpoint for many brands, maintaining app quality with new features and updates is critical. That's why many brands and organizations prioritize test speed.

Prioritizing test speed allows companies to reduce their release intervals so they can push critical updates faster, maintain the user experience with quality updates, and stay ahead of competitors by delivering new features. The result: a mobile app that is reliable, relevant, and more likely to drive revenue.

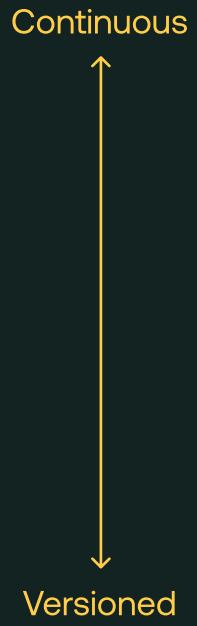
Y-AXIS Release velocity can be plotted between two ends of the spectrum:

Continuous Releases

This method is more typical in SaaS environments, where updates can be pushed multiple times daily. While it may require companies to adjust their traditional development practices, continuous releases allow them to be more innovative, more iterative with user experience improvements, and more responsive to emerging security issues or regulatory requirements.

Versioned Releases

This is the traditional model for software releases, which sees software updated at scheduled intervals. Versioned releases can offer more stability and control, ensuring each update is thoroughly tested before deployment.



Release Velocity:
Version versus Continuous

Even for organizations that use a versioned release approach, it's important to leverage a testing strategy that reduces bottlenecks so that versions can be released with a faster cadence – continuous-lite, if you will. Eliminating slow, lengthy test processes and delays in feedback loops lets developers keep their apps up-to-date while still meeting compliance requirements.

By evaluating your position on the release velocity spectrum, you can **identify opportunities to optimize your test strategy and increase your release cadence.**

A major financial organization leveraged a comprehensive automated testing strategy to **increase its deployment frequency by 4x while improving its app quality** thanks to a higher testing volume.

Read the case study

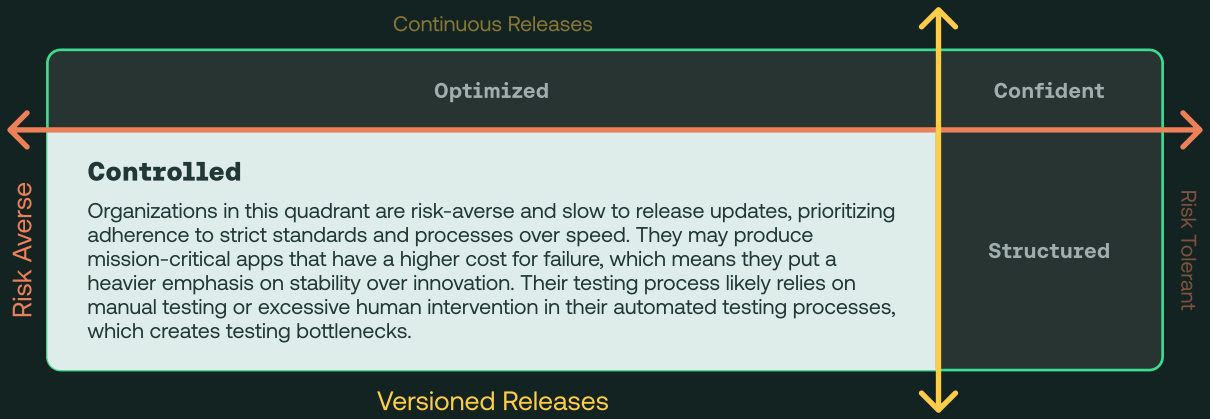
Defining the Continuous Quality Operating Model Categories

Each quadrant represents a different approach to risk management and release velocity, along with distinct testing strategies for balance coverage, risk tolerance, and speed. Here is what each quadrant entails, along with tips for improvements.

Defining the Continuous Quality Operating Model Categories

Controlled

Risk Averse + **Versioned Releases**

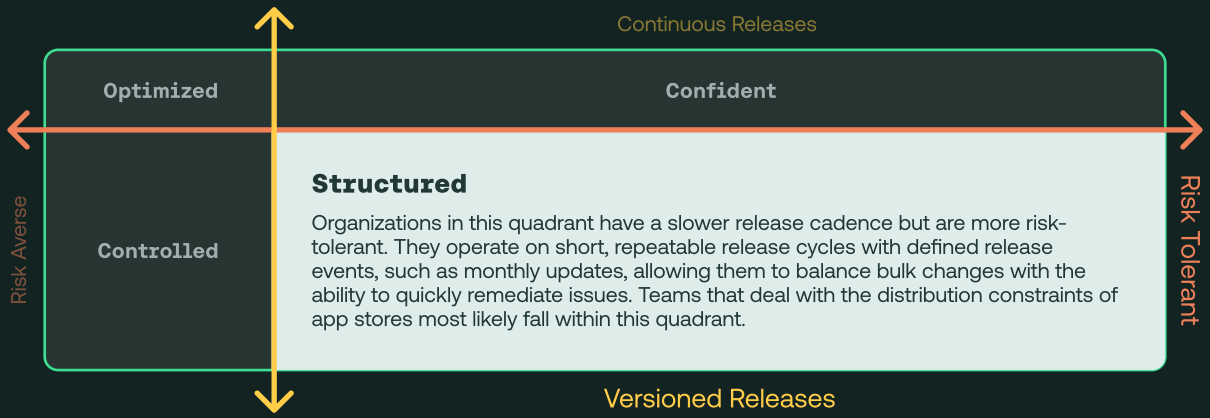


These organizations can reduce risk by:

- Transitioning from manual to automated testing using the Real Device Cloud for scalability
- Increasing efficiency by accelerating and simplifying test setup
- Providing basic test analytics dashboards for enhanced visibility into test coverage and results

Structured

Risk Tolerant + Versioned Releases



Structured

Organizations in this quadrant have a slower release cadence but are more risk-tolerant. They operate on short, repeatable release cycles with defined release events, such as monthly updates, allowing them to balance bulk changes with the ability to quickly remediate issues. Teams that deal with the distribution constraints of app stores most likely fall within this quadrant.

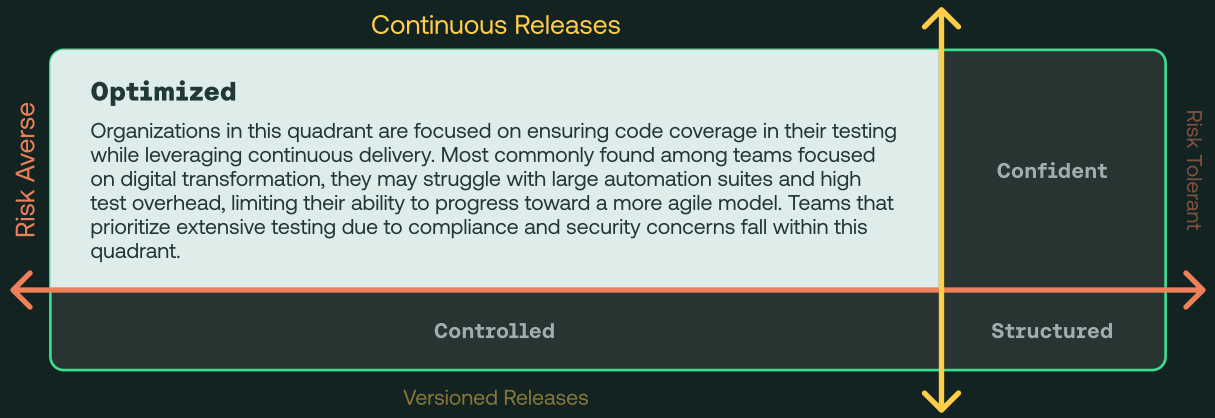
Organizations in this quadrant can shift testing left by using Sauce Labs to:

- Identify issues earlier in development with emulators and simulators
- Use Visual Testing to ensure flawless UI and consistency across devices, critical for user-facing features in mobile banking apps
- Optimize the testing pipeline for faster feedback loops with Sauce Insights

Defining the Continuous Quality Operating Model Categories

Optimized

Risk Averse + Continuous Releases

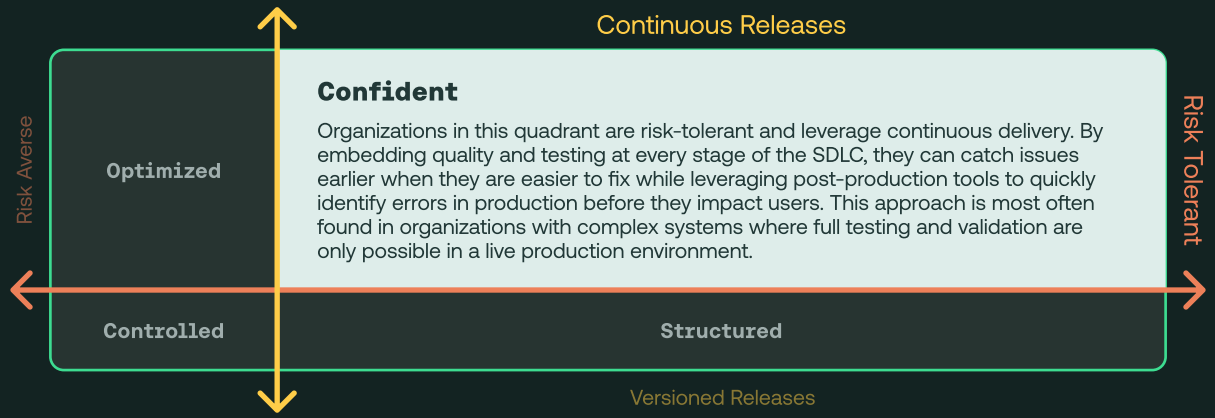


Organizations in this quadrant can catch more issues in production by using Sauce Labs to:

- Optimize beta testing and gain actionable feedback from small user groups with Mobile App Distribution
- Detect, prioritize, and resolve critical bugs in production with Error Reporting
- Identify redundant test cases and increase focus on critical areas with ML-powered Failure Analysis

Confident

Risk Tolerant + Continuous Releases



Organizations in this quadrant seeking to improve their entire continuous quality process can use Sauce Labs to:

- Leverage Sauce Insights across the SDLC to identify opportunities to optimize and enhance velocity across the development lifecycle
- Test user-critical workflows with the Real Device Cloud to ensure mobile app performance under real-world conditions
- Monitor live systems for issues and enable proactive mitigation of risk
- Rely on virtual devices for fast, scalable testing
- Ensure UI consistency and prevent visual regressions that could impact user trust and satisfaction

Where does your organization fit?

You may already intuitively understand where your organization sits on the Continuous Quality Operating Model. If not, take this quiz to help map your position.

Continuous Quality Assessment Quiz

Use this quiz to determine your organization's position across the Operating Model Matrix

How does your organization prioritize testing efforts?

- 1 PT ☐ We test every feature thoroughly, regardless of usage or criticality, before releasing
- 2 PTS ☐ We focus primarily on critical features and rely on post-prod monitoring for lower-priority areas
- 3 PTS ☐ We prioritize testing for key functionalities but still cover most areas before release
- 4 PTS ☐ We focus our testing efforts on high priority areas, and rely on production safety nets

How does your organization balance testing speed and risk?

- 1 PT ☐ We prioritize thorough testing over speed, even if it delays releases
- 2 PTS ☐ We try to balance speed and risk but often lean toward more comprehensive testing
- 3 PTS ☐ We optimize for speed, relying on targeted testing and real-time monitoring
- 4 PTS ☐ We are fully optimized for speed, focusing testing on high-priority areas and resolving risks in production

How automated is your testing process?

- 2 PTS ☐ We rely primarily on manual testing
- 3 PTS ☐ We have automation for key areas but still depend on manual testing
- 4 PTS ☐ Most of our testing is automated
- 6 PTS ☐ Testing is fully automated, with pipelines optimized for speed and scalability

How does your organization handle bugs in production?

- 1 PT ☐ We have a zero-tolerance policy for production issues and focus heavily on preventing them upfront
- 2 PTS ☐ We acknowledge that some production issues will happen and focus on minimizing their impact
- 3 PTS ☐ We occasionally encounter production issues and prioritize root-cause analysis after they occur
- 4 PTS ☐ We accept some production issues as a trade-off for faster releases, and rely on quick remediations

How frequently does your team release updates to production?

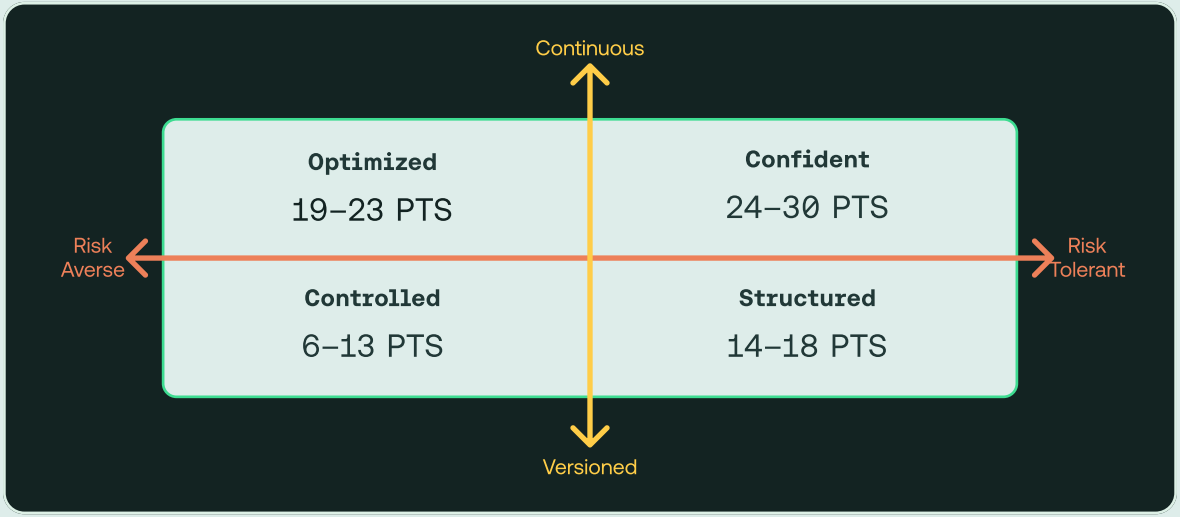
- 2 PTS ☐ Quarterly or less
- 3 PTS ☐ Monthly
- 4 PTS ☐ Weekly
- 6 PTS ☐ Multiple times per day

What best describes your release pipeline?

- 2 PTS ☐ We follow a versioned approach, with scheduled release cycles and detailed approvals
- 3 PTS ☐ Our releases are versioned but with faster approvals and some automation
- 4 PTS ☐ We mostly rely on continuous delivery, but we use versioned releases at times
- 6 PTS ☐ We have a fully continuous delivery pipeline, pushing changes to production frequently

Regardless of where your organization is today, there's always room for improvement.

By aligning your testing and release strategies with your business goals, you can achieve greater efficiency, increased customer satisfaction, more revenue, and a significant competitive advantage.



Sauce Labs is the only enterprise-ready automated testing solution that meets the complex needs of diverse industries such as financial services, retail, and more. Sauce Labs enables you to incorporate testing across the SDLC in a way that is complete, scalable, and secure with:

Mobile app distribution:

Optimize beta testing processes and streamline distribution and management with our secure, all-in-one platform.

Post-production error monitoring:

Use our platform to capture, prioritize, and quickly resolve errors post-release while identifying the root cause so you can permanently correct the issue.

Insights:

Use Sauce Insights to gain the visibility that you need into your testing trends to improve your processes across the SDLC.

Automated testing:

Sauce mobile solutions include a comprehensive Real Device Cloud with public and private devices, emulators and simulators, and integrated visual testing, making it simple to test any device/OS at any time.

See it in action

Learn more about how to optimize your software testing strategies with Sauce by taking our platform product tour.

Start product tour