Optimizing Software Releases with Risk-Based Testing

Deliver cost-effective, defect-free applications by testing only what matters.

Author: Kaarthick Subramanian, QA Practice Lead, emids Technologies

emids
experience partnership.

Contents

Overview

As the demand for technologies continues to grow, organizations are under more pressure than ever to deliver new and defect-free software applications on a faster, more frequent basis.

Along with developing higher-quality applications in shorter project cycles, they must also ensure applications work for an increasingly diverse set of users who access them through different platforms such as web browsers, mobile devices and other channels.

Companies spend lots of time, money and effort trying to test every possible scenario before each software release, yet many still end up with defects in production. Most believe that the more they test an application, the fewer defects it will have, but that rarely happens. The majority of defects are typically caused by a small set of issues.

Through risk-based testing, organizations can uncover the most significant defects early at the lowest cost. This test optimization approach targets the most critical risks, including those that could potentially have the biggest impact on the organization, such as a loss of reputation or revenue.

Prioritizing these risks helps organizations determine what to test when and how much testing is sufficient, allowing them to create the most efficient testing strategy. Though it's impossible to anticipate every risk—especially those that arise during and after testing—this strategy helps organizations find the most pressing defects early enough to give development teams time to fix them before the application is released.

Not only does risk-based testing improve customer satisfaction by minimizing defects, it also helps organizations to reduce their cost of quality by reducing repetitive, redundant testing and improve the quality of testing by aligning it more closely to the perspective and priorities of the business.

Why Risk-Based Testing Works

Figure 1 illustrates the prevailing belief that the relationship between testing and reducing risk is linear, and rigorous testing leads to a completely defect-free application.

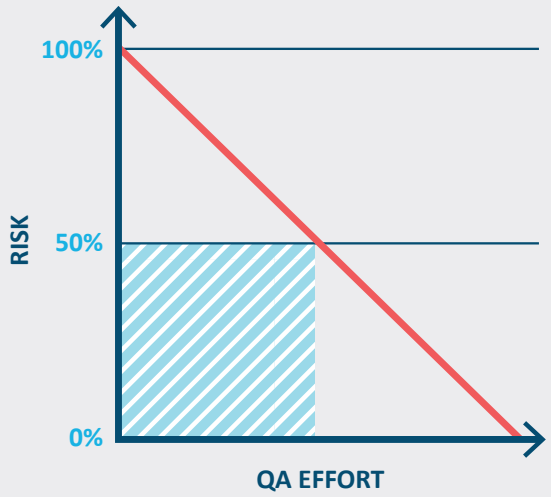

Figure 1: Belief that reduced risk is linear. Actual figures may vary depending on the application, team and maturity of the process.

Figure 2 shows how testing actually reduces risk. Risk-based testing helps QA teams mathematically arrive at a point where they can decrease a large percentage of the risk—up to at least 50 percent—at a quarter of the effort without experiencing diminishing returns.

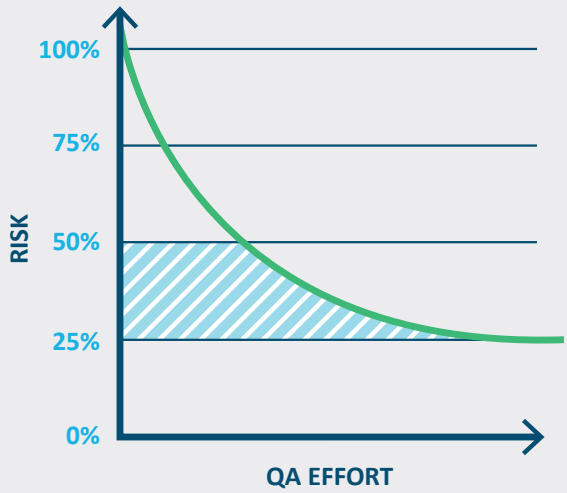

Figure 2: Reduced risk with risk-based testing. Actual figures may vary depending on the application, team and maturity of the process.

Approach to Risk-Based Testing

The most common technique for conducting risk-based testing is arriving at a risk rating for each feature of the application under test. Once risks are identified and scored for each feature, test cases are then developed and mapped to each one. Tests are executed in order of priority, from critical to low, based on risk scores.

Arriving at an accurate risk rating requires the collective input of testing, business and development teams as well as those at the organization familiar with issues important to the customer. Potential risks may range from disruption of a critical function of the application to a defect that complicates its use for end users.

Risk Rating Formula

The risk rating depends on two factors: business risk (B) and probability of failure (P). Multiplying these two together equals the risk rating (R = B X P). When determining business risk and probability of failure, the following values need to be considered.

Note Risk factors values are rated on a scale of 1–5.

BUSINESS RISK (B) VALUES

- **Business criticality**: Measures the importance of the feature in relationship to the business need being served by the application under test.
- **Visibility**: Measures the visual impact on the business should the feature fail.

PROBABILITY OF FAILURE (P) VALUES

- **Complexity**: Measures the complexity of the code written to implement the feature the test case is evaluating.
- **Change control**: Measures how often requirements or the code written changed prior to being tested.
- **Development time**: Measures the time taken to write the code implementing the feature the test case is evaluating.

Charting Risk Rating

Figure 3 shows how risk rating works. The chart illustrates the risk rating for a few sample features and the recommended order

	Business Risk (B)		*Propability of Failure (P)*			*(R)*	
Service Workstation (1.5) Test Areas	**Visibility**	**Business Criticality**	**Complexity**	**Change Control**	**Development Time**	**Risk Rating**	
Contacts	1	2	4	2	3	27	
Related Links	2	1	1	1	1	9	← Low End
Boo of Business	3	5	3	5	2	80	← High End
Actions	1	4	5	1	4	50	
Events	2	1	1	2	1	12	
Interactions	4	3	1	4	1	70	

Figure 3: *How charting risk rating (B X P = R) works. Actual figures may vary depending on the application, team and maturity of the process.*

for executing tests. The test case with the highest rating, “Boo of Business,” must be executed first, while the test case with the lowest rating, “Related Links,” can be executed last or dropped from subsequent retests once it has passed the first time.

Developing a Test Execution Schedule

Once risk rating has been calculated for the feature under test and the test cases mapped over, the next step is to prioritize them into groups and develop a test execution schedule, as ***Figure 4*** illustrates. All test cases that fall as *critical* should be executed first across modules. Developing a test execution schedule will help the testing team find and address the biggest issues first and allocate resources most efficiently.

Mitigating Risks

Once test cases are executed, the final step is to come up with solutions that address each of the risk scenarios should they arise. Developing a contingency plan can help organizations respond quickly to risks that are likely to become a reality and minimize their impact.

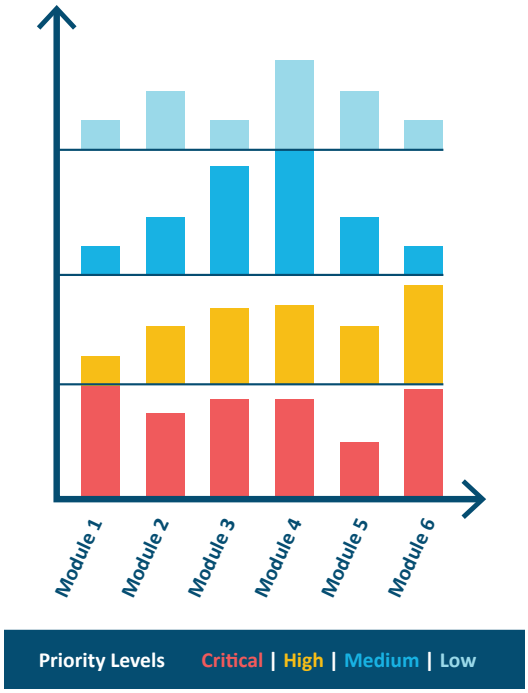

***Figure 4**: Example test execution schedule.*

Benefits of Risk-Based Testing

Risk-based testing offers several competitive advantages for organizations looking to drive efficiency in testing and accelerate their time to market.

Reduced Testing Costs

Organizations can avoid repetitive, redundant costs for testing and reduce spending in this area by up to 50–70 percent.

Improved Efficiency

Organizations can know what to test when and more accurately project how much testing is sufficient before releasing the application.

Better Customer Satisfaction

Organizations can identify and fix critical defects of the application early, allowing them to deliver a nearly defect-free application.

Higher Quality Testing

Teams can better align testing to the needs and risks of the business instead of just evaluating the application from a technical perspective.

Best Practices for Implementing Risk-Based Testing

When conducting risk-based testing, it helps to simplify the process as much as possible. Risks should be confined to the most likely scenarios and tested based on what matters most for the organization, whether that is saving money, improving a business reputation or reaching new customers.

Guidelines to Remember

Get an Early Start

The sooner organizations can start identifying and managing risks in quality assurance testing, the better. Risks that are overlooked or underestimated will eventually resurface if they are not addressed early on.

Build the Right Team

Selecting the right stakeholders to gather input from is crucial during the risk identification phase. Make this process as collaborative as possible to increase its accuracy and effectiveness.

Use Objective Data

Avoid basing risks on subjective criteria. Look for historical data when possible and objective criteria to draw from when formulating risks.

Plan Accordingly

Build in enough time for testing, with reasonable start and completion times. Too much planning leaves too little time for testing.

Conclusion

All software projects have risks, and none can be completely risk-free, no matter how much testing is done. Few organizations have the resources to test everything before each software release, but fortunately they don't need to. Risk-based testing helps organizations improve efficiency by knowing what to test when. Not only does this save time and money, but it also drives customer satisfaction by ensuring applications are nearly defect-free when delivered.

About the Author

Kaarthick Subramanian, QA Practice Lead, emids Technologies

Kaarthick is a seasoned technology leader with a winning track record of establishing and sustaining value-producing IT operations focused on process improvement, technology enablement, and quality assurance to reach the next level of performance. He offers in-depth industry knowledge, actionable insights to design and execute IT strategies that drive business growth, minimize costs and leverage new revenue streams. Kaarthick brings a track record of success in building world-class quality assurance practices for Fortune 500 companies and has worked extensively across range of business verticals.

About emids

emids is a premier provider of healthcare IT services and solutions. emids enables healthcare entities to achieve accessible, affordable and high-quality care by providing custom application and data solutions. Our clients experience true partnership with us as together we navigate the challenges of a rapidly changing healthcare industry.