Luxoft
A DXC Technology Company

Risk-based requirement prioritization — save project money, time and effort

by Yogesh Sanjeevan Kshirsagar,
Principal Consultant, Banking and Capital Markets

Risk-based prioritization (RBP) is about identifying requirement risks, understanding testing needs and ranking requirements in line with business priorities.

Testing has its own set of risks including ambiguous requirements, test environments, unavailability and not having proper stubs or automation tools for testing. Crucially, if you don't address a particular risk, it could become a costly issue in production.

RBP can streamline your entire software development process if carried out efficiently and effectively. It can also save money diverted to testing bugs you could have captured earlier in the software development lifecycle (SDLC).

Fix the risk **before it becomes a production issue**

The entire team is responsible for developing crystal clear requirements at the beginning of your project. If team members don't understand a requirement, it will cause defects in production. It's extremely expensive to fix anything in production. After all, you'll need to stop applications, risking losing reputation points because your system is down (imagine your bank system failing for 5 minutes just when you want to make an important transaction).

The cost of fixing defects increases exponentially if you allow them to propagate from one phase to another. Let's say a requirement issue surfaces in the design phase; it would be much timelier and more cost-effective to contain it in the requirements phase itself.

It pays to know what's what

If programmers, testers and managers know precisely what they'll be implementing as a team, there will be very few change requests, reducing time-to-market and saving a great deal of money and effort. Clearly, if your requirements are frozen with next to no changes, you can stick to the project scope and schedule, and deliver on time or even early.

The challenge is whether you can test, prioritize and categorize your requirements to optimize and test development.

RBP is a process that enables you to clarify, redefine and rank your requirements according to importance, even before you begin development and testing. This article looks at identifying non-testable requirements and making them testable, plus the practical benefits of risk-based prioritization.

Are your **requirements testable?**

You rarely encounter project supervisors or test managers at pains to understand whether your requirements can be tested or reviewed as a team exercise. It's one of test management's critical challenges and a threat to your entire testing process.

To ensure your requirements are testable, they must be:

Complete | Concise | Correct | Feasible | Unambiguous

These five characteristics help team members understand and interpret requirements consistently.

So, when you're in the sprint planning or analysis phase, business analysts gather requirements so the testing team can go through the lot and check the testing criteria. If you haven't fully identified your testing team, make sure you have a test manager, plus some test engineers and developers who can work on clarification.

Making requirements **testable**

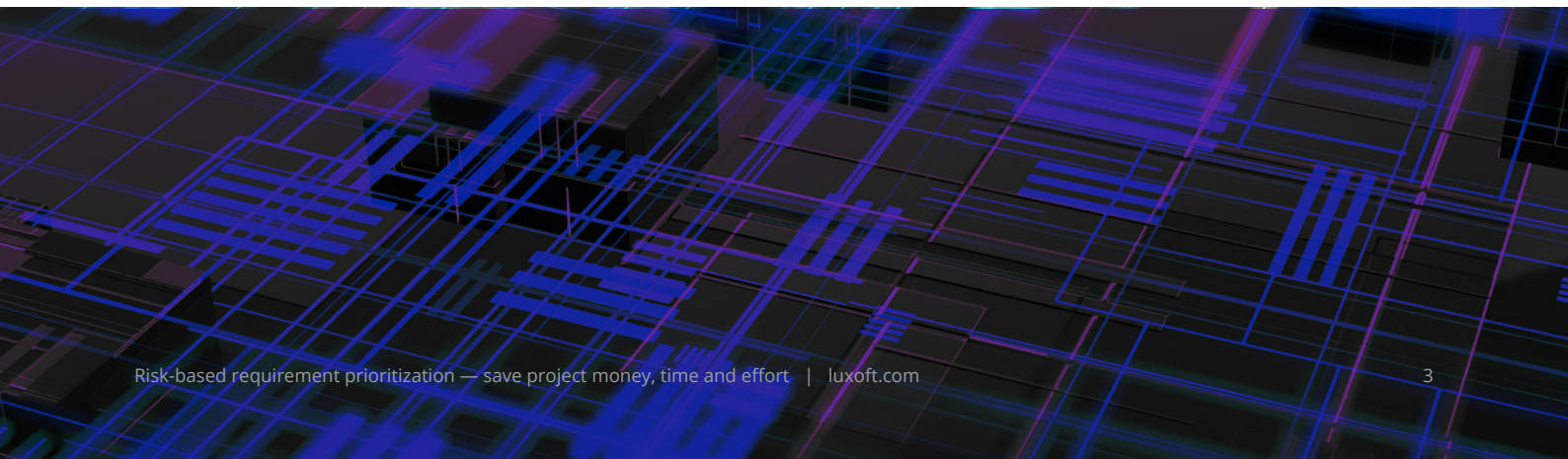
If you see that you can't test some of your requirements, try to make them testable by applying risk-based prioritization. For instance, your requirement definition might say, *"The application should show a status message or disconnect a user if there is no activity for about 60 seconds."*

The statement raises the following questions:

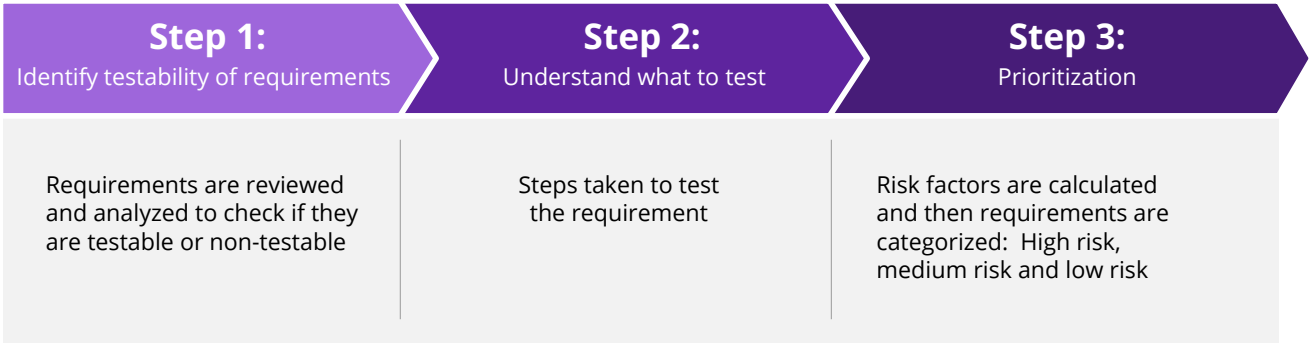
- Which application are we talking about?
- Should it show a status message, disconnect a user, or both?
- If the interval is *'about 60 seconds'*, is it okay to disconnect a user if there's no activity for 50 seconds?

You cannot test ambiguous requirements. Inevitably, different people interpret things differently. Developers might implement one thing, and testers test something else entirely. Then the resulting product will have little value and create no customer satisfaction. Consequently, you must make requirements crystal clear.

To make it testable, you have to work with your business analysts to clarify the ambiguities, update the requirements, and circulate the updated documentation throughout the team. Once you've completed this initial exercise, you can start the core process of risk-based prioritization.



The risk-based **prioritization process**



The three-step risk-based prioritization process:

1. **Checks your requirements are testable.** Earlier, we covered how to assess your requirements against the five testability principles and convert them into testable requirements.

2. **Helps you understand what to test.** Every requirement in the SDLC has an element of risk. Take "Change the font on your company's website." Even though this appears to be a straightforward requirement, it still carries a risk. Familiarize yourself with the steps needed to mitigate that risk.

3. **Prioritizes requirements based on business criticality.** This is the core element of risk-based prioritization.

Number two input parameters (impact and probability) for each requirement. "Impact" gauges the ramifications of your requirement failing in production. "Probability" indicates the chance of failure or errors creeping into production. These two parameters are generally rated from 1 to 3 (low-high).

Multiplying the impact and probability ratings together gives you the requirement risk scores (1-9). Having identified the risk scores, we divide your requirements into high-risk, medium-risk and low-risk requirements by setting up some guidelines as follows:

Scoring guidelines

Risk score	Priority
<=3	Low
>3 <=6	Medium
>6	High

Parameters and the scoring model can vary depending on the company you're working with.

Requirements categorization based on the guidelines

Impact	Probability	Risk scores	Category
1	1	1	Low
1	2	2	Low
1	3	3	Low
2	1	2	Low
3	1	3	Low
2	2	4	Medium
2	3	6	Medium
3	2	6	Medium
3	3	9	High

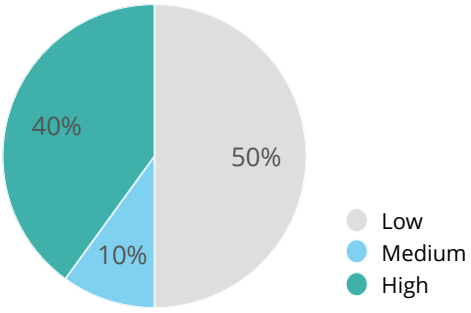
The left table shows the guidelines set for requirements categorization. The table on the right shows permutations and combinations along with basic categories like high, medium and low.

The test manager consolidates all the inputs and circulates the results, detailing the number of high-, medium- and low-risk requirements as follows:

Prioritization of requirements

Total requirements	10
High	4
Medium	1
Low	5

Requirements categorization



Req. ID	Requirement definition	Input parameters		Risk score	Priority
		Impact	Probability		
Req.1	Verify login to the website	1	1	1	Low
Req.2	Verification of functional aspects of the screen	1	2	2	Low
Req.3	To be able to add a form for a change of details	1	3	3	Low
Req.4	To be able to select all cases	3	3	9	High
Req.5	Verify that the user is able to change the font	3	3	9	High
Req.6	Verify that the admin is able to delete a user	3	2	6	Medium
Req.7	The report should show the people registered on the website	1	3	3	Low
Req.8	The report must be sent to the key stakeholders daily	1	3	3	Low
Req.9	The values should be truncated to 2 decimal places	3	3	9	High
Req.10	An email should be sent to all stakeholders by EOD	3	3	9	High

Risk scores based on scoring model **Prioritization based on scoring model**

Based on these inputs the team starts working on development, testing and implementation.

Benefits of risk-based prioritization

In summary, risk-based prioritization helps:

- Identify untestable requirements and make them testable
- Focus on testing activities right from the analysis phase
- Keep all stakeholders up to date
- Identify critical requirements so you can increase their testing
- Communicate the quantifiable risks in your projects
- Optimize UAT by ignoring low-priority requirements (you'll need business-team buy-in)
- Streamline your entire testing process

The bottom line

The risk-based prioritization testing process allows you to scrutinize requirements in the gathering phase. It enables you to focus on testing and minimizing defects that penetrate the system due to inaccurate requirements.

At the same time, it helps you identify priorities and category requirements, and optimize your UAT. RBP provides a strong foundation for testing, automatically improving your development process quickly and with maximum business value.

Find out more about how Luxoft can help you streamline your software development process by visiting **luxoft.com/capital-markets** or **contact us** — we have many more insights to share with you.

About **the author**

Yogesh Kshirsagar
Principal Consultant,
Banking and Capital Markets

Yogesh has 19 years of IT experience in banking and finance. Before joining Luxoft, he held leadership positions across the UK, United States, Singapore, Malaysia and India, working with clients like Standard Chartered, Credit Suisse, American Express, CLSA, Natixis, Bank of Ireland and MUFG Securities. He specializes in regulatory reporting, anti-money laundering, client lifecycle management and investment banking. Yogesh also writes and speaks about these topics. Any spare time is spent with his daughter, jogging, reading or experimenting with new ideas.

About Luxoft

Luxoft is the design, data and development arm of DXC Technology, providing bespoke, end-to-end technology solutions for mission-critical systems, products and services. We help create data-fueled organizations, solving complex operational, technological and strategic challenges. Our passion is building resilient businesses, while generating new business channels and revenue streams, exceptional user experiences and modernized operations at scale.

luxoft.com