BrightScan

Achieving cost-efficient and effective software testing

Table of contents

Introduction

In this whitepaper, we share key insights gathered over the years while crafting a methodology to empower software delivery organizations to realize their full potential. These insights were predominantly acquired during assessments, also known as BrightScans, conducted for numerous enterprises struggling with the repercussions of the digital revolution and the organizational shift towards methodologies like Agile and DevOps.

A recurring theme among these organizations was their shared objective of wanting to enhance software testing to make it more efficient, mitigate the costs associated with poor quality, whilst maintaining effectiveness.

In response to this challenge, we have developed several effective solutions over the years. In this whitepaper we will primarily focus on the foundational steps toward achieving first-time-right software development. With a particular emphasis on sharing insights gathered from our BrightScan methodology.

Cost-efficient software testing

The prevailing misconception among decision-makers across all industries regarding software testing is the belief that it is an avoidable expense. They look at testing as an expendable cost, always seeking ways to minimize it.
But the opposite is true, software testing can generate substantial financial advantages, both directly and indirectly. And luckily many decision-makers within organizations share this perspective.

The main reason some organizations continue to perceive testing as a cost centre is their inability to manage it efficiently and effectively. Like many worthwhile endeavours, achieving cost-efficient and effective software testing requires a solid foundation and a clear organization-wide strategy. This necessitates an initial investment to bring software testing costs to a bare minimum whilst staying effective, resulting in significant benefits and cost savings across the organization.

All of this raises crucial questions:

- When should an organization make this investment?
- How to implement it?
- Where to begin and stop?

1. What is a reasonable cost for software testing?

Let's start with the basics. When talking about software testing costs, it's crucial to establish what constitutes reasonable expenditure. This determination depends on a multitude of factors, including:

- Business complexity
- Technological landscape and requirements
- Software significance (business-critical or not)
- Industry type (e.g., healthcare)
- Personnel (experienced or not)
- IT and team organiaztion (roles, availability, ratio, ...)
- Maturity level of the current environment
- And various others

Software testing should always bring financial benefits when developing software.

The above-mentioned factors contribute to the challenge of pinpointing a single figure to represent the cost of software testing. Investigations typically reveal that, on average, software testing costs fall within the range of 15 to 25% of the total project cost. However, in certain instances software testing expenses can rise much higher, constituting up to 50% of the total project cost.

Why is it so challenging to establish a reasonable cost for software testing?
Based on our experience, a good method for determining the reasonableness of software testing costs involves comparing these expenses to the costs resulting from production incidents. Or how we call it, the cost of poor quality. To do this, we must define the average cost of incidents in production, as thorough software testing plays a pivotal role in averting late-stage issues in the development life cycle. And consider all indirect costs beyond the rework by development, which is an essential factor in the cost of poor quality, but often forgotten.

Ultimately, software testing should consistently bring financial benefits during the development of software.

1.1 The cost of poor quality

Defining the cost of poor quality can be very complex. On the other hand, defining the cost of fixing various types of production incidents is a task that every organization should undertake, tailored to its own unique situation. The direct cost of such incidents is influenced by several factors outlined below.

Effort spent on:

- Root cause analysis
- Development to fix incidents
- Deployment of the fix incidents in various environments
- Regression and re-testing
- Adjustment of existing test sets (manual and/or automated)
- Adaptation of user manuals and other documentation
- Modification of analysis or user stories
- First-line support for affected users or customers
- Mitigation actions in production
- General communication

Additionally, it's essential to consider the cost of project delays resulting from diverted time away from other tasks.

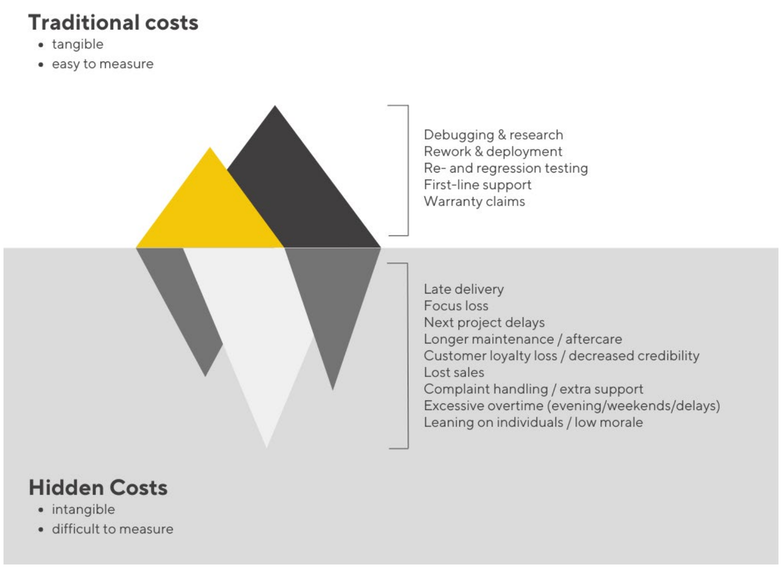
Creating first-time right software has never been more crucial.

The above does not consider potential revenue loss due to production incidents. Think of scenarios where errors lead to inaccurate price calculations or customers abandoning malfunctioning products. Moreover, software failures can severely tarnish a company's reputation, resulting in direct hits to stock prices and diminished customer loyalty.

Indirect cost associated with production incidents are challenging to quantify. These may include:

- Late deliveries and delays (even of next projects)
- Focus loss from current work
- Future loss in product sales
- Decreased credibility
- Negative reviews or publicity
- Internal frustration leading to people leaving

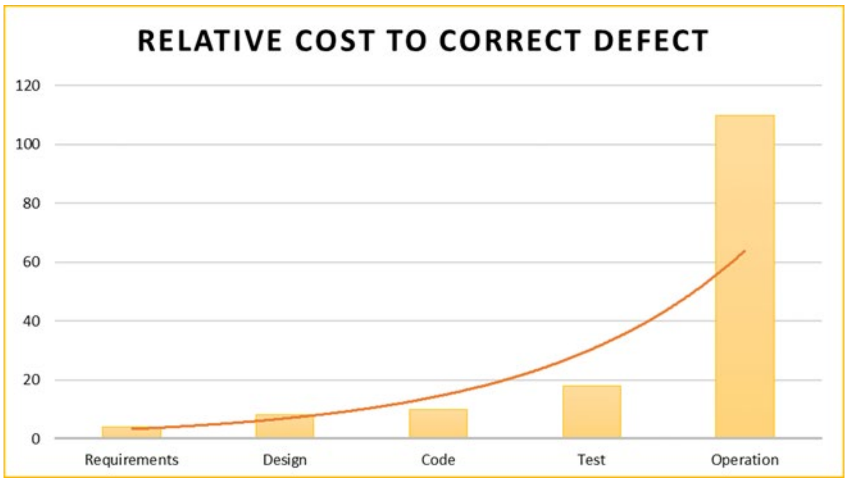
In today's landscape, ensuring first-time-right software, is more crucial than ever. Consider, for instance, the launch of a mobile application. If it fails to function properly upon release, it risks being abandoned by users indefinitely.


Traditional costs
• tangible
• easy to measure
Debugging & research
Rework & deployment
Re- and regression testing
First-line support
Warranty claims
Late delivery
Focus loss
Next project delays
Longer maintenance / aftercare
Customer loyalty loss / decreased credibility
Lost sales
Complaint handling / extra support
Excessive overtime (evening/weekends/delays)
Leaning on individuals / low morale
Hidden Costs
• intangible
• difficult to measure

1.2 Relative cost of defects

Over the past decades, the field of software testing has seen significant innovations. A numerous insights have been collected, and various studies have been conducted to better understand the dynamics of software development and testing. These studies have delved into a wide range of topics, including the relative cost of defects in software products.

This wealth of information is now widely accessible through various channels, from academic literature and industry reports to online articles and blogs. They provide valuable insights to guide organizations in their software testing strategies.



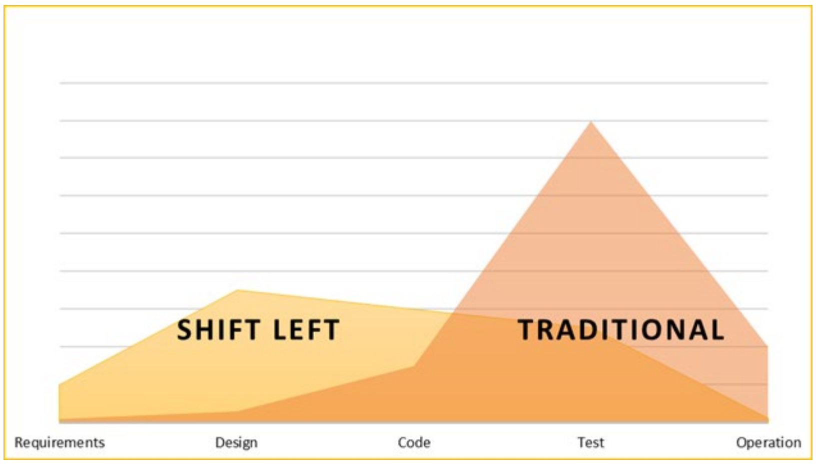
As you can see in the graph above, the relative cost of defects refers to the fact that the cost of fixing a defect increases exponentially as a product moves through the stages of development. Fixing defects in the post-production stage can be significantly more expensive than during the design, coding or testing stages.

1.3 Shift-left

Given these insights, it's common sense to focus on the early stages of software development to prevent as many defects as possible. So, one should find a higher number of defects before the testing stage than during it. This mindset shift may be challenging for some teams or organizations, but it does not require magic.

However, it's not simple either. If creating flawless software was easy, there would be no necessity for any form of verification or validation. Secondly, shifting left is not just a procedural or technical change, but a cultural one. It requires a mindset shift across the organization, not just within development and testing teams.

In a nutshell, the shift-left approach suggests that 'testing' should not be delayed until the final days leading up to a release. Instead, it should be shifted left on the project timeline, emphasizing the importance of frequent and early testing.



This doesn't imply that testing towards the end of the development cycle is forbidden. These testing stages should still be executed as planned. However, since most issues would have been prevented or detected earlier, they should be less significant and quicker to resolve.

The shift-left approach integrates testing and quality checks throughout the entire development life cycle. It ensures that all resources and roles involved in the project contribute to and take responsibility for the overall quality of the system or project.

Some practical examples to implement the shift-left approach:

- Testers participate in design sessions to inquire about customer usage, potentially leading to design modifications
- Testers pair up with developers to test new features on their machines, or the development environment, before they are developed
- Testers engage closely with developers to pose questions, generate test ideas, and devise 'what if' scenarios
- Developers execute extensive unit (integration) testing and code reviews
- Analysts write clear acceptance criteria and assist in testing them early on
- And so on

These are just a handful of examples. Depending on your organization's specifics, you can take many more steps to facilitate early testing and quality checks.

The shift-left approach primarily mitigates these risks:

- **Less resources and planning issues**, as testers play a more active role in planning and estimation.
- **Less effort and resources waist**, given that issues in requirements and design are identified and resolved at an early stage. Debugging and related tasks such as identification, localization, rectification, and regression testing become increasingly complex as more code is written and integrated with other systems.
- **Less technical debt** since defects aren't pushed to the next increment or version. There is more time to address bugs discovered early in the development lifecycle.

This mindset shift may be challenging for some teams or organizations, but it does not require magic.

The main advantages of shift-left testing:

- **Improved quality**:
  Since testing is conducted throughout the development process, there are more opportunities to catch and fix issues. This results in increased coverage and a higher quality product.
- **Cost savings**:
  Detecting and addressing issues early in the development process is more cost-effective than trying to fix them after the product has been released.
- **Increased efficiency**:
  Testing is integrated into every stage of development, which leads to more efficient workflows and shorter development cycles, saving time and resources It also enables a faster time-to-market with a high quality product. Additionally, efficiency and effectiveness can be increased by a wide range of tools and technologies supporting a shift-left approach, e.g. automated testing tools and CI/CD pipelines.
- **Better collaboration**:
  Shift-left fosters a culture of quality and better collaboration between developers, testers, business analysts, and other stakeholders from the early stages on, leads to better communication and understanding within the team, and shorter feedback loops. This allows for rapid changes based on real-time

insights, and ultimately a better product, which is more aligned with user expectations.

- Reduced risks:
  By catching and fixing defects early, it reduces the risk of project overruns, missed deadlines, and poor-quality products. Also, teams learn from their mistakes and prevent them from recurring in future development cycles, fostering continuous improvement.

Implementing shift-left requires a certain level of skill and expertise among team members. It's essential to invest in training and skill development to ensure that everyone is on board and capable of contributing to the shift-left approach.

2. Most common mistakes

Our extensive experience in software quality over the years has given us a lot of insights into how organizations handle software testing. We've gathered valuable lessons from successful initiatives, but perhaps we've learned more from initiatives that didn't quite hit the mark.

We've compiled the most frequent missteps organizations make when attempting to optimize the cost of software testing.

2.1 Underestimating the importance of testing and inadequate planning

As said before, some organizations tend to view testing as an afterthought or a necessary evil. Rather than an integral part of the software development process. This leads to insufficient budget allocation, inadequate testing, and potentially costly defects in the production environment.

Often linked to this underestimation of testing, is the lack of a proper testing strategy and plan, resulting in disorganized and inefficient testing. Leading to wasted time and resources, and increasing the cost of testing.

2.2 Lack of skilled testers

Often, organizations try to cut costs by hiring less experienced testers. While this may save money in the short term, it can lead to poor quality testing and higher costs in the long run.

For example, numerous organizations have opted to outsource (all) software testing tasks to offshore or nearshore partners. Lured by the immediate cost reduction of up to 50%. However, they often overlook the potential cost of poor quality, which can significantly exceed the initial software testing costs.

Sure, certain well-defined, repetitive tasks can be efficiently outsourced. But, as said before, cost-efficiency is rarely achieved by testing towards the end of the development cycle. If an organization doesn't consistently maintain an effective defect prevention strategy, the long-term costs will probably be substantial. It's crucial for organizations to retain control over testing activities and governance. So, they should invest in a clear test strategy and high-level expertise onsite to utilize offshore or nearshore resources effectively.

Another example is letting developers or analysts do the testing. Some organizations choose not to hire specialized testers, assuming developers or other team members can handle everything that involves testing. This is a misconception, it's essential to have dedicated testers on your team for the following reasons:

- Perspective:
  Testers have a different mindset. They view the system as a whole and consider various end-to-end and application specific user scenarios. Developers, being close to their own code, may be more focused on individual features or components.
- Expertise:
  Testers have specialized skills in creating test cases, identifying edge cases, and using testing tools and methodologies that developers may not have.
- Efficiency:
  By allowing developers to focus on writing code to create applications and testers to focus on identifying defects, both can work more efficiently within their areas of expertise.
- Unbiased testing:
  Developers may unconsciously avoid testing areas of the application, for whatever reason. Independent testers do not have this bias.
- User experience:
  Testers often act as the first line of users, providing feedback on usability and user experience that developers may overlook.
- Reporting:
  Testers are skilled in documenting defects, providing clear reproduction steps, and prioritizing issues based on severity and impact, which is crucial information for effective debugging and fixing.
- Risk mitigation:
  Testers are trained to think about and test for potential risks,
  helping to prevent future issues and improve the overall quality of the product.

This is why organizations should maintain this segregation of duties for optimal results.

2.3 Neglecting or overestimating automation (with AI)

Automation can significantly reduce the time and cost of testing, but it requires an upfront investment. Some organizations shy away from this investment, resulting in higher long-term costs due to manual testing.

On the other hand attempting to automate everything, trying to eliminate manual testing, is a flawed approach. While it may work initially, the evolving nature of software products impacts the maintenance of automated test sets. The focus should be on automating business-critical paths, repetitive tasks, and tasks that are time-consuming when done manually.

Effective automation requires significant effort, the right strategy, and appropriate tools. Initial investment can be high before reaping the benefits. Yet, it’s worthwhile if done correctly.

Many organizations are paying substantial licensing costs for test automation tools without understanding their coverage or return on investment. We’ve seen organizations automate everything without linking automated test scripts to system requirements. This leads to significant time wastage when errors occur. In some cases, test scripts were excluded or modified to pass, resulting in major production issues and a loss of trust in automated test scripts. Or even situations where automation was a bottleneck for a high-performing DevOps process due to the sheer volume of automated test scripts.

Over-automation can lead to a false sense of security. Aspects like user experience tests or parts of an end-to-end process, which are not automatable, can be overlooked.

2.4 Ignoring non-functional testing

Organizations often overlook non-functional testing, but it’s just as critical as functional testing. Non-functional aspects like performance, security, and usability directly impact the user experience and the overall quality of the software. Ignoring

these aspects can lead to serious issues that, if found in a live environment, can be highly expensive and damaging to fix and recover from.

Planning for these non-functional tests from the beginning is crucial. It should be part of the initial test strategy and design phase, not an afterthought. This way, any issues can be detected and fixed early in the development cycle, which is significantly less costly and less risky than fixing them after the software has been released.

2.5 Poor communication

Often, we encounter environments where testers, developers, and other (business) stakeholders aren't communicating effectively. Misunderstandings lead to incorrect development, testing, missed defects, and rework, all of which can inflate the cost of development and testing.

Including testers from the beginning of the project is crucial. They can provide valuable insights during the requirement gathering and design phases, helping to prevent ambiguities and misunderstandings that can lead to defects later on. Early involvement of testers also allows for the creation of a robust testing strategy, aligning it with the development process and business requirements.

Business stakeholders also play a vital role. They need to be fully invested in their projects and take ownership. They should clearly communicate their needs, expectations, and any changes in requirements. This helps ensure the development and testing teams have a clear understanding of what needs to be achieved. Regular meetings, constant communication, and transparency among all parties can facilitate this process.

Remember, software development and testing are team efforts. Everyone involved, from developers and testers to business stakeholders, plays a crucial role in the project's success. By fostering a culture of open communication and collaboration, you can greatly improve the quality of your software and the efficiency of your development and testing processes.

3. How to get started with cost-efficient testing?

The situations mentioned above are far from ideal, and unfortunately, they are realities in many organizations. In such organizations, software testing is often viewed as an avoidable expense.

Move well before moving fast.

The rationale behind this is straightforward: to maximize cost-efficiency in software testing, your organization needs a customised, solid foundation and a coherent, organization-wide testing strategy. The best way to establish this, is by examining the processes, methods, people, and tools involved in the software development lifecycle. It boils down to a simple yet potent mantra: move well before moving fast.

The initial step is to start moving well. We've discovered that an outside-in perspective of your software development organization and testing activities is the most effective way to lay a solid foundation. By considering the organization's unique needs, allocated budget, and required maturity level, it's possible to devise a custom approach based on best practices for any organization.

BrightScan

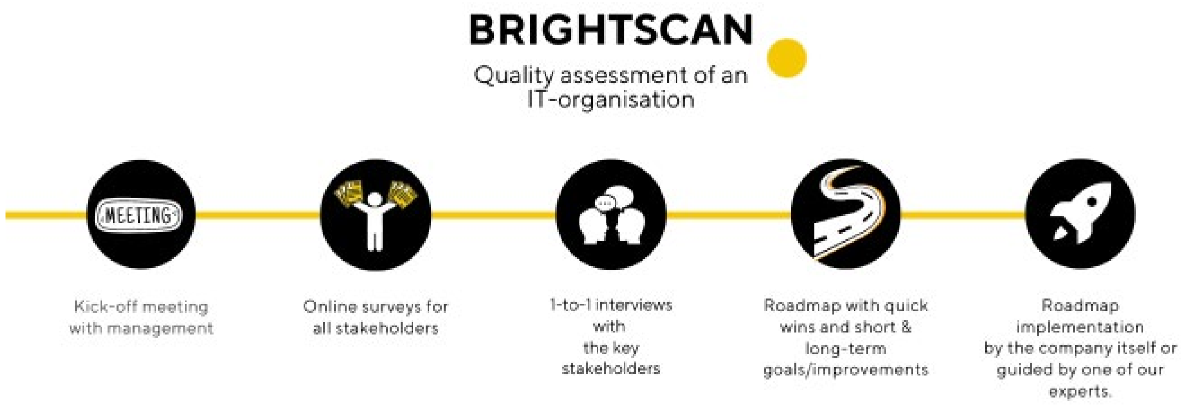
A BrightScan is a highly effective evaluation carried out by a team of experts, aiming to provide an objective analysis of an organization's current software quality structure and to identify future enhancement initiatives. This enables organizations to elevate their testing maturity, accelerate their time-to-market, while minimizing their quality costs.

By addressing needs rather than problems, we're able to offer organizations valuable advice on enhancing their operational processes.

Being an independent specialist, Brightest is not restricted to any specific vendors or methodologies. This allows us to concentrate on what's best for your organization, seeking durable and cost-effective solutions that are tailored to your specific needs. By addressing needs rather than problems, we're able to offer organizations valuable advice on enhancing their operational processes.

1. Efficient process

Over a period of 2 to 4 weeks, we carry out online surveys and engage in discussions with stakeholders to gain a comprehensive understanding of the strengths and weaknesses of the existing (test) organization. Depending on the specific requirements, our focus extends beyond testing to encompass the entire process and delve into more technical aspects.


BRIGHTSCAN
Quality assessment of an IT-organisation
MEETING
Kick-off meeting with management
Online surveys for all stakeholders
1-to-1 interviews with the key stakeholders
Roadmap with quick wins and short & long-term goals/improvements
Roadmap implementation by the company itself or guided by one of our experts.

1.1 100% or partially remote

Our specialists have meticulously refined the process mentioned above over the years, ensuring that every step in the BrightScan procedure can be carried out remotely. In addition, our specialists have extensive experience working in remote settings. Many organizations today collaborate with offshore or nearshore software development teams. The combination of our robust process and skilled specialists means we can guide any organization through a valuable assessment, regardless of geographical boundaries or time zone differences.

1.2 100% tailored

Each organization faces unique challenges. Our approach incorporates a holistic perspective, focusing on various aspects tailored to benefit your organization. For instance, if an organization seeks a more technically driven assessment, we can readily modify our process to meet the organization's specific needs.

We are knowledgeable about various types of organizations and can customize our results to suit their unique needs:

- Startup:
  Might need focus on scalability and speed of delivery, and a scalable testing strategy that can grow with the company.
- Large enterprise:
  Might need focus on improving collaboration and efficiency across multiple teams or departments.
- Software company:
  Might need focus on improving code quality and reducing bugs.
- Healthcare company:
  Might need focus on ensuring regulatory compliance and patient data security, whilst wanting to improve efficiency and cost of testing.

By tailoring the approach to the specific needs and context of each organization, we can provide solutions that bring the most benefit to that organization.

2. Result driven

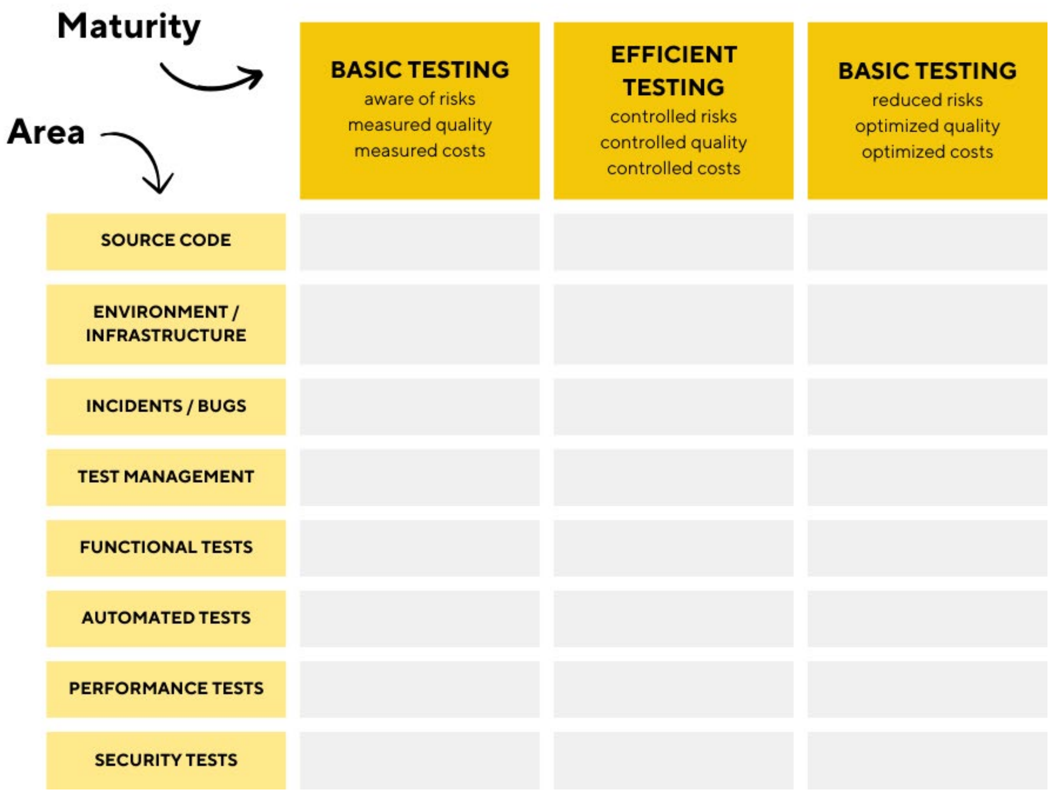
Following a thorough evaluation, we generate a strategic roadmap that highlights immediate gains (quick wins) and outlines objectives for both the short and long term. Our goal is to tie each objective to a quantifiable value for the organization, giving insight into the Return On Investment (ROI) for each enhancement.

2.1 3-step maturity model

We use a 3-tier maturity model to determine the current state of your testing organization and outline the next steps in the roadmap. In general, advancing to a higher level isn't feasible without mastering the preceding one.

Achieving proficiency at the basic level enables any organization to guarantee a minimum standard of software delivery quality. This might be sufficient, either for the time being or even for an extended period. Everything hinges on your organization's current circumstances, aspirations, and capacities.

However, it's entirely possible to excel in one aspect while giving less attention to other aspects. But this could also potentially signal that your organization's focus might need efficiency improvements.



Maturity Area	**BASIC TESTING** aware of risks measured quality measured costs	**EFFICIENT TESTING** controlled risks controlled quality controlled costs	**BASIC TESTING** reduced risks optimized quality optimized costs
SOURCE CODE			
ENVIRONMENT / INFRASTRUCTURE			
INCIDENTS / BUGS			
TEST MANAGEMENT			
FUNCTIONAL TESTS			
AUTOMATED TESTS			
PERFORMANCE TESTS			
SECURITY TESTS			

Every organization should aspire to reach the third level of maturity, in its own time. Our experience shows us that progressing to the next maturity level involves organizational change, which doesn´t occur overnight. The larger the organization, the more time and resources are required to ascend to the next level.

The following table presents realistic timelines for organizations or teams ranging from small to large:

ORGANIZATION TYPE	BASIC TESTING	EFFICIENT TESTING	CONTINUOUS TESTING
SMALL (1 - 2 agile teams)	1 - 3 months	3 - 9 months	9 - 15 months
MEDIUM (3 - 8 agile teams)	3 - 6 months	9 - 18 months	15 - 36 months
LARGE (>8 agile teams)	6 - 9 months	12 - 24 months	24 - 36 months

The timelines mentioned above are influenced by various factors, such as

- The number of distributed teams
- Expertise level
- Individual experiences
- Budget
- The willingness and support from the C-Level executives

A significant factor in successfully implementing this type of organizational change is integrating software quality into the organization's strategy.

2.2 Improvement areas

The identified areas for improvement are associated with 4 key enhancement areas. Crucial questions are addressed for each during the analysis phase.

Organization
People
Methodology
Tools

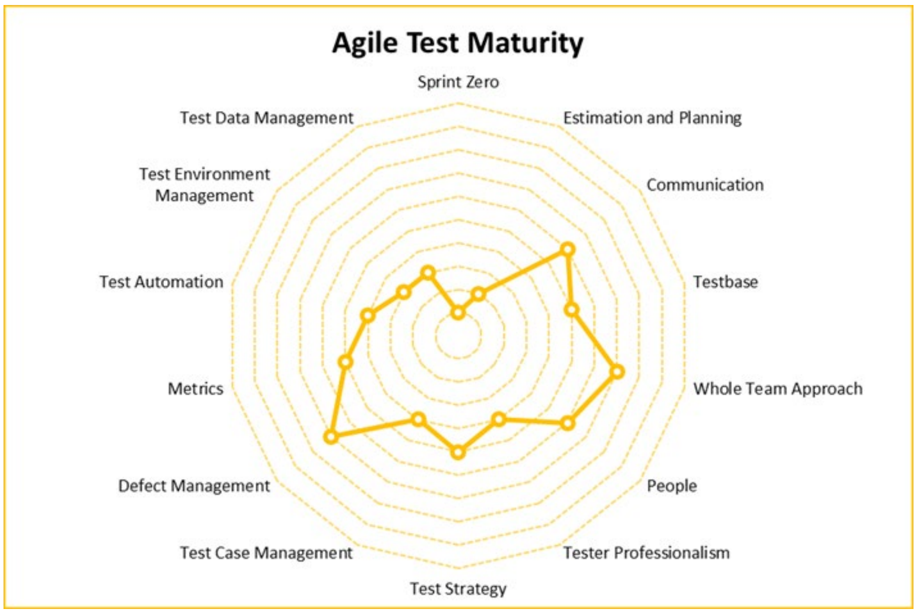
- Organization:
  How can the existing organization structure be enhanced to suport a more efficient strategy?
- People:
  What improvements can be made to the current human resources to align them with the future work methodology?
- Methodology:
  What methods can be optimized or implemented to facilitate efficient software delivery?
- Tools:
  What are the optimal tools to improve efficiency and software quality in the forthcoming roadmap?

2.3 Implementing assessment results

Software delivery organizations can either self-implement the proposed roadmap or enlist expert support for parts of their transformation.

A key advantage of the BrightScan methodology is its ability to translate findings into manageable, actionable improvement steps. These steps are connected to the 3- tier maturity model, ensuring appropriate prioritization.
This is why software delivery organizations can either self-implement the proposed roadmap or enlist expert support for parts of their transformation journey. It's important to note that some improvements will need to be addressed by the organization directly.
We advise designating a responsible individual within your organization before

initiating a BrightScan. This person will be in charge of implementing the roadmap after the results are shared, with the option of guidance from hired experts, if necessary.



3. Top 3 advantages

3.1 Powerful side effect

The BrightScan process has demonstrated its effectiveness in fostering employee engagement within software delivery organizations. It appears to heighten everyone's focus and appreciation for software quality, particularly those involved in any part of the software delivery process. The proposed improvements and long-term roadmap are perceived as solutions to their personal challenges, not just those of the test team, the organization, or management. This sense of ownership and relevance enhances their commitment to the process.

3.2 Broader than software testing

Even though our surveys and interviews are primarily centred on software quality delivery and testing, they often provide deep insights into the broader aspects of the organization. By offering an impartial and independent platform, we enable individuals to express their challenges and concerns about their roles or situations. This often reveals a great deal about the organization's processes and methodologies, or the absence thereof.

Thus, a BrightScan usually generates additional valuable insights for the organization, alongside recommendations on software testing-related activities and strategies.

3.3 Success guaranteed

Thanks to the built-in success factors of the process, the resulting quick wins and long-term objectives will be specific and backed by all key players in the software delivery process. Even if the organization doesn't manage to implement all suggested improvements, they will at least have accomplished the most critical steps towards enhancing their working methods and communication.

More information

A BrightScan can be conducted for a wide range of organizations, from startups to multinational corporations. The pricing for a BrightScan varies based on the organization's complexity, and the number of people needed in the workshops.

Read all about how we can help you on our website.

For more information, don't hesitate to reach out to us via info@brightest.be.