ACT 2 LEAD

Software Testing Leadership Handbook

Marko Rytkönen & Kari Kakkonen

Marko Rytkönen & Kari Kakkonen

ACT 2 LEAD

Software Testing Leadership Handbook

1st edition, published on 23.9.2024

Design: Marko Rytkönen and Kari Kakkonen
Cover: Adrienn Széll
Authors: Marko Rytkönen and Kari Kakkonen

Publisher: Dragons Out Oy / Leanpub.com

This book is for sale at: https://leanpub.com/act2lead
ACT 2 LEAD homepage: https://www.act2lead.net/

Helsinki, Finland, 2024
ISBN 978-952-65567-0-3 (PDF)
ISBN 978-952-65567-1-0 (EPUB)

Table of contents

Acknowledgements

We want to thank our families and loved ones for all the support and help you have given us. We also want to thank all our pilot readers. Thanks to your feedback, the book gained more professional depth and different perspectives. Your feedback also encouraged us to continue writing and publishing the book. Special thanks to Esko Hannula, Mikko Vaha and Fredrik Åström. Thanks to Karoliina Airaksinen, Veikko Lohi, Leena Saari and Matti Vuori. Thanks to Emilia Rytkönen and Julia Rytkönen. Thanks to Phil Royston, Debra Friedenberg, and Sue Atkins for reviewing and improving the English text.

Thanks also to our employers, Hidden Trail and Knowit, for their great attitude towards our book project.

Thanks to the cover illustrator Adrienn Széll.

Thanks to Reino Myllymäki for the help with the layout.

Thanks to you, the reader, for deciding to learn more about testing by reading this book!

1. Foreword

Testing - one of the most misunderstood activities in software development?

We don't know if the above statement is true. It may be a bit of an exaggeration, but in our experience, it contains a grain of truth. And it's no wonder because testing is easy to misunderstand. Many people test or lead testing without having studied it.

We have written this book for anyone who wants to understand what testing and testing leadership are all about. A better understanding will help to lead and improve your testing - whether testing is outsourced or in-house. CEO, CIO, or CTO - this is the testing book for you, too! It's also for anyone who works with software: product owners, project managers, system analysts, developers, testers, test managers, and many others, including students. So, this is not a book that teaches you how to test.

The book is written as individual questions so that it can be read chapter by chapter and in any order. We have chosen the questions based on our experience. There are not always people to ask for support for testing at work. The book then acts as a support and sparring tool, a source of new ideas and options, and a confirmation of your thoughts. We have also included our own experiences we have seen or heard to help clarify the text.

Remember that the book is not full of truths. Instead, it is full of our ideas and common knowledge about testing. We don’t claim that our answers are absolutely correct or comprehensive, as there are many opinions. The answers may also change over time.

We have learned about testing and software development through work and by studying it through various media. We have read dozens of books and attended many conferences and events. We've had countless discussions and learned lessons from others. With these experiences, you don't often hear the source of information. We have included the sources that we know. At the end of the book, we have listed courses, books,

conferences, and other sources of information from which we have learned.

We hope that the book will help you consider testing in your daily work and decisions so that you will ultimately see testing reflected in your business.

Remember to ask questions and challenge everything you read, including this sentence - as a good tester should always do.

2. How did Marko become a leader in testing?

I graduated from Kuopio Technical Educational Institution in 1992, and back then, I thought I would become a PC support person. Those people were often hired at that time because computers were rare, and their use required new skills and help for users.

I had been using computers for about ten years by then and had been able to help my relatives and acquaintances with them. My first computer was a VIC-20, and at that time, computer games were mostly written by ourselves. The source code was read from a computer magazine and written line by line on the computer. The game had to be tested to see if the code was written correctly. This was the beginning of my career as a software tester, although I didn't know it at the time.

But I didn't become a PC support person. In 1994, I got a job as a software test engineer at Ericsson, where my job was to test telephone exchanges. Since then, my responsibilities have changed many times, but I have always been involved in testing and quality assurance. I have tested different products, managed and led software testing in international projects, been responsible for improving testing at the IT department level, managed in-house and outsourced testing, and created, sold, and bought testing services. I have also led an international testing business.

I've had the chance to work for good companies on interesting projects with new technologies and surrounded by great people. They have taught me a lot about testing and software development. Without them, this book wouldn't have been written. In fact, the lessons in this book are very much what I have learned and heard from my colleagues during my career. I have learned by doing and by making mistakes. I want to share these lessons and ideas through the book and offer readers “Aha!” moments to help in daily work.

On my first day at Ericsson, I learned that the dumbest question is the one you don't dare to ask. I have tried to live up to this lesson and asked

many questions throughout my career. Sometimes I have asked even if I knew the answer. Sometimes, the questions are more important than the answers, as questions force us to think, and thinking leads to new insights. It is from these learnings that the idea for this book and its structure was born.

3. How did Kari become a leader in testing?

I got my first computer in 1984. It was an Amstrad CPC 464 with a cassette drive. I was immediately drawn into the world of computers. The school computer club had Apple II computers, and we wrote programs using the BASIC programming language. I tried to get the software I had programmed to work and learned to test at the same time.

I got a seat at the Helsinki University of Technology to study industrial management. I was interested in computer science, so it was an obvious technical minor. I learned to program in three different programming languages and read about data structures and algorithms. In programming exercises, I tried out different ways to see if the code worked or not. Later on, I learned that this was unit testing. The last chapter in one of the textbooks was about testing, but reading it was optional. In 1996, when I went to work at the Software Engineering Center alongside my studies, I was tasked with selling software and doing consulting work. I particularly fell in love with a test automation software called SQA. I also took a summer course in software testing at the Helsinki University of Technology.

I was enthusiastic about improving the world through the student organization AIESEC. I wanted to improve the world in my field and found software testing as a way to do that. Software testing could improve software quality, and good quality software doesn't kill people as some of the famous software defects used to occasionally do. So, I decided to devote my time specifically to software testing.

In 2002, I was able to focus fully on testing after changing jobs to a startup company called Conformiq. At the same time, the world was ready for testing communities, and I co-founded a testing community in Finland, TestausOSY (Finnish Association of Software Testing[1]). I also

[1] https://testausosy.fi/

co-founded the Finnish Software Testing Board (FiSTB[2]), a founding ISTQB member[3]. ISTQB has grown to become the largest testing organization in the world and has offered me many attractive international opportunities.

It was great to see that testing and quality were getting more attention. Unfortunately, quality was all too often overshadowed by project schedules. I promoted the testing and quality but didn't limit myself to those. I also improved the development process as a whole so that testing could start early enough. I became an advocate of Agile and DevOps.

[2] https://fistb.fi/en/
[3] https://www.istqb.org/

4. What does testing mean?

Testing is checking and exploring a test object and its environment by appropriate and case-specific means. It aims to help others to improve quality by discovering new and relevant information about the test object and its working environment. This information, i.e., the test results and other similar information, should be passed on to those who will decide on the next actions. These decisions may relate, for example, to whether to fix a defect or change the feature or the project schedule. These decisions are not typically made by the testers but are mainly the responsibility of leadership, the product or project manager, or the entire software development team. These decisions are made to improve quality and business and to satisfy customers. Through improved understanding, the quality of the test object can be improved. Testing, therefore, delivers value to the customers and the business, just like all other software development work.

Testing is a reactive activity. It is done when the test object and its defects already exist. For example, a document is reviewed when it has already been written (fully or partly), and functionality is tested when it has already been programmed and deployed into a test environment for testing. Testing in itself doesn't improve the quality of the test object, but it does improve people's understanding of the test object. Neither does it decrease quality since the defects in the test object already existed before testing started. Quality will be improved or decreased only if the product or its environment changes.

> *Once, a leadership team discussed what testing is about. The team said that testing ensures and improves quality. They were asked to test a coffee cup for a while and tell how the quality improved. The team found that testing only improved their understanding of the cup's quality. The defects in the cup were already there before they started testing, and the defects were not fixed with testing.*

The test object is often software, but it can also be a service, document, a requirement, data, people's understanding or business idea, or how users use the product. In other words, testing can, and indeed should, be done all the time, for example, during planning, design, and production use. Testing is, therefore, an ongoing activity, not a time-bounded phase.

Testing consists of two different work-types: checking and exploring[4]. They are not alternatives to each other; they complement each other. In our experience, checking is often easier than exploring, and good testers are distinguished by their ability to explore the product.

Checking focuses on verifying known or desired cause-and-effect relationships, such as whether the software works according to documented requirements or business processes. In simple terms, once the username and password are provided and the login button is clicked, you check whether the user is redirected to the home page according to the requirement. Due to the nature of the work, test automation can be utilized for checking so that it can be done quickly and at any time (see Chapter 32). Checking alone can create a false sense of certainty of the quality.

Exploring, on the other hand, explores, challenges, and questions the test object, focusing on finding unknown cause-and-effect relationships. This is needed as only a small proportion of causal relationships are known beforehand or are disclosed in the requirements. To simplify, it is not the functionality of the login button that is tested but the login feature as a whole. What exactly happens when you log in to the service? Does the login work when done in different environments? Humans are good at exploring by nature as we explore the world around us all the time. Exploring often reveals more observations and defects than checking.

[4] Hendrickson, 2010

There are an infinite number of possible tests, but time for testing is limited. Therefore, we have to make some assumptions in testing based on limited information about the test object.

> *You may have test-driven the car before deciding to buy it. As you have only test-driven it for a short time, and under the current conditions, you may have made some assumptions. For example, if you test-drove a car in the city in summer, you may have assumed that it would be as good to drive on the highway in winter. You may realize afterwards that your assumptions were wrong. If you found problems during the test drive, you wouldn't have had to make similar assumptions because then you would have known for sure that the car had the problems you found. The test drive neither improved nor decreased the quality of the car. But it was very valuable for you as it gave you more insights about the quality of the car for the purchase decision.*

Sometimes, the assumptions are correct, and sometimes they are not. For example, if a login feature works today, we can only assume it will still work tomorrow. Or, if the login is done in a different environment than the one where it was tested using a different browser or username, we can only assume that the login will still work.

> *In 2019, there was a weird bug in the Finnish Tax Administration's systems.[5] Out of a batch of 27,000 letters printed, 2,836 were incorrect. All letters were sent successfully, but the incorrect ones contained incorrect personal data. According to the newspapers, the bug was caused by an overflow of the maximum value of a variable in the printing software. From a "source code and its variables perspective", each printing transaction is unique as each print uses different data and variable values. The assumption that every printout always works in the same way wasn't true.*

[5] https://www.ts.fi/uutiset/4667142

If we want full confidence that the software will work correctly under all possible conditions, we should test all those possible conditions and combinations. In practice, there is never enough time or money to do so. Testing is sampling[6], i.e., only a subset of all possible tests are selected for testing. The selection is based on factors such as expertise, risk, current understanding, or even the time available to test. Each tester tests individually, selecting and executing a different set of tests to the ones selected by other testers. Just as different users use a product in different ways, then different testers will find different defects. Different types of software are also tested differently based on factors such as business and risks. For example, a medical device is tested much more deeply than a free game. The company's quality culture also affects the quality and amount of testing.

[6] Weinberg, 2008

5. What is test leadership?

Test leadership means taking care of testing and testers throughout the lifecycle of the test object. It aims to ensure that testing is diverse, timely, and of good quality aligned with the business need. Testing leadership cannot be separated from other leadership work, so testing cannot be siloed from the rest of the business. It helps when leaders understand that software development and testing are problem solving processes (to understand and solve user needs) and not just a delivery process (to build a predefined product).

Testing should be led at an organizational and operational level. These cannot be separated; both are needed. At the organizational level, i.e., company or department level, testing can be led by a Head of Testing. At the operational level, i.e., product, project, or team level, testing can be led by a Test Manager, Test Lead or the whole software development team together. The line between these two levels is blurred, and the differences depend on the context, such as the company's size. Both levels can include tasks such as:

- Enabling testing
- Advocating testing
- Recruiting testers and selecting suppliers
- Acting as a superior for testers
- Leading learning and competence development
- Budgeting
- Line management and coaching
- Change management

Leading testing is, first and foremost, about leading people. As always in leadership, each person must be led in a way that is appropriate to their competencies and aptitudes. This can be considered through the three-level Shu-Ha-Ri learning model[7]:

[7] Adkins, 2010

- Shu, "follow the rules": When a person doesn't know how to test, they are at the Shu level of testing. They need detailed instructions on what to do and how to test. They cannot question the instructions because they don’t know the underlying theories and principles. At this level, the test process, detailed test cases, and similar instructions may work. A tester at the Shu level should be taught how to test in practice.
- Ha, "break away from the rules": when a person has been testing long enough by following given instructions, they start to question the instructions. Why use detailed test cases? Can I test without test cases? They learn new test approaches (see Chapter 25) and the underlying theories and principles. Testers learn outside their work too, for example, from other testers, a course, or this book, and they put new ideas and learnings into practice. A tester at Ha-level must be coached in testing and learning.
- Ri, "go beyond the rules": once a person knows the underlying principles and has used new practices, they will learn from their work and experiments. They create the best ways to test in their context and may not know why they test in the way they test because testing and decisions come naturally. A tester at Ri-level should be mentored and encouraged to find the best ways to test. It helps when the working environment is safe so everyone can try different ways to test without fear of failure.

At any given time, the leader will need to understand which of the above levels the people being led are at so as to support and lead them correctly. Leaders should also support people to grow towards the Ri-level. Leaders must also be aware of which level they personally are in terms of know-how and skills in testing so that they don’t require others to perform actions that would only help the leaders themselves.

> *A group of directors asked a test manager to create one generic test process that dozens of development teams would use. The test manager didn't think a generic test process would help the*

teams. Later on, the test manager realized that it was because these directors were at Shu-level in testing. If the directors had been the ones doing testing, then the generic test process might have been useful for them, but it wouldn't have helped the professional testers in the development teams, because they were at the Ha or Ri-level. The generic test process demanded by leadership would also have removed the responsibility for quality and testing from the development teams.

5.1 Organizational test leadership

Test leadership at the organizational (company or department) level is strategic test leadership. The aim is to ensure that the organization understands what testing is about and why it is essential to invest in it so that testing is considered in all decisions. It involves defining the short-term and long-term visions of testing, advocating for it, and enabling it by influencing and challenging people. Often, the aim is also to create visibility of the quality of testing across the organization and to support people in improving testing practices.

The work may involve the creation of an organization-wide testing guideline, for example, in the form of a test strategy. The strategy can include vision, objectives, general principles, and practical tips on what to consider at an operational level, i.e., in all projects and teams. These may include policies on recruitment, subcontracting, tools, standards, and regulations to be considered in testing, test environments, and test data management. The strategy should be regularly updated and communicated to the organization, including leadership. Implementing the strategy often requires repetition of the message (strategy, guidelines), discussion with people, and follow-up to ensure that needed actions are taken.

Test leadership belongs within all leadership levels and existing decision-making mechanisms, budgets, and leadership teams. It requires actions and decisions that cannot fully be delegated to others. These

decisions can relate, for example, to testing budget, recruitment, selection of subcontractors, and competence development. Successful test leadership requires at least a strong understanding of modern testing practices and sometimes even practical knowledge.

Bigger organizations may have one person to lead testing at an organizational level. Their title might be Head of Testing or QA Lead. This role should be defined as one of the company's core competencies if testing is important to the business and if there is a need for continuous testing. This person can be subcontracted if an in-house employee cannot be hired. The person's job description may include the following tasks:

- Consultation and discussion of business needs to ensure that they are considered in the company’s test policy or strategy.
- Creating and communicating common testing guidelines.
- Defining short-term and long-term vision for testing and challenging people towards them.
- Mentoring and supporting managers, test managers, testers, quality assistants, and superiors working at the operational level. Helping them with testing-related daily issues and matters, such as reviewing contracts and plans (project and test) and assisting with recruitment, subcontracting, and competence development. When supporting, it is important that responsibility for testing remains at the operational level.
- Advocating for testing.
- Improving understanding and skills related to testing.
- Selecting testing vendors and improving testing in cooperation with them.
- Creating visibility of the quality of testing practices across departments and teams.
- Acquiring and maintaining common testing tools and ensuring the necessary competence development and training for tool usage.

- Supporting career development of testers and those wishing to become skilled testers.
- Organizing information sessions on testing and quality assurance.
- Building and running a testing community (see Chapter 28.2).
- Managing a testing center of excellence. The center of excellence may be a home base for all testers and quality assistants in the company who are working in different development teams across departments.

Kari analyzed the client's testing practices. Based on the analysis, the client wanted to put more effort into organizational-level test leadership. A head of testing was subcontracted to promote testing culture and mindset to the company's leadership and development teams. The first tasks were to create a test strategy and testing community. As a result, more people became interested in testing, and testing became more professional and efficient. Later on, the consultant was replaced by the company's own personnel.

5.2 Operational level test leadership

Operational-level test leadership puts testing into practice in day-to-day work so that it is aligned with the context and organizational-level test strategy. This may involve resourcing, planning, organizing, improving testing, and managing testers, for example, for a software project or product. As with organizational-level test leadership, operational-level test leadership should be part of the other leadership work for this level. Testing should be led with the same timings and practices as the project, product, and software development leadership. This enables testing to become a part of the other software development activities so that testing is not siloed from the other work.

The leadership of the software development process, such as the business area manager or product manager, usually decides on the biggest testing policies, such as resourcing and budget. They don’t usually

lead the day-to-day work of testers at the operational level but may act as a superior for testers and test managers. The head of testing at the organizational level often works in collaboration with leadership at the operational level.

In a traditional waterfall software project, a test manager often leads testing. They may plan the testing and lead a separate testing team. They often monitor and report on testing progress within their team and coordinate testing with business users involved in testing and other development teams.

In an agile team, test leadership can be the development team's responsibility, so a separate test manager may not be needed at all. The team must then have sufficient test management skills and decision-making authority. Leadership is facilitated by people's autonomy and transparency in testing and software development. The team must find the best ways to test and deliver good quality software for the business. The team can have a dedicated test lead who ensures all necessary testing is done on time. This can be a permanent or rotational role, either part-time or full-time.

5.3 ACT 2 LEAD heuristic

We have summarized test leadership in the ACT 2 LEAD acronym, a heuristic for test leadership. It reminds us that test leadership means taking action; words are not enough. No matter how smart the people around you are, you just cannot completely outsource or delegate test leadership.

The acronym contains eight main points to keep in mind when leading testing.

- **Add**: Add testing to everything you do so that testing is not a separate function, phase, or process. Lead it as a holistic effort.
- **Context**: Understand your context and lead testing accordingly. There is not just one correct way to test or lead.

- **Transparency:** Demand and create visibility into testing, regardless of whether testing is outsourced or not. The better the visibility, the better it can be led.
- **2**: Ensure that both humans and test automation test your software. Humans are good at exploring the software, and test automation is good at checking the software - both are needed.
- **Learn**: Learn to test and test to learn. Use learnings to improve testing and test leadership. Learn also to improve the software being tested and the business.
- **Enable**: Enable good quality testing by creating a quality culture where the importance of testing and quality is understood. Boost the culture via your daily work and the decisions you make.
- **Adapt**: Adapt testing to risks. Focus testing on high-risk areas and mitigate risks by testing those areas first.
- **Diverse**: Ensure that testing is diverse. The more diverse testing you have, the more diverse information you will get.