Altom

HOW TO GROW AS A TESTER

A Learning Guide Used at Altom

by Alexandra Casapu

TABLE OF CONTENTS

WHAT TESTING IS AND IS NOT ABOUT

Testing is not about applying a magical recipe over and over again.

Testing is thinking. It's learning how to analyze and solve all sorts of problems.

If testing is thinking, how do you learn how to test? You learn to think better. You learn to learn better. That's not easy stuff. And it sounds like it's not so specifically, uniquely related to testing.

Looking around, I see testers get information from a wide variety of domains. When I scan the bookshelves in our office, I see a lot of books from the area of social sciences, programming, management, critical thinking, business, systems thinking, math... And I think most of them can be useful in my testing work in one way or another. However, there is no definitive rule on how much to use which.

Every tester I met has a unique background and set of skills. I don't know of any recipe for becoming a tester in one day/week/month/year/etc., because there is no standard tester that I know of. I think it's unrealistic to expect a standard script to work in shaping a person that operates in such a creative domain.

But still, if you are a tester at the beginning of the road, where do you start? What do you focus on in order to improve?

You have control (and responsibility) over where you want to start and the direction you want to improve in. There is no universal starting point. However, there are sets of ideas you can get inspired from, and adapt to your own interests. This is what we want to offer here.

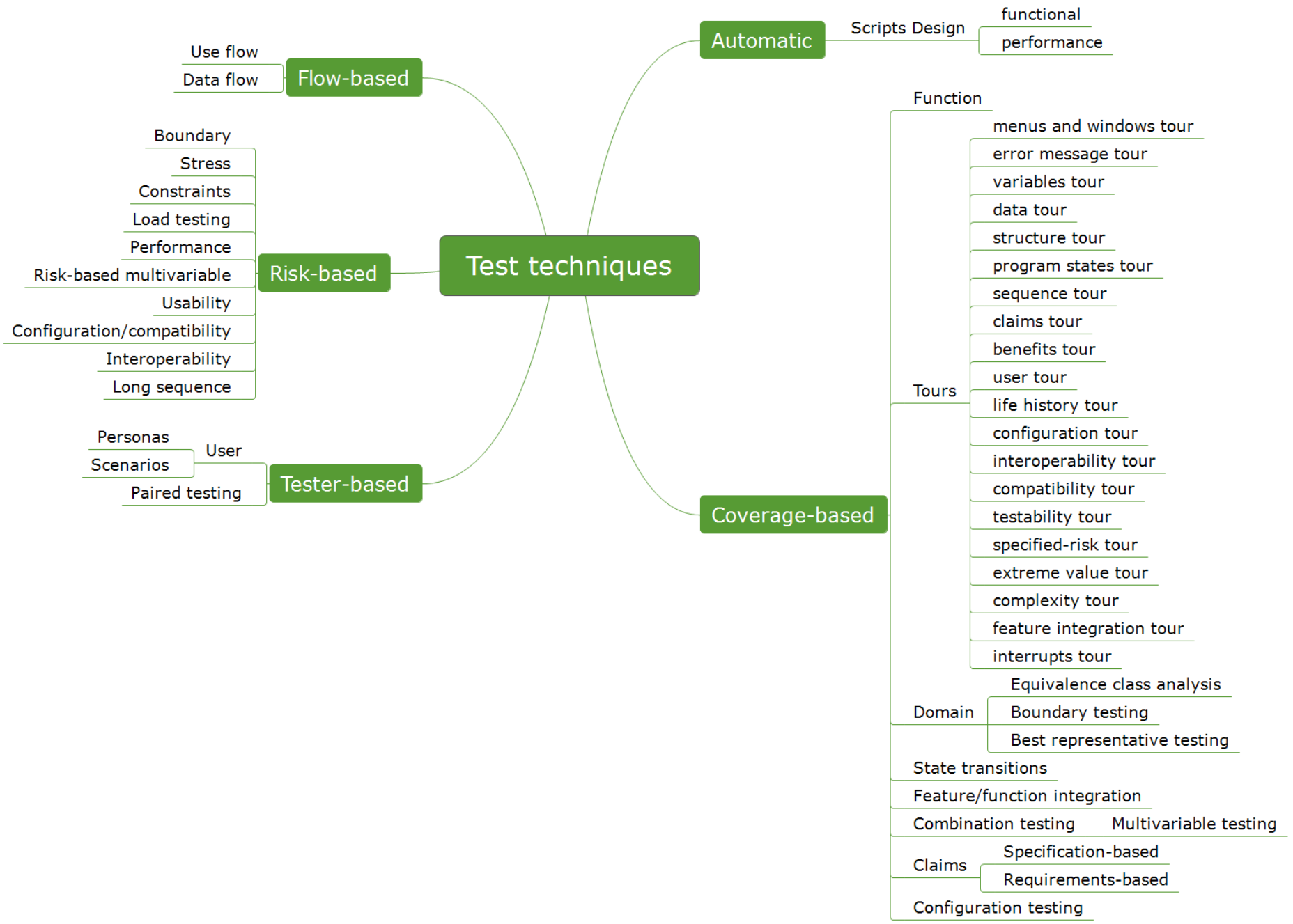
AN IMPORTANT GUIDELINE:
Regard these resources with critical eyes and ideas.

WHAT HELPED ME IMPROVE IN TESTING

A good source of inspiration for me is looking back at what I did, in particular what worked for me, or I thought was helpful. This is not a definitive list, it's always changing as I get new experiences.

- **Request feedback on my work** from people whom I trust, and who can give me useful feedback.
- **Make my work visible, to make receiving feedback easier** – offer to create and share a dashboard with the areas, charters or mini-missions I covered.
- **Code my own projects.** This helps not only to become better at coding tasks, but it also helps a lot to better understand the product I'm testing. As I learn more from programming on my own projects, I discover how to interpret the behavior of the program when I'm testing; or to search for test ideas. Because I understand from the inside in what ways software can fail. It doesn't have to be a super ambitious project. It can be as small as a script that helps you create test data faster, or a tiny web app with a server and a client side.
- **Read books** to understand better the technology I work with, my thinking, testing approaches and techniques.
- **Participate in online courses from connected fields that interest me** - e.g. on Coursera - courses on algorithms, machine learning, model thinking, behavioral psychology.
- **Watch talks from other testers** to find out how they test and how they tackle challenges.
- **Make lists of things I missed and revisit them.** From time to time I would miss a bug, or miss a relevant detail about a feature. So I created a list where I would note down some of these, which I thought were interesting and could stumble upon again. I tried to note down the underlying thing I missed, or a more general question I failed to ask, so that I could check the list when testing in other parts of the project and apply the ideas there as well.

- **Choose a test technique and focus on it for one or more testing sessions.** Then pick another one. Think of using a variety of test techniques. To help me choose the techniques, I studied several lists of test techniques I found and categorized them in a way that made sense to me. I found some in the book Lessons Learned in Software Testing, and also in the Domain Testing course. Starting from those, I created my own list of techniques I wanted to practice with and thought were applicable on my project:



Q&A WITH A TESTER AT ALTOM

To give valuable tips for testers who are at the beginning of their career, I tried remembering what and how I learned back when I started, and what I would have liked to know to help me.

I also wanted to find out the perspective of someone who has recently started learning testing.

So I talked to Diana, who has been testing for a few months now, to find out what helped her learn, and what are some of the things she learned.

Here's what I learned:

What helped you learn since you started working in testing?

- It was of great help to compare my notes with others'. This way I see a diversity of approaches and I can collect ideas to incorporate into my own work.
- Getting feedback at the end of a testing session. Discussing what I did helps me better understand my work, and to question some conclusions.
- Collecting interesting ideas and putting them in a visible place, to stumble upon them easily. For example, I read about a bug that happened when a long sequence of actions occurred. I put a note of that idea near my monitor, and when I test on my project, I think of how I could search for a similar bug.
- I made a glossary with testing terms and concepts that were new to me. When adding an item, I would also try to clarify it as best as I could, to make sure I understand what it means.
- Reading the documentation of the technologies I work with and going through tutorials.
- Comparing more sources, instead of stopping at the first one where I find an answer for my question. There might be more views on some matters.

At Altom we use Session-Based Test Management on several projects. What's essential to this method of organizing testing work is that the tester splits the testing into chunks of time, each with a focus on a mini-mission, or charter. Since joining, Diana has practiced with this method, she saw how other colleagues use it on their projects by participating for a few weeks in their testing, and now she is using it on her own project.

What did you learn since you started organizing your testing work in sessions?

- I control the time of a testing session. If during a testing session I realize that the originally planned time, say 60 minutes, is not enough, I can adjust the scheduled time. At the same time, I also adjust my expectations – testing work is dynamic, and it's OK to change the initial plan.
- With time, once I gather some experience doing testing sessions on a certain project, the session time estimation becomes more accurate.
- If at first I focused a lot on the time, and less on the task, now I am able to channel most of my energy to the testing task during a session.
- I now take notes in such a way so that I can use them as a trigger and reminder of what I wanted to do. This is a great habit that frees my attention from having to remember what I wanted to do after I finish something, to the current test idea I'm working with.
- Organizing my work and taking notes provided me with a better understanding of the product under test.
- I worked on improving my critical thinking in order to have a structured algorithm in my mind when a problem occurs: formulate clear questions, take into account the implications, and think of multiple points of view.

GOOD HABITS OF HIGHLY SKILLED TESTERS

To gain expertise in testing, we need deliberate practice and good feedback.

Unfortunately, practicing with new things only once or twice does not mean that what we learn will stick. I noticed that when faced with a challenging situation, my first instinct is to take the beaten path: apply something that I'm comfortable with. Being aware of that, I can focus on growing some habits that help me avoid the beaten path, and allow me to be open to improving.

Here are some good habits I observed in testers, which enable them to practice more and better, and to get more and better feedback.

#1 **They make a test strategy** when starting work on a project, and update it as they get new information and as the goals change. Test strategy doesn't have to mean a big document that no one will ever read. Sometimes it's enough to create a list of ideas that help you answer the question "How will I test?"

#2 After they do a task, they ask themselves: "**what have I learned?**" and "**what impact can this have on my work?**" When you experiment, the experience and the outcomes can be surprising. Pausing to think about these two aspects can mean valuable feedback.

#3 **They create safe situations for experimenting.** In order to test better, from time to time they tweak how they do things. For example, the second time they use a technique, or test the same field, they try to do it somehow in a different way. This way they can discover new things and may also learn new methods of testing. Doing this in situations where it's no problem if they "break" something encourages them to add the variation.

#4 **They keep notes** of what they've learned, what they find intriguing, what new ideas they have discovered. Through this they avoid quickly forgetting, and it allows them to later use those ideas as triggers for new ones.

#5 **They focus on one challenge at a time.** Separating points to focus on means focusing your energy on one thing at a time, which can help you avoid interruptions and learn faster.

#6 **They practice with a wide variety of testing techniques.** Developing expertise in testing means also figuring out what solution to choose for a problem. If you only know of one possible solution, it's probable that it will be the only one you choose, but having a large toolkit can help you choose more appropriate solutions to the current challenge.

ACTION POINTS

A checklist of things you can do to improve your testing abilities

Different testers have different learning styles and abilities. Some people I know hate watching tutorial videos. They don't have the patience to watch such a video from start to finish, and prefer to go over written tutorials, where they can quickly scroll through the parts they are not interested in. Others don't particularly enjoy taking notes or thinking about how to improve their note-taking. We all have our quirks, so we will pick learning strategies that fit our own needs.

Here are a few action points that you can pick from, based on your preferences.

You can start with 3 things, focus on each a time, then choose (or add) a different set of 3.

- ☐ When you start a new sprint, or set of tasks, create a test strategy - before doing tasks, think about how you want to achieve your goal.
- ☐ After you've done a task, ask yourself - what have I learned, and what impact can this have?
- ☐ To encourage experimentation, create safe situations - no consequences for failing - e.g. create a mock database to practice on, if you're testing database migrations.
- ☐ Keep notes of what you've learned, what you find intriguing, what new ideas you have discovered.
- ☐ Practice with testing techniques. Create your own list of test techniques you want to consciously use and vary. Choose them based on what you think is appropriate based on your own context and interests.
- ☐ Read a testing book, or one from a connected domain.

LEARNING PRACTICES WE HAVE AND ENCOURAGE AT ALTOM

Who is around you is also important. The group around you is part of your learning environment and can influence how you improve in testing.

That means you can influence their learning process too! You can take initiative in your company and kickstart a new learning practice that will help you and all of your colleagues improve.

Here are some practices that formed in time at Altom, many of them from individual initiatives.

#1 Say out loud what you think you understand about a new idea

Explain to someone what you think you understand about a testing concept or about a task or a goal.

#2 Peer learning

Review others' bugs, have others review yours and do pair testing sessions.

#3 Participate in valuable courses

Our testers usually start by taking the BBST courses:

Foundations **Bug Advocacy** **Test Design**

#4 Internal workshops

Organize and participate in internal activities that introduce you to new technologies and perspectives. Some examples of workshops we had at Altom:

- Git workshop on a Friday
- One day of code retreat – practice TDD and rules of simple software design.
- Cryptoparty – learn about mail encryption.
- Internals of web workshop – learn about the history and current state of the web, practice with client side technology, and with setting up a server.
- Testing sessions on open source software – we chose an open source project to test, put together a team, and met after work to test, share our findings and have fun.

#5 Share resources among the team

We frequently share talks, webinars, blog articles, tools, anything we find useful in our work.

#6 Make books available

We have an office library with books from many domains, including testing and coding. I once made a list of the most useful books that me and my colleagues have read to keep track of them. I'm sharing this list with you on the next page.

MY LIST OF BOOK RECOMMENDATIONS

Books about test techniques and testing approaches

- **The Domain Testing Workbook**
  by Cem Kaner, Sowmya Padmanabhan and Douglas Hoffman

- **Introduction to Software Testing**
  by Paul Ammann and Jeff Offutt
  This textbook deals with technical aspects related to testing and coverage. It includes examples and exercises.

- **Lessons Learned in Software Testing: A Context Driven Approach**
  by Cem Kaner, James Bach and Bret Pettichord
  A practical book, with examples from the authors' experience

- **Explore It!: Reduce Risk and Increase Confidence with Exploratory Testing**
  by Elisabeth Hendrickson

- **How to Break Software: A Practical Guide to Testing**
  by James A. Whittaker

- **Software Testing Techniques**
  by Boris Beizer

- **The Craft of Software Testing: Subsystems Testing Including Object-Based and Object-Oriented Testing**
  by Brian Marick

Software systems

- **Code: The Hidden Language of Computer Hardware and Software**
  by Charles Petzold

BBST Workbooks

We keep several copies at the office for colleagues taking the course and for reference after finishing the course.

- **Foundations of Software Testing**
  by Cem Kaner and Rebecca Fiedler

- **Bug Advocacy**
  by Cem Kaner and Rebecca Fiedler

- **Test Design**
  by Cem Kaner and Rebecca Fiedler

- **The Domain Testing Workbook**
  by Cem Kaner, Sowmya Padmanabhan and Douglas Hoffman

Project Management books

- **Measuring and Managing Performance in Organizations**
  by R D Austin

- **Peopleware: Productive Projects and Teams**
  by Tom deMarco and Tim Lister

- **Exploring Requirements: Quality Before Design**
  by DC Gause

- **The Mythical Man-Month: Essays on Software Engineering**
  by Frederick P. Brooks Jr.

- **The Effective Executive: The Definitive Guide to Getting the Right Things Done**
  by Peter F. Drucker

- **A Framework of Software Measurement**
  by Horst Zuse

Books on problem-solving and critical thinking

- **Tools of Critical Thinking: Metathoughts for Psychology**
  by David A. Levy

- **Proofs and Refutations: The Logic of Mathematical Discovery**
  by Imre Lakatos

- **Critical Thinking: Tools for Taking Charge of Your Learning and Your Life**
  by Richard Paul and Linda Elder

- **Perfect Software: And Other Illusions about Testing**
  by Gerald M. Weinberg
  A light read, this book presents relevant aspects on software testing and quality.

THE MINDSET WITH WHICH TO APPROACH ALL THIS INFORMATION

I like to use in my testing work a framework that was inspired by a book on critical thinking.

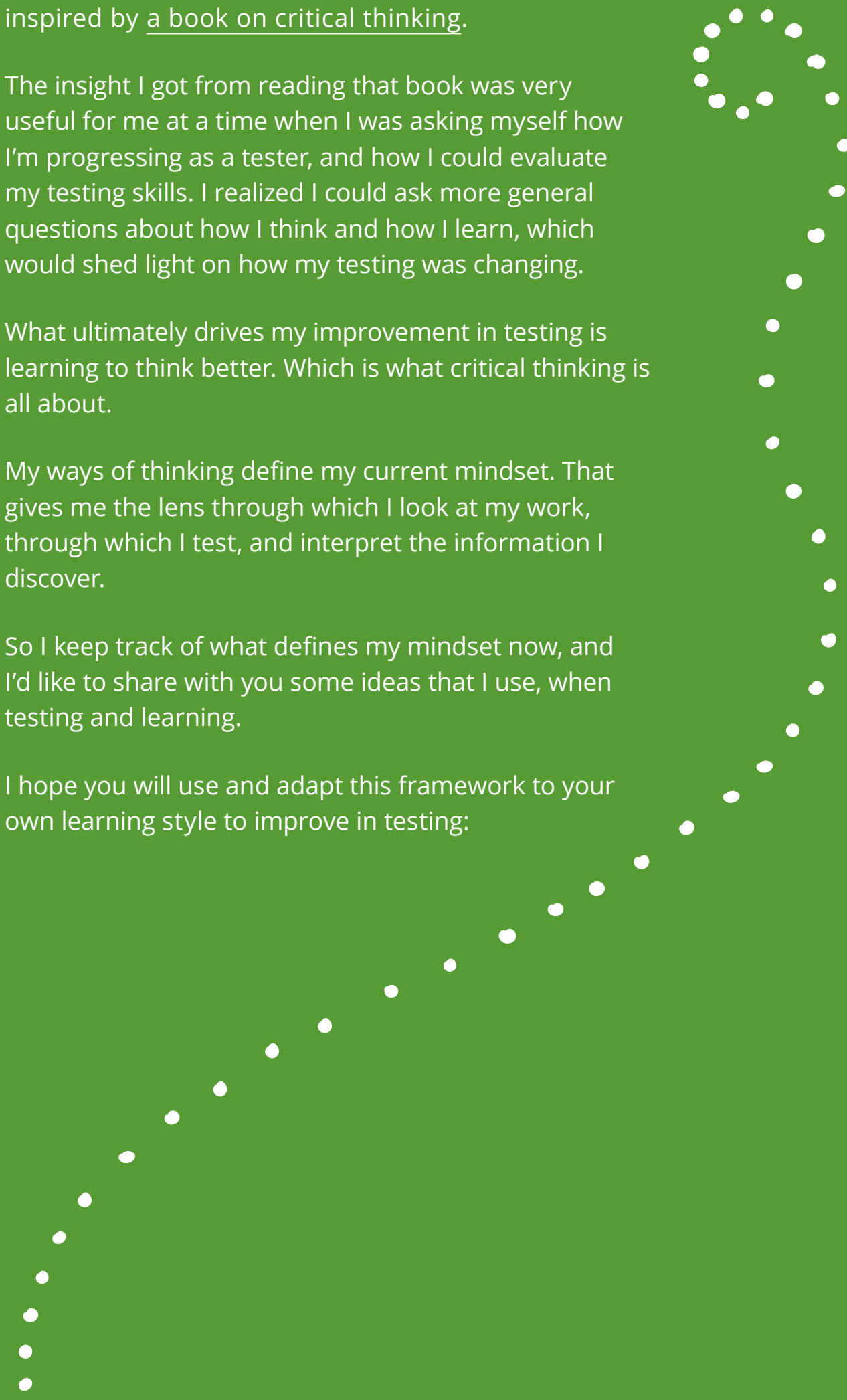
The insight I got from reading that book was very useful for me at a time when I was asking myself how I'm progressing as a tester, and how I could evaluate my testing skills. I realized I could ask more general questions about how I think and how I learn, which would shed light on how my testing was changing.

What ultimately drives my improvement in testing is learning to think better. Which is what critical thinking is all about.

My ways of thinking define my current mindset. That gives me the lens through which I look at my work, through which I test, and interpret the information I discover.

So I keep track of what defines my mindset now, and I'd like to share with you some ideas that I use, when testing and learning.

I hope you will use and adapt this framework to your own learning style to improve in testing:



Start from your goal, your purpose.
Define it in a clear way. What skills do you want to achieve?
What is your way for achieving that purpose?
Explore questions, issues, problems.

Then **start looking for diverse information**—presentations, workshops, conferences. There is a lot you can find online. But depending on what the subject is, the rhythm of change can be different. For example, some technologies are more dynamic than others.

Also, searching for something may lead you in multiple directions. Keep your curiosity awake and dig into related materials. You might discover useful ideas.

Try to truly understand what you read, what you work with. Extract the concepts. Come to conclusions about what you learn.

Be aware of some of the assumptions you made to reach the conclusions, and their implications.

Avoid absolute truths.
Keep in mind the idea that everything might be different than you think.

Be skeptical of absolute truths: choose something you are sure of and try to find possible reasons why it might turn out not to be true all the time or not to apply all the time.

• • •

Thank you for reading!

ABOUT THE AUTHOR

Alexandra Casapu does software testing at Altom. The environment she works in has facilitated her learning on the importance of context in testing, the exploratory approach, and caring a great deal about improving her testing skills.

In 2014 she became an instructor for the BBST courses, enjoying the experience of going in depth with valuable learning material and having more interactions with fellow testers.

Her current interest is at the interface of hands-on testing and practiced reflection on testing activities.

ACKNOWLEDGMENTS

Special thanks to my colleagues Diana Magdas and Bogdan Szabo for their contribution to this guide.