# Contents

**Preface** xi  
**Acknowledgments** xv  

## Chapter 1  Product Development Landscape Today  
Roles in a Product Team 7  
Do We Still Need a Software Development Life Cycle? 9  
Did You Know? 11  
References 12  

## Chapter 2  Influencing a Change in Software Testing  
Why Change Is Inevitable 13  
Change Drives Innovation and Innovation Is Important to Both Survive and Thrive 14  
Change Breaks Monotony 15  
Change Breaks Fat, Dumb, and Happy (FDH) Syndrome 15  
Change Is Driven by External Factors 16  
Change Is Normal 16  
Resistance to Change 16  
Sense of Insecurity 17  
Lack of Trust 17  
Lack of Visibility into What Is Coming 17  
Comfortable with Status Quo 17  
Fear of Failure 18  
Manner in Communicating the Change 18  
Managing and Succeeding in Implementing a Change 18  
Do the Required Homework 19  
Maintain Required Levels of Transparency 19  
Communicate Frequently 19
## CONTENTS

Show Commitment in Making the Change a Success  
Empower People Involved in the Change to Succeed  
Software Testing Is Changing Too  
Software Development Methodologies  
Testing Attributes  
Technology  
Market and End-User Requirements  
Service-Oriented Mind-Set  
Markets and Compliances  
Did You Know?  
References  

### Chapter 3  What Did We Do So Far in Software Testing?

**How Did Testing Fit into the SDLC of the Past Years?**  
**How Did the STLC Itself Look?**  
- Test Strategy  
- Test Planning  
- Test Design  
- Test Execution  
- Defect Management  
- Test Sign-Off  
- Other Activities in Which Testers Were Involved  
**How Test Attributes Were Tested**  
- Functionality  
- UI  
- Usability and Accessibility  
- Performance  
- Security  
- Globalization  
- Other Testing Types  
**Testing Metrics**  
**Case Studies from Our Past Years of Software Testing**  
- Case Study 1  
- Case Study 2  
**Analyzing the Past**  
**Software Testing Career Options**  
**Did You Know?**  
References  

### Chapter 4  Testing in the Present Times

**What Has Changed in Today’s Style of Development?**  
**DevOps and Changing Role of Test in DevOps**  
**What New Technologies Have Impacted the Overall Test Strategy and Effort?**  
- Cloud  
- Mobile  

Click here to order "Software Testing 2020: Preparing for New Roles"  
© 2017 by Taylor & Francis Group, LLC
Wearable Computing and Augmented Reality 59
Social Media 60
Analytics 61
SMAC 61
Computing Everywhere, Internet of Things, and Context-Sensitive Systems 62
What Other Things Are We Doing as Testers? 63
Market Watcher 64
Innovation Seeker 65
Quality Empowering Collaborator 65
Doubling on One’s Role 65
We Are at the Crossroads 66
Case Studies 69
Case Study 1 69
Case Study 2 72
How Do We Thrive in Today’s Environment? 74
Current Trends That Will Also Set the Base Moving Forward 75
How Is a Tester’s Career Shaping Today? 76
Did You Know? 77
References 77

Chapter 5  What Does the Future Behold for Software Testing? 79
We Will Start Off by Talking about a Very Controversial Question: “Is Testing Dying and Will It Cease to Exist in the Coming Years?” 79
The Dynamic Landscape Will Continue to Become Increasingly Dynamic 80
App Development Will Soar 80
Testers Will Coexist with Crowd Users 81
Testing Fraternity Will Work to Seek Balance in a Number of Areas 81
Increasing Collaboration in All We Do 82
Manual Testing Will Not Disappear but Will Become Completely Niche 83
Nonfunctional Areas Will Become Very Important Including Compliances Related to Them 83
Automation Will Become an Increasingly Integral Part of Software Testing 84
Agile Will Become a No-Brainer but Customizations Are What Teams Will Need to Understand 84
Testing Will Not Be Confined to Just Your Core Hours of Work in Office Premises 85
Will Metrics Still Be Used? For Such Short Releases, How Do We Connect the Dots with the Past? 85
Independent Testing’s Future 86
The Merge between Development Testing and Software Testing  
A Twist to Testing Centers of Excellence (TCoE)  
Cover Note and Client Overview  
Shaping Up the Future of QA in the Evolving Technology Space  
Client  
Challenges  
Solution  
Approach  
Client Benefits  
Did You Know?  
References

Chapter 6 Are We Empowered for the Present and Ready for the Future as a Fraternity?  
Is Test Automation Becoming Mandatory?  
Will Manual Testing Cease to Exist?  
Do I Have to Be a Domain Expert?  
How Do I Stay Current and Look ahead to Give Myself an Advantage?  
Will a Tester Be Embraced and Accepted into the DevOps Fold?  
Is Independent Testing Dying?  
Independence Need Not Come in Just from Testers  
Understand the Dotted Line in Role Delineation between Developers and Testers  
Understand What Role a Tester Should Assume at This Time  
Other Additional Best Practices to Adopt to Empower Ourselves for the Present and Prepare for the Future  
Stay Hungry, Foolish, and Continuously Challenged  
Adopt Continuous Improvement  
Work Is Not Just a 9–5 Job  
Did You Know?  
References

Chapter 7 How Is the Tester’s Role Changing?  
What Can a Tester Give away to Another Team Member?  
What Can a Tester Take on His Plate Instead?  
A Fluid Role  
Changing Facets in Software Quality That Will Additionally Define a Tester’s Role in the Coming Years  
A Constant Innovator  
Did You Know?  
References

Click here to order "Software Testing 2020: Preparing for New Roles"
© 2017 by Taylor & Francis Group, LLC
CONTENTS

Chapter 8 What Do Industry Veterans Have to Say? 119

Tom Churchwell’s Input 119
  Tom’s Thoughts on Future of Quality and Software Testing 120
  The “Ante” for Products 122
  The Cloud, the Crowd, and Distributed Teams 123
  Bringing Quality Closer to the Customer 123
  The “Ante” for People 125

Ross Smith’s Input 126
  Ross’s Inputs on Where We Are Heading 127

Rahul Viswaroop’s Input 129
  Awareness of Touch Interfaces and How They Impact Usability and Performance 130
  Mobile First Workflows 130
  Reliance on Community and Social Platforms to Solicit Feedback Proactively 130
  Agility in Release Cycles 131
  Compressed Duration for Endgame Certification 131
  Connected Workflows 131
  Security 132
  Wearable Technologies 132
  Internet of Things 132
  DevOps 133

Concluding Thoughts Summarizing the Sentiments of All Three Experts 133

Did You Know? 134

References 135

INDEX 137
Preface

The dynamics of past, present, and future are often very profound on all disciplines, and software testing is one that has seen it all. The discipline has taken shape from very humble beginnings into one that has tremendously influenced how a product development effort is taken up. Testing processes, tools, methodologies, technologies at play (both in terms of what we test and how we test), and tester roles have all been going through a paradigm change. Having been closely associated with the software testing and quality assurance domain from the very early years, both in the capacity of a tester in a large product organization and a founder and CEO of a testing services organization, all of this has been very intriguing to me. My thoughts about the past, the present, where we are heading in the future, how all of these are interconnected, and how some elements of the past reappear are what I discuss in detail over the course of this book. My inspiration to do this at this time has been multifold:

1. We often look at the present and future, but variables from the past are equally important to consider in defining our path into the future. The current young testing crowd, especially, may miss out on this not because they do not care for the past but due to lack of consolidated information where the details are often ignored as a “thing of the past.”
2. Software testing is at a very important crossroad, where we are going back to the roots on certain fronts. For instance, test automation is growing in prominence, but manual testing is becoming a niche; we are increasingly collaborating with the developers, breaking the bounds of unrealistic independence in testing, and bringing in true conscious quality. At such an important stage, it is important to take stock of the past, present, and future to define both the direction the discipline would take as well as the careers it would entail for testers.

3. I have been having in-depth discussions with several veterans in the industry, both from clients we work with as well as people who we engage with on the evangelism front. This has been a topic of detailed discussion in recent times. I wanted to use this opportunity to bring thoughts of such leaders to the forefront.

These are some of the core reasons I started working on this book in 2015. You may be aware of my earlier books on software testing: Are You Smart Enough to Be a Software Tester? and Leverage the Wisdom of Crowd in Software Testing. Both are available on Amazon. The second is also available from CRC Press. In continuation of those two books and the learnings from them, I have sequenced chapters in this book in a logical order that is easy to read, follow, and effectively impart the message. We will look at a range of topics covering where we are in the product development landscape today, what are the varied disciplines at play, what are the influencing factors bringing in a change in software testing, why is such change important, what did the past look like, what is the current decade turning out to be like, and where are we heading. As for the future, we will look at it from both near-term and long-term standpoints. We will also see in detail as to whether we, the testing fraternity, are ready to take on such changes and are empowered enough or are there gaps that we need to fill. We will talk about what all of these mean to a software tester’s role and wrap it up with inputs from industry experts on what is in store for the software testing discipline and community in the coming years. I will also discuss case studies from QA InfoTech, along the way, to bring practical relevance to the points we discuss.
After reading this book, I am confident, that you as a reader, will be not just excited but also confident to take on what is in store for us in the coming years. This preparedness will be wholesome, with the required knowledge of the past and present, taking relevant elements into the future times. Once again, I am very excited to be doing this at this time, which I believe is the perfect opportunity for a topic of this nature, helping us as an industry move to the next level, influencing the change not just among us, but also at the product engineering industry level. Thanks for taking the time to read, and I look forward to your comments, thoughts, and feedback. Please feel free to write to me at mukesh@qainfotech.com.
Everyone has their own calling, but not everyone is looking for the phone, or either they missed the call, or just not answered it.

—Anthony Liccione

This quote applies not just to individuals but also to organizations and industries at large. Software has become omnipresent. Companies are branching into untested territories to take on new challenges and bring in unforeseen solutions. Google, for example, has multiple such projects including Google X, Fiber, and Calico all focusing on diverse portfolios from Google’s flagship products around online search, advertising, and mobile operating system. The parent company, Alphabet, was formed in August 2015 to bring together the multiple brands it encapsulates into this umbrella (Figure 1.1). Google has been traditionally known to support an open culture to innovate. Twenty percent of every employee’s time is set aside to innovate. Such innovations help individual employees and the organization remain nimble and dynamic that have become their imperative characteristics for staying competitive in today’s marketplace.

Organizations are beginning to understand that such dynamism does not mean randomization and overwhelm, which were some of the core reasons they have often resisted big changes in the past. They are increasingly able to bring in the right focus needed for their core businesses yet focus on futuristic trends. This is a very positive outlook for the software industry at large.

Besides understanding how product development landscape today is, the other key trend to note is how global it has gotten to be. A few years ago, globalization largely meant only one of
the following three aims—for product companies from the West (especially North America):

1. Making their solutions available for global consumption.
2. Outsourcing one or more of their development needs (such as development, testing, support) as service requirements to developing countries that had the talent and less cost. This would include reaching out to countries such as India and China to get the work done offshore.
3. Setting up global development centers, again to leverage the global talent and low costs.

While this is not untrue even today, the welcoming trend is that the growth of local product companies is on the rise. Alibaba, for instance, is an e-commerce giant in China today. Flipkart and Snapdeal are making the market competitive for global leaders like Amazon in India. InMobi is competing with the Googles of the world. A number of product start-ups are being funded and homegrown, including people returning to their home countries to be a part of such revolutionary changes.
How have all of these impacted the traditional product development cycle and landscape that we have gotten used to? As a “fraternity” in the software development world, we have all come to accept the downsides of the traditional waterfall life cycle, especially its alignment with today’s requirements. While there were a lot of “teething” and acceptance issues for the agile life cycle in the early 2000s, the industry has learned to implement agile in varied shapes and forms to meet specific needs. Organizations have gone agile, teams have learned to practice it to promote better collaboration with focus on result and end product, and users have started increasingly involving themselves in the software they consume even when it is still under development.

However, the question we now need to answer is, “Are these changes sufficient to meet the current needs of dynamic and global development?” The industry as a whole is attempting to answer this at this time and simultaneously take on newer practices to keep pace with the latest and greatest in the development landscape. The following are some noteworthy evolutions:

1. Develop a user-centric and service-oriented development model: Products and applications are no longer being built with a one-sided approach of handing a deliverable to the end user. Organizations are increasingly adopting a user-centric approach in looking for value-added opportunities to exceed end-user expectations. The whole focus of the product development effort is now on user satisfaction, thereby taking in a service-oriented approach. Thus, even a product company now brings in a service focus that further makes them more nimble and amenable to end-user wants. Users are closely involved in the development cycle, be it to review designs, evaluate the product implementation before market release, and provide inputs for future implementations, making it all a very tightly coupled development effort despite how geographically far reaching the market may be.

2. Adopt a hybrid development methodology: When teams started moving into the agile life cycle in the early 2000s, one of the main troubles they had was the rigidity and lack of complete
alignment to any one given model. While certain aspects of a scrum implementation may work for them, they may not be able to embrace it fully. Similarly, they may better align to specific characteristics of other models such as Kanban and XP. Over time, teams have started understanding the value of hybrid development methodologies, where they take the best of varied life cycles and create a custom model that works the best for them. Such a hybrid implementation has become an inevitable need of the day to focus on the deliverable that they are building as opposed to the rigidity of implementing a specific development methodology.

3. **Build cross-functional teams:** These are teams that run horizontal across product groups. Building and maintaining cross-functional teams continue to be a chicken and egg issue over time. In the days of the waterfall model, special functions such as performance, localization, and security, especially in software testing, were cross functional, since people with such skills were not very easy to hire, demanded high pay, and were not needed throughout the development life cycle. With the advent of agile, they had enough tasks to take on throughout a release that they were seen as resident experts in specific teams. However, cross functional has a new meaning in recent times. These experts are again being leveraged across teams given the know-how they bring to the table. Additionally, there is so much cross functionality between product teams that needs to be leveraged over the course of development—whether this is cross functionality from a product, process, or resource standpoint, organizations are looking at collaboration at all possible places and encouraging teams to become cross functional. This also helps them develop a strong sense of appreciation for the larger product context instead of looking at the specifics of a module or application they may be working on.

4. **Increased use of APIs:** This is seen as an upcoming trend. In the software world, we are not new to APIs. But what is becoming increasingly popular in the cross sharing of APIs? There is so much public knowledge and implementation available such that software systems are becoming increasingly API centric, leveraging such existing knowledge sources.
Researching for such existing reusable snippets of code will become more valuable, given how time and cost constrained teams are. Despite the value it holds, most teams often fail to leverage such common resources, even within the same organization—this is often due to lack of due diligence from their end. The ones that will differentiate themselves in delivery and implementation will be the ones that increasingly use APIs both developed at their end and also leveraged from external sources.

5. Embrace open source: A long-standing debate in the software world continues to be “proprietary or open-source.” While this question will continue to prevail, the answer to which is very specific to an organization’s individual requirements, the debate is easing out in favor of the open-source world. In addition to the cost-effectiveness and vast community knowledge open source brings in, it is also becoming increasingly sophisticated in its feature set and quality, in recent times, giving proprietary software a good run for money. In a recent panel discussion we moderated at the Next Generation World Testing Conference, this debate was taken up. Interestingly, more than each group taking its own side, they were seen touting the benefits of the other side. For example, the commercial tool vendors, such as the ones from IBM, were talking about their contributions to the open-source world, whereas the open-source proponents were talking about using commercial tools to develop open-source software to ensure robust quality. The panel was moderated in favor of a collaborative existence, although at a large level, most teams are using open source in possible places. Open source builds on their agility and productivity, without any bureaucracy around approvals. Given that open source is also soon catching up on its quality, comprehensive feature set, and the range of functions it offers, it is certainly a time to embrace it, in development efforts.

6. Integrate individual modules into a wholesome ecosystem: The industry as a whole is moving to an ecosystem mode. While individual modules have their own relevance, the industry has started acknowledging the benefits of an ecosystem. To take a simple example, we need no introduction to the
social, mobile, data analytics, and the cloud computing environments. However, a new technology around an integrated system called SMAC combining the four to bring in better business and user value is on the rise. Synchronizations between modules and individual systems such as these will only continue to grow, forcing development teams to think at a wholesome scale.

7. Understand DevOps end to end: In the integrated work that has become necessary in the development landscape, teams have come to appreciate the need for DevOps. However, they have only partly understood what DevOps stands for. While they see the piece of integrating development and operations, they still do not fully see the cultural shift that is needed to embrace DevOps comprehensively. How software quality fits into DevOps, the kind of tools that support this implementation, the team level collaboration that is needed to practice this concept in its entirety are all still loose ends that the industry will tie up together in the coming years.

8. Secure systems: While cross collaboration and integration has a lot of benefits to offer, it certainly opens up a whole new problem around application security. A number of touch points between systems directly translate to the number of vulnerabilities the system is exposed to. Hackers are smart in leveraging newer vulnerabilities by the day that the software development team has to increasingly focus on hardening the system and closing varied threat entry points. In the coming years, securing systems will be an important element of the development landscape.

9. Focus on improving application performance: Global user base for products and applications is on the rise. Obviously, this means more users are using the product round the clock. Digital consumption across the world has gone up significantly, where development efforts need to specially focus on application performance. This also includes the scalability of the underlying system that supports the application, server side performance, client side performance, and greater focus on capacity planning to ensure seamless availability all around. Thanks to advancements in technology and infrastructure, especially
in the space of cloud computing, performance is improving and keeping pace with the growing user base. However, this level of focus should continue to rise in the coming years for organizations to have a competitive edge in the marketplace.

10. **Strengthen online identification**: A great application may miss market presence because it is not able to connect with its relevant and target users. Similarly, a substandard application may pass off with a decent acceptance given its right reach with end users. Herein, the importance of search engine optimization needs to be emphasized where development efforts need to account for the right presence and optimization even as the product is being engineered.

11. **Develop collective ownership for quality**: Although the tester is still responsible for overall product quality, product teams are taking on responsibility for quality in possible ways. For instance, developers are taking on unit testing, build engineers are leveraging automated sanity tests, and everyone is interested in defects reported from the field. These are all good signs allowing testers to focus on bigger and newer tasks related to quality, enabling the team to achieve the required test coverage within the short release cycles they have to work with.

Development landscape, although agile in its implementation, will have to accommodate these evolutions into its fold, at this time. You may have already been using one or more of these evolutions, but these put together form an important set for constructively shaping your development landscape in the coming years. When teams are able to bring these into their scaffold, they are able to reap a shortened development cycle, cost-effective development approach, global reach, user connect, and a technology advantage, all of which together will be able to differentiate them from their competitors.

**Roles in a Product Team**

A multitude of functions come together in building a product. Roles associated with these functions are diverse and each has its own significance. Traditionally, regardless of the development methodology
used, the core roles in a team that are instrumental in building a product include the designer, the developer, the tester, the build engineer, and the product manager. These are time-tested roles that have existed since the inception of software development although the functions they perform have been subjected to change over time. For example, around the late 1990s or early 2000s, the concept of independent testing was not very popular. A developer would himself or herself take on the testing and a tester would at the most come in later in the game, just before release, for a round of acceptance testing. While in the later years independent testing became popular, we are again in a phase now where everyone collectively owns quality. The tester is still responsible for quality and issues if any, but the team as a whole understands its role in contributing to product quality in possible ways. Figure 1.2 is a simple view of how software testing as a function has evolved over time. Other disciplines have also undergone similar such transformations, to align themselves with the need of the day.

Today, in line with market needs, the major roles in a software development team continue to be those of a product owner, a project manager, a technical lead, a developer, a tester, a UX designer, an architect, and a build and operations engineer. Each of these roles further has its own specializations. Specifically in the context of the

![Software testing has evolved immensely over the last 50 years, especially in the last two decades](Image)

**Figure 1.2** Evolution of independent testing.
testing, a tester could be any one or more of the following types: manual functional, automation, performance, localization, usability, and security. Such specializations continue to build a niche for the individual as well as the role they play all together strengthening the product under development. Additionally, a tester has to extend his or her function beyond what the core testing job entails. He or she has to collaborate with the developer, designer, build engineer, and others on the team to take on a bit of their tasks as well; for example, how can the tester take on unit testing, what kind of design suggestions can he or she provide, and how can he or she help the build and operations engineer debug issues from the field are all new factors testers that are starting to pay heed to.

With time, some of these roles may split or merge again either at an industry level or at an individual organization level, mostly driven by the needs of the product. For instance, recently, an acquaintance of mine, who has been into quality engineering at a leading ISV, talked about how everyone owns quality now, in his team. Changes such as these are often large enough that they have a huge impact on organization structuring, individual careers, and the overall quality and acceptance of the product in the marketplace. This is exactly what we will see over the course of this book: to understand how software testing looked, what it is today, and what is expected to shape in the coming years.

**Do We Still Need a Software Development Life Cycle?**

Software development has come a long way. The processes, methods, technologies, and models have all changed over the years to align with the need of today. The software team has certainly become increasingly self-managing, thanks to the agile practices they have been exposed to in the last several years. They understand the importance and relevance of fast time to market, competition, and user feedback besides the core focus of product development and delivery. Given these positive changes that have set in, the question is, “Do we still need a software development life cycle (SDLC)?” Does it not add extra weight into the overall software implementation, making the overall approach more rigid and inflexible? What is the true value an SDLC brings to the table and is it justified to continue to use one or
more of these models in the coming years? This is an important question to answer upfront, to understand how software testing will look like, and what testers will take on in the future.

At the core, an SDLC is a process to plan, build, engineer, test, and release a software system. This process helps teams enforce several important traits that are invaluable to the success of the project. To this extent, the model becomes a framework in real time to support end-to-end project execution. This includes the following:

*Meeting and beating deadlines*: An SDLC imbibes the sense of time in the team. The team is able to appreciate the need to meet and beat deadlines, track red flags, and bring in corrective course of action, when they have a certain model to follow.

*Enhancing team productivity*: SDLC model promotes knowledge and resource sharing, which is a great way to enhance a team’s productivity. While individual team members are capable of their own research to explore ways to improve their productivity, an SDLC model instills better sense of collaboration and cross sharing, which greatly improves team’s hands on productivity.

*Establishing long-term vision*: Without an SDLC, teams, although focused on the current development, often fail to have a vision into the future. An SDLC, as a model-based approach, forces them to think what is next for the next several releases, thus giving them an overall perspective into the future and long-term vision.

*Enforcing adherence to a certain code of discipline*: An SDLC brings formality to the team’s operations. The onus is on the implementing team to either leverage this formality to bog down the team with mundane bureaucracies or use it to give the required structure to the effort, to bring in a positive discipline to encourage the team to deliver. When used well, the discipline and structure the SDLC brings in can further bond the team well, hence empowering them to work toward a common goal.

*Balancing project constraints*: Every project, however well funded it may be, faces constraints around time, cost, and quality.
It has to balance these out over the course of the project execution to ensure they are in control and positively impact the delivery of the project. In the absence of an SDLC, these constraints are more likely to get skewed making the effort very chaotic—all of these only add to the likelihood of the project to fail.

Having discussed the need of an SDLC, it is important to acknowledge that organizations are slowly and steadily moving away from a rigid single SDLC model execution. Hybrid style of operations where they adopt more than one model depending on the project they work on is gaining prominence. As long as organizations understand where the development landscape is heading, what project they are working on and what model to implement, and the flexibility and structure the SDLC offers despite its innate rigidity, they will already be half set for a successful implementation.

Did You Know?

1. Agile succeeds three times more often than waterfall² (Figure 1.3).
2. Software maintenance costs between 40% and 90% of overall project costs.³
3. Search content is available in 35 non-English languages in Google.⁴ This is only continuing to grow showing the scale of global product development today.

![Figure 1.3](a) Waterfall versus (b) agile implementation success rates. (From The CHAOS Manifesto, The Standish Group, https://www.mountaingoatsoftware.com/blog/agile-succeeds-three-times-more-often-than-waterfall, 2012.)
References