Human Error and High Quality Software

Ross Smith
Director of Engineering
Skype Division, Microsoft Corp

How has Defect Reproducibility Changed Over the Years?
Rajini Padmanaban
Page 4

And many more...
Table of Contents

Human Error and High Quality Software.................................................................2
How has Defect Reproducibility Changed Over the Years?........................................4
Recent Technical Events – Prastuti 1 and DevSummit...........................................5
A Sample Set of QA InfoTech's Articles on Techwell...........................................9
Testing Tips & Tricks.........................................................................................11
Did You Know?.................................................................................................11
Tester Speaks.....................................................................................................11
Human Error and High Quality Software

Throughout history, technology has always been a driver of change. Clay tablets, the wheel, the aqueduct, movable type – humans have always been moved forward by advances in technology. Today’s world of rapidly changing mobile software is no different. And yet, throughout our history, humans have always been prone to making mistakes. Errors and our ability to learn from failures are part of what makes us human. As we think about the rapid pace of the world today, some simplistic techniques to reduce human error can go a long way towards improving the quality of our users’ experience.

Overview

Humans have learned through trial and error for generations. Mistakes are part of human nature. The rapid pace of development and production in the 21st century demand high quality delivery, at scale, immediately. Organizations constantly struggle with the trade-off between production and protection – how fast can we deliver versus how much risk do we accept in order to protect our users? One easy place to look is to prevent simple human error. We tend to look to process optimization, development models, compilers, automation systems, and many other quality improvement models – all of which are great – and yet we tend to overlook simple, basic human error as a contributor to poor software quality. If you’ve ever written a late night email only to discover that you forgot to pluralize, or perhaps you’ve left your car keys in the ignition after a long day at work. Maybe you were distracted by a phone ringing as you arrived home and as a result, discovered the next day that you misplaced your purse. These mistakes are natural – and they are preventable. Software quality can benefit from a similar approach focused on reducing human error.

Process Changes

Software development process changes over the last few years have shifted from a “waterfall” model, complete with a long stabilization phase, to an always connected, always updating capability model, where we can use the power of big data to detect anomalies and patch them almost immediately.

The cloud services of today have a broader reach – across consumer and enterprises alike – and IT departments can look to the cloud to monitor and improve the quality of their users’ experience, without having to worry about administering it themselves. As with other historical shifts, there are pros and cons, and learning and risk assessment to go through. Changes continue to influence the landscape.

Human Error

In the 1830’s, a Fellow of the Royal Society named Charles Babbage developed his “difference engine” – the precursor to the modern computer. This machine could help with astronomical and mathematical calculations. In mid-June, 2015, Apple announced new developer capabilities for its Watch, Music, and Siri services. Microsoft announced its new Surface Hub for business conferencing. Twitter block lists, government data leaks, Facebook messenger, and the Internet of Things are all in the news every week.

From Charles Babbage through Claude Shannon and his explorations in binary arithmetic, to early mainframes – Grace Hopper and the moth that was the incarnation of the term “bug” – through the Commodore PET, IBM PC, DOS, Windows, the first Mac – all the way through to the internet, mobile devices, and cloud services that we have today. The one and only constant that impacts a successful user experience with software – the one that transcends generations, platforms, technologies, programming languages, hardware, society – is human error!

The early human who strays from the safety of the cave in search of late night food who becomes late night food on the Serengeti because their guard was down from lack of sleep is not a whole lot different from the mobile developer who checks in buggy code late at night - because their guard was down from lack of sleep and they want to go home.

A great (and easy) way to improve the quality of your software – to truly impact the customer experience – is to look at your workplace culture and how you’re engineers work day-to-day. Remove distractions, reduce pressure to submit / check in code against a deadline. Add a reflection and evaluation period.

James Reason has done transformational investigation into the study of human error. Reason has built on Jens Rasmussen’s work on skill, rule, and knowledge-based performance. His categorization of errors and examination of causation and correlation apply as much to software as the industrial accidents that formed the basis of his studies. Rules-based errors are addressable by the organization – whether they be the rules themselves, or communication-based – if rules are in place that prevent errors/bugs, and errors/bugs still happen, then the rules should be altered.

When we think about software quality in the 21st century world of always online, always updatable – the opportunity is in the skills and knowledge-based performance.
How do we improve our skills so that human error is less likely?

The opportunity for magic is in the ability to improve the knowledge-based performance of our engineers – and the causes and response to human error in this domain. How do we learn from mistakes, how do we teach others?

**Example Recommendations and Next Steps**

Build a culture of trust! – It’s OK to make mistakes as long as learning from them is part of the culture. Trial and error are how humans learn.

Reduce QA friction for developers who write better code, increase friction for those who don’t.

Share lessons learned broadly so everyone learns from mistakes.

Eliminate distractions – give people a time and place to focus. Open space offices are great for collaboration, hard on concentration. Provide alternatives.

Share user experiences and feedback – make people aware of the cost of their mistakes.

Investigate bugs per KLOC for code checked in after 10PM vs. early in the day – find the tipping point where quality drops off and establish a rule

Investigate bugs per KLOC for early cycle vs. 2-3 days before milestone lockdown

Investigate day of week quality and productivity metrics and optimize processes to emphasize. Are more bugs found on Friday than Monday? Is buggy code written on the weekend or on Thursday?

**Additional Reading**

James Reason Swiss cheese Model


James Reason, Human Error

http://www.barnesandnoble.com/w/human-error-james-reason/1116961447?ean=9780521306690

James Reason, Managing the Risks of Organizational Accidents


The Practical Guide to Defect Prevention

www.defectprevention.org

Ed Catmull, Creativity Inc.
http://www.creativityinbook.com/catmull/

Gary Hamel, Future of Management
http://www.amazon.com/dp/1422102505/?tag=mh0b-20&hvadid=3484532662&hvlocmedia=e&hvmed= c&ref=pd_sl_12ok5r77cs_e

Ross Smith, FRSA
Director of Engineering
Skype Translator - Microsoft Skype Division
rosss@microsoft.com  LinkedIn  Twitter  Skype  Facebook  Web Blog

Ross Smith really enjoys getting a paycheck to “play” with software for 25 years now, over 20 at Microsoft. In September, 2014, he was nominated and accepted as a Fellow of the Royal Society of the Arts. He is one of the authors of “The Practical Guide to Defect Prevention” and holds six patents. 42projects has aspired to promote cultural change, “bring buzz and laughter to the hallways”. He is a member of the leadership council for the Anita Borg Institute. He was also part of the organizing committee for TEDxSeattle, and has recently been working closely with UrbanTeen.org and Spreeha.

In addition to his passion for creative techniques to improve the quality of the experience of using software, he’s explored organizational trust, enterprise gamification, management innovation, diversity and the future of education through games and with the Skype in the Classroom program. In December 2011, he was invited to the White House for a discussion on women in STEM. He was the keynote speaker for the American Road and Transportation Builder’s Transovation event in fall 2014. The work of his teams have been mentioned in Forbes, The Economist, the Wall Street Journal, PSFK, the American Journal of Play, Harvard Business Review, and the London School of Business. He is a blogger for the Society of Human Resource Managers WeKnowNext site and regularly posts on management innovation. He is most excited by the current work on Skype Translator.
How has Defect Reproducibility Changed Over the Years?

As testers, finding defects is one of the core responsibilities we bear in helping build a quality product. The industry is beginning to acknowledge that we cannot track down a tester’s performance purely by numbers although, a tester who finds more defects, is definitely seen as a better contributor than the rest. Of course there are other factors as well at play here including the kind of defects reported, whether they are valid defects or not, how well the tester articulates the bug in his report, how thorough he is in following it up to closure etc. In all of these, one of the challenges a tester has traditionally faced in defect management is defect reproducibility. How to bring in predictability in reproducing defects that he has seen once or a few times – especially those nasty looking defects, that he knows exists in the system but are tough to reproduce. When the tester gets deeper into the system, be it performance issues, database level issues, system crashes, that have all happened once, can especially get tricky when he tries to line up all ducks in a row to simulate the issue again. How many of us as testers, have been thrilled to see a blue screen, but have sat down wondering how to reproduce it to report it as a defect? As testers, we have all been in this spot and continue to face this day in and day out in our testing jobs. There are several interesting and helpful articles online on how a tester can get better at defect reproducibility. Alongside these tips, a tester also needs to keep in mind that the evolutions in technology need to be leveraged in increasing the chances of reproducing a defect. What are some of those key things a tester can take advantage of?

1. A tester is no longer working on an isolated test machine. Most often, these days, he is testing on the cloud, with machines that have cross connectivity amongst team members. This enhances the chances of someone else reproducing the issue, even if he is not able to, as long as he has provided the required details in the defect.

2. Also, the tester currently tests not just on one device – given that he tests on multiple devices, the chances of reproducing a defect on one of the several devices he tests on is much higher than in the past.

3. The tester also has access to devices outside of the office premises. This may not be the case for all projects, but in most app testing engagements in the current day, a tester becomes a real user of the app, outside of work.

- Technology has become readily available and accessible anywhere, that a tester is able to try realistic user scenarios even outside of his work premises – some of these such as varied bandwidth, varied scenarios, can enhance the chances of reproducing the defect, he may be having trouble with. Also, data is often synchronized across these devices, making it easier to try reproducing the issue. Such access to data and devices is a huge boon that was not available to the testers of the yester years. For instance, say later in the evening, at home, I am able to recall a certain test path or want to try something, I am in a better position to do so, unlike the testers earlier on, who would have to wait until the next morning to get to work to try the test. By then, chances are high that the tester may have lost track of what he wanted to try.

4. Social forums have made it so much easier for testers to stay connected and discuss issues. Say for instance, the tester has seen a defect around a third party integration but is having difficulty simulating it, there are several online forums, including the more popular ones such as LinkedIn, Yahoo! groups, where the tester can seek professional help and check if anyone else has been facing similar issues. Such information may not only help give ideas for newer test scenarios but also help try reproducing some of the more difficult bugs.

5. With newer apps and tools that are available now, the tester is empowered to take on better defect debugging and analysis, improving the overall chances of seeing the defect again, under the same circumstances.

6. Better tools for more detailed code coverage analysis, help the tester, understand test code paths to re-trace the steps to seeing the issue again.

7. Keyword based searches in defect management systems are more sophisticated now, enabling testers to search for similar issues that others may have logged, to get additional ideas to reproduce.
8. The idea of group bug bashes (within the organization, with crowd testers), have all improved the chances of nailing down that hard to reproduce defect, which has been sitting strong in the system. Herein, the change in team mind set, that everyone can contribute to quality and that it is not just the tester's responsibility, is welcoming to see Testers have themselves evolved in their roles over the last decade. The key is to remember that they do not work in an isolated manner. If they need help, there are other ways and people to leverage, rather than bogging oneself down. While all these techniques are valuable in reproducing that hard to find defect, a tester also has to use these logically. Let's say, it is a very low priority issue with not much user impact – it really does not make sense for the tester to use these options in nailing the issue down. Use these as your supplemental techniques on the right issues, rather than using them all, at all times – this will not only save time for everyone on the team, but also help you prioritize your tasks well and hold your reputation high as a sensible tester.

**Recent Technical Events – Prastutti 1 and DevSummit**

In April, QA InfoTech launched its first technical summit, Prastutti, in an effort to bring together testers and share knowledge on a range of topics, relevant to them and current in the industry. This offered a great platform for testers to present in front of a large audience and also ramp them up for larger such opportunities in the organization and in the industry. Here’s a quick peek at what we hosted in the first of such a series of Prastutis to come.

**Client Side Stress Testing Framework for Mobile Applications – by Ajay Kumar**

End users today expect mobile applications to not only provide rich functionality and a nice user interface, but also render results quickly without much distress on the overall device performance. And as we iterate through the mobile application development cycle it is important to weigh the performance of the application from both client as well as server side to understand the performance bottlenecks and thus optimize the system for a better user experience.

In current development scenario of mobile applications and the number of high-to-low end configuration devices available in the market, users invest more time waiting for client side rendering of the content than the server side processing. Along with the server side optimization of the applications, efforts should also be made to optimize the client side as end user monitoring of the apps has never been so important.

We have developed an automation framework to stress test the application from the UI side and calculate the CPU statistics such as memory, CPU, battery at runtime. The stress test automation framework covering cross platform mobile UI tests, calculation of memory and CPU % consumption, loading speed of the app, consolidated reporting during run time is discussed in detail in this presentation.

**Internet of Things – by Deepak Daroch**

Transformation is a part of our life - technology is also transforming and evolving. Internet has also transformed over the years making itself better and smarter which is driven mostly by expansion on Internet through the inclusion of physical objects combined with an ability to provide smarter services known as Internet of Things (IoT).

Internet of Things also known as Internet of Objects is a computing concept where everything we use in our daily life will be connected to the internet smartly e.g. Smart refrigerators that texts you when you are running short of supplies, smart homes that turns the heat down as soon as you leave the house, smart cars that operate around traffic jams. Internet of things promises all these advances without requiring human-to-human or human-to-computer interaction.
World is expected to have around 26 billion to 30 billion IoT devices by 2020. In order for all these devices to work without any issues, all these devices must be tested to the core.

Testing IoT is going to be a huge challenge considering millions of devices connected to each other. Testers need to be aware and ready for the upcoming challenges and before we start planning test strategies we need to keep few things in mind e.g. what could possibly be the testing scenarios (need to think out of the box), testing a technology which will be changing rapidly, user scenarios etc...

This presentation, talks about how IoT is going to change our lives completely and how we, as testers, should be ready to make things simple, easy and qualitative for the rest of the world.

**App Testing – Present to Future – Mobile to Wearable – by Karandeep Singh and Akhil Bansal**

Mobiles and tablets are no more a set of gadgets that are owned only by a few. With the Android and iOS market so wide and open, we have more than 1 million apps each on Google Play Store and Apple Store. And here is where an organization like QA InfoTech comes into picture to build quality into the apps used by millions each day.

Downloading, launching and using the app seems like a piece of cake! But, testing an app is never so simple.

There are various types of apps available including native, web and hybrid; then to add to the complexity there are various types of testing involved such as functional, layout, performance, coexistence, end-to-end, usability, automation, etc.; and then to make it even more complex add the number of devices that are there in the market along with different versions of operating systems that run on them. Hufff!

Seems a cumbersome process! Well, we with our experience have streamlined the complete process of testing mobile apps effectively and efficiently. This presentation talks about insights into various aspects of mobile app testing and how all of this is transforming to the next level which is moving from mobile app testing to wearable app testing.

**Big Data Testing – by Vikrant Singh**

Data quality challenges can be tackled by deploying a structured testing approach for both functional and non-functional requirements. Applying right test techniques and following best practices will improve the testing quality which will help in catching the defects early and reduce the overall cost of implementation. Big data testing is a specialized stream and the testing team should be built with diverse skills including coding, white-box testing skills and data analysis skills for them to perform a better job in finding quality issues in data. For mid-sized data, the data can be exposed as HBase tables. Also it can be verified from input data set by applying business logic. For large scale data, big data testing requires engineers with unique skill sets that are used for testing large and complex data sets.

As large number of organizations select Big Data as their data analytics solution, they are finding it hard to define robust testing techniques and set up the right test environment. Big Data includes processing of huge volume of structured/unstructured data across different nodes using languages. A robust testing technique needs to be defined well in advance in order to ensure that the functional and non-functional requirements are met and that the data conforms to acceptable quality. This presentation walks the audience through important big data testing techniques.
**Automation: Doing it the Human Way – by Sandeep Singh**

Being a manual tester, have you ever wondered "what if my manual tests execute automatically without writing code to automate it?" In this presentation, we will discuss a framework that allows testers to write test cases using predefined steps in a very simple language and execute them without writing even a single line of code – essentially, the "human way" of doing test automation.

This approach enables manual engineers to automate ~75-80% of their acceptance test cases while they are writing it, including some complex test scenarios. More complex test cases can be automated with minimal coding. This framework utilizes the capabilities of Ruby, Selenium Webdriver, Cucumber (Gherkin) for behaviour driven development (BDD) to automate tests and Appium to execute test cases on Android/iOS mobile web and native apps. The whole setup requires less than 15 minutes to build and allows parallel execution of tests on multiple desktop browsers and mobile web. Hence, it is cross-platform, open source and free. This presentation walks the audience through the approach to building test automation the human way.

**Audience Response System – by Heena Feroz and Juhi Agarwal**

How would you feel if an impersonal hall becomes an active discussion section wherein the audience interact dynamically and stay awake!?? With the rise in technology, the tech giants have come up with an idea of a new technological trend known as "Audience Response System (ARS)".

ARS is a way to change a one-way lecture into a two way dialogue that enables the presenter and the audience to interact dynamically in minutes. Through this system, the presenter is not required to wait any longer to understand his audience or to know their feedback.

It provides the freedom of choice in the room to meet the needs of every presenter and the audience as a simple, reliable and pedagogical tool.

In this presentation, the speakers outline the key aspects and

**Web Application Security Testing – Tools and Services – by Pulkit Thapan**

The number of web and mobile applications being developed continue to grow by the day. In addition, organizations are opening up applications (or portions of applications) that were previously for internal use only to customers and partners in the interest of both transparency and productivity.
Knowledge sharing has been an integral component of the DevLabs culture since its inception in 2010. All along, the organization was small enough where employees could easily disseminate knowledge about new development tools, practices, technologies, case studies etc. However, we realize the importance of more formal and sustainable knowledge sharing practices as we scale in size and the DevSummit we hosted on June 13th, 2015, was the first such initiative to accomplish this. This DevSummit was our first ever developer conference with speakers from varied technology backgrounds, diverse topics, meeting the needs of not just co-developers, but also testers and anyone in the software development industry. The topics were carefully handpicked to align with the latest in the industry and were backed by both our experience in projects that we work on as well as the research that we undertake internally.

Here’s a quick rundown of the presentations we had: “Augmented Reality Experiences” introduced the new technology to the audience as well as the internals associated in developing an AR application. Amit Joshi and Nandan Chabbra handled this as a co-presentation. Deepak Joshi, walked the audience through the whole nine yards of “Digital Signage”. Ashish Jindal picked a simple but very important topic on “Developer Tools” to enhance developer productivity and efficiency. Sansar Mor presented “MEAN stack” to develop robust and maintainable JavaScript applications, covering Mongo DB, express framework, Angular JS and Node JS in a nutshell. Rahul Chakrobarty picked “Angular JS” to present it in greater detail. Upendra Gareri picked an up and coming topic, “Parse” where backend is leveraged as a service. Saurabh Arora talked about the relevance and application of “AS400” in today’s development world. Arti Grover picked a very specific topic on “GradeCam” in line with educational projects that we work on and how GradeCam applications will gain prominence in the coming years. And finally, Nitesh Garg talked about the varied “Message Delivery Mechanisms in developing Mobile Applications”, to cover a large range of app development options.

All in all, the sessions were very well received by the audience, which included both developers from DevLabs and testers from QA InfoTech. The speakers included some of the most senior people at DevLabs as well as some of our more junior people, reiterating that at the end of the day, what matters is the knowledge that we share, regardless of the years of experience a person may hold. Our CTO, Kunal Chauhan, including a few others from the management team were closely involved in choosing and coaching the DevSummit speakers. At the end of a successful first DevSummit, we look forward to many more such sessions at more frequent intervals in the months and years to come!

UPCOMING EVENT
QUALLOQUIUM 2015
How Testing Is Changing for the DevOps World

DevOps is becoming increasingly prevalent in the agile world of software development. Product teams are gaining better understanding and confidence in implementing DevOps, and most are getting better at it with every new product release. The role of software testing and quality assurance is especially continuing to gain comprehension in a DevOps setup. This position has undergone a lot of change in the testing practices used, tools leveraged, and the shift in skill set and mind set of practitioners.

Testing practices have changed vastly, even from the days of the agile development cycle. Because testing happens in a very collaborative and continuous manner when integrated in a DevOps lifecycle, there is a need for a lot of automation. Test-driven development has become increasingly popular, and near 100 percent code coverage is also becoming inevitable. Testers have a lot to learn from the DevOps environment. Given the core drive to collaborate, it offers a lot of best practices testers can use to beef up the quality assurance effort.

Leveraging standalone test automation tools by themselves does not meet the bar anymore. Testing tools aimed at promoting increased integration—in terms of both test coverage and ease of use for the diverse group of developers, build engineers, and testers all using the product—is becoming important.

Testers are having to spend a lot of time getting the automated systems and suites to a fully reliable and maintainable state up front. They end up spending much of their time from then on monitoring these systems, and free time is set aside for exploratory testing. This is blurring the lines between a manual tester and an automation tester. In fact, even non testers have to test within their scope of operations. So, the tester is assuming the role of a leader, defining the scope of quality assurance and empowering the entire product team with the tools, metrics, and practices for monitoring the overall product quality.

Read More: http://www.techwell.com/2015/07/how-testing-changing-devops-world

Testers need to learn to understand the larger context of the project and the information they bring to the table. The real benefit is that testers’ work helps the product team make release decisions based on the product’s quality. This flows into quality being an information service, rather than purely a bug-finding numbers game. Once this maturity sets in, the drive to find bugs will have true meaning and will help not just the tester, but also the product under test.

Are You Choosing the Right Testing Tools?

A tester’s productivity and efficiency increasingly is being driven by the set of smart tools used over the course of the testing effort. A decade ago the tool choices were limited to a few commercial solutions. Gradually, as open source tools came into the picture, the question of whether to use commercial or open source testing tools became prominent.

This is not an easy question to answer, and the debate continues to this day. Commercial players and open source software makers are increasingly strengthening offerings and addressing weak areas to make it a very competitive marketplace. For instance, commercial players are usually looking to offer the most affordable price packages, while open source players are bringing in more professional experiences in the tool’s reporting capabilities, end-to-end test coverage, and coverage for niche areas such as automation, performance, security, and test management.

Read More: http://www.techwell.com/2015/06/are-you-choosing-right-testing-tools
Are Your Testing Practices in Line with Today’s Needs?

Practices in any discipline need continuous review and churn to ensure they are still effective and in line with current requirements. Software testing practices are no exception—the development landscape is highly dynamic, requiring periodic review and updating of practices.

However, this is often easier said than done. Where should testers start? Should the practices be revisited with a forward or a retrospective outlook? Should it be a test team or a product team activity? Should it be technology-centric or product quality-centric?

These are all valid questions, and the answer is that all these parameters need to be considered when defining what testing practices need to be.

A good place to start a review is at the end of the calendar year, when new predictions for the discipline are released by experts and analysts. Such a list is a great checkpoint to see if the team’s testing practices align with what is coming up in the year ahead. For instance, the trends for 2015 predicted an increase in social, mobile, analytics, and cloud testing; security and performance testing; test automation; and testing centres of excellence. While some of these ideas are new and forward-thinking in nature, others are retrospective practices that are regaining prominence.

Given how global product development has become, support across platforms, users, and devices has grown. This is necessitating tighter collaboration among product team members and stakeholders in order to deliver a scalable product in the marketplace. To accommodate this, decision-driven test management and keyword-driven testing are becoming increasingly important as core testing practices.

Read More: http://www.techwell.com/2015/07/are-your-testing-practices-line-today-s-needs

Building a Testing Framework for Digital Accessibility

Software testing frameworks are popular tools in some areas of testing. Many mistakenly believe frameworks are applicable only to test automation because they provide easy test implementation, enhanced productivity for testers, and empower dependable quality for the product. However, testers should look at building test frameworks in other areas of testing, as well.

Accessibility is a rich ground for frameworks to be built and leveraged. We continue to see discussions that support manual testing over automation in the space of accessibility, so let’s explore how and why to create an accessibility framework.

A robust accessibility framework is one that encapsulates testing methodologies, best practices, tools, compliance, checklists, and core scenarios under one umbrella. This creates a holistic implementation of accessibility from both regulatory and end-user standpoints. Each part of the framework need not be very verbose but should cover the core points in full essence, ensuring that the framework is scalable and maintainable.

Testing methodologies: Include processes such as promoting paired accessibility testing, implementing manual reviews on static code as well as dynamic implementation, leveraging accessibility toolbars such as WAVE, bringing in realistic end-users to test, etc.

Best practices: Include providing accessibility training to engineers (even ones who do no work on accessibility), keeping track of technology advancements and newer devices, engaging from early stages of product design, building a strong and productive business case, specifying information to include while filing accessibility defects, defining good use cases for automated accessibility testing, and researching accessibility devices and software to invest in.

Tips and Tricks:

1. Group test cases, wherever possible for quick and effective regression
2. The risk in jumping straight to automation is that you're only interested in getting a working script. Allow collaboration between functional testers and automation engineers (if possible make them one and same) so testing extends beyond the 'positive & happy flow scenarios'
3. A tester's role extends all the way until a bug is fixed, it does not stop when the bug is found
4. Make negative path testing a part of your test efforts
5. In God we trust, everything else we TEST – continue to look for ways to maximize your test coverage
6. Never stop collaborating with your programmers, analysts, customers and team mates
7. Testers should have a destructive mind but a constructive approach
8. Try a graphical representation of results, whenever possible

Books Shared by Your Colleagues:

- The Art of Software Testing, by Edward B. Burger, Michael Starbird
- I am a Bug, by Robert Sabourin
- Software Testing Techniques, by Boris Beizer
- Beautiful Testing, by Adam Goucher

Contributors: Karan Arora, Ankit Soni

Did You Know?

In 1979, Glenford J. Myers (American computer scientist, entrepreneur, and author) correctly hypothesized that there must be a distinction between debugging, which means identifying and eliminating bugs in the software code and actually testing the software in real world settings.

There is a world cup dedicated to software testing. For more information, visit http://www.softwaretestingworldcup.com/

Did you know, the famous quote “Testing shows the presence, not the absence of bugs” was reportedly first spoken in a conference by the NATO Science Committee in Rome in October 1969

Contributors: Adobe LOC Team, Karan Arora
About QAIT SHARE+

Welcome to the quarterly online magazine by QA InfoTech. As players in independent software testing for over 11 years now, we hope to use this as a comprehensive medium to share the latest and greatest in software testing with the community at large.

Team

Content | Rajini Padmanaban | Saurabh Ganguly | Akanksha Angrish
Design | Paras Ahi

QA InfoTech, A-8, Sector-68, Noida, 201309