

THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

# Tea-time with Testers

JUNE 2015 | YEAR 5 ISSUE IV

**Leadership Special Edition**

## Articles by –

Jerry Weinberg

Mike Lyles

Rajesh Mathur

Paul Crimmins

Matt Heusser

Anna Royzman

Ruslan Desyatnikov

Justin Rohrman

Rahul Verma

T. Ashok



Over a Cup of Tea with Dr.Cem Kaner

# Delivering High Quality Mobile Apps with Test Studio

R2 2015 Release Webinar: July 30, 11 a.m. ET

REGISTER NOW

According to Gartner, by 2017, mobile apps will be downloaded more than 268 billion times, generating revenue of more than \$77 billion. The first interaction your prospects have with you is increasingly likely to be on a mobile device. Are you prepared?

The new Test Studio R2 2015 release allows you to test your mobile, web and desktop applications easily and reliably. Join the release webinar which will cover everything that's new in the Test Studio collection including:

- Record and playback functionality for iOS native applications
- Predefined layout presets - giving you flexibility to change the Test Studio layout, depending on the tasks you are working on for ex: Test recording, Debugging, Execution, etc.
- Enhancements in web and desktop testing

## About The Presenters



Daniel Levy  
DIRECTOR,  
TELERIK ALM DIVISION



Shravanthi Reddy  
PRODUCT MARKETING MANAGER,  
TELERIK



Andy Wieland  
SOLUTIONS ENGINEER,  
TELERIK

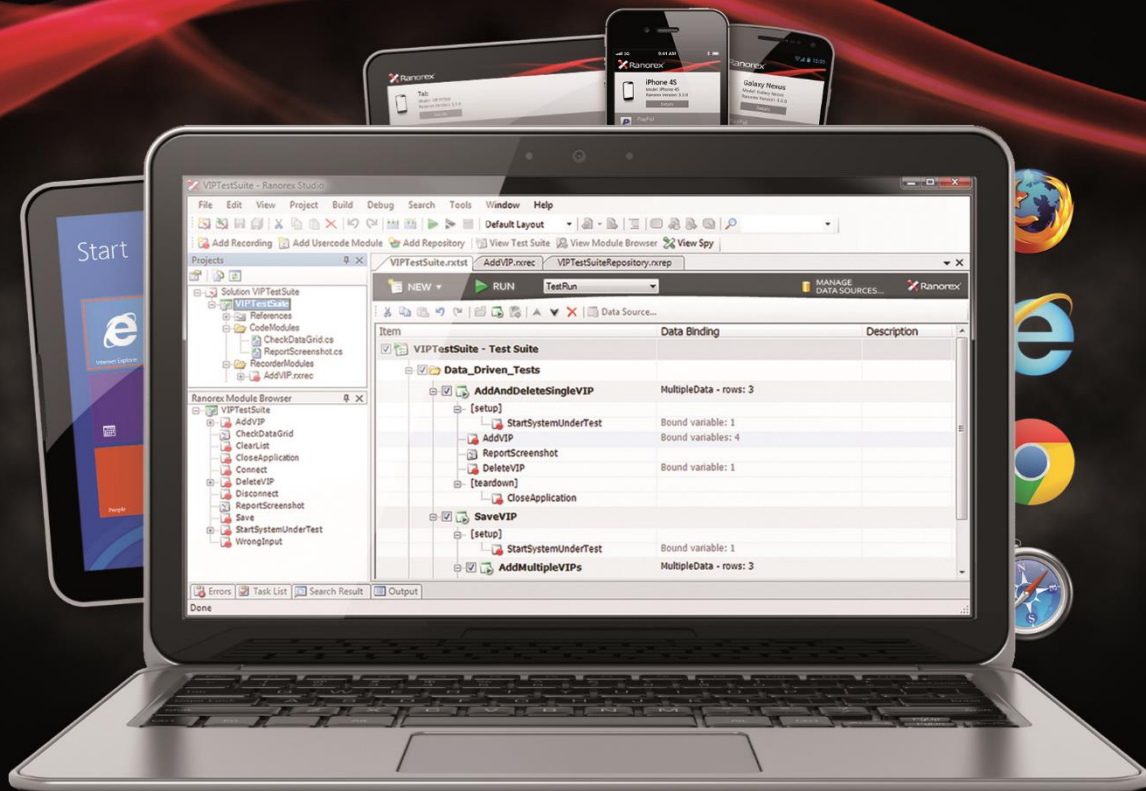


Iliyan Panchev  
PRODUCT MANAGER,  
TELERIK

Register



# Automated Testing of Desktop. Web. Mobile.



Robust Automation



Broad Acceptance



Seamless Integration



Quick ROI



Why Use Ranorex

[www.ranorex.com/why](http://www.ranorex.com/why)





# TeamQualityPro

## SEE EVERYTHING

## GET A FULL VIEW INTO ALL YOUR DATA



## TAKE THE FIRST STEP IN SAYING:

## YES TO:

- Real time reporting
- Instantaneous data gathering
- Objective, vetted data
- Comprehensive data
- Data that mirrors your process
- Data that is always current

NO TO:

- Manual reporting ○
- Data warehouse projects ○
- Subjective data ○
- Missing data ○
- Not aligned data ○
- Wrong time frame data ○





# TEA-TIME WITH TESTERS

**First Indian testing magazine to reach 115 countries in the world !**

Created and Published by:

**Tea-time with Testers.**

B2-101, Atlanta, Wakad Road

Pune-411057

Maharashtra, India.

Editorial and Advertising Enquiries:

- [editor@teatimewithtesters.com](mailto:editor@teatimewithtesters.com)
- [sales@teatimewithtesters.com](mailto:sales@teatimewithtesters.com)

Lalit: (+91) 8275562299

Pratik: (+49)15215673149

This ezine is edited, designed and published by **Tea-time with Testers**. No part of this magazine may be reproduced, transmitted, distributed or copied without prior written permission of original authors of respective articles.

Opinions expressed or claims made by advertisers in this ezine do not necessarily reflect those of the editors of **Tea-time with Testers**.

# Editorial



## The Leader is YOU!

In my last editorial I talked about how some people are (mis)using DevOps as a tool to kill testing. My special thanks those who wrote me letters and expressed their opinions in support of my views.

Out of some 30 odd letters, one of it has made me seriously think about one key problem. An excerpt from that letter is below:

*I'm a lone tester in my team and somewhere I feel that my independent voice/opinions as a tester are conveniently getting ignored and I'm being forced to do things that I do not recognize as real testing and that is only to ensure that we are shipping the product faster and faster.*

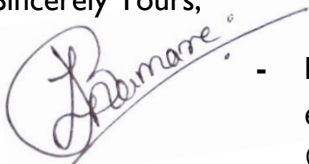
*I tried to explain my stand to our lead but no luck. He had no reasonable answer to justify why should I continue working like that. I'm sure that if our stakeholders get to know my concerns they will consider it. But I'm stuck. What should I do? Wish I could have a better lead...*

Well, this is not the very first complain of this nature I have come across. I know of some very good testers who want to see things around them changed for better but they lack support or expect someone else to take the lead. I believe this is the fundamental problem with majority of testers even today. They are aware of what kind of project environment can enable and empower them to do their job better but then they expect someone else to take stand for them or they are afraid of speaking for themselves or they probably don't know how to do it.

Some of my friends say, "Wish we too could get leaders like you have got, Lalit". Sure, having a powerful leader in organization who gives voice to testers and fights for meaningful testing has advantages of its own and it inspires actions too. But again, nothing is permanent. People change places, they change jobs and one can't have same kind of people everywhere. What does that mean then? In my opinion, that means only you can help yourself in the end, if you genuinely want to see something changed. From my personal experience I have realized that you don't always need someone with authority (aka leader) to stand by you to get heard and to get taken seriously. I have realized that you don't always need someone with authority to speak on your behalf. All you need to understand is that leadership does not only come with authority or bunch of people reporting to you. At any level of designation, there are plenty of possibilities to show leadership. In context of testing, I feel that saying no to bad testing is leadership, saying yes to adopt to required changes is leadership, proactively learning new skills is leadership, communicating things with courage and conviction is leadership, becoming first one to care and do something about things is leadership. Basically, every step, be it large or small that you take towards changing things for better is an act of leadership.

For motivation, you can always choose the role model for yourself and get inspired by their actions. Remember the [Eklavya's](#) story? If you keep on waiting for your messiah to come, probably nothing will happen. All I want to say is, the leader is (also) in you. You just need to wake her up...for yourself and for those who need someone to look up to...

Sincerely Yours,



- **Lalitkumar Bhamare**

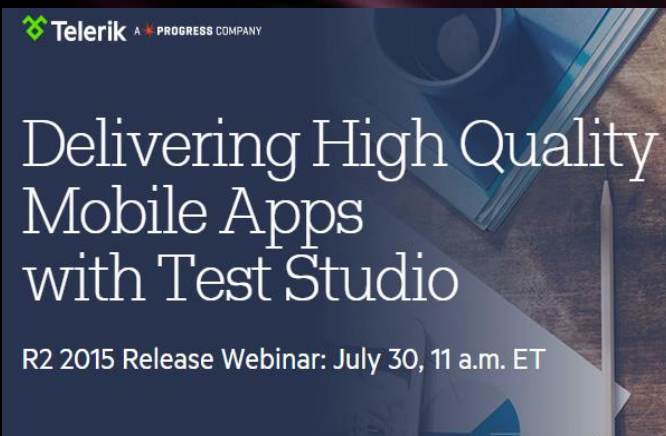
editor@teatimewithtesters.com

@Lalitbhamare / @TtimewidTesters





# QuickLook



## Editorial

### What's making News?

### Tea & Testing with Jerry Weinberg

### Speaking Tester's Mind

**Leading the Next Generation of Testers – 23**

**The Thing Called Leadership -27**

**What Type of Leader Are You? – 32**

### In the School of Testing

**Leading Beyond the Scramble – 35**

**Building & Maintaining A Successful QA Team -39**

**On Testing Leadership – 42**

**Notes on Test Automation -45**

### T ' Talks

**Unleash Your Potential, Unlock Others' Potential-66**

### From The Special Desk

**Over a Cup of Tea with Dr. Cem Kaner**

### Family de Tea-time with Testers



## On-Demand Webinar: [How Manual Testers Can Break into Test Automation](#)

Adoption of automated testing has not happened as quickly as many organizations have required. As more companies move toward implementing agile development as their software development lifecycle, more features are being implemented and released more quickly. This leaves less time for full regression testing of the system, nonetheless this should still be done. Manual testers need to transform into test automation testers as well.

Many manual testers believe they have to learn a development language in addition to the functionality of a specific tool to be effective. Add to that the in-depth or SME knowledge one must have about the system under test along with the development and management support required and it may not seem at all clear where to start.

Jim will cover the following in this session:

- The challenges faced by many organizations beginning the test automation journey
- Early stages of adoption and adding to the value of work handled by a manual test team with little programming knowledge.

[Learn how to make this jump as a manual tester](#) and focus on the right areas first e.g. automation test structure, object recognition and results interpretation.



Jim Trentadue has more than fifteen years of experience as a coordinator/manager in the software testing field. As a speaker, Jim has presented at numerous industry conferences, chapter meetings, and at the University of South Florida's software testing class.

Brought to you by -  **Ranorex®**



# STATE OF TESTING

# 2015

Results of the Largest Worldwide Testing Survey are out!

**Are you curious to learn more about the testing community? We believe that all testers are!**

The State of Testing Survey-2015 results are finally here and we are sure that you'll love the report when you see it. More than 860 testing professionals participated in this survey which helped us gather a lot of interesting insights on questions around the current state and the future of software testing.

You can download the Survey Report from here →

[Download Survey Report](#)

You can also get free copy of previous survey report – [State of Testing Survey 2013](#)

A big thanks to all our contributors

We want to thank all the QA & Testing bloggers as well as the tweeps who have been helping us reach more testers worldwide by sharing this survey and its results with their audience and followers.

**State of Testing** is brought to you by **Tea-time with Testers** in association with **PractiTest**.





# TESTEA

# TALKS



Evolution of the Art and Craft of Testing cannot be discussed without citing credits to the work done by **Dr. Cem Kaner** in last few decades. Dr. Cem's contribution to this field has had a significant influence on the way testing is being done by thinking and adaptive testers today.

Conversing about testing with Dr. Cem, knowing more about his journey and knowing more about his opinions on various subjects in the field was on our wish list from long time. I got to speak with Dr. Cem on various aspects of his life, expertise, opinion and suggestions recently. And this 'series' is a result of what we have been discussing from quite some time.

In part 1, we've covered Dr. Cem's career journey and his thoughts around testing education. Watch out for more in part 2 and 3 in coming issues of Tea-time with Testers. Enjoy the ever informative read from our conversation snippets!

- Lalit Bhamare

## Over a Cup of Tea with Dr.Cem Kaner





## World of software testing needs no new introduction of Dr.Cem Kaner but we are curious to know about your journey in software testing.

When I was young, I worked in my father's clothing stores. Many of my earliest memories are set in the stores. I was the son who was supposed to take over the business. One of my tasks as a teenager was to systematize our inventory control system, to settle our processes in order to make them suitable for computerization. We took the big step in 1970, contracting with a company that installed fancy computerized cash registers that sent data to be processed by sophisticated custom software. The result was a disaster. Their system was very expensive but compared to our manual system, it was less accurate, provided less information, and took twice as long (two weeks). Even worse, the cash registers were too hard to use. They interfered with the work of our salespeople. Eventually, we pulled the cash registers out of the stores and made do with a manual system for another decade.

### Any lessons that you learned from this experience?

I learned three important lessons from this:

- (a) This system was worse than worthless but the central problems were design errors, not coding errors. Since then, “works as designed” has never meant “acceptable” to me.
- (b) The most expensive problems involved un-usability. The system was so badly designed that our cashiers and salespeople kept making “user errors” that would take long times to work around. These people had been very effective in their jobs before the computerized cash registers and they became effective again once we threw out the technology. The “user error” problems were not problems with the users. They were problems of a system design that set these people up to fail. This was one of many occasions in our business when I would learn that recurring problems indicated failures of our systems, not of our staff. Years later, I would study Deming and realize that many of the lessons he was teaching were generalizations of the same lessons my family had been learning through experience as we rebuilt a business from nothing (one bankrupt store) to a national chain.
- (c) We could not get the system improved. We could not even cancel the contract. The best we could do was to stop using this terrible system (and keep paying for it). We learned that our traditional expectations about how contract law governs service contracts didn't apply to software contracts. Over several years, as I saw many more examples, I would eventually realize that when customers have no power to hold their suppliers accountable, many suppliers will cheat their customers. And when the legal system protects dishonest suppliers, the most dishonest suppliers will be able to drive the honest ones out of business.

As our family's friends computerized their businesses, I paid attention to their stories. The details were different but the same core problems appeared again and again.

**... CONTINUED ON [PAGE 61](#)**

# Tea & Testing



with

# Jerry Weinberg

## **What Is Leadership, Anyway?**

If you are a good leader,  
Who talks little,  
They will say,  
When your work is done,  
And your aim fulfilled,  
"We did it ourselves."  
- Lao Tse

Leadership is like sex. Many people have trouble discussing the subject, but it never fails to arouse intense interest and feelings. If you have trouble discussing the subject of leadership, this article is for you. Everyone says you should enjoy sex, so whom can you talk to when it doesn't work right? If you find leadership messy, embarrassing, and sometimes painful, you are not alone, though it may seem that way.



People who look really sexy are often great disappointments when it comes to actual performance. It's the same with people who look like leaders. They believe they're supposed to do it well by instinct, not by practice—and certainly not by reading books. If you are disappointed in your own performance as a leader, this article (and my [book](#)) brings you a simple message of hope: It doesn't have to be that way.

## THE RELUCTANT LEADER

According to Freud, our prejudices about sex are formed in early childhood. I think it's the same with our feelings about leadership. If you have always felt there was something slightly wrong about one person telling another person what to do, perhaps your experiences were like mine.

In grammar school, I was one of the smart kids. In the teachers' eyes, this made me a leading student, but in the students' eyes, it made me a ratfink. Whenever the teachers singled me out in class, the students punched me out in recess—if I was lucky. If I was unlucky, they wouldn't play with me at all.

With that kind of training, I soon learned about the dangers of being a leader.

Although school taught me that every good citizen is supposed to lead, the schoolyard taught me to be ashamed of any desire to lead. I learned to try not to become a leader. If leadership was thrust upon me, I always put up determined resistance. Whenever possible, I dealt with the question of leadership by pretending it didn't exist. And to make doubly sure I would never have to deal with leadership questions, I chose a career in computer software.

It didn't work. Whenever I did a reasonably good technical job, my co-workers learned to respect me a bit more. Because they respected me, they looked to me for advice, for leadership. If I'd been smarter, I might have isolated myself from them, refusing to give or receive information.

But I was naive and besides, I liked to be asked.

Sometimes I was asked to teach courses—a form of leadership. I was asked to sit on technical review committees—leadership again. I was put in charge of a project team, then a larger team. I had ideas I wanted to share even further than my own office, so I wrote papers and books—more leadership. Each time I realized what was happening, I backed off. Sometimes I was violent

Nobody was going to make me into a leader, so I was snared in a paradox. The more I struggled against becoming a leader, the more I was setting my own direction—and the more I was becoming a leader.

After all, isn't a leader someone who isn't satisfied with taking the direction set by others?

I grappled with this paradox for several years by withdrawing from anything that might lead other people. This withdrawal was like dealing with sex drives by pretending they don't exist. The leadership was still there, but I wasn't determining its direction.

Sometimes the direction was random, but most of the time I was easy prey to manipulators. In the end, I had to face the leadership issue, no matter how embarrassing it was.

## FACING THE LEADERSHIP ISSUE

I have a curious way of dealing with difficult issues. Whenever I want to learn about something, I arrange to teach a course on the subject. After I've taught the course enough to learn something, I write a book.

After twenty years of running leadership workshops, I think I've learned enough to attempt a book. Although I still have many unanswered questions, I have learned that I'm not alone. There are others out there who are tortured by leadership questions in their own lives:

- Are leaders really as stupid as they sometimes behave?
- Can I be a leader without becoming like those other people?
- How can I be a leader and keep up my technical skills at the same time?
- Is there a place for a leader in high-tech society who never had any technical skills to begin with?
- How much of my technical expertise do I sacrifice?
- What will I get in return?
- If I'm a leader, will I have to boss people around?
- Can I learn leadership from reading books?
- What else can I do to learn?
- Why do people see me as a leader, when I don't feel that way?
- Why don't people see me as a leader, when I feel quite capable?
- What if I don't want to assume leadership responsibility?
- What is leadership, anyway?

These are hard questions. Perhaps the last is the hardest of all. What is leadership, anyway?

## A CONVENTIONAL BUT FLAWED VIEW OF LEADERSHIP

Psychologists and management theorists have dozens of models of leadership, with a typical one of their texts offering this explanation:

There are two principal ways to identify the leaders of a group:

Asking the members to identify which members they regard as most influential in directing the group, or

Asking observers to name the most influential members, or to record the frequency of effective influencing actions.



Although they appear to be scientific, these models are based on the opinions of the members or the observers, and on their ability to observe "effective influencing actions."

Over the years, I began to see some flaws in that approach.

For instance, a company recently retained me to help a group of computer programmers improve their problem-solving techniques. The company was losing thousands of dollars of sales each passing day because of a subtle error in its software product. Until the programmers could find the error, the product was useless. To help the group, I videotaped them as they struggled to find the error.

In one hour of observation, the "effective influencing actions" of the four programmers involved looked like this:

Arnie 112 actions

Phyllis 52 actions

Weber 23 actions

Martha 0 actions

Martha's actions were easy to record. She sat like a zombie through the entire hour, studying the printout of the erroneous program. She said nothing, made no gestures, and didn't even smile or frown. Without question, she had no influence on the group whatsoever.

After consuming an hour with their effective influencing actions, the other group members were no closer to solving the problem than when they started. All of a sudden, Martha lifted her eyes from the listing, pointed a finger at one line, and said, ever so quietly, "This word should be '87AB0023', not '87AB0022'." Then Arnie, Phyllis, and Weber resumed their agitated discussion. They terminated the meeting ten minutes later, after they had convinced themselves that Martha was indeed correct.

When I asked the group who had been their most influential member, they all said, "Arnie." Then I played the videotape, asking them to be especially alert to the method by which their problem was solved. After watching the tape, Arnie, Phyllis, and Weber changed their answer to "Martha." Why? Because in terms of solving their problem, the table of effective influencing actions should have read

Arnie 0 actions

Phyllis 0 actions

Weber 0 actions

Martha 1 action

Without Martha's contribution, the meeting would have gone nowhere, yet non-programming psychologists would have probably missed Martha's role entirely.

When such nontechnical psychologists observe our workshops, they are consistently befuddled by the dynamics of the teams as they solve technical problems. It's as if the psychologists were watching people from another planet, people whose culture and language look and sound superficially like ours but are entirely different.

## CONTRASTING MODELS OF THE WORLD

In order to recognize leadership in a group, you must have a model that somehow matches the group's culture. For instance, if their model of "problem solving" is too simple, psychologists will have trouble understanding leadership in technical environments. Someone once said that the central dogma of academic psychology is that there is one and only one correct solution to every problem—and the psychologist knows it. Any psychologist who believes that simple model will have trouble defining leadership in a way that works in real-world situations. For one thing, such a person would certainly never recognize Martha as a leader.

There are many models of how people behave in the world. Even within the discipline of psychology, there are dozens of major models and hundreds of minor variations. The sociologists' models differ from those of the psychologists, as well as from the anthropologists, the economists, the executives, and the janitors. The reason there are so many models is that each of them is useful, but only in some contexts. The problems arise when we try to apply a model that doesn't match the situation in front of our eyes.

In my book 'Becoming a Technical Leader', I have used and developed a number of models for understanding that slippery phenomenon we sometimes call "leader- ship." To be an effective leader, you will have to have many models at your disposal, and be able to switch appropriately from one to another as the situation demands. Most of the models I favor may be considered organic models, in contrast to linear models, but there are times when I can be quite appropriately linear.

Organic models can be contrasted with linear models on several dimensions: the way events are explained, the way a person is defined, the way a relationship is defined, and the attitude toward change. Let's compare the two types of models on each of these in turn, then see how they affect the way leadership is defined.

### Explanation of an event

Linear models get their name from the assumption of a linear relationship between events; that is, one effect stems from one cause, and vice versa. Organic models may be characterized by "systems thinking": the belief that event X is the outcome of hundreds of other factors, including the passage of time.

The strength of linear models lies in the large number of events that can be well understood in terms of a single cause their weakness arises from events of greater complexity, which include, unfortunately, most critical events involving people.

The threat/reward model is an example of a linear model with morality added; there is one and only one right answer, and anyone who cannot see it must be either dumb or bad. When we use this model, we tend to feel stupid and ashamed in the face of events we don't immediately understand.

The strength of organic models, by contrast, is that they enable us to be comfortable in complex situations that we don't fully understand. When we use these models, we're able to open our minds to dozens of possible explanations (many of which can be true simultaneously) until we have sufficient information to fi make an appropriate choice.

One weakness of organic models is that they may prevent us from acting at all. Effective leaders often have to act even when they don't understand all possible factors. In order to use organic models, you must be able to live with the occasional error.



## Definition of a person

Linear models tend to place individuals in categories. Organic models define people in terms of their uniqueness, that is, their sameness plus their differences. The useful side of the linear approach is that it allows us to deal with people quickly and efficiently. We can order a cup of coffee in the morning without considering the waiter in his full-blown individuality.

The useful side of the organic model is the way it allows different people to find a common basis for working together in complex situations. People who abide by the organic model tend to see other people as sharing the same life force, the same spiritual base, and the same kind of relationship among their unique individual parts. They don't compare people to some standard, so they are not tempted to shape people to some ideal image. Such people tend to see the job of a leader as getting people in touch with their own inner harmony.

Linear models become less useful when they slip over into defining people in terms of what they should be. If people differ in their thinking, feeling, or acting from this ideal, they may be "treated" with attempts to cut them down to size, or stretch them out. The threat/reward model is a linear model that says people's actions are completely defined by the threats and rewards confronting them. When we operate out of the threat/reward model, we tend to see the leader's job as issuing threats and doling out rewards.

When we hold to the threat/reward model, we tend to have low feelings of self-worth and the worth of others. We give ourselves—and others—messages such as "don't work hard enough," "I talk too much," "I'm too fat," and "I can't get people to do what I want." These messages often lead to feeling frustrated, angry, and unworthy, though if asked, we will deny these feelings.

## Definition of relationships

Linear models tend to define relationships in terms of roles rather than people: the boss rather than the person actually exerting influence. The organic model tends to define relationships in terms of one unique person to another unique person.

One useful side of linear models is that they allow planning of large-scale operations, where it would be impossible to consider each relationship in its full glory.

Linear models are less useful when they are extended to one-to-one relationships, where individuality becomes critical in understanding the interaction.

People who adhere to the threat/reward extreme tend to see power as existing in the role, rather than the relationship, so they put great store in titles as a way of defining relationships. When things get tough, they are likely to invoke their "authority," or yield to someone else's. Although this view of power can be useful statistically, it falls apart when applied to one-on-one situations. In love, or teaching, or leading, it's not too useful to think in terms of one person always on top and the other always on the bottom. When we do think this way, we tend to experience emotions of fear, anger, aggression, guilt, and envy toward the other person.

The usefulness of organic models is most clear in one-on-one situations. The two persons, regardless of their roles in the current situation, are presumed equal in life significance. Organic models lead toward problem solving in which everyone benefits.

When we act this way toward other people, our most common emotion is joy of discovery. Sometimes, however, we get so wrapped up in this joy that we fail to get the job done, if there is a job to do.

## Attitude toward change

When we examine the processes of change, linear and organic models are in stark contrast. Linear models tend to see change as an orderly, one-thing-at-a-time process.

Underlying organic models is the fundamental idea of systems thinking: "It is impossible to change just one thing at a time." Linear models tend to be most effective in relatively stable situations, but when things start changing, they get us into trouble.

One kind of trouble is that when change doesn't fit our model, we try to stop it from happening. When faced with change, we may feel paralyzed and helpless. People holding to organic models need security just as much as anyone else, but they obtain their security by taking risks and by tolerating ambiguity.

Under the influence of the threat/reward model, we may try to assure our security by struggling to keep all people and relationships forever the same. If we do feel the need to change, we usually direct it at someone else. And we usually try to change them by "removing" their "bad" behaviors.

Organic models expect and accept change as a normal part of the universe. Some organic models go even further, and welcome change as an opportunity to go into the unknown and grow. They have faith that growth is a natural process by which our wonderful potential is realized, in the same way a seed must grow to realize the wonderful potential of the flower. We'll sometimes refer to such organic models as seed models.

## AN ORGANIC DEFINITION OF LEADERSHIP

That's a very rough sketch of the difference between linear and organic models. Obviously, no person uses either type of model one hundred percent of the time, and that's one reason why the definition of leadership is so hard to pin down.

Linear and organic models lead to contrasting ideas of what constitutes leadership.

In the extreme cases, the threat/reward model of leadership may be characterized by the words "force" and "judge," and the seed model, "choose" and "discover." In the seed model, Leadership is the process of creating an environment in which people become empowered.

For example, before Martha made her observation, Arnie, Phyllis, and Weber were getting nowhere with their problem-solving techniques. After her observation, the environment changed so that the same techniques became powerful.

But Arnie, Phyllis, and Weber were also exercising leadership, in a surprising way, by creating an environment in which Martha was free to work in a style that was powerful for her. Some people in groups simply cannot tolerate one member not "participating," by which they mean doing a lot of talking. Talking wasn't Martha's style, and the others knew it so they left her alone. That's also leadership. Instead of leading people, as in the threat/reward model, organic leadership leads the process, leading people requires that they relinquish control over their lives. Leading the process is responsive to people, giving them choices and leaving them in control. They are empowered in much the same way a gardener empowers seeds—not by forcing them to grow, but by tapping the power that lies dormant within them.

Leadership in the seed sense is creative and productive through other people. It is an organic definition, because it works through creating an environment rather than confining itself to a few focused actions—threats or rewards—in a few specific instances to create a few specific results.



To people ensnared by linear models, this organic model of leadership may seem vague and wishy-washy, but it actually lends itself to more precise quantification than the more conventional models. It's especially useful in technical work because, unlike the more linear models, it allows us to take innovation into account. Innovation is concerned with redefining a task or the way the task is done. Linear definitions of leadership assume that observers have a perfect understanding of the task. Such definitions filter out innovations that the observer hasn't seen before or doesn't understand. Such blinded observers obviously cannot see the possibility of leadership through innovation. In an age of high technology and discovery, such constraining definitions are practically useless.

The organic model of leadership covers all sorts of work, especially the highly technical. It does not offend technical workers, and can actually be used to measure innovative contributions such as Martha's. Psychologists might not agree with my approach, but I have found it a practical way to describe technical leaders and technical teams.

## QUESTIONS

- Observe someone you consider a leader. How is this person's life different from yours? Which of these differences are a result of being a leader? Which of them are a cause of being a leader?
- How would you expect your life to be better if you increased your leadership skills? Which of these improvements will arise from your changed behavior, and which from recognition of the changed behavior of other people?
- How would your life change for the worse if your leadership skills increased? Will these changes be worth the rewards? How can you change, yet behave in such a way that these changes do not affect you so adversely?
- Make a list of situations in which your presence seems to increase the productivity of others. Alongside this list, identify situations in which your presence seems to decrease the productivity. How can you characterize the differences between these situations? (For example, increases in productivity might involve working with people you know well, or working on a problem that is new and different. Or perhaps just the reverse is true for you.) What do these lists tell you about yourself and the environments that empower you?
- Based on the two lists from the previous question, are you statistically an asset to groups, or a liability? Do you seek out situations in which your leadership will be positive, or do you more often look for situations in which you can learn to do better? Do you, in fact, learn from these situations, or do you just keep repeating yourself?

# Biography

**Gerald Marvin (Jerry) Weinberg** is an American computer scientist, author and teacher of the psychology and anthropology of computer software development.



For more than 50 years, he has worked on transforming software organizations. He is author or co-author of many articles and books, including *The Psychology of Computer Programming*. His books cover all phases of the software life-cycle. They include *Exploring Requirements*, *Rethinking Systems Analysis and Design*, *The Handbook of Walkthroughs*, *Design*.

In 1993 he was the Winner of the **J.-D. Warnier Prize for Excellence** in Information Sciences, the 2000 Winner of **The Stevens Award** for Contributions to Software Engineering, and the 2010 **SoftwareTest Professionals first annual Luminary Award**.

To know more about Gerald and his work, please visit his Official Website [here](#) .

Gerald can be reached at [hardpretzel@earthlink.net](mailto:hardpretzel@earthlink.net) or on twitter @JerryWeinberg

People who look really sexy are often great disappointments when it comes to actual performance. It's the same with people who look like leaders. They believe they're supposed to do it well by instinct, not by practice—and certainly not by reading books. If you are disappointed in your own performance as a leader, **Becoming a Technical Leader** brings you a simple message of hope: It doesn't have to be that way.

If you aspire to be a leader, this book is for you. Its sample can be **read online**.

Know more about Jerry's writing on software on **his website**.

**Becoming a  
Technical Leader**  
an organic problem-solving approach



Gerald M. Weinberg

**TTWT Rating:** ★★★★★

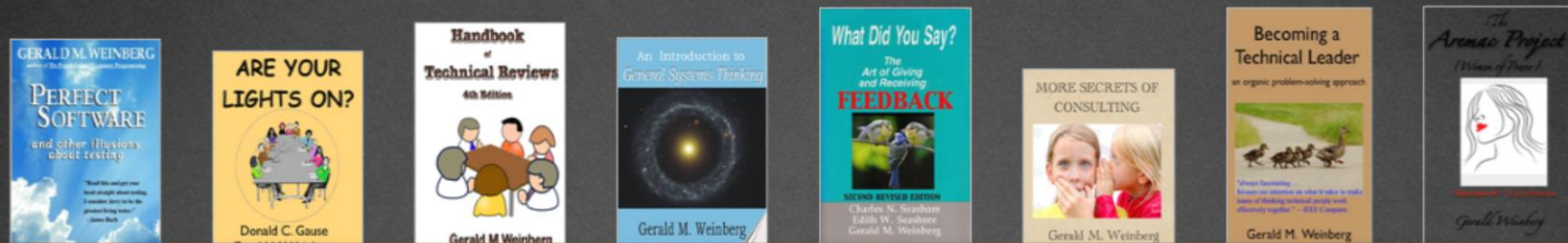


# The Bundle of Bliss

Buy Jerry Weinberg's all testing related books in one bundle and at unbelievable price!

## The Tester's Library

*Sold separately, these books have a minimum price of \$83.92 and a suggested price of \$83.92...*



The suggested bundle price is **\$49.99**, and the minimum bundle price is...

# \$49.99!

Buy the bundle now!

**The Tester's Library** consists of eight five-star books that every software tester should read and re-read. As bound books, this collection would cost over \$200. Even as e-books, their price would exceed \$80, but in this bundle, their cost is only \$49.99.

The 8 books are as follows:

- Perfect Software
- Are Your Lights On?
- Handbook of Technical Reviews (4th ed.)
- An Introduction to General Systems Thinking
- What Did You Say? The Art of Giving and Receiving Feedback
- More Secrets of Consulting
- Becoming a Technical Leader
- The Aremac Project

[Know more about this bundle](#)

A green pendulum bob hangs from a thin wire, positioned centrally over a circular pattern drawn in the sand. The sand is light-colored and textured, with the pattern consisting of concentric, slightly irregular lines. The entire scene is framed by a dark blue border.

# Speaking Tester's Mind

- straight from the author's desk



# Leading the Next Generation of Testers



by Mike Lyles

Every parent, coach, counselor, manager, mentor, teacher, or advisor, has experienced the complexity of guiding others in the right direction. If you've been one or more of these, then you have likely felt the desire in your heart and your mind to be part of a positive change in someone else's life. This becomes the core of your passion. The joy of knowing that you were part of helping someone else grow is usually more than you can imagine.

If you've been in IT for long - especially if you've been in software testing for long - then you have experienced the exponential growth in technology over the last few decades. You've seen the world move from large mainframe computers to desktops, to laptops, then to smartphones and mobile devices, and now wearables. Everywhere we turn there is a new software package that can "help you test better". The strategies for software testing are changing. Yet, as with the roles listed in the introduction of this article, we must understand the core values and goals of our craft. And we must instill this knowledge to the next generation of testers that are joining the workforce every day.

At the time I am writing this article, the top news in the United States is around Zappos. The company has moved to a new organizational structure called Holacracy ([www.holacracy.org](http://www.holacracy.org)), in which managers have been removed and the team is empowered to work together. Zappos CEO, Tony Hsieh is quoted as saying, "Think of every employee as a mini entrepreneur. Which is not really for everybody; some people like to be told the 10 things to do". He's absolutely right. In fact, it was proven this week as 14% of the staff accepted a buy-out to leave the organization because they didn't like the new structure.

I'm honestly surprised that it was only 14%, because I would think many of us want to have some direction and guidance. In fact, even if you're a CEO, you look for direction from your board members, customers, and supporters.

If you're a test manager, or a test leader in your organization, your responsibility to the community is heavy. And below are just a few of the things you should consider as you lead your team:

## **1. First Things First**

The responsibility of a leader of a testing team starts with ensuring that everyone is in sync and that they are speaking the same language. Nothing will kill an organization faster than having a team that is divided on how they plan to leverage the testing practice in your organization. Talk with your team. Get to know what they believe, how they think testing should be done. Understand their ideas, philosophies, and their beliefs on the goals of a strong team. If they are not aligned, then it is your responsibility to ensure something is done to sync the team and align the organization.

## **2. Education**

A strong testing team is one that is not satisfied with coming to work every day and practicing the same thing day after day, year after year. If you're a parent, you want your kids to have the best education they can get. If you're a coach, then you want your players to study and learn their position on the team and you want to ensure they are studying the right methods. If you're a teacher/advisor/mentor/counselor, you feel compelled to instruct your students on the best suggestions you can think of to ensure success for them. The same applies to leading a testing team. It is your responsibility to ensure that your team understands the schools of testing, understands which work, as well as which may not. It's important that they study in the testing community. They need to know why one practice is not the best for your team, and why another needs to be followed. They need to understand the value of testing and why your team is following a certain set of goals, methods, and guidelines. Give them the direction. Show them what they should study, where they should study, and advise them on what you think works, and what you think will not work.

## **3. Play Well With Others**

If you're a parent, you've likely had that moment when your child is having a difficulty connecting with other children. Or maybe they are debating with another child and you have to step in. If you're a coach, you want your players to win, but you want them to win fairly and by playing with a respectful conduct. Sometimes the best parenting and coaching comes with leading by example; by showing those that we are leading that we, too, can do the same and we conduct ourselves in a manner that they can imitate. This same philosophy holds true with testing organizations. As a test manager, you need to ensure your team sees how you want the testing group to interact and engage with sponsors and stakeholders. Show them the best way to engage and seek support from their customer. And most importantly, place them in situations where they are able to show that they are learning this practice, and give them opportunities to grow their engagement.

## **4. A Fool With a Tool is Still A Fool**

Over the years, we've seen teachers on the news complain that kids have become so dependent on calculators and smartphones that they have lost the simple core concepts to doing things on paper and using their brain to solve problems and answer questions. As parents, we've watched as our children no longer open a book made of paper, but instead they use their mobile devices to read books. There's something to be said about remembering the basic concepts of using our most powerful tool (our brain) to accomplish things. As a test manager, you have surely heard all the debates over test automation, testing tools, and new software and services to "improve your testing organization". But it is important



for us to remember that while tools may simplify a process, and they may even be the best fit in certain contexts, a good testing team understands the core concepts of testing without the need for a tool. Drive your teams to understand and believe that before any organization can invest in a tool, they need to have the processes and practice in place before that tool can add any value.

## 5. Send Them To Summer Camp

When I was growing up, every summer, there was a summer camp for kids. Parents would allow their kids to spend several weeks at the summer camp, learning new things, having fun with other kids, and returning home with new ideas, skills, and understanding. This may surprise you if you know me, but I was a very shy child, didn't socialize a lot, and didn't want to be part of the camps. I never attended one. My parents were willing to let me go had I wanted to go, but I was not interested. I spent my summer breaks elsewhere. As a test manager, I strongly suggest that you give your team this "summer camp" opportunity by suggesting them to attend conferences, testing meetups, workshops, and extended trainings. This gives them an opportunity to meet people from other cultures, other companies, and other disciplines. It gives them the opportunity to engage with leaders in the industry, build their skills, and, most importantly, bring back those skills to help your team.

## 6. Nothing Fails Like Success

The first time I ever heard the phrase "**Nothing fails like success**" was from Stephen Covey. He was holding his arms so that his fists are facing each other, and then as he said those words, he moved his right fist upward, leaving his left fist in the same position. He said that what works today may not be good enough for tomorrow. I'll never forget those words, because it reminds me daily that even if you're successful as an organization today with a set of practices, it may not be good enough tomorrow. The world is changing daily. And it is our responsibility as test managers to not only keep up with the change, but also to coach, mentor, and empower our testing teams to recognize and react to those changes as well.

## 7. Share With The Whole Family

Teach your team to "share with brothers, sisters, aunts, uncles, parents, and elders". What do I mean by this? If I learned anything about mentoring and coaching, I learned one important fact...in my lifetime, I have almost always learned more from those I mentored and coached than I ever expected to teach. Encourage your team to not be afraid to share their ideas to their peers, their bosses, their stakeholders, their business partners, and other organizations. We can all learn from one another, and the more we can influence in every direction, the more aligned the organization will become. Some of my best ideas have come from those that work for me. And I hope that in my career, I've been able to influence and guide peers and senior management to think differently about an approach to testing.

## 8. Find What You Love

In a memorable address to the graduating class at Stanford University, Steve Jobs said, **"You've got to find what you love. And that is as true for your work as it is for your lovers. Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. As with all matters of the heart, you'll know when you find it. And, like any great relationship, it just gets better and better as the years roll on. So keep looking until you find it. Don't settle"**. Seek testers who live, eat, and breathe testing. Seek testers who are curious and want to be part of a quality product. Seek testers who are

passionate about their career in testing. For those that may not seem to love testing, work with them, inspire them, give them opportunities to fall in love with the practice. Sometimes you have to accept the fact that not everyone is built for testing. We know that testers are a special breed of people. You may have to help them make the decision to stay or go. Would you want your favorite sports team to be filled with players who want to win the championship or would you want it to be filled with people who are just happy earning a paycheck for each game? We want winners. And if you are a great leader and test manager, you will find them, motivate them, and win.

I have so many other suggestions, ideas, and thoughts on leadership. I plan to release a book on leadership itself in the coming year. Keep your eyes open for it. It has been a pleasure to share this information with each of you. Please reach out to me anytime.



**Mike Lyles** is a Quality Engineering Program Manager with over 22+ years of IT experience, gaining exposure through all aspects of IT in various roles – software development, program management office, and, ultimately, software testing. He has led various aspects of testing: functional testing, test environments, software configuration management, test data management, performance testing, test automation, and service virtualization. In his current role, he is responsible for defining and driving the efforts for test automation, performance, environments, and tools, and leading a Quality Management Office for the organization.

Mike has been an international/keynote speaker at multiple conferences and events, and is regularly published in testing publications and magazines. Mike's passion to help others improve and grow in the field of testing, leadership, and management is his key motivation. You can learn more about Mike on Twitter @mikelyles, on LinkedIn at <http://www.linkedin.com/in/mikewlyles>, or his website at: [www.MikeWLyles.com](http://www.MikeWLyles.com)



# THE THING CALLED



by Paul Crimmins & Rajesh Mathur

"What do you mean by leadership?" Paul retorted when I told him that I was writing an article for the magazine on Leadership, especially test leadership.

"Well, what is your understanding of it or your experience with your leaders?" I replied calmly. The conversation was not going where I wanted it to go. It appeared to me as if I had touched a nerve.

"What does it mean to me?" Paul Said. *"Firstly let me start with something that happened the other day that lead me to ask this question to you. I was in a meeting and there were some unhappy people in the room. One party was expecting that the project was already well under way for the last week while the other advised that they had not even started yet. After some discussion about how this had occurred it came down to some processes in place were not being followed. What came to light was the people not following the processes were actually the ones that asked for a process to be in place."*

He waited to catch a breath.

*"One of the key management "or leadership values that immediately comes to my mind is "Leading by example" which clearly was not happening here. This lead to frustration among those affected and made me think 'If the management is not following the process then why should we?' This is obviously not the correct attitude to have but why do management act this way and what can we do about it? I for one am not really sure as it seems we are not allowed to tell management or the leadership team that they cannot have something because they are not following the correct process."*

Aha! I thought. This is going to be a nice discussion.



Paul continued, *"Other values of leadership and management that I would look for and like to emulate are; being supportive of the people you are managing, respect and mentoring. I do not believe the leadership by fear or deceit is the right way to go, my own past experiences have found that I am far more willing to work for and put in extra effort (hours??!!) for people who have some or all of the values I have mentioned above. The last thing I will add is that unfortunately in my experience I see far more of the fear and deceit than respect, mentoring and leading by example."*

Now, let me ask you, was Paul right about his expectations or his understanding of leadership?

I am sure he is right about his expectations and desires. He wants direction, respect and mentoring from people who lead teams. We all do. No matter whether we are a junior tester or a senior Vice President of Software Engineering department. What is not right in Paul's understanding is that these people are mostly just managers, not leaders. Leading a team does not make you a leader. Leadership is hard work.

Managers manage others, Leaders inspire others.

I will not get into what is the difference between management and leadership. For that, either read something at Harvard Business Review or look at pictures on LinkedIn. Everyday someone is posting some vacuous junk to tell us what the difference is. Posting a picture is easy. Practicing what you preach is hard.

Who is a leader?

Is someone with a lead or manager title a leader? Not really. A title is just a title. If you look closely, all world leaders started with no title. They were selfless, and they were inspiring others who followed them. Look at Abraham Lincoln, Mahatma Gandhi, Martin Luther King Jr., Nelson Mandela or any other world leader; they did not do what they did for power or title. They not just inspired others, but also empowered others to do the right thing. They were not born special, they took help from others to achieve their goals. Leaders take help from others. Do not shy away from asking for help when you set out to achieve something which makes a difference. Leaders make a difference.

Leadership is a relationship, not a title.

If we go back to the discussion that I had with Paul, you will notice (and hopefully realize) that we all seek certain qualities in the leaders and one of them is leading by example. We also call it 'walking the talk'. We have all come across those leaders or managers who are untrustworthy, deceitful, secretive and plain intolerable. These people are mostly self-centered, competitive and insecure. It is obvious that we cannot expect them to walk the talk. Now you may consider either working around them or just by defying them; depends on your particular context.

I worked with a manager who was not just manipulative, but also kept changing the versions of the conversations we used to have. This person changed his stand on testing often and that put the team on the toes because they found it hard to present value to the stakeholders. He sometimes followed testing standards and sometime followed what he called fit-for-purpose testing. For example, he once told people

what User Acceptance testing was and next day disagreed on the definition he gave a day before. This leader (?) was more problematic to his team than difficult stakeholders teams worked with. I know that no one likes to work with people like that person. So if you think that you have any of these symptoms, it is time to get rid of those. This is how you will move towards the path of good leadership.

What we expect is different. We feel good when we have a leader who is there to support, motivate and guide us; a person who is there to watch our backs and make us feel empowered. Do you remember the core elements of definition of Exploratory Testing - Personal freedom, responsibility, optimizing value and learning? Do you?

See the definition below:

*Exploratory software testing is a style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the value of her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project.*

A good leader will give you personal freedom so that you can deliver value. S/he will let you learn and experiment. Good leaders are coaches. They tell you what you are doing wrong and correct your mistakes. But they also want you to learn. Now it is up to you to use that personal freedom to optimize value personally as well as professionally.

When I started writing this article, I was in a dilemma. I had two possible options to write. One, to tell others how to be a good leader; and the other was to share my experience. Considering that I am still a learner leader, I skipped that. I do not want to tell others how to be a good driver while I am still learning driving. That would be dangerous. So I chose the other way. I decided to tell you what I have experienced so far, what I have learned and what I want to learn and do. The last point about learning is also a wish list or we can call that expectation. The expectation of what I want to see in leaders.

See, what I have noticed so far is that leaders follow the practices that let their followers transform ideas and vision into reality, silos into a collaborative team and problems into a solution. They keep communication open and honest. In many cases I have seen that talking is one way, but that does not mean that communication is going to be one way too. Effective leaders talk, but they make sure that people listening to them understand what they talk about and take appropriate action. This is what open communication is: keeping it two-way even if just one person is talking. It is up to you to not just listen to the leaders, but understand and act upon.

Talking is not communication. Leaders communicate.

When leaders communicate, they give others opportunity to challenge them. Since people feel empowered to take action, they also feel empowered to challenge. Testing used to be considered as a boring, repetitive process where people followed templates and some standards and that was all. Context-driven testing community challenged that notion. People like Cem Kaner, James Bach, Brian Marick and few others stood up against mindless application of processes. They described testing as a cognitive complex activity which needed application of critical thinking and constant learning. They emphasized on

communication and value to stakeholders. We see them as leaders in testing community because they exhibited the leadership qualities.

Leaders look for opportunities to change the current situation. They search for opportunities for innovation and challenge the status quo. They take mistakes as learning, explore the unexplored and take risks. James Bach could have stayed at Apple Computers as Test Manager. I am sure he was doing fine there. He could have used ISO, CMM and IEEE processes. But he decided to do something different there.

Breaking free from daily routines is vital if you want to be a leader. You cannot become a leader if you follow the status quo. If you do, then you will be a follower. You must question the status quo. Search for challenging assignments. Easier assignments and projects are for followers, not for leaders. Personally I have mostly been handed the complex, difficult and challenging assignments. I took them as rewards. Remember, only those with leadership abilities are given complex tasks to manage. Why? Because other leaders trust that they will be able to sail through the difficult situation without breaking a sweat. Every job is an adventure. Try to enjoy it.

As a leader you do not have to reinvent the wheel; but you do not have to accept the process that is 'just working'. Why not challenge the process to achieve better results? How many leaders have you seen who say that, "We have always done it this way?"

Leaders challenge the process.

My manipulative manager did not have a vision. He possibly had an agenda, a personal one. My good manager had a vision, a shared one. I think all good leaders have shared aspirations. They have a vision and they see others in that vision. They see a good future and they want others in that good future. They mobilize others so that everyone can achieve the shared goal. Leaders talk about the shared vision and they ensure others join them in visualizing that. Did you ever hear the famous "I have a dream" speech which Martin Luther King, Jr. gave? Look at Wikipedia for it. His dream or vision mobilized millions which became a shared vision.

If you work for a company, align your vision with your employer's vision. Your vision should be realistic and you must be passionate about it. Talk to your teams and colleagues about it so that they can envision it. Inspire them about the positive outlook and hopes that you have about your vision. If others can't see it or do not find it attractive enough, it will not move them forward.

When I joined my current employer, it was a process factory. Testing was a routine job for all. Overarching Test strategy was a hundred page document that no one had ever read. I had multiple informal meetings with the team and I told them stories about all the international conferences and meetups I attended, I shared my knowledge of what I learned from context-driven community, I told them how we could make it a more respectful job for all of us where everyone can achieve a lot more than a mere salary. The testing team is a strong and dynamic context-driven team. We practice ET, SBTM along with scripted checking, we write our own automation tools and we learn together. We are considered by senior management as trusted advisors. But we have not stopped yet. We have a roadmap which guides us forward. Where required, we change direction to align to the business objectives. Team members are self-motivated; they learn and share their knowledge, work collaboratively and have trust and mutual



respect. Politics and bureaucracy does not bother them much because it does not reach them often. They challenge each other on complex tasks and also provide coaching where required. Each one of our team members is responsible for their own work and uses discretion and judgement. We delegate as well as communicate often. There are no secrets in the team.

Leaders inspire a common vision, mobilize others to act and build effective teams.

Our jobs as testers are hard. We are required to learn more than others. We are expected to find issues in someone's creative work and report that to everyone within and beyond the project team. And we are still expected to keep a good relationship going with everyone else. Due to the nature of our job, we have to face more conflicting situations than others. Leaders know how to deal with conflicts and manage conflicting situations. I wrote a blog post about this subject some time ago, which can be read here:

<http://www.dogmatictesting.com/2013/03/understanding-conflict.html>

Having said all that, it is not impossible to be a good leader. I do not think people are born with leadership qualities. Maybe some people have characteristics and qualities to be a leader, but leadership is not genetic. It can be learned. Do not accept it when someone says that you can't be a leader. The moment you accept that you don't have qualities of becoming a leader and that your destiny is to be a follower, you will lose all the possibilities of learning the art of leadership. No research has ever proved that leaders are born.

Leadership is an acquired skill which can be learned by observing the traits and abilities of other good leaders, by coaching and support from others. This skill can be improved by constant practice, learning and feedback. You may fail to be a perfect leader (no one has ever become one), but you can be good leader which will be good enough as there are not many around. All you need is desire and motivation.

### About the Authors:

**Paul Crimmins** is a context driven tester who is continually learning about the craft. He is an aspiring leader who would like to emulate the attributes that he has mentioned in this article. He works at AHPRA as a Team Lead – Performance Testing. His Twitter handle is @Paul\_Crimmins

**Rajesh Mathur** is based in Melbourne, Australia and working as Head of Testing at Australian Health Practitioner Regulation Agency. An International speaker and writer, Rajesh is a context-driven tester who believes in practical approaches of software testing that provide value to the business and stakeholders. Rajesh has held senior test management positions with Cathay Pacific Airways, Nokia and Sopra-Steria Group amongst others living and working in US, UK, India and Hong Kong China. He can be approached at LinkedIn and on Twitter @rajeshmathur.

# What type of leader are you?

by Anna Royzman

Contrary to the common understanding of the term 'leader', the nature of leadership is complementary. The leader of any given group is chosen by the group to perform a certain function: to solve the group problems. Such dynamics can explain why one person may well be the leader within one group, yet not a leader in another group. The most interesting phenomena is observed when your role expects you to exhibit "leadership" qualities. For example, it is defined with such term as "Lead", or your responsibilities include the word "Leading..." in the description. When management puts such emphasis on the 'leader' role definition, you are oftentimes judged by the wrong set of metrics by your superiors. For example, there is a standard perception on how 'leaders' behave, instead of differentiating whether the leader fulfilled his/her function in respect to the group goals. Such discrepancy oftentimes results in co-existence of an "official lead" and an unofficial leader in the same group. Where can you take it from here? First of all, if you want to become an effective leader, do understand your context:

- What are the goals of group that you want to lead?
- What are the group's problems?
- How can you solve these problems effectively?
- What skills do you need to solve the group's problems?
- How do you know that you are successful?

Understanding the context will help you in selecting the most appropriate leadership style. For example, the newbies group may need "the tyranny of the expert" – someone with the unquestionable authority and expertise, making all the decisions. Make sure you fit into this role, otherwise your decisions will be questioned and someone else will be taking over. In another example, the group of disgruntled experts may need someone to motivate them. In this case, your task will be entirely different. Mostly, you need to stir this group away from whining into more personally fulfilling activities. I will leave your creativity come up with the answers; hopefully, you already get the idea.

My preferred style of leadership is best described in Jim Collins' book "Good to Great". Jim calls it Level 5 Leadership, and he found, through the research of 1,435 companies, that the organizations which were most successful are headed by leaders exhibiting specific qualities. I can summarize such qualities using Lao-Tzu's quote: "A leader is best when people barely know he exists, when his work is done, his aim fulfilled, they will say: we did it ourselves".

Best of luck in finding your personal style in leadership!



A context-driven scholar, professional tester for over 15 years and a thought leader, Anna is always on a quest for quality. Her passion is in discovering new techniques and creating environments that allow people to be most effective at what they do. Anna is a renowned trainer and an invited speaker at international Agile, Testing and Quality conferences. Anna serves on Community Advisory Board for Software Test Professionals Association.



# Love to Write?



Write For Us

*Your ideas, your voice. Now it's your chance to be heard!*

*Send your articles to [editor@teatimewithtesters.com](mailto:editor@teatimewithtesters.com)*



A photograph of a classroom scene with several students raising their hands. The students are seen from behind, wearing light blue, red, orange, and green shirts. They are facing a dark chalkboard. The image is framed by a thick black border.

# In the school of Testing

*for your better learning & sharing experience*

# Leading Beyond The Scramble

## Part I: The Scramble

by Matt Heusser

Image credit - ptcs-photos

After nearly twenty years of working in software, I have come to see some things repeat - the same type of event happens over and over, at many companies.

One of them is what I call the scramble.

It starts with the employees are frantically working to get something done in a very tight timeline. People ignore rules, work late, stop talking to each other, and there is a constant sense of tension, combined with a belief that if we can just get this done, great things will happen.

Exactly what we are scrambling over depends on the company, but you know it when you see it. Here are a few examples:

*At a Software Product Company*, there is a huge potential customer who will buy, but only if we can add certain key features. Or, perhaps more likely, the customer already bought. And we already promised them the features - by a specific date.

*In an Information Technology (IT) organization supporting a not-software company*, an executive sponsor is unhappy with our work and will cancel a project unless something changes—fast. In other cases, the IT system is a critical part of a new service that company has already started selling ... but it doesn't exist yet.

*At a Consulting Company*, there is an urgent need for a tester (or a group of testers) with very specific skills, on-site at a client - right now. If we can't line up candidate, location, time and price, we won't win the contract.

In all of these cases, someone decided that we need to do something, right now, and took some action. That action created a sense of urgency, which led to frenzied activity.

You'll notice I didn't say "results", as the scramble rarely has results. In a few cases, looking back, there are some small wins - we do win this contract, or that contract.

It is *possible* to benefit from the scramble, perhaps even benefit greatly - the people making the Macintosh and Facebook scrambled to great benefit. Yet in the cold light of the morning after the scramble ends, as we scratch our heads and ask "what really happened here? Was it all worth it?"

Most of the time, we realize ... we sure didn't build Facebook. And the million companies that scrambled and failed, we probably haven't heard about.

It is more likely that we pay a terrible price for it in terms of morale, and turnover, but also in derailing all our existing work. Everything we dropped to work on the scramble is now a week (or two or three) behind. We don't remember where we were, and have to get back into the work.

How much better would it have been if we realized that during the scramble? What if we had not scrambled?

In most cases, we would be in the same condition, without the overtime and pain. In many cases, we could even be in a better condition, as the frenzy of the scramble actually hurts team performance. We get buggy code, and it arrives late.

### **Leaders Act. Followers Re-Act.**

The person that created the scramble, that called everyone into the conference room, explained the problem, and made it clear that "now is not the time for chatting", that we need to "get to work" and "solve this problem" was certainly a leader - as they took an action and that action changed the behavior of the rest of the group. Perhaps they were misguided; perhaps the scramble was a bad decision - but they **acted**.

*What did you do?*

Forget about your title. I don't care if you were the senior director of a center of excellence or a janitor. Did you act, defining how you understood the world to work, or did you react, doing what you were told or what was expected of you?

Furthermore, if you did act, did anyone follow you?

The being followed part is the definition of the leader. The action might be a good start.

There are plenty of positive responses to the scramble. That is, there are good things we can do. You might work a little harder, take a slightly shorter lunch, work a little later in the evening - but keep your cool. Don't throw out good habits, and encourage others to keep theirs. I wouldn't advise telling the big boss to "stay calm", as that appears condescending. You might ask them to tell you about it, then ask how you can help. That help is likely to be the communication and coordination that gets thrown out early in a scramble. All of that is likely better than jumping in to the same frantic frenzy that is consuming everyone else.



But here I am, telling you what to do. And that's sort of the point. The test leader has a vision that is compelling, and suggests actions to others that inspire them — makes them want to comply with the vision — whether that person is the test intern or the Vice President of Software Engineering.

So think back to the last scramble you experienced. You might be living in one now. Most people in that situation re-act. That's okay. But challenge yourself - what would have to change you to work **on** the situation, instead of **in** it?

How we act in the face of a scramble is one dividing line, one litmus test, for if we acted as a leader or not. Be prepared though, as you might not appreciate the face looking back in the mirror.

After dozens of scrambles over nearly twenty years, most of the time, most people follow. Of course you can have an excuse. Most people will. They'll say "That's not my job", or "Our situation is more complex", or "Matt, you don't understand." All of it's true, I'm not doubting it. And you can go back to the articles that define the virtues of a leader, and read about integrity, and courage, and compassion, fairness and creative thinking, and all of those are good things to work on.

But after reading this, in the back of your mind, I hope you hear those nagging question: What did I **do**? What should I be **doing** now?

### Wait - Isn't that Disobedience?

One of my peer reviewers pointed out that not giving in to the scramble might be an act of "civil disobedience."

Hold on a minute here.

Nobody said out loud "everybody PANIC!" The very leader who is pushing the scramble, if asked in clear tones "does this mean we should panic?" would say "of course not." The problem is the leader is suggesting by *action and tone* that everyone should panic.

In that case, failing to panic is not disobedience. It *might* be leadership. It can backfire politically. During one scramble a friend of mine kept his wits about him and encouraged others to do so, and got a reprimand for his attitude. Reprimand or not, I'd say he was a leader, and I suspect you would too if you were there.

If the scramble happens a great deal, and you lead and get punished for it, well, maybe that isn't the department for you. You realize that and take action (act) or hide for it (re-act), that's up to you. This issue is about the acting, I think.

### The Good News

If you've been in the scramble before and followed, that does not mean you "are" a follower. It means you behaved as one at one point. You can change that next time.

And, again, after doing this for more than a little while, I think it's fair to say that there will be a next time.

I'd like to close with a few words from Rudyard Kipling. I hope you'll overlook the dated language and read for the point the author is making, about moving from doing what is expected to moving to the beat of your own drum, and, perhaps, along the way, inspiring a few people to follow you.

If that ain't leadership, then folks, I'm sorry. I don't know what is.

*To be continued in next issue...*

**"If", Rudyard Kipling (First Published in Rewards and Fairies, 1910)**

*If you can keep your head when all about you  
Are losing theirs and blaming it on you;  
If you can trust yourself when all men doubt you,  
But make allowance for their doubting too;  
If you can wait and not be tired by waiting,  
Or being lied about, don't deal in lies,  
Or being hated, don't give way to hating  
And yet don't look too good, nor talk too wise;*

*If you can dream—and not make dreams your master;  
If you can think—and not make thoughts your aim;  
If you can meet with triumph and disaster  
And treat those two imposters just the same;  
If you can bear to hear the truth you've spoken  
Twisted by knaves to make a trap for fools,  
Or watch the things you gave your life to, broken,  
And stoop and build 'em up with worn-out tools;*

*If you can make one heap of all your winnings  
And risk it on one turn of pitch-and-toss,  
And lose, and start again at your beginnings  
And never breath a word about your loss;  
If you can force your heart and nerve and sinew  
To serve your turn long after they are gone,  
And so hold on when there is nothing in you  
Except the Will which says to them: "Hold on";*

*If you can talk with crowds and keep your virtue,  
Or walk with Kings—nor lose the common touch;  
If neither foes nor loving friends can hurt you;  
If all men count with you, but none too much;  
If you can fill the unforgiving minute  
With sixty seconds' worth of distance run,  
Yours is the Earth and everything that's in it,  
And—which is more—you'll be a Man my son!*



**Matt Heusser** is the managing consultant at Excelon Development and editor in chief for Stickyminds.com. In addition to nearly two decades of professional work in software delivery, Matt is also the creator of the TestRetreat series of conferences, the creator of the lean software delivery courses and methods, lead organizer of the workshop on technical debt, the workshop on self-education in software testing (WHOSE), and the workshop on Teaching Test Design ("WhatDa"). A former member of the board of directors of the Association for Software Testing, Matt led the initial effort to create a grant program and headed the grant committee for its first two years. Space does not permit a full list of Matt's accomplishments, but you can learn more about Excelon at [www.xndev.com](http://www.xndev.com), follow Matt on twitter @mheusser, or email him using [Matt@xndev.com](mailto:Matt@xndev.com).

Back To Index



# Building and Maintaining A Successful QA Team



by Ruslan Desyatnikov

Building a new QA team can undoubtedly be a daunting task. If you've just been hired at a new company and tasked with creating or restructuring their QA department, then you have a lot of homework to do. I've seen new managers come in like a whirlwind, moving people and process around with no real direction or goal. It's as if they feel the need to change things just for the sake of change, not taking into consideration the thoughts and feelings of the individuals they're working with. An approach like this can lead to confusion and resentment, creating division and making the team even less effective.

In my 20 year career, I've set up over a dozen QA teams of varied sizes from scratch, in the US and abroad. Most of the companies I worked with had limited budgets and I wasn't always able to hire people with the experience I needed. Success was still achieved, even with teams of people who were not initially experts, QA savvy, or had little to no IT experience. Here are a few things I learned along the way.

## Investigate Products and Process

First things first, you need to learn all about your new company, their products, their development process, and current QA process - if there is one. It's important to fully understand what you're working with before you can determine what needs changed or how to change it.

Do they have automation in place anywhere? How much and what tools are used? Is there a bug tracking system, and if so, is it integrated with anything else? Do any members of the team work offsite or even offshore? What's the level of communication and the general process for communicating? Is QA aligned with business and given the space and authority they need?

These are all pretty standard things to discover in your first couple of days, no big revelations there. But the most important thing to focus on is determining where the gaps are in the existing process. Is there a technical gap? (Missing skilled personnel) Or possibly software or hardware gaps? (Missing defect tracking or test environments) Find out the strengths of the process along with its weaknesses. After



all, leading the charge by proclaiming that everything is wrong might not be the best way to make friends.

The approach shouldn't focus solely on the QA members. Finding out what the developers, as well as staff outside of software development think about the current process can be very insightful. If the process is changed, the others will be affected as well, so their input is important. Determine what people think is working, and what they feel is not working.

Knowing what products the company offers and their planned future offerings will provide a great deal of useful information. What is their customer base? Who is the target audience? How mobile-friendly are they? What are the growth expectations?

Once you've established the way things currently are and what the future expectations are for the company, then you can move to the way things should be in order to meet those future goals. Are they using waterfall when they should be using Agile? Are they using Agile when they should be using XP? Should there be multiple teams, or should the multiple teams join into one?

### **Identify Roles and Assess Skills**

Process is important but the people involved in the process are even more so. Assessing the roles of each current team member, their feelings about their roles, their skills, and future goals is vital to understanding what you have and what you still need as a team. A new leader needs to sit down with all of their current team members and find out the answers to all of those questions. Does everyone know and understand their role on the team? Do they have goals to move into different roles? Do their skills align with those goals, and if not, what can be done about that? It may very well be that some roles should be switched and it's the job of the new team leader or manager to figure that out. Finding the balance between what the team needs and what the team members need can be difficult but doing so will create a strong and successful unit.

Not only will this step help utilize the strengths of each team member, but it will go a long way towards earning their trust and respect if they feel you're invested in them as individuals as well as a team. During some initial discussions with a new team I had taken over years ago, I discovered that one of my testers had both the skills and aspirations to be a developer. While I didn't want to lose one of the stronger testers on the team, I still needed to encourage him to pursue his true goals and give him the opportunity to prove himself. I queried other development managers and when the opportunity arose, I made sure he knew about it. He's now a happy developer within that same company.

### **Fill the Gaps**

So, you know what you have, and have identified what you need. Now it's time to start pouring new talent into those gaps. Finding the right people will be a huge determining factor in your team's success, but it's important to focus on more than just skills on a resume. While skills are important, a good fit is equally so.

Since you should know the goals of the team and the company, as well as the process you're going to take to meet those goals, you can interview with that in mind. Share some of your vision with the

prospective team member and gauge how they would fit into it. A team I used to work with was a very relaxed one, with few rules. We could pretty much make our own hours and we were very self-directed. A woman that was hired onto the team, while greatly skilled, was the type of person that wanted and needed more structure, rules, and direction. There's absolutely nothing wrong with that, but she was never completely comfortable with us. Her skills matched what we needed, but she didn't really fit the environment she was walking into.

For instance, are you going to run an agile shop but the interviewee has a dislike of Agile? Are you planning on putting all team members in an open office environment but the person you're interviewing expresses issues with that kind of setup? Skills and experience are important, but if the new hire has problems with your vision from the beginning, then they may never completely fit in and the whole process of hiring could be a waste of time for all involved. Sometimes you can't afford to hire the skills you need, but that's not the end of the world. I've paired junior team members with senior resources to help guide them, and then provided personal training and coaching to help move things along. I monitored and offered support, constructive criticism, and continually raised the bar higher and higher until they flourished into experts.

## Lead

You have your team and your vision, now lead them through it. A good leader sets expectations for the process and the team, but allows for a level of self-direction at the individual level. You should place some trust in each of your team members after all, you hand-picked some of them. If you've let everyone on the team know what their role is, what value they bring to the team, and what the goals of the entire team are, then it's much easier for the team members to figure out how to work together to achieve those goals. Also important to keep in mind, when an engine runs smoothly and all of the parts work together, it can handle the loss of a key resource with less difficulty, should someone decide to leave. Consistency, communication, and empowerment are three tools that will help any team reach a shared vision for success. Each person on the team should feel important and vital to the team, understanding their impact and willing to take ownership of the work they do. Regular one on one meetings can help a leader ensure that individuals are still aligned with the vision they have for themselves, as well as the vision for the team. It also gives you an opportunity to assure them of your faith in their work, or to provide constructive criticism. It's important to encourage and coach team members to continue moving forward and broaden their skillset. Another consideration to help keep testers fresh is to set up a rotation through different projects so they aren't seeing the same things week after week, year after year. There's no one right way to create a successful QA team, but keeping these suggestions in mind and maintaining the delicate balance between individuals and team will aid the process. I've found my greatest success using good people skills, a little psychology, and genuine concern and understanding for each of the team members.



**Ruslan Desyatnikov** is Founder and CEO at QA Mentor, Inc. He brings nearly 20 years of Quality Assurance, Quality Control, Process Improvement and Software Testing experience. He is responsible for growing QA Mentor as a trusted partner for software testing, independent verification and validation, and strategic consulting for client base around the world.

Contact Ruslan via LinkedIn  
<https://www.linkedin.com/in/ruslandesyatnikov>



# On Testing Leadership



by Justin Rohrman

When I think about the common threads of leaders I respect in the testing community, I realize that I didn't follow any specific path. And also, they all had picked their own direction. Some of them spent part or most of their careers managing big companies developing new generations of testers, others went independent and created their own unique brands of testing by planting seeds of knowledge all over the place. So far, at least. I've never run a conference, or written a book. I haven't contributed new, useful ideas to the field, I've only helped spread the ideas of those that came before me, mostly through channels that were already built.

And yet, here I am, writing about leadership.

For most of my professional career, I've been a lone wolf. Initially, blogs and BBST classes from Association for Software Testing drew me in. From there, I started writing a little bit on my personal blog, here and there when I had the motivation, and then teaching BBST. Things progressed quickly.

So, I can't talk about traditional paths or ideas on leadership but I can describe the few defining things that work for me.

## Leadership VS Management

Early on in my career, I had management and leadership mixed up in very fundamental ways. Just starting out, most of the obvious power came from roles with titles like 'lead' or 'manager'. Management is sometimes a role defined by activities like facilitating work, helping employees grow and develop in their field, and negotiating salary. There is also power associated with the role of a manager, some of that comes from the perceived authority.

Leadership comes from a relationship rather than perceived authority. It means that someone finds something about you valuable enough to follow. It only takes one person for you to be a leader. Leadership is a lot like quality, it means someone values something in you.



## Writing For Clarity

Writing completely changed, or more specifically, developed the way I think. I read *Becoming A Technical Leader* by Jerry Weinberg a few years. It took a while for me to realize how clever the title is. This isn't a book about becoming a leader in a technical context, it is a book about what the leadership role can be like regardless of title. Technically, a person can be a leader in any position in a company.

Jerry made a recommendation in the book that struck me; people wanting to be leaders should write daily, even if it is just for five minutes. This isn't because being able to write clearly is important, and it is. But, because writing is an activity focused on developing and clarifying ideas. Clear writing is one sign of a well-developed idea. Sloppy writing, on the other hand can be a sign of sloppy thinking.

I certainly don't spend 5 minutes every day writing, discipline isn't my strong suit. But I do manage to get at least 1000 words written and sometimes reviewed every week.

Most of the time when I talk with someone and leave impressed with well thought out argument and clear thought process, they spent some time before hand talking and writing about the subject.

## Hard Work

Expertise is hard work, it is something that goes away the minute you think you're there and stop working on self-development.

In Japanese lean philosophy, there is a concept of kaizen, constant improvement. I think there is a lesson there for us.

In the book *This Is Lean: Resolving The Efficiency Paradox*, the author tells a story of having one of the creators of lean tour his factory in Sweden. They move station to station on the shop floor and at each place the author asks "Are we lean yet" and the other person would simply reply "hmmm..." At the end of the day, the author asked the same question and the visitor finally replied "How could I possibly know that, I wasn't here yesterday."

The point is that we should be able to see where we are right now, where we want to be, and figure out how to get there.

Pursuing a goal is a lot like testing. Along the way, we discover something interesting and decide to explore that for a little bit. Wilhelm Wundt wrote about this topic and called it the Heterogeny of ends. The concept explains how while taking action to achieve a goal, can sometimes change the end goal.

## Thinking About More Than Testing

Leadership happens in layers. A person can be a leader to one individual, to a group, to a division, and to an entire company. One part of reaching outside your current sphere of influence is studying more than the craft of testing. Having a deep understanding of testing is of course important, but studying the language of the business can make a person valued outside of their immediate circle. There are two extreme examples of where we sometimes go wrong.

The first is a new example where a new tester is talking to a director about how testing is coming along. The tester says that 874 out of 900 tests have passed so far and that things are looking pretty good. The reality is that the number doesn't mean anything. Neither of them realize that no actual information was communicated.

On the other side of this is a fully is a fully engaged and very excited tester that starts talking about heuristics , oracles, models, and measurement validity when someone a few levels up asks about testing status. I have done this more than once. The result was eyes glazing over, and again, no real information being delivered.

Understanding the business means being able to discover what that person really wants to know. A lot of the time, in a situation like this, the real question, is something like "do you see any reasons why we shouldn't be able to ship on time". Understanding that question means that you can answer with risks you are aware of (often in the form of documented bugs), risk you are concerned about and need a little more time to investigate, and any problems that are slowing testing progress.

Decision makers value clear, concise answers to their questions.

## Do Something

In the beginning, working constantly on our own to develop skill and understanding in our field is important. But, leaders are people that touch someone and impact their life. It could be anyone, even one single person. To do that, we have to actually do something. More than that, we have to do something meaningful. This means somehow changing from being a consumer, to a creator that contributes back to the community some of the things they have taken away. That means something different to everyone.

This is part of my personal story and the things that stand out to me.

Find what works for you.



Justin has been a professional software tester in various capacities since 2005. In his current role, Justin is a consulting software tester and writer working with Excelon Development. Outside of work, Justin is currently serving on the Association for Software Testing Board of Directors as VP of Education helping to facilitate and develop projects like BBST and WHOSE. He is also a student in the Miagi-Do school of software testing, and facilitate sessions for Weekend Testing Americas.

Justin is deeply interested in software testing and delivery, and also in helping organizations fix problems in measurement and metrics programs.

# Notes on Test Automation: Test Encapsulation



## *Enabling Self-Aware Automated Test Cases : Part 1*

by Rahul Verma

### Introduction

Test Automation Frameworks (**TAFs**) do not get the treatment in terms of design that they deserve. Spending time on understanding the precise requirements and then designing/choosing the framework becomes much more important when a general-purpose framework is required. Such a situation is unavoidable when your organization is looking at the possibility of a single underlying TAF that could be employed by multiple teams involved in strikingly different products; their testers are conversant with different programming languages, and they choose or need to employ different tools to support testing.

This article is focused on the technical aspects of designing a scalable, general-purpose, object-oriented TAF. This in turn means that the contents would mainly comprise concepts that directly translate into an implementation in an object-oriented language. One could choose to write the same thing in the traditional programming way, but there would be further work required on part of the reader to translate the design.

The article will not attempt to answer the questions of why to automate, when to automate, which tests to automate, automated tests versus manual tests, ROI of automation, etc. There are quite a few books available on the subject of test automation, and these topics are discussed at length in them. So, if the reader chooses to follow this article, it is assumed that s/he already understands the importance of test automation. This article series focuses on how to automate, whereby the “how part” is not about using an existing tool or framework, rather about designing one.



## The Problem of Dumb Tests

In the design of TAFs, most of the important information that a test requires is kept outside the TAF, as many a times such frameworks are designed using traditional programming rather than employing object-oriented programming. A simple example is grouping tests into platform-based folders, after which the scheduling logic copies in only the tests meant for a test execution machine based on a specific platform. There are multiple problems with this approach. One is redundancy, multiple copies of tests across folders. If you think about it, we could have a common test platform and then specific platform folders, but my question is how many such “commons” would you have? Even if you have the patience, what happens when a test gets modified and is not meant to be run on Platform A? How do you know to which folder you should go to change it? So, there should be an external means to map tests to folders! By not making the information available within the test about which platforms it should (or should not) run on, you have the following situation:

- You need a complex folder structure, which is virtually impossible to maintain as the number of supported platforms grows
- You need an external means to map tests to the above folder structure.
- If the test writer is not the test framework administrator, it means the person who requires this is different from the one who implements it.
- You have a “dumb” test, which does not know on which platform it can run. So, if anything goes wrong, you have an incorrect test case that could crash the system or, at worst, lead to an incorrect result.

So far, we have only talked about the platforms. We have not talked about categorization of tests, priority values associated with tests, known bugs, API version checks and so on. Can you see the complex and impossible folder structures that these options would lead to if kept outside the test?

This article focuses on the concept that a test is meant to be the most complex part of the framework in terms of its power and flexibility. Test encapsulation is at the heart of building a general-purpose framework, which can be used by testing and development teams as a common test automation platform for white-box and black-box tests. To know about what exactly is meant by test encapsulation, its approach and benefits, keep reading!

## Introduction to Test Automation Frameworks

As usual in the software testing world, even the term “test automation framework” is interpreted differently by different individuals and in different contexts. Here are some general points which arise when one discusses terms:

- **Data-driven test automation approach:** This separates test data from the test execution code. This is helpful when the same scenario is to be driven with different sets of data. This is also helpful in enabling changes being made to the test data for an existing scenario without modifying the test execution code.

- **Keyword-driven test automation approach:** This is a command-based model. Commands exist outside the test execution code in the form of plain text files with corresponding parameters. The execution logic reads from the files, interprets the commands and calls corresponding functions in the test automation framework. This is mostly used in combination with the data-driven approach.
- **Generation and mutation based test automation approach:** Instead of relying on hard-coded data provided, these frameworks rely on data generated at run-time. Another difference from traditional data-driven frameworks is that these contain commands for mutations directly within the input configuration files, thereby incorporating a flavour of key-word driven frameworks as well. These are more complex than both the above categories and their implementation is usually found in the world of “fuzzing” frameworks.
- **Multiple test nodes on a single test runner machine:** (Test node is the thread or process running tests on a test machine). This is mostly seen in performance testing frameworks, which use a single test runner machine and multiple test nodes connected via multi-threading (mostly) or multi-processing. This is made possible only for non-GUI testing and testing in which no system-wide changes are made by a test. In recent developments of headless browsers as well as automation frameworks like WebDriver, it is possible for functional test automation for the web applications as well.
- **Central execution control:** The framework should be able to trigger, control and report the results of multiple tests from a single point of control. For example, CPPUNIT Lite, a unit testing framework for C++, can execute all tests via make command.
- **Distributed test execution:** Testing frameworks can also be thought of as providing the functionality of central execution control from one or more “controller” machines, which can trigger, control and report the results from multiple test node machines.
- **Enhanced reporting via GUI:** As an extension, frameworks can support GUIs (stand-alone executables or web-based) to support customized reporting. In performance testing frameworks, for example, features for plotting and analysis of performance statistics is essential. Also, different stakeholders would have different needs in terms of the test report they want to see with respect to details and contents.
- **Hardware/operating system infrastructure:** In some contexts, where hardware and operating system requirements are precise (or even otherwise), some refer to the complete set-up (i.e. test automation code + hardware and operating system set-up) as the test automation framework. This can be seen, for example, where the functioning of the TAF depends on external tools that are pre-installed on the test runner machines, which cannot be done as part of the test set-up.

#### Note:

*Some texts enumerate the first two bulleted points above as the “Types of test automation frameworks”. I beg to differ. I do not consider these as types of test automation frameworks, because that would suggest they cannot co-exist in a single framework. The first three are approaches towards a TAF design in terms of enabling the TAF to execute tests via data and commands present in clear text in various forms, such as plain text files, XML files, databases etc. All three can be built on top of a general-purpose framework. In simple words, a test framework can provide any and all of the features mentioned above, by choice.*

Following is the list of other general features, on which decisions have to be made when choosing/building a test automation framework:

- Which language should be chosen to build the base frameworks? For which languages would extensions be supported?
- Should the test automation framework be generic or specific to a product under test, testing tool or a language, in which tests can be written?
- How much complexity do you want to hide from the end user (the one writing/executing/reporting a test)? Many times, hiding the complexity might result in providing less flexibility as well. Some performance testing tools, for example, provide for the recording of the test scenario as a GUI-based tree representation with a limited amount of controls for customization – ease of use with less flexibility. Others might choose to provide both.
- Should it be cross-platform? What browsers it will support? Which versions of the product can it test?
- How do you manage framework files (e.g. core library, configuration files) and the user contributed files (scripts)? Are you going to opt for version management? What would be the protocol? Should it exist along with the development repository for the product under test, or be separate?
- What kind of regression strategies would it provide? Does it provide for bug regression, priority based regression, author based regression, creation date based regression, API version based regression, etc.?
- Should it support physical and/or virtual test runner machines?

The above is an incomplete and a very high-level list of features that one might need to consider when building a test framework. So, it shouldn't be a surprise if one sees testers dissatisfied with the testing tool/framework they currently employ, because there would be one or the other feature missing.

*To be continued in next issue...*



Rahul Verma is a consulting software tester, author, speaker, coach and a serial entrepreneur from Bangalore, India. He is the founder of [Test Mile](#) and [Talent Reboot](#).

He is known for his practical and unified view of the software testing subject. He has been awarded multiple Testing Thought Leadership awards for his contributions to software testing community.

You can visit his website [www.rahulverma.xyz](http://www.rahulverma.xyz) to know more about his work and get in touch.







# From Special Desk

## Why Test Automation?

In today's fast moving world, it is a challenge for any company to continuously maintain and improve the quality and efficiency of software systems development. In many software projects, testing is neglected because of time or cost constraints. This leads to a lack of product quality, followed by customer dissatisfaction and ultimately to increased overall quality costs.

The main reasons for these added costs are primarily:

- poor test strategy
- underestimated effort of test case generation
- delay in testing
- subsequent test maintenance

Test automation can improve the development process of a software product in many cases. The automation of tests is initially associated with increased effort, but the related benefits will quickly pay off.

Automated tests can run fast and frequently, which is cost-effective for software products with a long maintenance life. When testing in an agile environment, the ability to quickly react to ever-changing software systems and requirements is necessary. New test cases are generated continuously and can be added to existing automation in parallel to the development of the software itself.

In both manual and automated testing environments test cases need to be modified for extended periods of time as the software project progresses. It is important to be aware that complete coverage of all tests using test automation is unrealistic. When deciding what tests to automate first, their value vs. the effort to create them needs to be considered. Test cases with high value and low effort should be automated first. Subsequently test cases with frequent use, changes, and past errors; as well as test cases with low to moderate effort in setting up the test environment and developing the automation project are best suited for automation.

### **Optimization of Speed, Efficiency, Quality and the Decrease of Costs**

The main goal in software development processes is a timely release. Automated tests run fast and frequently, due to reused modules within different tests. Automated regression tests which ensure the continuous system stability and functionality after changes to the software were made lead to shorter development cycles combined with better quality software and thus the benefits of automated testing quickly outgain the initial costs.

### **Advance a Tester's Motivation and Efficiency**

Manual testing can be mundane, error-prone and therefore become exasperating. Test automation alleviates testers' frustrations and allows the test execution without user interaction while guaranteeing repeatability and accuracy. Instead testers can now concentrate on more difficult test scenarios.

### **Increase of Test Coverage**

Sufficient test coverage of software projects is often achieved only with great effort. Frequent repetition of the same or similar test cases is laborious and time consuming to perform manually. Some examples are:

- Regression test after debugging or further development of software
- Testing of software on different platforms or with different configurations
- Data-driven testing (creation of tests using the same actions but with many different inputs)

Test automation allows performing different types of testing efficiently and effectively.

**[Learn how Ranorex can help with your Test Automation...](#)**

### **ABOUT THE AUTHOR: LARISSA STOISER**

Larissa Stoiser is Team Lead of Ranorex Quality Assurance. She has profound experiences in combinatorial testing and applies its approaches.



# Delivering High Quality Applications in a Mobile World



by Shravanthi Alimilli



Over the past couple of years, there has been a tremendous increase in worldwide sales of smartphones. According to [Gartner](#), Smartphones grew 20 percent in the third quarter of 2014, and the overall smart phone sales surpassed a billion units in 2014 as per [this](#) report.



Mobile devices, including smartphones, are transforming how our society engages with the world, how business gets done and how we communicate throughout our everyday lives. With an upward trend in the mobile phones market, consumers are spending more time on their apps than ever before, and the App economies have continued to grow as well. Over the past half-decade, the proliferation of mobile devices has transformed us into an app-driven society.

According to Gartner, by 2017, mobile apps will be downloaded more than 268 billion times, generating revenue of more than \$77 billion and making apps one of the most popular computing tools for users worldwide.

## User Experience and Customer Satisfaction

The pace at which consumers are adopting web on mobile is commendable, and the majority of transactions for many e-commerce companies happen on mobile devices now. Businesses are using mobile apps as a key part of their user engagement strategy, and the prediction is this will become even more significant moving forward.



However, how these applications actually work on the mobile devices can have a major impact on customer satisfaction, and is an indicator of business success.

*Neglecting to test these applications thoroughly before launching can have negative effects for the business and will also affect customer loyalty in the long term.* With the growing demand for mobile applications, consumer expectations are also high. An overwhelming majority of mobile apps are deleted within seconds because of poor user experience. Additionally, research indicates only 16 percent of users would give it more than two attempts if the app failed to work for the first time. In fact, the only guarantee in mobile app development is you'll get a guaranteed 1-star rating and passionate bad reviews, if you ship an application with bugs.

At the same time, testing mobile applications is not an easy process. There are many common challenges that must be considered before testing a mobile application.

## **Mobile Testing Challenges**

### **1. Hardware and software configuration:**

Hardware: Different screen sizes, hard keypad, virtual keypad (touch screen), resolution, memory and so on. Also, you must account for the different mobile vendors such as Samsung, Apple or Nokia, as well as all variations these configurations introduce into your testing scenarios.

Software configurations: Different operating systems, such as Android, Windows, Blackberry, iOS and so on, along with various versions of each (O/S iOS 5.X, iOS 6.X, BB5.x, BB6.x). Don't account for them all and you'll hear about it!

### **2. Device fragmentation:**

The device matrix is growing constantly, especially for Android and iOS. As mobile applications can be deployed across multiple devices with various OSs such as iOS, Android and the like, and across various versions of an OS, compatibility and consistent user experience across devices becomes an issue when it comes to mobile testing. Validating a consistent UX across devices is probably the biggest problem experienced in mobile testing today.

### **3. Native, Hybrid and Mobile Web Apps**

Building and reusing tests across projects becomes much more difficult when you factor in the various development styles. Tests built for hybrid apps, for instance, won't meet the mark when presented with a native scenario. This puts additional pressure on testers and hampers productivity.

### **4. Network challenges**

Multiple type of Networks (GSM / GPRS / Wi-Fi / Wi-Max etc.), unpredictable data transfer times and inconsistent speed of connectivity across different regions are just a few issues when it comes to mobile network connections. Designing and validating mobile applications to accommodate these network variations poses unique challenges.

## 5. Gestures

Scroll, zoom, swipe and pinch are just a few gestures that govern touch screen and mobile devices. Testing against these gestures on multiple mobile devices to provide a consistent user experience can be very challenging.

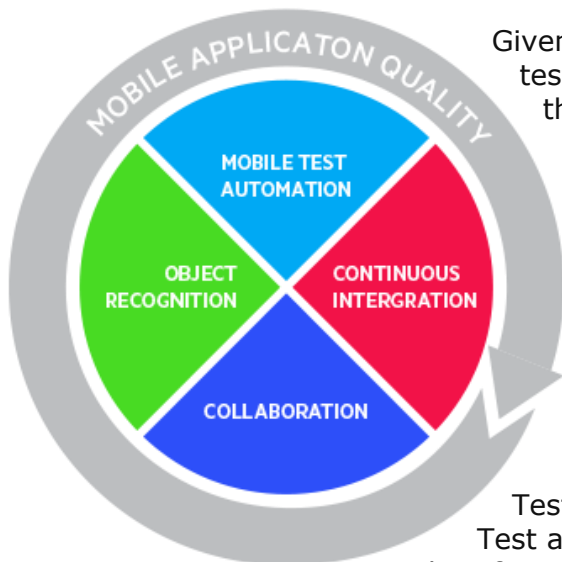
Delivering high-quality applications in such complex mobile environments requires a new approach. At Telerik, we constantly try to improve your testing experience by stepping into the shoes of development and quality teams to solve these real-time issues, and develop features and functionality to help you deliver high-quality apps on time. To that end, we are introducing Mobile Test Automation with Test Studio solution.

## Selecting a Mobile Testing Tool

When it comes to mobile application testing, manual and exploratory testing should be included in the test methodologies; however, test automation can save both time and costs, and help you avoid the impact of defects, both internally and on a business.

Here are some key points to consider before selecting a tool for Mobile Testing:

### 1. Object recognition:



Given the complexity of mobile applications, it is important to select a testing tool that can run a single test on multiple devices--a tool that supports one central object repository. Telerik Test Studio solution enables your test elements to be stored in a shared Element Repository on the project level, and when an element is used across multiple tests, it can be referenced from this central location, without duplicating tests. And, with the built-in Intelligent Element Find Logic, you can edit an element's find expression at only one location to update it in all tests and steps in the project.

### 2. Mobile Test automation:

Test Automation is great for mobile application development. When Test automation becomes an integral part of your agile process, it saves a lot of time and cycles for developers by enabling you to fix an issue before the next iteration. Test automation also offers significant value by enabling you to test in parallel with development, leading to greater team collaboration. Mobile applications can be tested instantly and effectively with Test Automation. A record and playback approach makes automated testing much easier. Record your interactions with the device (or emulator), then run the recording on device is a well-accepted approach to automate your mobile testing efforts. This approach, when incorporated into your mobile application development strategy, can increase effectiveness, efficiency and coverage of your software testing.

### 3. Continuous Integration

Continuous Development and Continuous Integration has great benefits for software development cycles, allowing defects to be identified and tracked early on. It also enables constant feedback on software status, typically resulting in problems being resolved early on. This is especially beneficial in the mobile world, as the mobile market is constantly changing, and keeping pace requires early deployment and release cycles. A continuous integration approach during development and testing phases can accelerate the release cycle. Test Studio suite's integration with Source Control systems facilitates collaboration between QA and developers by enabling them to check-in at the same time and independently.

### 4. Collaboration

Integrating test automation into the overall development lifecycle requires collaboration between teams, leading testers to tools that optimize Tester-Developer coordination. Test Studio solution has always helped to enable this collaboration, so testers can design and maintain tests and pass them to developers through source control to assist with more complex, edge-case scenarios. We are bringing the same great stuff into mobile testing.

### Conclusion

Testing is crucial to ensure success in the highly competitive and complex world of mobile apps. It is important to include testing as a part of app development, especially for agile teams. The earlier you test, the earlier you find faults—and the earlier you fix those problems and win the trust of users. Poorly tested applications can directly affect the business and lead to undesired results. Automate your mobile testing and mobile app, and go to the marketplace with as few defects as possible.

### What next?

In the next release of Telerik Test Studio solution (third week of July), we are adding mobile testing functionality to support mobile test automation. Stay tuned for further updates on the release and a complete solution for all your testing needs: Test Studio for web, desktop and mobile testing.

There will be a mobile testing eBook published around the July/August timeframe. [Register](#) to receive the eBook and product release announcements. There will also be a release webinar in September during which we'll talk in detail about all the new features of Test Studio suite.



# Delivering High Quality Mobile Apps with Test Studio

R2 2015 Release Webinar: July 30, 11 a.m. ET

[REGISTER NOW](#)





# TEA-TIME WITH SMARTBEAR



## About this column...



**SmartBear Software** not only provides testing tools to help development and testing teams accomplish their software quality goals, it is also a hub of information and news for the software testing industry. From workflow methodologies to discussions on industry practices and tech conference coverage, SmartBear has become a source for testers seeking quick access to a wide variety of content.

SmartBear's goal in creating this column in **Tea-Time with Testers** is to empower software testers around the globe by helping them become more informed about the current state of the software testing industry.

# First Impressions Count with Mobile App Quality

- by **Nikhil Kaul**

“Let’s get that mobile app out of the door quickly!” Heard that from your marketing team before? Speed of application delivery and time to market often takes precedence while rolling out mobile applications. However, if the app is buggy or broken, users will often end up deleting it. To make it worse, [TechCrunch](#) reported back in 2013 that only 16% of users will give the app another try after it has failed once.

To overcome fickle consumer behavior driven by less tolerance to buggy code, the first crucial step is realizing mobile necessitates a different testing strategy than what is traditionally used to test desktop and web applications. Applying the same testing strategy is doomed to fail, primarily because of the following:

- **Speed is important, but so is quality:** Mobile users have a lot of power to make or break the future of apps through reviews on app stores. This is significantly different than what happened with desktop or web applications. Remember those blue screens you used to encounter while navigating desktop applications. After a while, you got so used to them that you actually started caring less about them. Mobile users however aren’t that forgiving, when they encounter similar issues.
- **Highly Fragmented Marketplace:** Android is the big elephant in the room here, after having introduced a variety of operating systems, such as Jelly Bean, KitKat, and Ice Cream Sandwich over the past few years. Add to that, fragmentation resulting from screen size, device type, and hardware specifications, testing can easily make things messy. And with the Internet of Things, gadgets such as Fitbit becoming more prominent—added fragmentation means testing becomes challenging and time consuming. Rampant device fragmentation also means that [testing mobile applications](#) needs to be device specific, resulting in testing cycles longer than that of desktop or web.
- **Move to “system of engagement” from “system of record”:** Mobile has reinvented how users interact with applications and what they expect. Traditional desktop and web applications primarily could be classified as “system of record” information systems. These applications would typically record and report information saved on ERP systems and large databases. Mobile apps, on the other end, power “system of engagement”. These applications deliver individual personalized content that help empower in-the moment consumer discussions. Given mobile interactions are focused on people rather than on process, these interactions need to be time, context, and social aware, instead of just recording and reporting from a “system of record”. Harnessing the power of application programming interfaces (APIs) to drive this personalized context thereby becomes crucial when using mobile.
- **Role of network performance:** Mobile networks play a large role in determining the end user experience of mobile applications. It is entirely plausible that a mobile application may behave differently on a Sprint network than it does on an AT&T network.

## Where does testing stand as of now?

[Mobile testing](#) is still in its nascent stages and as a result much of it is being carried out manually. This can be really cumbersome, especially when it’s necessary to test in a shorter release timeframe.

For organizations that have automated testing in place, they focus most of their energies on automating front-end user-interface test cases, while paying limited attention to back-end services/APIs that actually power user experience at the front end. Even less attention is paid to ensure these APIs are monitored continuously after being implemented.

## How to release mobile apps within shorter release cycles?

- **Reduce testing time by improving code quality:** This is the most straightforward, yet tough to implement. By shifting testing to the left, organizations can accelerate attention to software quality from the inception of the software development lifecycle. Tools that help testers and developers collaborate on source code as well as requirement documents can be of great help here.
- **Prioritize test cases:** Releasing mobile apps in a shorter timeframe necessitates a strategic approach to testing. It's thereby absolutely essential to reduce rework, trim down unnecessary tests, and prioritize key functions without affecting coverage. Test management solutions can help prioritize testing efforts, account for risk, and plan for coverage.
- **Record gestures once and then use them across multiple mobile devices:** Testing solutions that help record repeatable accurate gestures and then replay these gestures in a consistent manner across multiple mobile devices can be helpful in overcoming device specific mobile testing, which results from device fragmentation.
- **Test backend services still being developed:** [API mocking tools](#) can power parallel API testing and development by enabling creation of a virtual service, while actual API is still being developed.
- **Automate not just the functional, but API testing as well:** Integrate API and functional testing tools with automated build systems such as Jenkins and other development tools to run functional as well as API tests as a part of an automated build.
- **Test for Network Performance:** Visualize and analyze how apps behave when assessed using worldwide carriers.
- **Reduce debugging time post deployment:** Use automated solutions that help you perform root cause identification at the code level in case an issue is encountered after production roll-out.

# Tea, Testing and



## **Exploratory vs. Scripted Testing: How to Strike the Right Balance**

Do you have a “camp,” when it comes to the tug of war between exploratory and scripted testing? Could you burn up social media debating the advantages of one over the other? Does it really have to be that way?

Scripted testers are typically seen as the older stalwarts of documented methodology, while the exploratory crowd represents the latest in entrepreneurial rebellion. But testing was actually very informal beginning in the 1960s, and it was not until the 1970s that more formalized standards and test scripts came into vogue. But process formalization does not always equal performance improvement, especially when there is high complexity in the system being tested. The flexible approach of exploratory testing is once again popular, especially when meeting the demands of an agile methodology.

In exploratory testing, test planning, test design and test execution occur in unison. A scripted test tries to take this process out of a test designers head and formalize it into a written document that any tester can follow. But by not having to script out the test in advance, exploratory testing puts more discretion at the hands of the tester and allows for a greater possibility of outcomes. Through not limiting the tester’s thought process to pre-determined documentation, we can empower the tester to take a more fluid approach to finding new problems quickly. The beauty of exploratory testing is that it is only limited by the testers’ imagination and insight into the application. As long as the tester has the skills to listen, read, think and report accurately and effectively, exploratory testing can be very productive and produce results not anticipated by a script.

However, scripted tests are still important in certain situations. For example, in a situation with high turnover or inexperienced testers, efficiency and repeatability are valued, and these are things a scripted test does well. If the tester does not know the product well and time is limited, a script can be the only way out.



## Exploratory and Scripted Crossover

The consideration between these forms of testing should never be an either/or proposition. There is a law of diminishing marginal returns in adding detail to a test script. Often, adding the detail to a manual test script to remove the last 5% of uncertainty in executing the script will double the time it takes to author it. So, there is a balance to be struck in most scenarios of how much to script, and how much should be left to be “explored,” even in a scripted scenario. An exploratory test will almost always have some boundaries in regards to a mission or charter to be performed. In many ways, this is a simplistic test script, and often times, it will be the basis of a more detailed test script to be delivered down the road.

Instead of a clear dividing line, think of a control knob that changes mixtures; on one extreme is scripted and on the other exploratory testing. The overwhelming majority of tests will have that knob turning back and forth, but always including elements of both testing formats.

### How to maximize both exploratory and scripted testing

Scripted testing works best early in the middle of the cycle, where theorized systems can be run through and architecture problems identified before design becomes code. The advantage here is an ability to clearly lay out the steps to thoroughly and completely test integrations between systems, application layers, etc. Often, there are a number of critical prerequisite items that need to be checked in a defined order during this stage to make sure that a dud of a build is not promoted.

Exploratory testing, with its non-structured format, is well adapted both to testing earlier and later in the cycle. In the beginning of the cycle, it can function to do a quick sanity check while the code is still fresh and able to be updated quickly. Later in the cycle, it can make sure that critical business scenarios are tested thoroughly and can pinpoint areas for further test coverage in future iterations.

The best advice is to realize that exploratory and scripted testing should not be in a contest. The two approaches not only work together, they work *well* together. It's why firms such as Microsoft will still commonly use both on the same project. While there is a clear contrast between the two, and while each can have their own advantages, it does not mean that each cannot help support the success of the other. It's finding the right balance that matters most in defining the success of your own testing initiative.



**Kevin Dunne** is the Director of Product Strategy at QASymphony where he collaborates with customers and industry experts to continue to provide innovative testing tools for agile teams.

You can find out more at [www.QASymphony.com](http://www.QASymphony.com)



Sharing is caring! Don't be selfish 😊

[Share](#) this issue with your friends and colleagues!



## Please tell us more about your teaching experience. Are there any special memories from university?

At university, I studied mainly mathematics and philosophy, especially Indian philosophy. I'm Jewish, as is most of my family, but my grandmother was a Buddhist. I was to spend about a decade coming to grips with my own belief system. A few parts of that seem relevant to my attitudes toward testing and the testing community:

(a) In the West, we take the "law of the excluded middle" as self-evident and as fundamental to logic. Under this rule, a proposition can have two states: something "is X" or "is not X". There is no middle ground. In contrast, several Indian logicians assumed four states, "is X", "is not X", "is both X and not X" and "is neither X nor not X". I puzzled over this for many years. I am not confident that my understanding of how this could be meaningful is the same as their understanding. But in human affairs, in interpreting people, how they interact, what situations they are in, and how to negotiate with them, I came to view the exclusion of the middle as a heuristic rather than a law or a rule. It is often useful to assume that "X or not X" are the only two possibilities, but when that is not productive, it is time to look for a third way.

(b) The texts that I read constantly challenged my perception of the world. Many appearances are just appearances. You see a snake, but when you touch it, it's a rope. I would eventually adopt a pragmatic realism (I interpret the physical world around me as real because that interpretation is very useful, and it is at least as likely to be correct as less useful-for-me alternatives). However, I also believe that much of what we perceive about the world is our own construction, our own model of reality. I see a specification, for example, as an incomplete and inaccurate description of someone's fantasy of a system. Such a description can be very useful, but anyone who takes a specification (or any model) too seriously is asking to be bitten by a rope.

(c) Some people present differences of religion as an excuse for intolerance, including the treatment of others with rudeness, dishonesty or violence. I view that attitude as a weakness. Unfortunately, some people are adept at gaining fame, power or money from pushing that attitude. That, I view as a disease. Some people in the testing community sometimes characterize differences in attitudes toward testing as comparable to hostile differences between religions. This can be an effective component of a marketing plan that portrays "us" as victims of the evil "them". I see such characterizations as destructive. In graduate school, I studied psychology, especially how people perceive (for example, how we see, hear, feel, and experience time), how we interpret, organize and communicate information, and how we can design systems that make their users happier and more effective.

Almost all of my research was computer-controlled. To do this, I had to learn to program. Most of my programming was in Assembler. I reverse engineered the Apple II operating system and rewrote parts of it and of its embedded version of BASIC to control equipment, collect data in real time (with precise timing) and do complex statistical analyses of large sets of data. I often had to defend my work, so I had to learn how to test and use test results to persuasively argue that my code was trustworthy. In seven years of graduate study, I spent more than half writing code, writing automated tests, writing articles for programming magazines, and trying to understand what made programs "good".

## So, what was the start point for you to focus on software testing and then how was the professional journey traversed for you?

I finished my studies in 1983, submitted my dissertation one morning and flew to San Francisco that afternoon. I had no idea who would hire me so I applied for every job that looked interesting. People offered jobs as a programmer, project manager, marketing manager, and software tester. The testing offer was at a company (named MicroPro then) that made the world's best-selling word processor (WordStar). They explained to me that WordStar had a difficult start on its new platform (the IBM PC) because they had not adapted its user interface to feel like a PC application. The program worked reliably as designed, but the customers didn't like it. Therefore, they said, sales suffered badly.





The company was expanding the scope of the testing organization to prevent this problem recurring. The job was as a supervisor in this expanding test group, heading their testing technology team. I would use my knowledge of human factors to evaluate software usability. And I would use my programming and testing skills to evaluate new test tools and processes, introducing the worthwhile ones into the department.

This was the most difficult-sounding job anyone told me about in any interview that year. It also sounded like the one I would learn the most from, and the one where I could do the most good right away. I jumped at the opportunity.

I had no experience managing testers at this point. I came to Silicon Valley a believer in formal processes, traditional documentation, and the value of designing tasks in a way that maximized delegation of work. Scripted testing (manual or automated) seemed self-evidently good. Those ideas mapped well to the software engineering textbook approaches, to recent books on software factories and talks on testing factories, and to the new IEEE 829 standard on test documentation. So I tried this out.

For some projects, such as testing an adaptation of a well-established program to a new platform, we benefitted from the time spent planning and documenting our testing approach and specific tests.

For many other projects, exploratory tests found most of the bugs.

As my ideas about testing project management were demolished, I spent more time seeking out other test managers to learn from their experience. For example, I joined the Bay Area Quality Assurance Club, went to their meetings and hung out with some of the people I met there. They had similar problems, though everyone reacted to them differently.

What I learned generally was that effective work groups paid much more attention to the quality of their products and much less to the niceties of their processes. The work groups that seemed most effective to me created very little formal documentation. The companies and work groups that seemed to produce the shoddiest software also seemed to rely on the most heavily on formal documentation. Similarly, heavily documented testing (scripted tests) provided weaker support for delegation of work than I had expected, at a much higher cost.

I evolved. Instead of advocating The Right Way, I trained my staff that it was their responsibility to do effective testing within whatever software process or corporate culture they found themselves.

The other surprise for me in California was the level of stress. I had expected a fast-paced environment but, coming from a somewhat more polite culture (Canada), I was unprepared for the harsh corporate politics that were common in the hard-driving culture of 1980's Silicon Valley. My older brother (also in San Francisco, soon to have his own Ph.D. in psychology) had formed a company that interpreted organizational dynamics as family dynamics. I joined them (part time) to learn how to interpret and navigate the interpersonal dynamics in this new-to-me corporate culture.

In 1984, as my way of making sense of the technical, professional and social dynamics of software testing, I started writing Testing Computer Software.

**What difference do you find in formal testing education given at universities as compared to commercial training?**





## FORMAL EDUCATION (UNIVERSITY) COMPARED TO COMMERCIAL TRAINING:

The university setting provides a different pace for studying testing. We can assign homework to students. We can require students to think about what we've taught, to try our ideas on realistically complex software, to try hard things and to question us when their efforts fail. It is very difficult to find time for this in traditional commercial classes, and even when there is time; it is common for students in commercial classes to expect the course to be easy. University students usually expect to learn new things when they take a course. There is no shame in the fact that they don't have that knowledge yet. Some commercial students, especially some managers, become angry when a course challenges their ideas or reveals their lack of knowledge or skill to other students. Course designers have to work around these differences.

Finding a way to bring the strengths of university education to commercial training has been my objective with BBST.

### What is your opinion about creating a degree in testing?

#### CREATING A DEGREE IN TESTING

There is a different type of interest that some people have in taking formal education in testing: they are hoping to get a degree in testing. I studied that option several years ago, as the chair of the curriculum committee in my department at Florida Tech. The school encouraged me to create a degree program in software testing. I designed a degree program that met all the requirements for a university degree in computer science AND all the requirements for a university degree in software engineering. Every graduate from this program would have taken the same required courses in math, computer science theory, software design and programming as the CS graduates who got jobs as programmers.

As I spoke with hiring managers and human resource managers, what I learned was that if we gave students such a degree:

They would almost certainly find it easy to get a job in testing, but

They would probably find it difficult to transfer out of testing into any other job in their company.

With a stamp of "TESTER" on their forehead, these students would have to get another degree (probably a Master's in CS) if they wanted to try their hand as programmers or designers.

A university degree should never lock someone into one type of job. It should help the graduate open many types of doors, not just one. And it should never be a barrier in front of one of those doors.

Because there seems to be a commonplace industrial prejudice that people who really want to do testing shouldn't be seen as suitable for other positions, we decided instead to create a "software engineering" degree that included lots of training in testing. Same courses, but the stamp on their foreheads is different. For now, this seems to be the better way.

*We conclude the part 1 here with Dr. Cem on his career journey with some interesting stories and lessons he learned; along with his views on testing education. Part 2 of this series will feature what Dr. Cem advises on becoming good and efficient tester and shaping up one's testing career...*

# Happiness is....

Taking a break and reading about **testing!!!**



Like our FACEBOOK page for more of such happiness

<https://www.facebook.com/TtimewidTesters>



Have technical skills, ideas, utilities or software tools to showcase to the world?

Write for next issue of **Software Tools Magazine**.

Your performance, our platform.

Let the show begin!



**YEAR I ~ ISSUE II**

---

A detailed photograph of a wooden workbench with various tools hanging on it, including a hammer, a wrench, a screwdriver, and a saw. The tools are arranged in a way that they appear to be part of a collection or a set.

# SOFTWARE TOOLS MAGAZINE

# T ' Talks



*T. Ashok exclusively on software testing*

## **Unleash Your Potential, Unlock Others' Potential**

Leadership to me means 'expanding influence'. The ability to have a far greater positive impact. To be able to motivate/inspire others, to enable others to harness the full potential, to deliver the highest value. It is title-less and not dependent on age or designation. It is anybody who evokes a strong sense of purpose that motivates us to deliver the best in us.

So how can I expand my influence? As testers, we are focused on doing great work. To come up with a good test plan, design good test cases, automate those that matter and hopefully discover interesting issues. Yes, you have done good work and need to be proud of what you have accomplished. Could I have done more? Possibly yes. Let us look at expanding spheres of influence that I could have on my team, my organization, my customer.

Taking ownership of issues that may arise in the team be it related to our style of working, process or filling in for somebody in a crunch situation, assisting team mates to enable them to solve technical issues that may arise are all examples where we go beyond ourselves in the larger interest of team delivery.

Understanding the larger context of the business in which this product/application is deployed beyond



testing delivers higher value to a customer. Let me illustrate this via three real life situations:

#### Situation #1: Contributing to business

A few years ago, we had a startup customer who was building an innovative product. Our team responsible on validation went beyond testing. They were participating in demos to prospective customers, doing pilots to win deals and later deployed and supported the early installs. This was greatly appreciated by the customer who saw the team deliver a greater value to his business.

#### Situation#2: Improving the system

Working with small product development company the testing team noticed a high incidence of defects that crept from earlier stages. These could have been easily caught by good developer testing, better requirements documentation/review and tighter build & management. When these were highlighted to the customer, we discovered that they were well equipped to implement the solution. The test team took it upon to assist in implementation and put together systems and tools to implement these.

#### Situation#3: Educating and changing the mindset

There was an interesting situation when a large multinational company called us to perform load testing and were keen to understand if we had license and skills in using Tool-X, as they were told by our competitor that Tool-X would be best suited for assessment of load/performance. I decided to probe further to understand the problem as what the product is, what the interfaces are and the key flows to be considered. After an hour I discovered that they did not need any tools, but a bunch of shell scripts which would do the job. We educated the customer, he was very happy as we had saved him a sizeable sum on license fees and offered him an easy and elegant solution.

I have been fortunate to have supervisors, peers and my team mates teach me leadership. They have taught, demonstrated and groomed me for which I am eternally grateful. Let me share two stories where my manager (actually a leader) blossomed a young engineer.

#### Story #1:

My company was awarded a project to migrate code from one platform to another, with the language on both these platforms different. The source code base was large and hence rewrite was costly. When I proposed automatic conversion and then intelligent manual customisation to my manager, a technologist at heart, he listened carefully, analysed my solution and despite not well versed with the tools of my solution, gave me the go ahead. Now it became more than just a project, it was about not letting him down after the faith he reposed in me. I faced challenges, he helped me discover solutions and eventually completed it successfully. What if I did not succeed, as this was my first major project? Well I guess he saw ability, intent and decided to lead than manage and therefore put his trust making me work the

hardest.

### Story #2:

As a senior engineer I would whine about lack of great quality of code to my boss a seasoned veteran. He was a smart man and decided to convert my whine into positive energy. He asked me if I would solve the problem by setting up a new focused test engineering team. I jumped at this opportunity, but put a condition. And this was 'Being the Director of Engineering with responsibilities of development and QA, you should not coerce me to release software that we as a team felt as unfit.' He agreed and supported me with a good budget to build a fine team.

These bosses were beyond just managers, they had a bigger vision enabling the individual to expand the sphere of influence. As I matured it was upon me to do what my wonderful bosses did to me. Spot potential and allow them to flower, to build new talent and leaders. Let me illustrate this with a third real life story.

### Story#3:

Years ago there was a young tester (three years of work exp.) in our company. We were a team of three working with a customer on a consulting assignment to validate their product and also improve their test practice. She was responsible to design test cases for this product, get this reviewed by the customer and then execute them. We did the work at our office and had a scheduled weekly meeting at customer's premises, which I was responsible for. After a few weeks, I noticed that she was doing a great job, but was not confident in presenting her work to the customer during the weekly meetings. I was keen on her building confidence and decided to drop out of one of our weekly meetings with the customer at the last time feigning ill-health. Well she had no time to back out, and with great trepidation went for the meeting. I still remember vividly the worrisome furrow lines on her face as she stepped out our office, the surprised glazed look of "how can you ditch me at the last minute?" When she came later in the evening to the office, I was delighted to see the confident radiance in her face. The fear was gone, she realised that she could handle the discussion alone. The last few hours had changed her. Well, she was surprised to see me at the office, realising quickly that I had set her up.

So what is your story? Think about situations where you have gone beyond, and pushed the envelope. Unleash your potential, unlock others potential and change the world around you.

Lead.



**T Ashok** is the Founder &CEO of STAG Software Private Limited.

Passionate about excellence, his mission is to invent technologies to deliver "clean software".

He can be reached at [ash@stagsoftware.com](mailto:ash@stagsoftware.com)





# testomaton

Quality Assurance via Automation

Software testing startup specializing in test automation services, consulting & training for QA teams on how to build and evolve automation testing suites.

## SERVICES

### Test System Analysis and work estimation

Review of client's system (either in development or on early stage) to produce a Test Plan document with needed test cases and scenarios for automation testing.

### Tests creation and Automation

Development of test cases (in a test plan) and Test Scripts to execute the test cases.

### Regression and Test evolution

Development of Regression test suites and New Features suites, including test plans and test scripts or maintenance of existing.

### Architecture Consulting

Review of software architecture, components and integration with other systems (if applicable).

### Trainings

Depending on the particular need, we can provide training services for: Quality Assurance theory and best practices, Java, Spring, Continuous Integration, Groovy, Selenium and frameworks for QA. For further information please check our web site.



We enjoy putting our experience to your service. Our know-how allows us to provide consultation services for projects at any stage, from small up to super-large. Due to our range of expertise we can assist in software architecture and COTS selection; review of software designs; creation of testing suites and test plans with focus on automation; help building quality assurance teams via our extensive training curriculum.

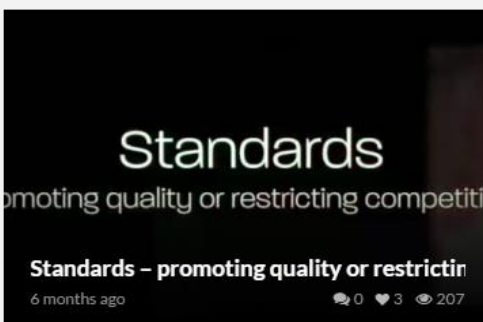
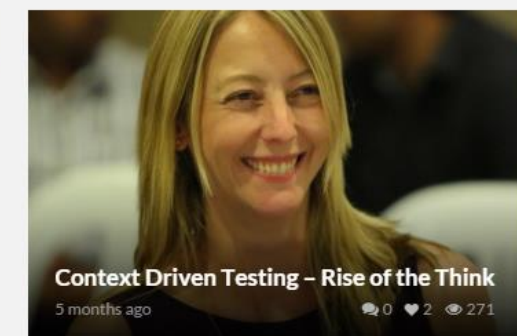
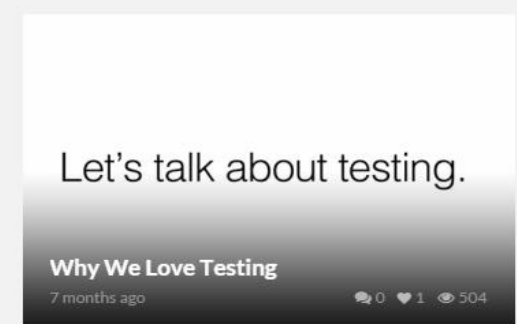
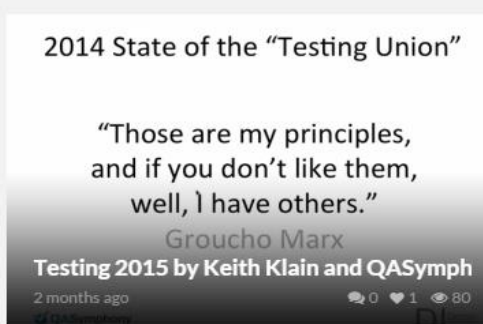
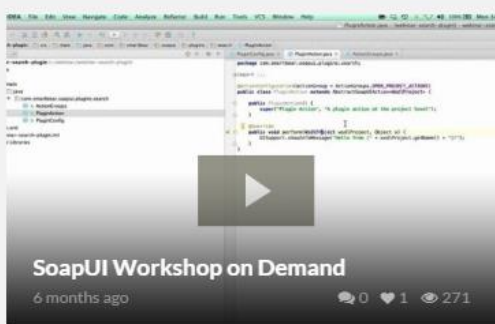
look us up: [www.testomaton.com](http://www.testomaton.com) - twitter: @testomaton



Got tired of reading? No problem! Start watching awesome testing videos...

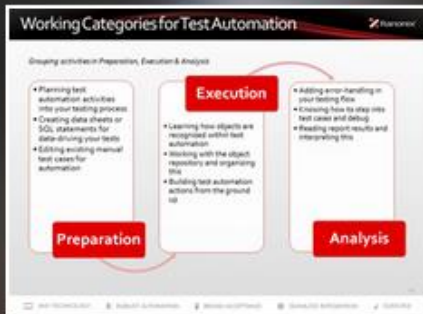
# TV for Testers

Your one stop shop for all software testing videos



[WWW.TVFORTESTERS.COM](http://WWW.TVFORTESTERS.COM)





Register for Jim's On-Demand Special Webinar

## How Manual Testers Can Break into Test Automation

We have recorded our well-attended webinar of Jim Trentadue, one of Ranorex' most experienced managers.

To watch it, please fill out the following form. You will receive an email with the links to the on-demand video and also the presentation slides afterwards.

**Adoption of automated testing** has not happened as quickly as many organizations have required. As more companies move toward implementing **agile development** as their software development lifecycle, more features are being implemented and released more quickly. This leaves less time for full regression testing of the system, nonetheless this should still be done. Manual testers need to transform into test automation testers as well.

Many manual testers believe they have to learn a development language in addition to the functionality of a specific tool to be effective. Add to that the in-depth or SME knowledge one must have about the system under test along with the development and management support required and it may not seem at all clear where to start.

Jim will cover the following in this session:

- » *The challenges faced by many organizations beginning the test automation journey*
- » *Early stages of adoption and adding to the value of work handled by a manual test team with little programming knowledge.*


Learn how to make this jump as a manual tester and focus on the right areas first e.g. **automation test structure**, **object recognition** and **results interpretation**.

Register for the On-Demand Webinar

Duration: **60 minutes**

Presentation language: **English**

**REGISTER NOW**



[www.talesoftesting.com](http://www.talesoftesting.com)

What does it take to produce monthly issues of a most read testing magazine?  
What makes those interviews and articles a special choice of our editor?  
Some stories are not often talked about...otherwise....! Visit to find out about  
everything that makes you curious about **Tea-time with Testers!**



# Advertise with us

Connect with the audience that MATTER!



Adverts help mostly when they are noticed by **decision makers** in the industry.

Along with thousands of awesome testers, Tea-time with Testers is read and contributed by Senior Test Managers, Delivery Heads, Programme Managers, Global Heads, CEOs, CTOs, Solution Architects and Test Consultants.

Want to know what people holding above positions have to say about us?

Well, hear directly from them.

And the **Good News** is...

Now we have some more awesome offerings at pretty affordable prices.

Contact us at [sales@teatimewithtesters.com](mailto:sales@teatimewithtesters.com) to know more.





Every Tester

who reads **Tea-time with Testers**,

Recommends it to friends and  
colleagues .

**What About You ?**



# in ne>xt issue

articles by -

Jerry Weinberg

T Ashok

Rahul Verma

Joel Montvelisky

...and others

# our family

## Founder & Editor:

Lalitkumar Bhamare (Pune, India)

Pratikkumar Patel (Mumbai, India)



Lalitkumar



Pratikkumar

## Contribution and Guidance:

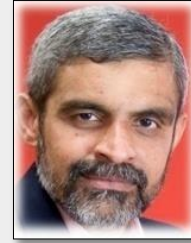
Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)



Jerry



T Ashok



Joel

## Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Cover page image – wallpapescraft.com

## Core Team:

Dr.Meeta Prakash (Bangalore, India)

Unmesh Gundecha (Pune,India)



Dr. Meeta Prakash



Unmesh Gundecha

## Online Collaboration:

Shweta Daiv (Pune, India)



Shweta

## Tech -Team:

Chris Philip (Mumbai, India)

Romil Gupta (Pune, India)

Kiran kumar (Mumbai, India)



Kiran Kumar



Chris



Romil

*// Karmanye vadhikaraste ma phaleshu kadachna /  
Karmaphalehtur bhurma te sangostvakarmani //*



To get a **FREE** copy,  
[Subscribe](#) to mailing list.

**SUBSCRIBE**

Join our community on

**facebook.**

Follow us on - [@TtimewidTesters](#)



[www.teatimewithtesters.com](http://www.teatimewithtesters.com)

