DEDICATED TO SHOWCASING NEW TECHNOLOGY AND WORLD LEADERS IN SOFTWARE TESTING

LogiGear MAGAZINE

The Training Issue

Black Box Software Testing:

Test Design Course

By Cem Kaner, Online BBST

Fun & Games:

Keep Training Lively

By Michael Hackett, Sr. VP, LogiGear Corporation

Coaching in Distributed Teams:

Challenge, Nonsense or Added Value?

By Jaroslav Prochazka, Tieto

February 2012 | Volume VI | Issue 1

LETTER FROM THE EDITOR

Worldwide Offices

United States Headquarters
2015 Pioneer Ct., Suite B
San Mateo, CA 94403
Tel +01 650 572 1400
Fax +01 650 572 2822

Viet Nam Headquarters
1A Phan Xich Long, Ward 2
Phu Nhuan District
Ho Chi Minh City
Tel +84 8 3995 4072
Fax +84 8 3995 4076

Viet Nam, Da Nang
7th Floor, Dana Book building
76-78 Bach Dang
Hai Chau District
Tel +84 511 3655 33
Fax +84 511 3655 336

www.logigear.com
www.logigear.vn
www.logigearmagazine.com

Submission guidelines are located at http://www.logigear.com/logigear-magazine/submission-guidelines.html

I have been training testers for about 15 years in universities, corporations, online, and individually – in both a training, managing and coaching capacity. So far, I have executed these various training efforts in 16 countries, under good and rough conditions – from simultaneous translation, to video broadcast to multiple sites, to group games with loud music; all for the purpose of skill-building for better quality software and better job satisfaction.

When I started working in software development, there were no YouTube videos, no Google TechTalks, only a small, but very good library by which to gain knowledge on testing. The best knowledge I received was from generous co-workers and by tribal knowledge learning - the "tips and tricks", and tools of the trade.

But back then, our development projects were slower. There was just as much pressure to work fast, but enough structure, process, approvals and manufacturing to provide a bigger buffer than we have today with, for example, SaaS immediate deployment. On the other hand, there was much less knowledge about testing! *Everyone* who tested was called QA. There were no differentiated job titles or skill sets, like "Quality Engineer", "Software Tester", or "Software Developer in Test" and the best test case management tools were spreadsheets. Today it's an entirely different world.

Software development has completely changed. New platforms, devices, programming languages, development tools, testing tools, and development lifecycles have all sped up product delivery to a blazing speed. Additionally, test engineers are expected to be fluent in all the technologies needed.

Luckily for all of us, the availability of learning and training is now at our fingertips. Onsite courses, university curricula, video, tool trainings, and white papers abound! Today you can get the skills, tools, ideas and knowledge to execute your job better, release higher quality software and be happier and confident in your job. Having the commitment, time and money are now the principal obstacles to overcome.

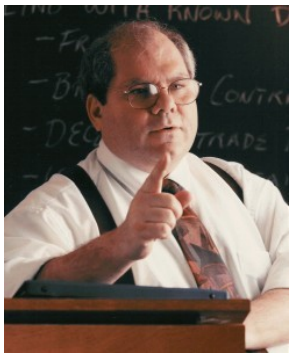
The problem for organizations now is how to provide skilled labor and continuous training in the most effective manner. Figuring out the *real* need is always the first step – what's best for each individual, what is best for your team, what's best for teams across an organization. What is the best method of delivery for the training? Live? Online? Hybrid? Training adults– especially knowledge workers – is obviously very different from educating young people. To have the training suit the needs of staff is not always easy!

In this issue, we will try to get you started along that path! First, our blogger of the month, Jaideep Khanduja, explains which soft skills are vital to project success. Then, I'll offer up the latest trend in knowledge worker training - training using games to increase knowledge retention. Cem Kaner explains the intense research, development and structure of his Black Box Software Testing - Test Design course. Jaroslav Prochazka will take you through the challenges of coaching and mentoring teams in distributed environments. I'll examine different team training techniques, while highlighting those that work and those that don't. And finally, I'll go through the results of the training section from the 2010 global survey.

These days, I often hear a phrase in software development: "Automation is not optional!" I agree. I also think "Training is not optional!" Training testers – home teams, outsourced teams, and offshore teams – and training managers, leads, and other software development folks in testing practices and principles, is no longer optional! Both hard skill and soft skill training is the key to success!

Michael Hackett
Senior Vice President
Editor in Chief

IN THIS ISSUE



Cem Kaner, Page 12

IN THE NEWS

Microsoft's Harry Robinson Video Interview on Training

At VISTACON 2011, Harry sat down with LogiGear Sr. VP, Michael Hackett, to discuss various training methodologies.

Harry Robinson is a Principal Software Design Engineer in Test (SDET) for Microsoft's Bing team, with over twenty years of software development and testing experience at AT&T Bell Labs, HP, Microsoft, and Google, as well as time spent in the startup trenches. He currently works with Bing teams on developing effective test strategies across the product. While at Bell Labs, Harry pioneered the test generation system that won the 1995 AT&T Award for Outstanding Achievement in the Area of Quality. At Microsoft, he created the model-based test technology which won the Microsoft Best Practice Award in 2001.

You can watch his interview by clicking here

Editorial Calendar Announced

We look forward to an exciting and full 2012 as its predecessor was a tough year for many in the software business. At *LogiGear Magazine*, we view 2012 as a brighter year filled with innovation and advancement. We will continue to bring you current articles and videos on the latest trends in software quality.

Our editorial calendar is filled with engaging themes addressing software developments, ongoing research and leading individuals in the industry.

As it is difficult to keep pace with the speed of which technology advances, we would like to invite those who have a story in mind and interested in publishing to our readers – be it an interesting perspective or memorable experience on software testing, we are interested in publishing articles or columns from professionals in the field. Our focus for 2012 is getting the next generation of software quality experts published and in the news! Please visit our website as we have just posted our 2012 editorial calendar. Our submission guidelines and contact information are also online here:

http://www.logigear.com/magazine/issue/news/editorial-calendar-and-submission-guidelines/

As the new year moves forward and software testing progresses, we at *LogiGear Magazine* will cover the highlights along the way.

Additional Training Resources

LogiGear offers a comprehensive selection of software testing training courses. A list of them can be found on our website here:

http://www.logigear.com/services/qa-training.html

Michael Hackett discusses the problem of training global software testing teams and highlights training as the solution to many of those problems. In this article, we describe some of the important facets of training and considerations for the solution, as well as outlining the necessary training topics. The full article can be found here:

http://www.logigear.com/2006/294-building-a-successful-global-training-program-for-software-testing.html

A full PowerPoint presentation on the topic can be found here:

http://www.scribd.com/doc/27314592/Build-a-Successful-Global-Training-Program-Effectively-Training

Former *LogiGear Magazine* guest writer, Bogdan Bereza, has a number of training resources on his website:

http://www.victo.eu/ENG/Training/index.html

BLOGGER OF THE MONTH

Jaideep Khanduja

Progressive software testing approach by acquiring soft skills

In software testing, we need to devise an approach that features a gradual progression from the simplest criteria of testing to more sophisticated criteria. We do this via many planned and structured steps, each of which brings incremental benefits to the project as a whole. By this means, as a tester masters each skill or area of testing, there is always something worthwhile to try next. Important aspects are:

Trust
Testing and development teams must build a trusting relationship with each other especially within the process of product development.

Constructive Communication
Initial communication between testing and development teams must be quite open, transparent, constructive and encouraging. This is the phase that determines the course of action for testing, as well as any alternatives. The future of the software, its accuracy, and its subsequent successful implementation, depend on good communication. But don't make this communication too formal.

Documentation
Ensure that testers have sufficient documents in hand before starting the analysis and testing the product. Appropriate analysis of these documents is very important to remove any ambiguity/confusion/loop-holes in understanding the customer's requirements. Test cases and scenarios will emerge from these requirements.

Open Mindedness
A tester has to go in with an open mind in all discussions. Only then will they be able to understand developers, the development process and customer requirements. Then they will plan the testing requirements accordingly.

Variables
Keep all important variables in the loop is important to testing and successful product development. Different variables could result in developers working on different modules, documents, testers, etc.

Learn from the Past
Keep learning from past experiences and try not to get caught with mistakes that have been made before. A common saying is – "Never repeat the same mistake twice". It's a mistake to test a product in a crude or unstructured manner. Learning from the past is perhaps the most potent means of improving your testing model over time.

Identify Shortfalls in Advance
Be clear about what is in the scope of testing and what is not. Clearly define what is not going to be covered in testing and understand what impact such omissions could potentially have on the product. Foresee any uncertainties or delays. It is important to understand the benefits you will derive well in advance rather than getting no benefits at the end of an exercise.

Keep Interacting
It is important that a tester continuously interact with developers throughout the entire testing process. Ultimately, the goal is to create a better product. To achieve this, both sides must work hand-in-hand to surmount all barriers, conflicts, bugs, faults and defects. ■

Jaideep Khanduja *has over 20 years of IT experience. Jaideep is currently working in the India Regional Office of a large versatile International group as Head of Quality Assurance and Project Management. Jaideep believes that innovation, team management, time management, skills enhancement, learning, knowledge management and mentoring are the best tools to grow.*

Fun & Games: Keep Training Lively

Activities and games to boost information retention.

By Michael Hackett, LogiGear

Training has to be fun. Simple as that. To inspire changed behaviors and adoption of new practices, training has to be interesting, motivating, stimulating and challenging. Training also has to be engaging enough to maintain interest, as trainers today are forced to compete with handheld mobile devices, interruptions from texting, email distractions, and people who think they can multi-task.

Many trainers cringe at the term *edutainment,* the necessity of education to be entertaining. But the fact is: if class time is boring – regardless of how important – your efforts will fail. Regardless of the content, to boost interest and retention, training has to be interactive.

Today, there is great innovation going on in classroom practices – and not only from online delivery. In class, I use as many games, exercises and examples as I can to engage and challenge students. It takes longer than a lecture, but it's much more effective.

What is clear about training adults is that lectures *alone* do not work. The content is easy to create and most efficient in terms of delivery time. But, in my experience, the retention rates are just too low! It's essential that there be learning *activities* – and the more the better.

There must be learning activities to:

- Verify the content delivered was received
- Get feedback and *replay* to gauge the level of understanding
- Challenge adults with having them immediately apply the information – a key achievement for adult learners.
- Make the training more diverse and interesting to keep attention and increase retention
- Entertain, make training more interesting and fun

This does not have to be complex, time-consuming or a big event. At a minimum it can be an instructor-led walk-through of a testing example. It could be relating a past project experience. Although an instructor-led walk-through will have less interaction, telling effective stories can expand thinking and application. It could also be a testing simulation applying a newly learned process from start to finish. Knowledge needs more than one method of delivery for retention and application.

It has become increasingly popular in recent years to have corporate classes designed as simulations. Within this format, the classes are conducted with numerous activities and learning games, which have proven to be the key method to learning and applying job skills.

A Scrum Master Class Full of Games

Let's talk about learning activities from an extreme. A few years ago I became a certified scrum master. I took a training class where the majority of the classroom hours, content and key learning topics were delivered using games.

Everyone in the room was highly motivated; no one was there *because their manager made them attend* – we were eating up the information, which was very well received. From these classes, I took away five important ideas about games in training:

1- I learned a lot! There was great participation. Games were fun and effective. In fact, I am using some of the same games now in my Testing in Agile class. [http://www.logigear.com/services/qa-training/testing-in-agile-development.html]

2- People were happy. A few participants did miss some learning points or the real essence of the activity. The games were time boxed and there was casual pressure to finish the tasks *on-time* (a key aspect of SCRUM). Some people were so focused on the task at hand and in finishing on-time that they missed the [key] learning point. So, "happy" does not mean all people learn.

3- Even with highly motivated people, the instructor had his hands very full answering individual questions, clarifying game rules, keeping his eyes on the clock and organizing things. Game activities often make instructing more difficult.

4- Just a few learning activities were played for the sake of *playing a game* where, with highly motivated participants, certain topics would have been more efficiently delivered as lecture and perhaps a discussion.

5- There were situations where not every participant was highly motivated, for example, at the end of the training day. Some people got distracted and it was clear attention wandered. In these cases, the retention and application of information was more problematic.

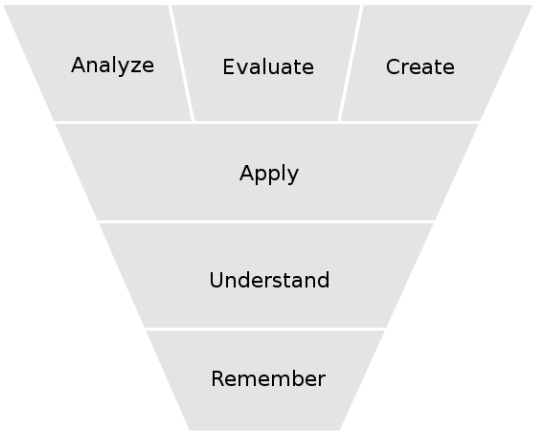
Learning Styles and Activities for Greater Content Retention

To get a fuller understanding of the impact of learning activities on retention, familiarize yourself with the following resources:

Learning styles: (http://en.wikipedia.org/wiki/Learning_styles)

Bloom's taxonomy which shows the progression of awareness in acquiring new skills. [http://en.wikipedia.org/wiki/Bloom's_Taxonomy]

Cem Kaner's seminar on Teaching Testers from BBST www.testingeducation.org/course_notes



Bloom's Taxonomy
Source: Anderson & Krathwohl, 2001. Via Wikipedia.

Clearly, the goal of any training, after comprehending the content, is to apply the gained knowledge to do a better job, improve quality, feel more confident, and be happier in one's work.

It's basic to every curriculum developer – be it a skilled trainer or a fellow team member putting a few slides together on the bug tracking tool – that training for adults should be both useful and compelling. Building training needs to be a mix of lecture, homework, reading, online video, exams, an array of exercises or even heady theoretical content. Vary the content delivery to accommodate learning styles, and climb Bloom's taxonomy past understanding - first to application of the knowledge and then higher to analyze, evaluate and create new work habits, ideas and processes based on the gained information.

Some might complain "These are adults. It's their job!", "They should be self-motivated.", "The instructor should be focused on *efficiency* – do a lecture! Follow up with some Q&A and be done with it!"

It's not so easy. Adult learners, motivated or otherwise, fall across a spectrum. Their optimal learning experiences vary widely. Each has a different retention level based on the type of content delivery. What I know empirically is that a lecture, in and of itself, has the lowest retention rate.

To learn more about learning styles, I'd start with Fleming's learner types – visual, auditory and kinesthetic learners. At a minimum, content must include reading, lecture with discussion, and learn-by-doing activities. Diverse delivery methods better suit the full range of learning styles and keep energy levels up. Use activities to actively engage participants' minds.

In the years that I've spent developing curricula, I can tell you that it is a lot easier to simply put together a bunch of slides for a lecture on some topic. Get the points covered, and communicated well. It's a nice, fast delivery – only problem is that it has the lowest retention rate. Great, challenging, useful, fun activities are much more difficult to develop and incorporate into limited-time trainings. While difficult, the results of activity training justify the added effort. Knowledge needs to come to life! Effective training needs walkthroughs, examples, exercises, games and learning activities.

Online Learners

This discussion on retention brings up one of the hot topics in training today: online learning. My work is mainly with product test teams or corporate training of individual testers working in that organization. With teams, there are many additional benefits to gaining knowledge – discussing immediate implementation of new knowledge, how to apply it with existing tools, toss out misguided tribal knowledge, even team building and better communication. These additional benefits are more difficult to obtain when groups rely on online learning.

I go into greater detail about the benefits and pitfalls of online training on page 24 of this issue.

Problems with Learning Activities, specifically Games

For an instructor or activity leader, I have a few words of advice: "Practice, practice, practice". Things can easily go wrong. To lead challenging activities you need to be skilled in encouraging, coaching, communicating, salvaging difficult situations, reinforcing the learning point, managing time, and sometimes dealing with difficult people.

Time Management

First, the time factor. Activities always need setup time. The more complex the activity, the longer the setup time. This time needs to be clear, efficient, and kept to a minimum. There have been times that I developed a good activity only to discover that the setup time was too long. Some groups may be slow in execution, some *individuals* may be slower, some may be so creative and engaged that time becomes problematic. Getting the exercise completed without getting distracted into worrying about time is important.

The post game discussion to reinforce the learning points and answer questions is the most important part of the game or activity. It is a mistake to cut this short. Also, games often do not go perfectly. They can run long, go *wrong* and need to be saved through a discussion of what happened and how to fix it if that happens in *real life.*

When setup includes people moving, or supplies or furniture moving around– whether it is balloons, basketballs, paper and scissors or chairs and desks, the momentary break can be a distraction. People may take it as an opportunity to make a call, text, or refill coffee. Time management can be challenging.

Lastly, the biggest failure in any complex, challenging, compelling learning activity is students missing the point! Remember that the activities are metaphors or representations of day-to-day work. Whether the activity is making paper airplanes, guessing the number of gumballs in a jar, or writing soap operas about distrustful family members trading stock on eTrade – whatever the game or activity, participants need to be kept focused on why it is important enough to take class time.

Let's use an easy example of the simplest form of activity – the icebreaker. Icebreakers rarely have good learning points. Even in a soft skill, communication, team building topic, of which I have many in *Working with Offshore Teams*, and *Leading Software Test Projects with Confidence*, there is no time or activity which do not have a clear and direct link to a skill or idea important enough to be in a time-limited class.

The essential aim or idea of every learning activity must be obvious in its application of the learning point. If the activity only gets people moving around or breaks up the monotony of a lecture - it's an inefficient use of time. Activity for the sake of activity is not effective.

Make every moment count. Reiterate the learning point, expand its application, and elicit questions and examples.

A key consideration for anyone developing training material on testing is - must all the learning activities be about software testing?

I like to use as many testing examples as I can. It's becoming more common for instructional course designers to build complex and long activities - devoting hours at a time - to solve problems common in their domain. My training usually does not have that time luxury. I most often cover a wide variety of topics in a couple of training days. For the sake of time, efficiency, convenience and fun, I most commonly use testing topics but occasionally mix in other topics.

My Experience

There are four examples that come to mind of effective learning activities in Testing and Leading classes I want to share. Some are directly about testing/software development. Others are not.

1- All my Children do Online Banking

Soap Opera Test Case Development - real-world, simulation. Testing related.

I've been teaching scenario-based testing (meaning workflow, user scenario, path testing) for a few years. The material was fine and it was interesting enough. Scenario based testing is a pretty straightforward topic. But actually writing effective, useful, potentially bug-finding scenarios is not as easy as it seems. After a couple of years, I updated the chapter exercise to a much more difficult, complicated exercise. In this more difficult exercise, the teams have to write a scenario. Then, the other groups in the class evaluate them; estimating their bug finding ability.

To do this exercise, I chose a well understood system; buying a book at Amazon.com or transferring money and paying bills in online banking; an easily understood system makes choosing personas, interesting and meaningful functionality and excellent data are all key component of the exercise. Then, the teams read their scenarios to their co-workers and get graded. The most interesting part, to me, as the instructor, is that at least one team's scenario will "fail" in the other teams' judgment. That is, it doesn't meet the criteria of great test scenarios.

Next, as a group, we fix the failed scenario. The change in participation, understanding, and fun was immediately palpable with the new, more difficult evaluation, scoring and fixing part of the activity.

The things that makes this game worthwhile and effective is that after the lecture and discussion, we immediately apply the knowledge to a testing problem, critique the other team's solution, score it - reinforcing *why* we are writing these "test cases" – to find bugs, and apply the knowledge again by fixing low scoring or bad scenarios. Reinforcing the knowledge is crucial to retention.

I regularly hear at the end of this exercise: "I thought writing scenarios was easy. Now, I know I have to go back and fix a bunch of end-to-end workflow tests I have been relying on."

As I mentioned above, adding exercises can significantly add time to the class. This is the challenge I've come across with this exercise. I need to keep the teams focused, keep it fun, keep it creative, and get it done in a reasonable time.

With many time consuming elements for this activity, it takes focus and attention to the clock to get it done as quickly as possible. This activity is fully related to testing

and applicable immediately to test case development.

2- Build a Bicycle!

Constraint analysis and critical path planning. Hypothetical game exercise. Not related to testing.

In my, "Lead Software Test Projects with Confidence class", there is content on resource planning and allocation. We do a game on constraints and critical paths analysis before making a schedule. The exercise is short but effective. The activity does not directly relate to testing.

The example I used varies from *making a cup of tea* to *repairing a flat bicycle tire,* among others. It is important to state, and re-state, that this exercise uses making a cup of tea but is not about *making* a cup of tea. It's about constraint analysis, defining and measuring the critical path. It also involves moving around and working fast. Teams communicate and make decisions in order to *beat* their co-workers' time to finish the task. It absolutely wakes people up. It gets the groups energized and involves each person. The additional benefits to the learning points are great for the class dynamic.

3- Blind-folded Basketball Coach

Coaching and Leadership game. Not related to testing.

In my class on *Working with Offshore Test Teams* we play a game on the topic of coaching, mentoring and delegating. I vary this game between throwing balls into a bucket - similar to shooting baskets - blindfolded, or building bridges out of only newspaper and tape. These games are always fun - sometimes hilarious - and at the same time, we never get away from the *learning point*. The activity is not about who can throw the most balls into the basket or build the longer bridge. It is about coaching offshore teams; coaching when you can't see the person, on the telephone, when you may not be located in the same office, when you may not be in the same country. Coaching face-to-face is difficult for some and coaching remotely is that much more difficult. A level of trust must be in place for remote coaching to be effective.

The reinforcement and discussion around these learning points are what makes this exercise useful and not merely a distraction or waste of training time.

4- Tell me a Test!

Lightning fast test development exercise, simulation. Related to testing.

In all my testing methods classes, whether it is the core, foundation level class - *Testing Computer Software* or *the Applied Testing Topics* advanced class, I very often do a rapid fire, instant test case development, lightening fast exercise. I set up the test situation and point my finger at each student for rapid fire test case development recommendations. The class *wakes up* and results in changes to their posture. And, most importantly, I get full participation as each person must answer immediately, whether it is a piece of data or test conditions. For example, in the Applied testing class, we use Google Checkout for an API testing example. After we setup and discuss the idea of passing parameters, we do rapid fire, lightening ideas for developing great test data.

Lightening tests - in a nice way - get everyone involved, assesses understanding, and are always fun. People come up with new, inventive, creative test data - fast. The added bonus is that it comes from co-workers, not me. I do this enough during long classes, that people joke with me, point their fingers at me and say: "Tell me a test!" When I have more time available for this exercise, I have other people in the class rate the test or test data as excellent (5 points), good (4 points), mediocre (3 points), bad (2 points), or will probably miss a bug (1 point).

Summary

Training is crucial to any test team's success. But it has to be effective in a way that is engaging to boost retention which will push staff to apply the gained knowledge. No one can afford to waste time, effort and money on mediocre training. Training for adults needs to be applicable, fun and interesting - bringing to life content introduced by reading or lecture. It must apply to work situations, provide real-world examples, and result in discussions about how to apply these concepts. Practice exercises and simulated work situations are great ways to convey these ideas. Use a variety of learning activities to climb to higher classes of Bloom's taxonomy - to analysis, synthesis and evaluation. This will foster application of skills for better testing with more confidence and less stress. ■

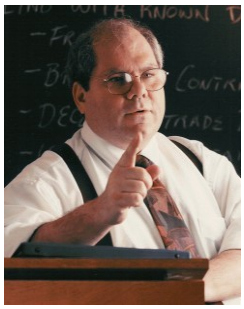
The classes referred to in this article have descriptions located at http://www.logigear.com/services/qa-training.html

Black Box Software Testing: Test Design Course

An examination of the development and structure of the third course in the BBST series.

By Cem Kaner, Online BBST

A few months ago, Dr. Rebecca Fiedler and I published BBST—Test Design. This third course completes the Black Box Software Testing (BBST) set. The other two courses are BBST Foundations and BBST Bug Advocacy. This article offers some information about the series, the design of the series (and the underlying instructional theory) and why you might be interested in it.



All of the existing courseware is available to you for free.

You can download the courseware directly from my lab's website, www.testingeducation.org/BBST or from our corporate website, www.bbst.info.

If you are reading this article a few years from now and if bbst.info and www.testingeducation.org are outdated, check the National Science Digital Library, www.nsdl.org. NSDL is an electronic library of instructional materials funded by the National Science Foundation of the United States. BBST has been accepted as a collection in NSDL.

The courseware is free, but not cheap. The underlying research and development cost over a million dollars over a period of 11 years. But the public has already paid for these materials, so there is no ethical basis for making you pay for them again.

The way this came about is that I returned to school (Florida Tech) in 2000 to improve my teaching. As a financially successful testing trainer in the 1990's, I was not satisfied with the results of my courses or the courses of many colleagues whom I knew and respected. In our short courses, we overwhelmed students with information, most of which they forgot. Our three days with them didn't give them enough time to practice and develop skills; to try techniques on hard problems and get feedback; or to apply the ideas to their own work and come back with critical questions when their first try at application didn't work. I wanted to develop courses that could make a more significant difference to the technical knowledge and cognitive skills of my students. Soon after I came to Florida Tech, the National Science Foundation agreed to support my research on testing and on online learning. In return, I agreed that courseware that I created with NSF support would be available to the public for free.

Our courseware includes:

- Videotaped lectures and their accompanying slides (currently 1065 slides).
- Readings: These are articles and book chapters (some specially updated for this course) authored by leading people in the field.
- Assignments and study guide questions to help you work through the material with your friends. We've included suggestions for using and grading the assignments and questions in our *Instructor's Manual*.

The best way to work through these materials is in a group. The members of the group work through the lectures together, do assignments and take tests together, apply the ideas to their day-to-day work and discuss the results. Some examples of how our courseware is being used are:

- The Association for Software Testing (AST) offers the three courses to its members at a low price. AST can afford such a low price because its instructors teach the course for free. (Not surprisingly, AST can only offer these courses a few times per year.)
- Some companies train one of their staff to be a lead trainer for BBST at their company. That person then coordinates classes at the company.
- Several universities use our materials in courses that they offer.
- Some people choose to form their own study group and work through the materials together.
- Some professional trainers offer this course to the public or to companies. These people charge normal commercial rates for the course.

Dr. Fiedler and I offer the course to companies who contract with us to train their staff – we often customize the course a little for the client company. (We charge normal commercial rates for this service.) Dr. Fiedler, Doug Hoffman and I are just finishing the *Black Box Software Testing Instructor's Manual*. We've been using drafts of this (now 400-page) book for three years to help people improve their online teaching skills and learn how to teach the BBST courses. We've added a lot of detail to this final draft. We hope that experienced trainers will be able to learn how to teach BBST from the book, even if they cannot come to one of our instructor training courses.

The book is now in final production. We expect printed copies at www.Amazon.com by June (maybe sooner). Electronic copies of the book will also be available at www.testingeducation.org, for free.

Dr. Fiedler and I are also writing a *Students' Workbook for the BBST Courses*. Our primary goal is to support people who are studying the material on their own. We also want to support instructors (and students) who find it easier to work with a course like this if it has a textbook. The *Students' Workbook* adds notes on every slide, adds many more references, review questions, exam questions

and suggests several more learning activities. The *Students' Workbook* won't be free – the research support funds that we relied on to help us develop the courses and *Instructor's Manual* just don't stretch far enough to subsidize the *Students' Workbook* as well. So this will published commercially at a price we expect to be reasonable for a technical book. We anticipate (but cannot guarantee) that the workbook will be available by September 2012.

Principles Underlying the Courses

BBST starts from the premise that software testing is a cognitively complex activity. We think people are pretty smart. That includes junior testers. We don't think that the difference between junior testers and seniors is that juniors should do routine tasks and seniors use their brains. Probably *every* tester has *some* routine, boring work to do. But we think that every tester, no matter how junior, should also exercise skill and judgment, critical thinking and creativity. More experienced testers can handle more complex tasks with less supervision, but in our opinion, even the most junior testers should be trained and managed to apply their own judgment and skill to their work.

We see testing as an empirical, technical investigation conducted to provide stakeholders with information about the quality of the product or service under test.
This definition of testing carries some important implications for our practice as testers and as educators:

- Different stakeholders need (or want) different information. One person might want to estimate the cost of supporting the software and what needs to change to reduce support costs. Another person might want to compare the quality to competing products. A third might want to compare the product to a specification in a contract. And a fourth might want to understand whether an accident that killed someone was caused by defects in this software. These are all reasonable things to want to know. An important part of our work, as testing-service providers, is to find the kinds of information that our clients want to know and to communicate that information clearly to them.

- Testing is an empirical activity – we gain our knowledge by running experiments (we call them tests). Humans have been thinking about how to systematically gain empirical knowledge for about 2000 years. We have a lot to gain from a study of the history and methods of science.

- Different testing techniques are better suited for exposing different types of information. For example, some techniques are very powerful for hunting bugs, but other techniques are more effective for highlighting issues that will cause customer confusion, dissatisfaction and calls for support. To do the job well, a tester must know many techniques and must understand these techniques' strengths and blind spots. In being so prepared, she is then in a position to pick the right set of techniques for collecting the types of information needed for whatever project she is working on.

Many courses focus on definitions. Definitions are easy to teach. Definitions are easy to test – especially on multiple-choice exams that can be graded by computers. But in our view, having knowledge of definitions is not very important. Knowing the definition of a technique does not tell you when to use it, or how to use it, and even if you can memorize a definition that includes a description of how to use it, you won't be able to use it well without successful practice. We present definitions because we must, but our objective is the development of judgment and skill.

We use multiple-choice tests as aids for review. Our multiple-choice tests are open book. Our questions are more difficult than most because we use them to teach and to stimulate discussion. When we teach the course to university students and to practitioners, we never use the multiple-choice test results to decide whether a student passed or failed the course.

We will publish a collection of multiple-choice review questions in the *Students' Workbook*. If you study the course material on your own or with a group of friends, don't worry if you get questions wrong. Use the feedback as a guide to learning more.

We use essay exams and assignments as our most important teaching tools. There is now plenty of research establishing that significant learning takes place when students write exams. We provide study guides with large collections of essay questions and design our exams to encourage students to create practice exams in preparation for our finals. With the assignments, we apply the course ideas to real products. For example, in the Bug Advocacy course, students join the test team of a significant open source project (typically Open Office or Firefox) and help the team replicate and clarify its bug reports. In the Test Design course, students assess specifications and documentation from significant projects (such as Google Docs) and apply risk analysis and boundary analysis to variables from a well-known product (such as Open Office). We design the assignments so that any company teaching the course to its

own staff can easily modify them for use on its own products.

The Courses

The BBST series includes three courses:

BBST Foundations presents the basic concepts of software testing and helps students develop online learning skills:

- We introduce the basic vocabulary of the field. The most important part of this is a review of (or for some students, an introduction to) basic facts of data storage and manipulation and of the flow of control in computer programs.

- We also present definitions of testing concepts because we need to establish a common vocabulary for the course. But along with these, we present our viewpoint that there are many definitions for most testing concepts and that the differences often reflect genuine disagreements about the nature of competent testing. We have a lot to learn, in our field, before those disagreements will be resolved. We teach testers that the best way to communicate with other people is to figure out what those people mean when they say something – ask them questions! – instead of assuming that they use the words the same way an instructor did in a testing course.

- We also introduce the key challenges of software testing:
    - The best strategy for testing a project (including the choice of techniques) varies. It depends on the informational needs of the project's stakeholders.
    - An oracle is something that helps you decide whether a program passed or failed a test. For example, the test's "expected results" are an oracle. However, all oracles are incomplete and they can be misleading. The pass/no-pass decision requires judgment, not just specifications and procedures.
    - There are hundreds of ways to measure coverage, each emphasizing different risks. We teach enough about programming for students to understand the simple code-structure coverage measures (such as branch coverage and multi-condition coverage) but we also explain why this is a narrow sample of the testing that can expose significant problems. Complete structural coverage is not complete coverage.
    - Complete testing is impossible. On the way to teaching this, we review some discrete mathematics to help students understand how to estimate how many tests it would take to fully test some simple parts of programs.

- Finally, we introduce students to software metrics, the theory of measurement, and the risks of measurements that are not skillfully designed. Most metrics of software quality and of the effectiveness or productivity of development and testing staff are human performance measures. We explain measurement dysfunction (the very serious risk of making things worse by relying on bad metrics). This is just an introduction. I teach a full-semester course on software metrics (about the same amount of content as *Foundations* + *Bug Advocacy* combined) and hope to bring that online in the next few years.

BBST Bug Advocacy presents bug reporting as a communications challenge. Bug reports are not just neutral technical reports. They are persuasive documents. The key goal of the bug report author is to provide high-quality information, well written, to help stakeholders make wise decisions about which bugs to fix.

- We define key concepts (such as software error, quality, and the bug processing workflow). Of course, we present the diversity of views. There are many different definitions of quality – not just different words. Different ideas about what "quality" means and what counts as a departure from high quality.
- We consider the scope of bug reporting: how should testers decide what findings they should report as bugs and what information to include in the reports?

- We present bug reporting as persuasive writing. The tester makes the case that someone should take this bug seriously – and hopefully fix it. Students learn how to write more persuasively. Within this topic, we apply psychological research on decision-making and decision-affecting biases to consideration of how project managers and executives make decisions about bug reports.

- And of course, we provide a lot of tips for troubleshooting bugs and for making them more reproducible.

BBST Test Design surveys over 100 test techniques. You can't select the right techniques for your context if you're not familiar with a wide array of techniques.

- We take a detailed look at function testing, testing tours, risk-based testing, specification-based testing, scenario testing, domain testing, and some types of combination testing. We consider their strengths and weaknesses, what they are useful for, what they are likely to miss, and how much work it takes to use them well.

- We suggest that a testing strategy combines a set of testing techniques that will be, together, effective for providing the stakeholders with the types of information they want to get from testing.

- We present ways to compare the strengths of different techniques, to help students combine techniques that have complementary strengths.

- And in the process, we present some tools for concept mapping, active reading, specification analysis, and combinatorial analysis.

Summing Up

Good training should help you and your staff become better testers.

Good exams ask the kinds of questions that job interviewers actually care about. People don't really care whether a tester knows the words of a definition of a technique (or, at least, they *shouldn't* care!). They care whether the tester can actually use that technique to find bugs or to find other useful information.

Good assignments provide authentic tasks – tasks that students recognize as real-world, and that they realize will help them prepare to do their own real work. Good assignments help students develop skills. (You get better at *skilled work* with practice and feedback.)

The BBST series is not a set of miracle courses. These courses do require a lot of work. But many of our students have found them invaluable. You can read the testimonials at http://bbst.info/?page_id=35, or just do a Google search on < software testing course "bbst" > and you'll find plenty of reviews online.

If you do use our courseware, we'd love to hear back from you on how you are using it and how it is working for you. This feedback (including negative feedback) will help us improve our course designs for the future. It is also useful for the organizations that support our research. ■

About the Author

Cem Kaner is a Professor of Software Engineering at the Florida Institute of Technology. He combines an academic background (doctorates in psychology and in law) with extensive software development experience (programmer, tester, manager, director, etc.) in Silicon Valley. Kaner is co-author of *Testing Computer Software* with Hung Quoc Nguyen and Jack Falk. He also co-authored *Bad Software* and *Lessons Learned in Software Testing.*

BBST is a registered trademark of Kaner, Fiedler & Associates, LLC

We acknowledge the support of NSF research grants EIA-0113539 ITR/SY+PE: "Improving the Education of Software Testers" and CCLI-0717613 "Adaptation & Implementation of an Activity-Based Online or Hybrid Course in Software Testing." We also appreciate the generous support of Texas Instruments, IBM/Rational and the Florida Institute of Technology and generous contributions of course content from Hung Quoc Nguyen, Douglas L. Hoffman, James Bach, and Dr. Rebecca Fiedler, Co-Principal Investigator of the National Science Foundation grants. Dr. Fiedler holds a master's in business administration and a doctorate in education. Along with leading the instructional design of BBST, she has designed online courses for academic students and government agencies. Any opinions, findings and conclusions or recommendations expressed in this article or in the courseware are those of the authors and do not necessarily reflect the views of the National Science Foundation or our other donors.

J.J.G. Van Merrienboer, *Training complex cognitive skills: A four-component instructional design model for technical training* (Englewood Cliffs, NJ: Educational Technology Publications, 1997).

Doug Rohrer and Pashler, Harold, "Recent Research on Human Learning Challenges Conventional Instructional Strategies," *Educational Researcher* 39, no. 5 (2010): 406-412, http://edr.sagepub.com/content/39/5/406.abstract; Eric Jaffe, "Will that be on the test?," *Observer of the Association for Psychological Science* 21, no. 10 (2008): 18-21, *http://www.psychologicalscience.org/observer/getArticle.cfm?id=2425*.

Coaching in Distributed Teams: Challenge, Nonsense or Added Value?

By Jaroslav Prochazka, Tieto

Have you ever tried to push heterogeneous groups of people to move in the same direction? It is quite a challenge, isn't it? And now, imagine this situation in a distributed environment, with people whom you've never met, and who must be persuaded by means of phone calls or email. Welcome to our world. We have been doing leadership coaching to developers around the world for 5 years, and in that time, have learned to rethink many of the traditional approaches to training. When it comes to working with people scattered across the globe, we're "not in Kansas anymore".

Let me introduce this topic by clarifying some terms. If you ask people what is meant by ***coach***, ***mentor***, or ***trainer***, they often intuitively come to definitions that are almost the same for all three. The same can be said for the terms ***offshore*** and ***nearshore***. But these small differences in definitions make big differences when implementing and sustaining change.

By the term ***coaching***, we mean a pure "challenging" approach, one that doesn't necessarily provide teams any solutions or hints. A coach helps a person or team clarify the goal or current situation, or identify the way forward. He or she does this simply by asking the right questions. A coach does not provide any solution or best practices, and often doesn't even need to have any knowledge of the subject. A ***mentor***, on the other hand, can use coaching as one facilitation method, but extends it by his or her knowledge of the subject, teaching techniques, hands-on implementation support, etc. Thanks to knowledge of the subject matter, a mentor can share best practices (to avoid teams "reinventing the wheel") and can gently coax mentored people to help them learn lessons quickly. Finally, a ***trainer*** is someone who conveys information and skill through instruction.

Offshoring strategy and terms are quite familiar for most readers. ***Nearshoring***, however, may not be familiar: it means a form of delivery involving teams in different countries but with only small time zone shifts, and with cultural and geographical proximity. Offshoring, by contrast, means delivery with teams settled in distant geographical locations and with bigger time zone shifts (with few or no overlapping working hours). People in offshore teams very often also have quite disparate cultural backgrounds.

Our team itself originated in 2006 from a nearshore location in the Czech Republic, beginning with local mentors focused on the implementation of specific software design (SWD) practices such as Agile planning, continuous integration, test driven development, and pair work. Today, we are coaches driving corporate change in a learning organization. This evolution has had significant impact on the way we deal with change. Most significantly, our mindset has shifted from trying to enforce a solution, to enabling change in a learning organization; from "the solution is obvious, why do they not implement it?" to "human change management optimizing the whole process", as is shown in Fig. 1.

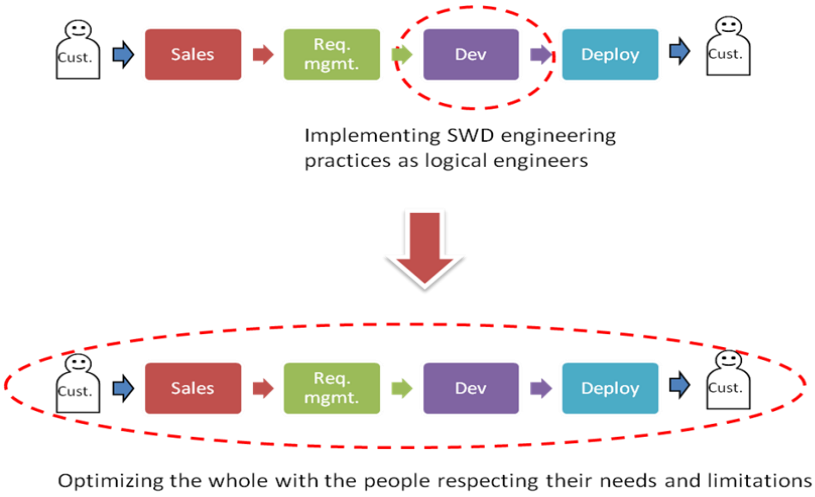


Figure 1: Shift of the mindset and implementation approach (from local mentoring to global coaching).

The key aspects of two key approaches to coaching in a distributed environment which we've learned, and now apply, are the following:

1. Having a Local change agent (a.k.a. internal coach) for hands-on support and sustainability.

2. Conducting a Kaizen workshop as a tool to overcoming the dispersion of teams, identifying common goals, and empowering the teams.

Local Change Agent

A local change agent can be a team leader, scrum master, architect, service manager, line manager or specialist in the local team having the right mindset. "Right mindset" in this case means understanding and *living* Agile and Lean principles. Position or role as such is not so important; more essential is having the "same language and thoughts". The change agent drives the change locally and provides daily support to local teams. Local change agents synchronize with us on a regular basis to ensure we're pulling the same end of the rope (sort of Scrum of Scrums). This synchronization among change agents only (not among all teams) is more efficient thanks to all possessing the same mindsets, lived principles, experience and vocabulary. Narrowband communication (phone, live meeting, email) used for coaches' synchronization is not such a big constraint in this environment. (You know, it is a pretty hard task to explain the value of Agile to a *waterfall*-ish guy over the phone).

Our first activities when travelling to remote locations are focused on identification and education of potential change agent candidates.

Kaizen Workshop

The Kaizen workshop is a very powerful tool for helping us overcome the distribution of teams, identify common goals, deliver value and, finally, empower teams for the change. The concept of the Kaizen workshop is nothing new; it is well-established as part of Lean thinking (see e.g., Liker: *The Toyota Way*). But the way we use it in distributed environments brings additional value. The workshop usually lasts for two days and has following agenda:

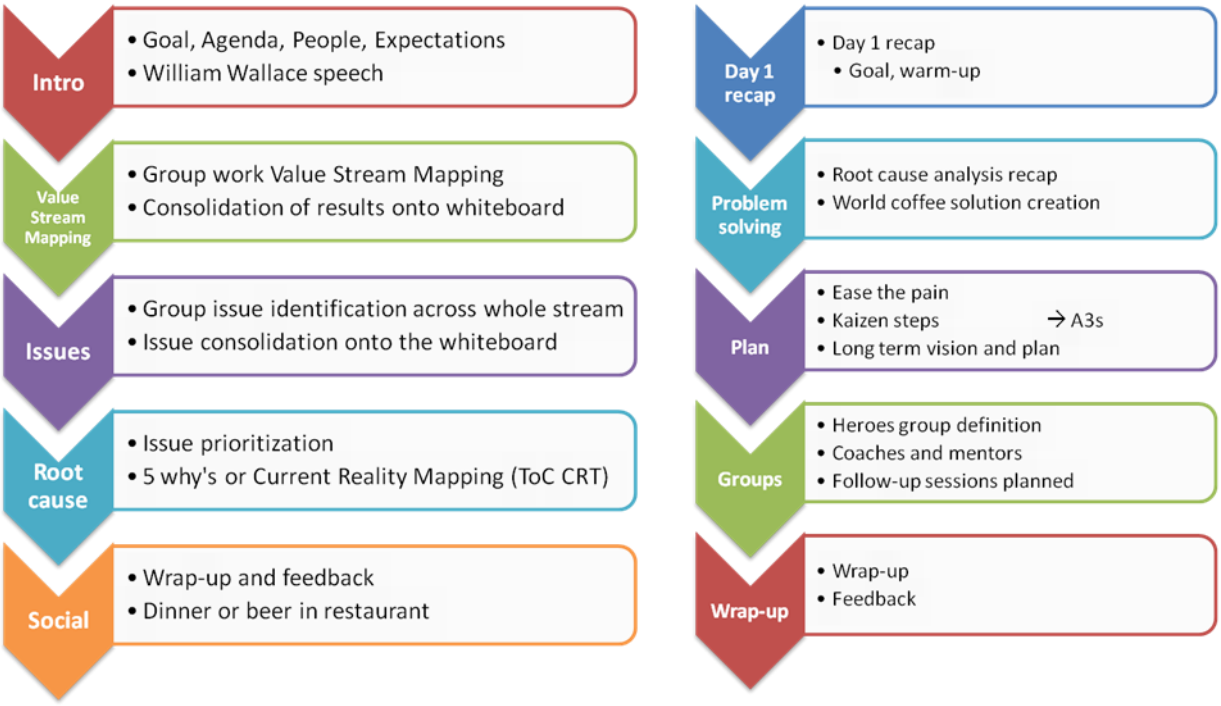


Figure 2: Kaizen workshop agenda.

One factor crucial to ensuring the success of a Kaizen workshop is to choose the right people. These are people who must be motivated for change, have decision-making power, play different roles (managers, technical people, developers, operations and testers) and offer different perspectives. We choose people from all locations involved in a project or service delivery in order to construct a full, big picture, and so that they may impact the entire end-to-end value chain. Ideally, that also means customer involvement, but if we are in really big trouble it may pay to sweep our mess up before exposing the customer to it. An efficient limit for a workshop is about 15 participants in total. Further communication of workshop achievements and actions among those not directly involved is the natural responsibility of the managers.

The first day of a workshop starts with a discussion and agreement on a common goal, and of what constitutes "real value" for the customer. Different stakeholders have different expectations, needs and interests, and we need to reach a consensus for a successful workshop. Without this, it risks becoming just another talking club with people digging their heels in on their positions.

After the common goal is established, we can continue with construction of the map of current reality for a selected area (e.g. the development process or problem management process). The outcome of this activity is a Value Stream Map (VSM), synchronization on current status and a list of perceived issues (see Figure 3).

1	ID	Issue	Area	PRI
2	13	Solution sharing to KB	Problem Mgt	8
3	11	Delays before work starts (in the beginning, before solution implementation - 60% of all problems)	Problem Mgt	5
4	10	Delays when L3 is outside Ostrava (project people)	Level 2	4
5	5	Worklog text/description clarity	Service desk	3
6	14	Reactive problem mngt only (no proactive steps)	Problem Mgt	3
7	4	L2/L3 does not solve incidents/SR quickly	Service desk	2

Figure 3: Value Stream Mapping and list of top perceived issues.

The next step is selection of two topics for further elaboration. This selection is important to the goals of staying focused and limiting work in progress – one of the key Lean principles. For these issues, teams investigate root causes using either the *Five Why*s or the *Fishbone* technique.

Example: 5 whys for issue *"Lot of different tasks (incidents, problems, projects, other) to handle"*

Missing appropriate prioritization of tasks

Priority assigned by team leaders (preference: incidents, projects)

Busy team leaders, not enough time to be with the team

Missing visibility (impact) of postponed problem solving

SLA measures forms the behavior of the teams (only incidents and service requests are measured); problem management and proactive proposals are not measured and proposals cause overhead of originator

The last step of the workshop is the definition of the ideal state and the related steps, called Kaizen steps leading towards that state. Kaizen steps are small actions that can help us achieve the ideal state. They can involve just a check of agreed attributes in the contract, creation of checklist(s), an update of the sales offering, or knowledge-sharing with the team. Usually such activities take from a few minutes to one full day at maximum. They should not be perceived as additional or extra work, because doing and improving the work is part of every position description. Some improvement actions will have already been identified during the discussions about the current flow. On top of that, we use the brainstorming or the world café technique to bring up additional possible improvements. Proposed Kaizen steps should address and improve uncovered root causes (Service Level Agreement (SLA) measures in our example). Respected actions designed to improve the situation are:

- Define needs for reports following trends across different teams. Have a few people committed to this activity in their respective teams.
- Implement defined trend measures in existing reports. Have one specialist managing reports committed to this activity.
- Define the *problem manager* role – discussion with HR.
- Define measures for problem management as well.

To summarize the Kaizen workshop's key points:

- It focuses on synchronization of people, uncovering common goals, and finding possible long term (ideally) as well as short term solutions (so called Kaizen steps).
- It focuses on the daily problems being experienced by participants, not on tackling high level management problems (***common goal***).
- It focuses on a few issues, but solvable ones. We regularly prioritize the list of issues to stay focused on only the few most important ones. The goal is to really implement some solution. The philosophy is that it is better to successfully implement something with a smaller impact than to attempt to tackle many big things with bigger potential results, the solutions for which will likely still be in progress by the end of the workshop (***limited work in progress***).
- Participants design small achievable steps (solutions) that can be applied today with not too much effort (***Kaizen steps***). No general solutions are proposed (you cannot ensure world peace in one week!), but pretty complex changes can happen when implementing incrementally small refinements (e.g., contracting, the way services are delivered, measurement mechanisms).
- Regular face to face as well as offline meetings for information sharing, and presentation of achieved steps and results, are key activities to get participants and teams engaged, and to successfully implement sustainable change.

Two of the commonly underestimated dynamics in software development and service delivery are social relationships and human nature. People need to meet physically in order to create personal relationships. We need to see faces behind names in order to build trust (it is often called "overcoming the layers of resistance"). Smaller groups are needed in the beginning to break the ice, set up a work hierarchy, and overcome natural human shyness.

Also, do not overlook the importance of social events as communication openers. We strongly recommend a joint dinner before or during the workshop; this is where groups become the teams and when open communication begins. You can find more about human nature in IT in one of my presentations, www.slideshare.net/xprochaz/human-it.

Conclusions

To summarize our lessons with coaching in a distributed environment, I would name following three points:

First, coaching is about coaching *people*. People and their needs should always come first, with the system and its limitations second. Although people see it as logical to perform this or that activity, they will not do it if it doesn't solve their issues. Soft skills are crucial; be a role model and practice what you preach. We use Agile and Lean principles and practices to support Agile and Lean implementations.

Second, our proven methods to overcome distribution challenges consist of following:

- Motivate people to ***pull*** (push is not sustainable even if you achieve some improvements… *really!*).
- Face-to-face intensive sharing kick-off or Kaizen workshops.
- Focus on built-in continuous improvement (driven by the teams, not by Agile/Lean coaches).

Third, remember our story and mindset shift. Actions need to correspond with people's mindsets, needs and environmental readiness; otherwise you push them into change that can overwhelm in an unsustainable environment. ■

About Jaroslav

Jaroslav Prochazka works as an Agile/Lean coach at **Tieto** Corporation and has been in IT for the past 11 years, starting as Java developer. He has more than five years of experience coaching and mentoring in distributed environments: coaching development, support and maintenance teams inside and outside Tieto. He has worked with about 400 people and has trained more than 800 to date.

Jaroslav earned his PhD at the University of Ostrava in 2007 and now teaches software development and information systems there. He speaks at international conferences like IBM RSDC 2009, Information Systems Development 2010, and the International Conference on Global Software Engineering 2011.

Jaroslav is an author of the recently published book, *Operate IT Differently: Agile and Lean Support and Maintenance of IT Services and Information Systems* (in Czech) and runs the blog *www.differ.cz*, which features articles, eBooks and templates in both Czech and English.

You can contact author at jaroslav.prochazka@tieto.com

Tieto *is a European Service provider delivering IT services and products to customers worldwide with the key focus on the Scandinavian market. We serve customers in various industry domains, namely telecommunication, banking and insurance, public or retail. The Tieto headcount is currently around 18,000 employees, with more than 5000 in near and offshore delivery centres in Middle and Eastern Europe and Asia. Our team, supporting important deliveries in Tieto, consists of 16 Agile and Lean coaches seated in the Czech Republic, Sweden and Finland. We help projects and services on a daily basis to remove obstacles, and to improve and start learning cycles. Our vision is to increase teams' productivity and remove delivery waste, and also to set up proactive cooperation with customers. We strive to change the way people think as well as their attitudes, while respecting their daily issues, limitations and human nature. Our services are Agile and Lean training sessions, conducted for various roles, as well as hands-on implementation support primarily through coaching. We regularly present our achievements and working methods at international conferences (e.g. XP2010, ICGSE2011, Lean IT Summit).*

Continuous Training for Test Teams

Engaging the whole team or organization in training will reap additional benefits.

By Michael Hackett, LogiGear

In this article, I share some of my experiences and observations on training teams, mainly in corporate settings. The operative word here is "team", not "individual". When training teams or groups in an organization, many of the considerations and benefits are different than those for the individual. We'll examine those differences and I will share successful solutions.

One must recognize that the needs of knowledge workers are different than those of traditional workers. Management guru Peter Drucker first used the phrase *knowledge worker* to describe "one who works primarily with information, or one who develops and uses knowledge in the workplace". To suggest how one might attain greater job satisfaction and better work effectiveness, Drucker lists four essential conditions: challenge, belief in a mission, results, and *continuous training*.

On that last point, Drucker isn't alone. The best modern management and leadership models emphasize the importance of continuous training. But is that indeed what is taking place in most professional workplaces today?

Continuous Training

Continuous training sounds big, and it is. Indeed, it can take many forms. It can come from a variety of sources – a test lead, a scrum master, an internal or outside trainer, a book, a video. It could be – among many things – technical, communication, process or tool training. As such, I'll only focus on delivery of hard and/or soft testing skills, leading and team content, and exclude tool and HR training (such as workplace safety, sexual harassment, diversity, etc.), since their goals and delivery are different from those of software testing content.

Thomas Davenport, professor of information technology and management at Babson College in Wellesley, Massachusetts, famously coined the term HSPALTA: *Hire smart people and leave them alone.* This indeed seems to be the common wisdom by which most knowledge workers are managed today. Hiring the "best and the brightest" is often critical to the success of an organization or project.

But leave them alone? Everyone needs training – even smart people! A laissez faire, "hands-off" mindset can quickly become a barrier to innovation and optimization. Yes, companies may indeed pull out all the stops to recruit and retain the most intelligent, skilled employees. But in our business, teams regularly make demands to do more with less, to work more efficiently, test new technologies, work with new tools, automate more of our workload, work with outside vendors and offshore teams... and the list goes on. Leaving staff members alone to identify their own training needs and acquire the required knowledge or information is just not good management. Only through proper, regular, well-organized, coordinated training can teams gain the foundation and skills they need and stay current with the know-how required to accomplish the personal, team and project goals.

Everyone wants to feel confident in their jobs, acquire the skills to do the job correctly and efficiently, minimize stress, communicate their work and, if at all possible, enjoy doing it all! When you hire people – *smart or otherwise* – and "leave them alone", you may very well be putting those goals further out of reach. Change happens at an overwhelming pace in our field. New devices are introduced, technologies evolve, development platforms change, tools are developed and quickly updated. Keeping up, even within a narrow slice of the tech world, is difficult and requires continuous training.

But how can a test engineer keep on top of the onslaught of new and ever changing information? The answer: continuous training! Sadly, though, I encounter so many teams who say they have no exposure to training, to new tools, and to new ideas, and are just too swamped with work to have the time to go searching on their own.

How can leads, managers, directors, and executives maintain confidence in the quality of testing when teams feel forced to engage in the same old practices month after month, year after year, because it's the only thing they know?

Inadequate Training

In the wide field of software development, there is one problem that seems to be unique to software testing and

A Training Glossary

Teaching
A formal process of bringing about awareness, conferring knowledge and instilling skills. Teaching focuses on knowledge.

Training
Prepares an individual or group to execute a skill. The focus of training is skill development.

Coaching
The process of aiding in the improvement and application of knowledge. Coaching involves assisting individuals in refining their skills and learning through practice.

Consulting
A consultant is one who is employed or involved in giving professional advice to the public or to those practicing the profession.

Live Delivery
In-person instruction wherein the instructor interacts with students/trainees in a live environment such as a classroom.

Online Training
Instruction in a learning environment where teacher and student are separated by space. The teacher provides course content through course management applications, multimedia resources, the Internet, videoconferencing, etc.

Instructor-led Training
Can be either online live or in-classroom live. Does *not mean* watching a video, viewing Flash content or reading.

Hybrid Delivery
Involves a combination of delivery methods - online and standup, live classroom instructor delivery. May include video, books or online instruction with group projects. Hybrid is a combination of delivery methods.

quality assurance.

In the course of working in a large variety of companies, what I have found to be a frequent issue is that many test teams do not have a common vocabulary. Often, moreover, they do not fully understand established common practices, or how to communicate them. Education is a big factor here. For teams in other fields, the opportunities for training and education tend to be more readily available. Teams engaged in software engineering, marketing, and engineering project management are usually composed of members with university undergraduate or even advanced degrees in their respective fields. It is much less common, however, for testers to have gone through a program of study focused on software quality and testing. There are some programs available, but they are rare.

Measuring coverage, for example, is a favorite problem area for me. Some teams can give you a number representing "coverage", but are hard pressed to tell you what it means, what it does *not* represent, what its limitations are, or alternative measures – or, for that matter, even if measuring testing is effective at all! Some testers can do a fine job of testing without understanding some quality basics or the reasons why we test. But without this knowledge, intelligently communicating risk, coverage, product stability, etc., to management is severely compromised.

Certification is not the answer for teams. Some people pursue certification only with the goal of getting a better job or making more money. Others get certified only because their companies pay for it. There are some major pluses to certification: one huge advantage of an entire team being certified is the idea of the whole team working from a common reference. And yet, I have never met a person who pursued a software testing certification for the purpose of gaining new skills; more often than not, the driving force was that piece of paper, and the potential job or salary benefits it promised.

Books can be a great source of information, ideas, and new approaches – *mainly for the individual.* Reading and circulating white papers to your group is certainly easy and beneficial.

But these solve skill problems for the individual – not for the team nor across the company. It takes a great deal of commitment and even "police work", to have a team read a book and have "book club" discussions about how to apply the lessons read.

Online Training

Online training is theoretically a good solution. It can be done at the individual's convenience, incurs no travel concerns, and online courses are usually cheaper than live delivery. Problems include: team discussions are not easy, participation is often problematic and retention is often low.

My involvement in online classes has not been positive. I have been involved in a few hybrid programs – hybrid programs being partial online delivery, partial live delivery of content. Three situations quickly come to mind. These same problems have been repeated by companies who have hastily decided that online is the way to go, without realizing the common drawbacks. In the first situation, most students had not done, or not completed, the online parts of the training before the live delivery. Those that had completed the work showed nowhere near adequate preparation or retention of the material needed to build upon in the classroom. In the second, the worst case example, only 2 out of 20 participants had actually completed the online training, which, not incidentally, had been quite expensive for the organization to develop. I am aware of too many organizations that have tried this - with the result being that only a small number of participants even "looked at" the online component. Retention was typically abysmal.

In the third case, another organization had "policing" in place to ensure that the online videos, flash content and reading were actually completed by participants. This hybrid program included homework and tests. Students were not at all happy about that. The overall impact on the team dynamic was a negative one and there were bad feelings all around. The result was that it became even more difficult for the organization to implement the training, tool and skill changes that were needed, due to the added complication of having to overcome ill will.

For all these efforts I have witnessed, none of the variety of online content

A Training Glossary pt.2

Self-paced
Learning at your own pace and convenience. Not done in a group. For example, reading a book and or watching a video. Where one can pause, review or begin again.

Skill Training
Teaching an individual how to perform the operations of a particular occupation; distinguished from personal adjustment training, work adjustment, and the acquisition of basic employment skills.

Hard Skills
These skills are based on facts, are universal and can be learned from books. These tend to be process, technical, measurement and scientific skills.

Soft Skills
A set of skills that influence how we interact with each other. It includes such abilities as effective communication, creativity, analytical thinking, diplomacy, flexibility, change-readiness, problem solving, leadership, team building, and listening skills.

Performance Improvement
Performance improvement is the concept of measuring the output of a particular process or procedure, then modifying the process or procedure to increase the output, increase efficiency, or increase the effectiveness of the process or procedure.

Team Building
The process of influencing a group of diverse individuals, each with his or her own goals, needs, and perspectives, to work together effectively for the good of the project. Such that their team ultimately accomplishes more than the sum of their individual efforts could otherwise have achieved.

delivery of test skill, tool and team soft skill training has been successful.

Other Sources of Training

Taking classes at a university or university continuing education program is great. Having designed and taught in these programs, I know that they have tremendous value. They can be useful to teams when employees are regularly sent through these classes. New employees are sent so that teams can work from a common reference, set of practices, foundation and glossary. This takes a great deal of commitment from management to stay vigilant in training. This may seem similar to certification, but it rarely involves as large a number of hours, high cost of certification and irrelevant courses, which often are of no benefit to one's specific work.

An important part of training teams in a large corporate setting is the conversation that takes place with learning new skills, methods, and communication tools. I have found this to be key to real change and the application of knowledge. But note that a serious commitment to both attending and paying attention in class is required, in order that it may be followed up by productive face-to-face explanations and discussions of ideas, practices and methodologies.

Foundation of Practice and Common Practices

It is important to have people on a team work from a common knowledge base, where teams can trust each other when their systems are integrated. People come into testing from very diverse backgrounds, be they subject matter experts, programmers, technical support or help desk personnel, or people hired because of their unique language abilities. How can you convey to people with very different backgrounds the technical skill sets, the communication skills, and the language and vocabulary of the job? Some teams try mentoring – and this can indeed work. But it takes the right personality and commitment. And in crunch times, this can be difficult.

When there are no universally agreed upon definitions of commonly-used word in software testing - for example, regression, acceptance or smoke - communication of our work is compromised. Most people wouldn't know what common or standard definitions are, or even where to find them. (You can go to www.IEEE.org and subscribe to their extensive glossary, and the Software Engineering Institute at Carnegie-Mellon (www.sei.cmu.edu)).

Having good training across teams in the organization can get everyone on the same page with complete and effective bug reports, meaningful metrics and fluid communication. The vocabulary they use to describe coverage and risk have a very powerful impact on the management of software development projects. This results in better testing, understanding, risk analysis and a better product.

Other Unintended Benefits

There is also a team building aspect to training that I've seen bear fruit on many occasions. When team members share problems, some often glean solutions from others venting about similar situations. There are additional benefits in having people who test, but who do not normally meet each other, share information and knowledge about test cases, tool usage, new tools, and new uses for tools developed in-house for particular applications. These programs can also provide forums for discussing project politics, how best to manage difficult situations, system problems, system limitations, and possible solutions. Investment in team training also leads to better team retention.

Another unintended benefit to team training is that it brings about the discussion of "educating up", since not everyone involved in software projects has a great understanding of why and how people test. I often hear: "This was fine – but my manager is the one who really needed to be here!"

Corporate Curricula

Developing corporate curricula has become increasingly popular, particularly with more software development work being distributed to various locations. Corporate curricula can ensure that each individual has a foundation of core competencies and essential skills (both hard and soft), uses a common vocabulary, uses test tools in the most effective way, understands development team goals or has a better awareness of users and their needs. Corporate curricula are most often a series of classes, each focusing on one topic developed for a specific need of the organization. They are *not* cookie-cutter, off-the-shelf classes. Chunks of content may be off-the-shelf, but they are combined in a way unique to your organization. I have been involved in developing and delivering a number of these programs and they have been very effective.

Corporate software testing curricula often includes - training across the suite of application lifecycle management (ALM) tools, quality basics, reporting, metrics and a common set of practices specific to the organization, delivered with the result being more confidence in the test effort across the company.

Brown Bag Lunches

Many teams do not share information across the organization. Also, you may be surprised at how little training actually takes place for test teams. I know many training coordi-

nators who say training budgets are often not fully utilized. Taking the time to do training takes commitment and time away from one's work. But there is at least one good alternative to missing entire days of work due to training: Brown bag lunches.

A brown bag lunch involves all participants bringing their lunches (the "brown bags") into a meeting room for approximately 1 hour of training or discussion focused on one topic. Such sessions can be a great way to conduct team training in your office. Training can be on a regular basis, (most commonly, as the name implies during lunch). I have conducted many series of brown bag lunch training sessions for Silicon Valley companies in one-hour segments, typically over the course of five or six months. We've almost invariably had an incredible amount of discussion and gotten significant amounts of training done - all without pulling people away from their projects.

This was convenient since I am in Silicon Valley. Of course, not everyone has access to good, local, competent, software testing trainers. But it does not have to be an outside company doing the training. With a few dedicated people, responsibility for topics can be divided up, and content development time set aside. Good lunchtime discussions can result in some truly great training and, moreover, the development of in-house experts.

Cross-team Round Table

In large organizations, it is often a sad reality that groups who test rarely spend time with each other. Server test teams, security test teams, localization teams, performance testing teams, functionality testers, etc., rarely cross each other's paths. And yet these teams very often face similar problems. Moreover, they generally use the same bug tracking, automation, test and project management tools, work under the same schedule constraints, and face the same political or communication problems. Test teams having round-table forums or discussions can be a great way to share information, increase tool usage, benefit from others' expertise and problem solve across the organization. They can also be a valuable and unique form of training and cross-training.

Training Offshore Teams

Training offshore teams is absolutely crucial to successful distributed software development. Training is the main solution to virtually every problem people experience with offshore, outsourced and outsourced-offshore teams. From cross-cultural communication to tool training, the amount of training for offshore teams must be significantly greater than that of a home team. If for no other reason than the lack of easy knowledge transfer across local teams, this is problematic to remote teams.

Software professionals in Western Europe, Japan and the US have been testing software for a long time. Over the decades, communities of testers have grown up with various skill sets and with varying degrees of exposure to training courses, books, and software testing conferences. But in many countries outside these regions, the situation is drastically different.

The number of languages in which contemporary books about testing is published is quite limited. Having conferences and access to training courses even in some of the current low cost centers (LCCs) popular for development these days is rare and new.

This often leads to people hired into testing lacking skills, and having limited or no access to the right skills to effectively test. These people have technical training, often having received software development training, but are not skilled enough to be developers, so they are hired as testers with no training specific to testing. Without good knowledge of test case design, test data design and reporting, there is no way that an effective test effort can be executed.

The most common experience I have when I go to an LCC to train teams on software testing is to be confronted by a profound variety of backgrounds (more diverse than you find in American development organizations). The level of understanding of technology and of business communication skills varies greatly. In my experience, the investment in training for offshore teams is directly proportional to the odds of a project's success. That is, teams that do not get trained adequately are significantly more likely to fail. Training offshore teams is such a broad topic that it's difficult to talk about it in a brief way. Training in such subject matter as technology, tools, process, platforms, SDLC, and corporate culture are essential, not optional, and this list is really only the beginning of the list of topics that offshore teams should receive training in.

For a full discussion on Training Offshore Teams see:

http://www.logigear.com/2006/294-building-a-successful-global-training-program-for-software-testing.html

Summary

What is most important when managing knowledge workers is *not* making the mistake to "*hire smart people and leave them alone*". Peter Drucker identifies continuous training as one of the four essentials of managing knowledge workers. This continuous training can take many forms, but be careful to chose wisely. In multi-product companies or a setting where work is distributed among numerous project teams, classroom delivery of cross-team corporate curricula is very effective. Also, be sure to devote extra training effort

Training Survey

__Michael Hackett__ discusses the results of the seventh installment of the Global Surveys focusing on common training in software testing.

Since people get hired into testing from diverse backgrounds, possessing unique skill sets, training is always necessary for proper execution on a test project. The degree to which people get trained in these skill sets varies. This section of the Global Testing survey deals with QA/Test training.

1. Have you ever taken a testing methodology or QA process course?

Yes	67.9%	72
No	32.1%	34

Analysis: 32% or respondents answering "no, I have never taken a class" is too high. To have close to a third of any job skill never to have had training in that area is absurdly high. I fear the number in the general testing population is probably much higher than 32%. In this survey, the respondents reached are those who are actively seeking additional information through education resources, magazines and other outlets. Corporations need to realize - for so many reasons - the response for this question must be "yes", 100%.

2. What skills are missing in the group? (you may select multiple answers)

QA/testing skill	31.6%	31
Coding/tool building	41.8%	41
Subject matter/domain knowledge	32.7%	32
Technical understanding of the code, system, platform, environment	34.7%	34
Automation skill	52%	51
Test tool (ALM, test case manager, bug tracking, source control, continuous integration tool)	22.4%	22

Analysis: These numbers speak for themselves. There are a lot of people who test that self-describe as lacking some skills to do their job. The accompanying lack of confidence will also contribute to other problems such as job satisfaction, morale, confidence about their work, ability to defend their work, among others.

Especially problematic is the people who feel they do not have the automation skills to do their job.

As a trainer, consultant and software developer, I see this as a major problem in our business. While many companies have testers automate their own tests, others leave the job of automating to specially skilled test automation engineers. These test automation engineers must have automation skills ready-to-go. Conversely, the current movement in testing is toward action-based or keyword-driven test automation. In action-based testing, an automation test engineer can build and support an automation framework where "subject matter expert" black box testers can utilize their technical expertise to easily get their tests automated. In TestArchitect, Fitnesse, etc., anyone who tests can have tests automated that use action or functions coded by someone else. Regardless of who automates, extensive knowledge of test automations is the only way to stay competitive in the software development world today. That 52% of surveyed testers feel they are lacking test automation skills clearly defines a need for more training in this most important area.

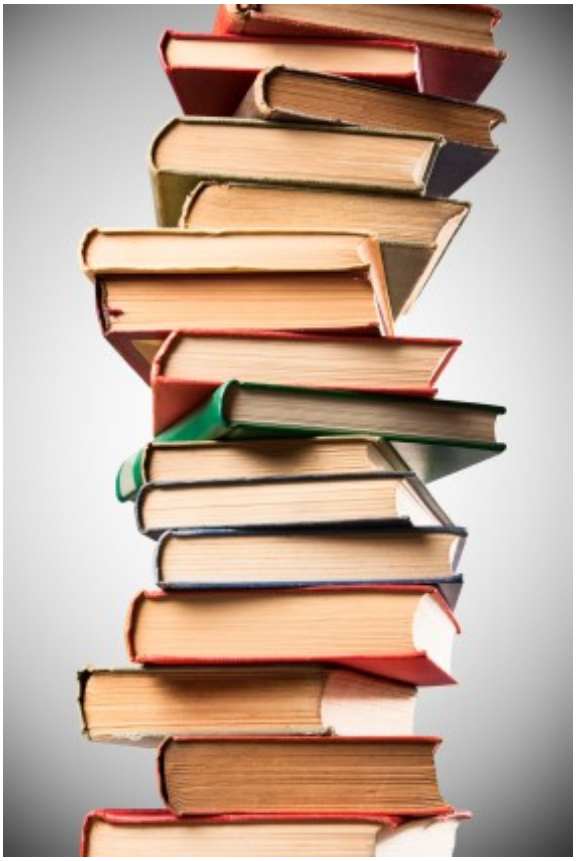
3. What types of training have you and/or your group taken within the last two years? Sources varied, including:

In-house
CSQE
ISTQB
University of California Extension
LogiGear Corp.
HP QC (tool training)
Cisco and technology specific trainings
Conference participation

4. How do you get job skills? (you may select multiple answers)

Read books	37.9%	39
Take a course	42.7%	44
"Brown bag" lunch, internal team training sessions, internal knowledge transfer	57.3%	59
Internet/online course	63.1%	65
I don't have an opportunity to increase my job skills	1%	1

Other answers included: Co-workers, read blogs, tech/tool trainings.



5. If you answered "books" in the above questions, please tell us what books you have read to get your job skills?

Of the 30 people listing books, three books to get more than one response:

Testing Computer Software (Kaner, Nguyen, Faulk)
Lessons Learned in Software Testing (Kaner, Bach, Pettichord)
The Art of Software Testing (Glenford Meyers)

6. If you took a course was it paid for by:

You, personally	24.5%	23
Your company	75.5%	71

7. Are you currently certified?
Note: of the 142 people completing the survey, 26 indicated they are certified. That is 18%

Yes	24.5%	26
No	67%	71

8. Yes, I was certified by:
Note: of the 142 people completing the survey, 26 indicated they are certified. That is 18%. Of the 18% of survey respondents who are certified:

ISTQB (CTFL)	66.7%	20
CSTE/ CSQA	23.3%	7
CSQE	0	0
CQA	3.3%	1
CSTP	6.7%	2
PMP (PMI)	6.7%	2
Six Sigma Black Belt	3.3%	1

9. Do you feel you have all the skills you need to do a successful job testing?

Yes	73.6%	78
No	26.4%	28

Analysis: These numbers speak for themselves. That a quarter of testers feel they are not prepared for their job is an industry problem. This group needs to be helped along the way and directed to the many training solutions available today! ■

VIET NAM SCOPE

Viet Nam's 'Golden Age'

Viet Nam is entering its economic 'Golden Age.' Will the final picture resemble Japan or Thailand?

By Brian Letwin

Viet Nam is entering what the United Nations Population Fund (UNFPA) calls its 'Golden Age', one characterized by an abundant labor force and low social welfare costs. It's a rare moment, an opportunity wherein a single generation is poised to shape the future of its country. Other Asian neighbors, such as Japan and South Korea, seized this opportunity and experienced exceptionally high growth rates and rapid industrialization during the second half of the 21st century. Thailand and Malaysia, on the other hand, having been graced with the same opportunities, were unable to fully realize their potentials and found themselves climbing the economic ladder awkwardly, only to stop somewhere in the middle. The contrast is evident today when comparing the downtowns of Tokyo and Bangkok.

Where in this spectrum Viet Nam will fall ultimately depends on number of factors, such as: responsible urban development, environmental preservation, adaptability to a changing climate and re-focusing of the economy from agriculture and manufacturing to high-skilled industries such as technology R&D and IT. As Ho Chi Minh City is the financial and commercial hub of Viet Nam, it must lead the way for these changes and serve as an example to the rest of the country.

There is a hopeful feeling in the city. When walking around in the afternoon, one is sometimes inundated by masses of just-dismissed school children running to their parents waiting on motorbikes. Now, as in the West, parents are focused on providing valuable education for their children with the hopes that, one day, they'll have the skills to compete in the international labor market. This is now becoming a reality as many international companies set up shop in Ho Chi Minh City, staffed by young Vietnamese.

Along with personal development, the benefits of a growing economy are evident in the city's skyline. Corporate towers are sprouting up like bamboo shoots in the city center, luxury apartments are rising in every direction, entire hi-income neighborhoods are being built from scratch. The first line of the city's subway system is currently being constructed with the hope that once complete, debilitating motorbike traffic will abate, much as was the experience of Taipei.

It feels like a mix of a rapidly developing turn-of-the-20th century New York and American idealism and consumerism wrapped in one package. Lively produce markets buzz in the shadows of office towers; cart vendors line streets offering everything from grilled corn to lottery tickets. Women balance entire food stalls on their shoulders with wood beams and straw baskets. Many sing in melodic voices to trumpet their wares. Smells, sounds and colors are everywhere, to be seen and taken in by everyone.

Generally, people seem happy. Absent is a sense of pessimism, something so common today in many other cities. This may be due to the positive nature of Buddhist faith (to which more than 90% of the country claims) or the fact that the economy is the best it's ever been. Most people now own motorbikes, food is abundant (in contrast to 30 years ago when there was strict rationing) and karaoke machines and flat panel TVs are common even in working class households. Consumerism is rising, as

evidenced by the increasing number of cars (despite a 100% import tax), designer clothes, and smart phones, all of which are used to trumpet their owners' status. It's the Vietnamese version of the 'American Dream'.

But this dream is mostly limited to urban dwellers and has not yet trickled out of the cities and into the countryside. Underfunded schools, and lack of basic infrastructure projects such as roads and water access, contribute to chronic underdevelopment. At some point, Vietnam will hit a critical mass wherein the success of the urban and financial centers depends on rural areas. Much of Vietnam's economy relies on these areas, based on the many products they produce - Vietnam is a top-3 exporter of both coffee and rice - and on the country's ability to bring these products to market. With a dilapidated infrastructure, along with old business practices, the current process is far from efficient.

Finally, and perhaps most important, there are the twin issues of the environment and urban development. Vietnam is one of the five countries forecast to be hardest hit by climate change in the coming decades and receives UN aid to mitigate against the potential devastation. Salinity levels in rice paddies are rising, shortening the growing season. If the future brings a one meter rise in water levels, 11% of the population (nearly 10 million people) will be directly affected. In addition to addressing this threat, more must be done about mass polluting of rivers, illegal logging, and haphazard urban development.

All of these issues seem to have been getting more attention from the government and press in the last few years. The hope is that this attention turns into effective action. One good example of a move in the right direction was the implementation of low-emission buses in Ho Chi Minh City and Hanoi.

The pieces are there. The question is: can Vietnam fit them all together to complete the picture it envisions for itself — Tokyo rather than Bangkok? In any case, it's certainly exciting to watch. ■

LOGIGEAR MAGAZINE
FEBRUARY 2012 | VOL VI | ISSUE 1

United States
2015 Pioneer Ct., Suite B
San Mateo, CA 94403
Tel +1 650 572 1400
Fax +1 650 572 2822

Viet Nam, Da Nang
7th floor, Dana Book Building
76-78 Bach Dang
Hai Chau District
Tel: +84 511 3655 333
Fax: +84 511 3655 336

Viet Nam, Ho Chi Minh City
1A Phan Xich Long, Ward 2
Phu Nhuan District
Tel +84 8 3995 4072
Fax +84 8 3995 4076