

November/December 2010

\$9.95

www.StickyMinds.com

BETTER SOFTWARE

The Print Companion to **STICKYMINDS.COM™**

TESTING GONE WILD
Regulated software
unleashed

THE OPTIMIST'S DILEMMA
Looking on the
(not-so-bright) side



**2010 SALARY SURVEY
RESULTS INSIDE!**

**“We need to make sure that
our company's most important
business gets done every
two weeks.”**

Erik Huddleston

**CTO
Inovis**



“Our business is dependent on tight turnaround times. **Rally makes it easy to juggle scope, hours and resources, so we can share up-to-the-minute project information with all of our stakeholders.** Everyone can see priorities, status and roadblocks to save time and money.”

See how development organizations like Erik's are delivering software the Agile way: www.rallydev.com/videos/

HOW HEAVY IS YOUR CLOUD?

Find out with **QA Wizard Pro**, now with **Load Testing**.

Will your cloud or web application be able to handle the load once it goes live? Will it crash or return errors to your users? Will it slow down for all users, or just bog down during specific functions?

Find out if your web app can handle the real world before you launch with **QA Wizard Pro's** new load testing functionality.

Load testing with Seapine's **QA Wizard Pro** lets you identify where your bottlenecks occur—memory, CPU, network, or database—and what it takes to break your application.

Now one application meets all your automated testing needs—regression testing, functional testing, and load testing.

Download **QA Wizard Pro** at www.seapine.com/betterLT and try it today!



Try our Jump Start Package Today
(FIRST 10 USERS FREE)
Call 1.800.439.7782 or visit www.techexcel.com

TechExcel

Knowledge-Centric ALM for Today's Enterprises

DevSuite



**With DevSuite, you can deploy one module, a multi-module solution
or the complete fully integrated ALM solution**

- DevSpec – Requirements management
- DevPlan – Project planning and tracking
- DevTrack – Defect tracking and task management
- DevTest – Test management
- Agile Studio – Agile planning and implementation
- DevTest Studio – Complete quality management
- DevSuite – Complete Application Lifecycle Management

1.800.439.7782

www.techexcel.com



CONTENTS



18

ONLY
In the
Digital
Edition

2010

SALARY
SURVEY

in every issue

Mark Your Calendar **4**

Contributors **8**

Editor's Note **9**

Virtual Resource Shelf **14**

Digital Survey Results **15**

Product Announcements **31**

FAQ **34**

Ad Index **36**

Better Software magazine—The print companion to StickyMinds.com brings you the hands-on, knowledge-building information you need to run smarter projects and deliver better products that win in the marketplace and positively affect the bottom line. Subscribe today to get six issues.

Visit www.BetterSoftware.com
or call 800.450.7854.

features

18 COVER STORY AGILE LIGHT BULB MOMENTS

Many of us have our personal identities wrapped up in our jobs, which can make change hard, particularly in agile environments. Recognizing the power of storytelling, Michele Sliger started collecting first-person stories about how adopting agile affected individuals and what their "light bulb moment" was like. Find out how agile adoptions have changed individuals—their perceptions of agile, their leadership styles, and even their personal lives.

by Michele Sliger

22 BUSTED: 5 MYTHS OF TESTING REGULATED SOFTWARE

Testing regulated software is often seen as a tedious job that generates stacks of documentation and is subject to crippling rules. See five of these assumptions exposed as mere myths, and learn how regulated testers can use the same approaches, techniques, and tools at any other tester's disposal while still passing a process audit.

by John McConda

26 AN INTRODUCTION TO SCALA

Scala is a programming language that blends functional and object-oriented language features. Scala programs run on the Java Virtual Machine and can easily interact with Java code. Learn how Scala can yield concise, safe, and compatible code and how you can start learning Scala on your own.

by Daniel Wellman

columns

11 TECHNICALLY SPEAKING THE OPTIMISTS DON'T MAKE IT OUT • *by Lee Copeland*

Optimism is normally viewed as a positive trait, but not when it comes to goals and estimates. Project managers who don their rose-colored glasses when faced with the harsh light of reality are setting themselves up for disappointment.

12 INSIDE ANALYSIS SOMETIMES PERCEPTION IS THE PROBLEM • *by Payson Hall*

High on a mountain twenty years ago, a wise man shared secrets of problem solving that have served Payson Hall ever since. In this article, Payson passes along a simple definition that offers insights into problems and potential solutions.

35 CAREER DEVELOPMENT WHICH OBSTACLE SHOULD YOU TACKLE TODAY? • *by Johanna Rothman*

As a lead and manager, your job to remove obstacles that impede work is most important. But of all the obstacles you find, whether they be people's perceptions, bottlenecks in the work flow, or an ill-fitted chair or desk, which do you tackle first?

MARK YOUR CALENDAR



software tester certification

www.sqetraining.com/certification

training weeks

www.sqetraining.com/public

Agile Software Development Week

November 14–16, 2010

Orlando, FL

March 14–18, 2011

San Diego, CA

April 11–15, 2011

Boston, MA

May 16–20, 2011

Chicago, IL

November 30–

December 2, 2010

Phoenix, AZ

February 15–17, 2011

Toronto, ON

February 22–24, 2011

Houston, TX

February 22–24, 2011

Seattle, WA

conferences



Agile Development Practices East

www.sqe.com/adpeast

November 14–19, 2010

The Rosen Centre
Orlando, FL

Agile Development Practices West

www.sqe.com/adpwest

June 5–10, 2011

Caesars Palace
Las Vegas, NV

STAREAST 2011 Software Testing Analysis & Review

www.sqe.com/stareast

May 1–6, 2011

Rosen Shingle Creek
Orlando, FL

STARWEST 2011 Software Testing Analysis & Review

www.sqe.com/starwest

October 2–7, 2011

Disneyland Hotel
Anaheim, CA

Better Software Conference

www.sqe.com/bsc

June 5–10, 2011

Caesars Palace
Las Vegas, NV

BETTER SOFTWARE

Publisher
Software Quality Engineering, Inc.

President/CEO

Drew Thoeni

Vice President of Publishing

Holly N. Bourquin

Editor in Chief

Heather Shanholtzer

Editorial

Managing Technical Editor

Lee Copeland

Online Editor

Francesca Matteu

Online Editor

Joseph McAllister

Production Coordinator

Cheryl M. Burke

Design

Creative Director

Catherine J. Clinger

Advertising

Director of Sales

Sonia Lavin

Customer Success Manager

April Evans

Circulation and Marketing

Circulation Coordinator

Jamie Green-Gago

Product Marketing Manager

Diara A. Sullivan

Marketing Coordinator

Stephanie Fender

A PUBLICATION OF
SOFTWARE QUALITY ENGINEERING



CONTACT US

Editors: editors@bettersoftware.com

Subscriber Services:

info@bettersoftware.com

Phone: 904.278.0524, 888.268.8770

Fax: 904.278.4380

Address:

Better Software magazine
Software Quality Engineering, Inc.
330 Corporate Way, Suite 300
Orange Park, FL 32073

Award-Winning Test Automation Tools

“Ranorex is best *Commercial Functional Automated Test Tool* for .NET and Flash/Flex applications.”
— 2010 *ATI Automation Honors Awards*





Object-based Capture & Replay Editor

- ✓ Maintainable recordings via the actions table editor
- ✓ Integration of Ranorex repositories



Automated Testing of Web & Windows Applications

- ✓ Winforms / C# / VB.NET
- ✓ WPF / Silverlight / Win32 / MFC
- ✓ Flash / Flex / Web 2.0 / AJAX /  / 



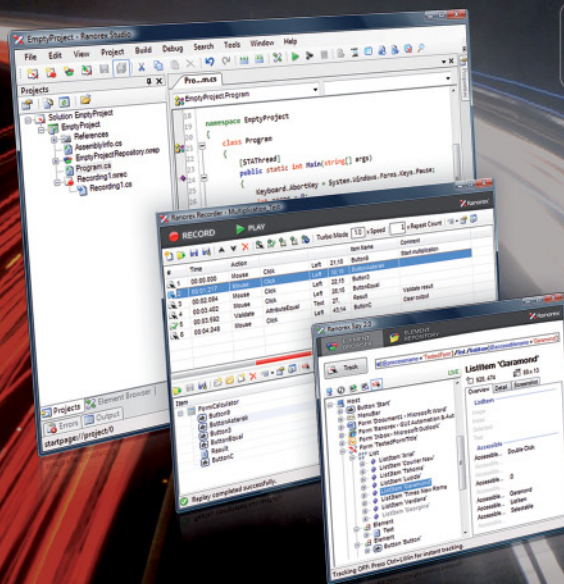
Maintainable UI Object Repositories

- ✓ Easy to maintain all types of UI objects
- ✓ Separate test automation from UI identification



Write tests in C#, VB.NET and IronPython

- ✓ Automatic and flexible code generation in C#, VB.NET and IronPython
- ✓ All-on-one test environment with code editor, code completion and debugging



Get your 30-day Trial
www.ranorex.com

Microsoft



Unreal Speed Made Real

Deliver solutions with incredible velocity.

Time-to-market is critical to driving business forward. Put your solutions in customers' hands faster using the total team toolset of Microsoft® Visual Studio® 2010.

VISUAL STUDIO 2010 SPEEDS DEVELOPMENT WITH TOOLS FOR:

- ▶ Real-time visibility into project quality and status
- ▶ Increased collaboration through a common interface
- ▶ Concurrent coding and debugging by incorporating test early
- ▶ Eradicating "no repro" issues with rich, actionable bugs
- ▶ Empowering manual testing and automating rote tasks
- ▶ Provisioning virtual labs for efficient test and build
- ▶ Eliminating waste with agile project management

Help eliminate waste and accelerate collaboration in your development and test processes. Visual Studio 2010 integrates test tools into the development environment, unites team workflow, and can help save time and money.

EXPLORE AND EXPERIENCE THE POWER.



Microsoft®

Visual Studio® 2010

Get proof at www.almcatalyst.com/test.

Buy at www.microsoft.com/visualstudio.



LEE COPELAND has more than thirty years of experience in the field of software development and testing. He has worked as a programmer, development director, process improvement leader, and consultant. Based on his experience, Lee has developed and taught a number of training courses focusing on software testing and development issues. He is the managing technical editor for *Better Software* magazine, a regular columnist for StickyMinds.com, and the author of *A Practitioner's Guide to Software Test Design*. Contact Lee at lcopeland@sqe.com.



PAYSON HALL is a consulting project manager for Catalysis Group, Inc. in Sacramento, California. Payson consults on project management issues and teaches project management. Email Payson at payson@catalysisgroup.com. Follow him on twitter.com/paysonhall.



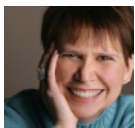
JOHN MCCONDA spent five years in the regulated world helping to create an FDA-validated testing process and cofounded WREST, the Workshop on Regulated Software Testing (www.wrestworkshop.com). A senior consultant for Moser Consulting, John currently leads a large testing effort for a federal government contractor, and gets his hands dirty with test automation, exploratory testing, and performance. John also enjoys teaching online courses for the Association for Software Testing.



DALE PERRY has been a programmer, analyst, database administrator, project manager, development manager, tester, and test manager. Dale's thirty years of project experience include large systems development and conversions, distributed systems, and online applications. He has been a professional instructor for more than fifteen years, including eleven years with Software Quality Engineering, and has presented at numerous industry conferences on development and testing.



Management consultant **JOHANNA ROTHMAN** is a regular StickyMinds.com and *Better Software* magazine columnist. She is the author of *Manage It! Your Guide to Modern Pragmatic Project Management*—winner of the 2008 Jolt Productivity Award—as well as coauthor of *Behind Closed Doors* and author of *Hiring the Best Knowledge Workers, Techies & Nerds*. Johanna is a host of the Amplifying Your Effectiveness Conference and has presented at numerous conferences. You can reach her at jr@rothman.com or by visiting www.jrothman.com.



As a self-described "bridge builder," **Michele Sliger's** passion lies in helping those in traditional software development environments cross the bridge to agility. She is the coauthor of the book *The Software Project Manager's Bridge to Agility*, which focuses on helping PMI-trained project managers make the transition. Michele is a Certified Scrum Trainer (CST) and a certified Project Management Professional (PMP). Michele can be reached at michele@sligerconsulting.com.



DANIEL WELLMAN is a technical lead at Cyrus Innovation (www.cyrusinnovation.com), a leading agile consultancy based in New York, where he leads development projects and coaches teams on adopting agile software development practices. Daniel has more than ten years of experience building software systems and is an expert in agile methodologies, object-oriented design, and test-driven development. Contact Daniel at dan@danielwellman.com.



A FIRST TIME FOR EVERYTHING

I became a first-time mom in July. While I thought I was prepared—I have a wonderful husband, I live in a safe neighborhood with good schools, I'm financially stable—the truth is I had no clue what we were in for. The first night in the hospital after the birth of our daughter, my husband and I looked at each other blankly and said, "What do we do now?"

It turns out there is a whole other level of tired reserved for new parents that I don't think anyone can prepare you for. I can't count how many times I was told to "sleep when the baby sleeps," but I was determined to go about my life doing daytime things during the day and sleeping at night like I always had. This, of course, resulted in my turning into a sleep-deprived lunatic.

My husband finally got through to me about two weeks in. One evening, as I tried to convince our daughter that it was time to go to bed, he turned to me and said (with apologies to Chuck Palahniuk), "The first rule of bedtime is there is no bedtime."

And it finally hit me: We were on baby's schedule now. Once I got that into my skull, life was much simpler and I was a lot less sleepy.

Life is full of these "aha" moments, and in this issue, Michele Sliger relates a few stories of when the proverbial light bulb went off for people who were transitioning to agile processes. After reading "Agile Light Bulb Moments," visit Michele's website to add your own agile-adoption tale. Michele hopes this compilation of real-world experiences will help others feel more confident as they begin their own agile adventure.

Also in this issue, Daniel Wellman offers up a brief introduction to Scala—a programming language that is designed to be concise, safe, and compatible. And, our very own mythbuster, John McConda, takes on five misconceptions we often encounter when tasked with testing regulated software.

As always, I hope you enjoy this issue of *Better Software* magazine. Email me to let me know how you've put this issue to work for you.

Happy Reading!

Heather Shanholtzer
hshanholtzer@sqa.com

STAREAST 2011

Crank it Up!

**STAR
EAST**

SOFTWARE TESTING

ANALYSIS & REVIEW

www.sqe.com/stareast

May 1–6, 2011

Orlando, Florida
Rosen Shingle Creek



**REGISTER BY MARCH 4, 2011
AND SAVE UP TO \$400.
GROUPS OF 2 SAVE EVEN MORE!**

**Choose from a full week of learning,
networking, and more**

SUNDAY

Multi-day training classes begin

MONDAY – TUESDAY

34 In-depth half- and full-day tutorials

WEDNESDAY – THURSDAY

5 Keynotes, 42 Concurrent Sessions, the EXPO,
Networking Events, Receptions, Bonus Sessions,
and more

FRIDAY

Testing & Quality Leadership Summit



Unreal Speed Made Real



Work together.
Win together.

Want flawless handoffs
and fast-paced finishes?

Integrate dev and test. Accelerate time-to-market.

Help increase speed *and* assure quality in your software development. **Microsoft® Visual Studio® 2010** revolutionizes the process of delivering business value with custom software development by fully integrating developers and testers in a single collaboration platform with shared tools.

VISUAL STUDIO 2010 IMPROVES PROCESSES: Software in customers' hands faster

Time-to-market is critical to driving business forward. Help eliminate waste and accelerate delivery with agile application lifecycle management enabled by the Visual Studio 2010 development system.

The integrated, comprehensive tools in Visual Studio 2010 are designed to enable more efficient development, better coordinated test, and end-to-end traceability throughout the application lifecycle. Testers and developers share a common, central repository for all tasks, code, and tests—and you can extend the collaboration tools across platforms to developers working with Java.

The centralized repository and built-in reporting enable those with oversight to access real-time reports at any time to monitor progress and help ensure business objectives are met. Visual Studio 2010 automates the delivery of status reports so you know that dev and test are working together to build quality into your software.

Microsoft built the Visual Studio 2010 family of products to help lower the risk from the process of delivering business value with custom software development. Get proof at www.almcatalyst.com/test.

"With Visual Studio 2010, our customers are able to identify and eliminate waste, and they can speed the delivery of real business value by 20 percent or more."

– **Steven Borg**, Principal Consultant and Co-Founder, Northwest Cadence

VISUAL STUDIO 2010 IMPROVES PROJECTS: More time for creativity in dev and test

Find, reproduce, and fix bugs faster with the advanced test tools in Visual Studio 2010.

Visual Studio 2010 brings dev and test into a common interface for improved communication and a unified experience. Testers don't need the full IDE to collaborate; **Visual Studio Test Professional 2010** equips testers with powerful new tools for automating tasks and for manual testing. Sharing results and reports across **Visual Studio Team Foundation Server 2010** is seamless.

Automating rote tasks frees up time for more creative manual testing. Testers can record clicks and fields to help developers reproduce bugs later; Fast Forward quickly brings developers to the point where test uncovered a failure. IntelliTrace™ streamlines the filing of rich, actionable bugs by recording the application's execution history.

Developers benefit, too, from time-saving build automation and built-in reporting tools in Visual Studio 2010. Uniting contributors—even across platforms—on the same foundation enables developers to focus more on the creative aspects of the job and less on "housekeeping." Visual Studio 2010 also has tools that enable fast, easy setup of virtual labs for test and build.

"We're seeing a 20 to 30 percent productivity gain with respect to quality assurance-related processes, such as testing and bug fixing."

– **Jorge Rodriguez Brizuela**, Development Supervisor, IT Finance, Flextronics

ACCELERATE AND EXCEL AS ONE TEAM.



Get proof at www.almcatalyst.com/test.

Buy at www.microsoft.com/visualstudio.

The Optimists Don't Make It Out

Goals describe the desired results of our actions. However, our goals may not become reality. Accept this now to save heartache later.

by Lee Copeland | lcopeland@sqa.com

There's only one advantage to delayed flights, missed connections, and extra nights stuck in hotels far away from home—you can catch up on your reading. The book at the top of my “to read” list was *Making it Big in Software* [1] by Sam Lightstone. It's a collection of interviews with various “biggies” in the software field mixed with good advice from Lightstone in the areas of education, getting a job, using your first work years wisely, gaining essential skills, career advancement, and others. All in all, it's an interesting read.

However, the interviews provided me with no usable advice. Some of the biggies' secrets of success were: join Microsoft in 1986; begin programming at age nine; start a successful company; become Google's twentieth employee; always work for a powerful, influential, and upwardly mobile boss; and have few friends and no family. In fact, when one interviewee was asked “How do you achieve a work-life balance?” he responded, “Why would I want to do that?” It seems that throughout my career, I've made all the wrong decisions to make it big.

In her interview, Diane Greene, cofounder and past CEO of VMware, explained the secret of the company's success. “We hoped for the best but planned for the worst, always.” That reminded me of an interview Jim Collins had with Vice Admiral James Stockdale as reported in his book *Good to Great* [2]. Stockdale was held in a North Vietnamese prison for seven years, beaten and tortured repeatedly, and later awarded the Medal of Honor. Collins asked Stockdale about his strategy for coping. Stockdale replied, “I never lost faith in the end of the story, I never doubted not only that I would get out, but also that I would prevail in the end and turn the experience into the defining event of my life.”

When Collins asked who didn't make it out of Vietnam, Stockdale immediately replied: “Oh, that's easy, the optimists. Oh, they were the ones who said, ‘We're going to be out by Christmas.’ And Christmas would come, and Christmas would go. Then they'd say, ‘We're going to be out by Easter.’ And Easter would come, and Easter would go. And then Thanksgiving, and then it would be Christmas again. And they died of a broken heart.” Stockdale added, “This is a very important lesson. You must never confuse faith that you will

prevail in the end—which you can never afford to lose—with the discipline to confront the most brutal facts of your current reality, whatever they might be.”

Are you one of those project managers who believes the project will be finished by Christmas, and if not then, by Easter, and if not then, by Thanksgiving? Have you been crushed again and again? If you continue down that path, Admiral Stockdale says that ultimately your heart and spirit will be broken.

In our work, it is vital that we do not confuse either goals or estimates with reality. Goals describe the desired end results of our actions—what we want to achieve. However, our goals may not necessarily become reality.

Estimates are an approximate calculation of the actions required to achieve goals—what we believe will need to be done. Our estimates are not necessarily reality, either. Reality is simply the way things actually are. It exists independently of our goals and estimates. Don't confuse them.

What could cause us to confuse goals or estimates with reality? Goals and estimates are often more positive than reality.

Ted Young, an agile development manager, believes it's fear—fear of disappointing people, fear of punishment, fear of failure, fear of looking foolish, fear of confronting others with the truth, fear of not being in control, and fear of our own inability. If our reality is full of fear, we substitute and believe positive goals.

Rick Scott, a Canadian philosopher-geek, generalizes this to “the positive consequences of believing something that can't possibly be true can be made to outweigh the consequences of believing something that is true.” In other words, it can be less painful to reject reality than to embrace it.

While never in the “Top Ten Attributes” of great software professionals, it seems that courage, the ability to overcome adversity in the face of fear, is vital to our effectiveness as software professionals. Remember: The optimists don't make it out; the courageous realists do. **{end}**

“In our work, it is vital that we do not confuse either goals or estimates with reality.”

Sticky Notes

For more on the following topic go to www.StickyMinds.com/bettersoftware.

■ References

Sometimes Perception Is the Problem

A favorite definition of “problem” helps Payson gather useful information about an issue before running off to solve a problem that may not exist.

by Payson Hall | payson@catalysisgroup.com

Anyone who has tried to retrieve understanding of a problem from another human’s brain knows the challenges and confusion involved. Twenty years ago, a hermit named Jerry shared a problem-solving secret that I now offer you through the lens of my experience.

“Problem” Defined

The most useful definition I’ve ever encountered was Gerald Weinberg and Don Gauss’ definition of the term “problem” in their book *Are Your Lights On?* [1] “Problem: A difference between things as desired and things as perceived.”

By this definition, there can be no problem unless *someone* has a desire. Who? What is his desire and how do we know that? How would we know if we satisfied the desire? How do we know it is not satisfied now? This gets to the other point, *perception*.

Perception is a powerful word because it reminds us that things aren’t always as they seem. How do we know the current state? What data confirms the current state? Whose perception of the current state are we relying upon?

This definition of “problem” helps quickly focus on gathering useful information about the nature of the issue before solving a problem that is ill defined or doesn’t exist. It also highlights options for solutions: change the desire, change the perception, and reduce or eliminate the difference between desire and perception.

This is one of the best tools in my kit, reminding me of richer options when I’m hunting for solutions.

Case Study

I was asked to help a public sector client with a “problem” of slow payment processing. First step: Identify who had the problem. The person willing to invest in solving the problem was the new head of administration for the department. She said, “Our vendors are complaining because we take too long to pay our bills.”

I asked how often complaints were received. She showed me a letter from a vendor detailing the payment history of

a specific invoice. She was new to the organization and had received no other vendor complaints.

My inquiry identified one source of her perception. If there were only one complaint, it might have been easy to mistake an incident for a trend. I took a copy of the letter and began my investigation.

When I met the nice people in accounting, they confirmed that prompt payment was a common issue and had been since the department was split a year earlier. When the department split, half of the staff had gone to the new department (my client) and half had remained with the old. Existing records had remained with the former department, making payment processing time consuming because old records weren’t readily available. It had taken months of overtime to copy and transfer the files. Payment timeliness had suffered as a consequence.

“How long does it typically take to pay an invoice now?” I asked.

“We don’t know.”

“Then how do you know that you haven’t caught up?”

“Because line managers keep calling with payment inquiries on behalf of grumpy vendors. We have dedicated several staff to doing nothing but respond to these inquiries.”

I followed the accounting manager on a tour of her facility. Mail was stacked everywhere. Staff worked frenetically. Phones were ringing. Staff apologized for payment delays.

I watched a worker process a stack of mail. He took an invoice, walked to a nearby filing cabinet, pulled a file, and riffled through the pages. He muttered, replaced the file, returned to his desk, and threw the invoice into the trash.

“What’s up with that one?” I asked, surprised.

“Duplicate,” he answered. “A third of these are redundant requests for payment. That invoice was approved for payment five weeks ago.”

I looked at the postmark on the envelope; it was two weeks old.

The rest of the investigation can be summarized as follows: There were two steps to the payment process—(1) Opening the mail and checking files to confirm the vendor was known

“Perception is a powerful word because it reminds us that things aren't always as they seem.”

to the system and funds had been budgeted; then sending the invoice to be approved by the line manager who had received the product or service, and (2) Processing confirmation from the line manager that the debt was authorized for payment, and paying the bill. Line managers often took thirty to ninety days to respond with approval.

Staffing was inadequate to perform these steps while also processing duplicate invoices and searching the stacks for work in progress to respond to internal inquiries. They had a bathtub problem—more work flowing in than they were able to process and drain out. It was making a mess and causing additional work.

I realized that even if things were processed instantly, payment would require thirty to ninety days because of the delay waiting for line manager approval. I partitioned the problem into three parts: processing time within the accounting department, processing time with the line managers, and the time required to cut checks.

Processing time within the department was constrained by the resources available. To speed that up we needed more people (not an option) or less work. Much of the work was a consequence of things taking so long—duplicate requests and inquiries about work in progress. This was a case where the perception of slow payment led to action (duplicate invoicing and inquiries) that led to even slower payment.

Part of the solution involved changing perception. The CEO sent staff a message explaining that accounting had gotten behind and was working to catch up, and that the time required for line managers to approve invoices for payment was beyond the control of the accounting department. The CEO established a one-week turnaround goal of for line manager approvals.

He pointed out the irony that inquiries about delayed payments were responsible for a significant portion of the workload, thus slowing payments, and asked that for ninety days no payment inquiries be made. The CEO announced that accounting would begin publishing weekly metrics on the number of invoices received and paid, and that interested people could watch these metrics to see improvement in throughput until the crisis was past.

A letter was created for vendors apologizing for delays and suggesting two steps vendors could take to have payments processed promptly: Make sure that future invoices were complete and correct, and don't submit duplicate invoices.

A process was established to log invoices as they were received and respond to the invoicing vendor with a copy of the letter and confirmation that the invoice had been received. This correspondence helped manage vendor perception and dramatically reduce the number of duplicate invoices and inquiries.

An effect of the logging process was to track the number of unique requests received and establish the input flow rate. A corresponding process was created to close out transactions when payments were sent. This provided raw data for the output flow rate and gathered data on total processing time for individual transactions. Transactions sent to the line managers for approval were logged out when sent and logged in when received, capturing metrics for turnaround time and enabling reporting on what contributed to the total duration of each specific invoice. Together, these metrics allowed the organization to monitor and diagnose the flow of work and also provide visibility into the process and its steady improvement over time.

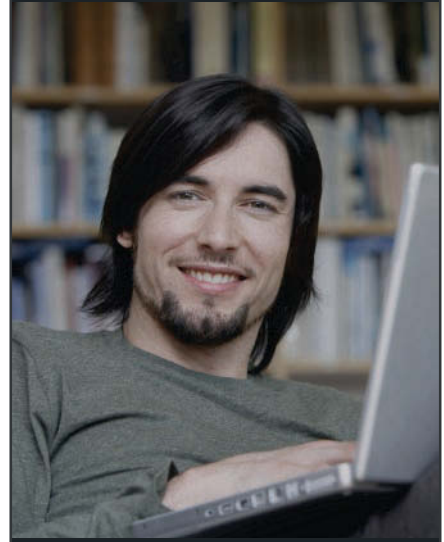
This is a favorite example of perception playing a role in both the problem and the solution. Try this definition of “problem.” If it serves you, thank Jerry Weinberg and Don Gauss. **{end}**

**Sticky
Notes**

For more on the following topic go to
www.StickyMinds.com/bettersoftware.

■ References

Go Live with Confidence!



The Load Testing Tool for all your web applications

NeoLoad, a load and stress testing solution for web applications, improves testing effectiveness. It enables faster tests, provides pertinent analysis and supports the newest technologies.

Test your Web application's performance easily and ensure trouble-free deployment thanks to NeoLoad!

Support for HTTP, AJAX, Flex, GWT, Silverlight, Java serialization, Push technologies,...

More details and free trial on
www.neotys.com

 **NEOTYS**
Make sure it works!

Virtual Resource Shelf

Author recommended books, blogs, gadgets, websites, and other tools for building better software

Q: How do you use social media?



I get most of my tech news

through Twitter—I'll follow people who have similar interests and read the stories they tweet. If I see a story retweeted several times in my news feed, then chances are good that the story will be important for me to read.

—Daniel Wellman



I'm active on Twitter and LinkedIn. If I have mutual LinkedIn connections with someone I meet or am about to meet in real life, I can discreetly check references to get a sense of who a person is. It's very helpful to get up to speed quickly on who you can trust. I sometimes use LinkedIn questions to gather data or get different perspectives on issues.



Twitter is like a turbocharger on my web surfing for software quality thought. The short format keeps messages succinct and it often gets used in the project management and software quality circles I monitor to post links to articles and blogs that I wouldn't always take the time to hunt down on my own.

My Twitter handle is @paysonhall, and you will see that I follow some software testing and quality luminaries, like the Bachs and Jerry Weinberg; project management gurus, like Johanna Rothman and Dave Garrett; and people who are just smart, like Dale Emery, Michael Kelly, and Naomi Karten. If any of these guys post a link saying, "Software estimation: This is worth reading," or "Here's an interesting take on status reporting," I make the time to check it out.

—Payson Hall

I use Facebook, Twitter, and LinkedIn. I update my status on all three when I blog. I mostly use LinkedIn as a way to connect with people.

—Johanna Rothman



facebook



I love the succinctness of Twitter! It's like having a team of readers who review articles, blogs, and videos for me and advise me of what should get my attention today. The more retweets and commentary, the higher it climbs in my to-read list. I try to return the favor and retweet things that I think warrant reading time, as well. It's also really helpful at conferences and helps guide me on which sessions to attend.

I have to give a shout-out to LinkedIn, as well. I rarely actively use it (I'm more of a passive user, reading LinkedIn updates but not really providing any of my own), but when I do need to speak with a specific person, it's a great lifeline.

—Michele Sliger

SALARY SURVEY

Better Software magazine
StickyMinds.com

2010

**BETTER SOFTWARE MAGAZINE
DIGITAL EDITION EXCLUSIVE!**

View the results of the
2010 *Better Software Magazine* &
StickyMinds.com Salary Survey.









userlytics[™]
WHERE AGILE MEETS USABILITY





www.userlytics.com

The right tool to:

-  reduce the time and cost of SW development
-  improve the customer experience
-  decrease QA and customer service spend
-  increase the ROI of SW development

Contact Userlytics today: (415) 449-0502

Simple:

set up a usability test in just
five minutes

Quick:

interpretable video-centric
results in hours

Powerful:

fine-tune Design, Development,
and QA

MASTER *the* ART *and* SCIENCE *of* SOFTWARE

Combine & Save on a Testing Training Week



The more training you take the greater the savings!

Maximize the impact of your training by combining courses in the same location. Combine a full week of training for the largest discount!

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
Software Tester Certification—Foundation Level			Mastering Test Design	
Systematic Software Testing*			Performance, Load, and Stress Testing Workshop	
Finding Ambiguities in Requirements†	Writing Testable Requirements†		Requirements Based Testing†	
Risk-Driven Software Testing**		Testing with Use Cases**	Test Estimation & Measurement	Just-In-Time Testing Workshop
Agile Testing Practices		Testing Under Pressure	Test Process Improvement***	Free Seminar (9 a.m. – 11 a.m.)
Test Process Management		Leadership for Test Managers	Note: Free seminar will be on Thursday in San Diego and Boston and on Friday in Chicago.	

*San Diego only **Boston and Chicago only ***Chicago only † San Diego & Chicago only

**SPRING
2011
SCHEDULE**

TESTING TRAINING WEEKS

March 14–18, 2011 San Diego, CA
April 11–15, 2011 Boston, MA
May 16–20, 2011 Chicago, IL

FREE SEMINARS

March 17, 2011
San Diego, CA
April 14, 2011
Boston, MA
May 20, 2011
Chicago, IL

TESTING

New
SPRING
2011
SCHEDULE
...

LEARNING OPTIONS:



Instructor-led
training in a city
near you



Live, instructor-led
classes via your
computer



Self-paced
learning, online



Instructor-led training
at your location

WAYS to SAVE

Take advantage of the different "Ways to Save" on training using our discount programs listed below. Purchase valuable software quality training for your whole team and save.



For more details on our discount policy, contact the Client Support Group at sqeinfo@sqe.com or call 888.268.8770 or 904.278.0524.

**EASY TO
REGISTER**



ONLINE:
www.sqetraining.com/register



PHONE:
888.268.8770



EMAIL:
sqeinfo@sqe.com



**PUBLIC COURSES
ON-SITE TRAINING
LIVE VIRTUAL
eLEARNING**

**PROVIDING EXPERT TRAINING
to SOFTWARE PROFESSIONALS**





AGILE
LIGHT BULB
MOMENTS
BY MICHELE SLIGER

ISTOCKPHOTO

I believe that the telling of personal stories around a specific theme is the best way to convey information to those who have concerns about that topic. In particular, those who've been through the transition to agile processes can tell stories that help agile novices better understand what it's like to adopt agile.

This seems quite fitting, given that agile development is full of nods to the importance of stories. For example, we use *metaphors* to help us understand our project goals. We use *user stories* to help us understand what the customer really wants. We write real-life stories as *experience reports* that we share as a way to help others understand what to try and what to avoid based on our situational experiences. Other real-life stories, such as *case studies* and *white papers*, help us understand broader agile issues and events. And now I'm encouraging the use of *light bulb moment* stories: stories of that seminal moment of pure clarity experienced when we finally understand what it truly means to be agile.

In this article you'll find several light bulb moment stories, with links to more. You'll have the opportunity to consider which, if any, resonate with you. And, you'll be able to add your own light bulb moment to help others travel the path with less trepidation.

But first, the meta story—how the light bulb moments came to be and why I believe in them so strongly.



Becoming Comfortable with Being Uncomfortable

I help teams with their agile adoption. After one such successful transition, my client said I should warn newbies that they “will experience at least three iterations of confusion, pain, and suffering. But, eventually, it really will get better.” This was eye opening for me. I knew that the first few iterations were hard, but I would not have used the word “suffering.” But clearly, for some, the change was very distressing, even traumatic.

I wanted to make my clients' agile adoption as easy as possible. I began to ponder what it would take to make that happen. The bottom line was that they were going to be uncomfortable with the transition, and that's normal. Learning something new, changing how we view ourselves in our role or career, not knowing what comes next—these all make us uncomfortable to varying degrees. What I needed to teach was the ability to become comfortable with this uncomfortable feeling; to accept discomfort as the norm and learn to ride the waves and not flail against them.

It was a nonprofit organization that had nothing to do with product development that gave me the tools I needed. In 2002, I discovered Judi's House (www.judishouse.org/). Judi's House is dedicated to helping children learn how to cope with the loss of a loved one by providing a safe place where kids can participate in peer-based support groups with children their own age who have had similar experiences relating to death. It was founded by Brian Griese, an NFL quarterback who lost his mother, Judi, to breast cancer when he was twelve years old.

After three days of formal grief facilitation training, I

started volunteering there. This training on how to companion others who are suffering, plus my personal experiences, helped me understand how to listen without focusing on ways to fix the problem. The kids simply shared their stories and derived strength from the knowledge that they were not alone.

I realized over time that Judi's House's facilitation and companion practices were applicable to my role as an agile team leader. Taking a lesson from the Judi's House peer groups, I realized that the best way to ease a wide-ranging community into agile adoption would be to offer a virtual space where those in an agile transition could find others who were going through the same thing or who had already been through it and come out on the other side. In this space, they could read about what others had to work through, what they learned, and whether or not it was worth it. They could read stories of hope and promise that might lift their spirits and reduce their fears. And they could share their stories, too—their light bulb moments.

Stacia Viscardi, my colleague and coauthor, contributed the first story—a wonderful story about her personal journey—that ended up being the introduction to our book *The Software Project Manager's Bridge to Agility*.



The Illusion of Having Power and Control

Stacia was a devout PMP at Primavera. As early as 1993, she was a Gantt chart expert and could do forward and backward pass calculations in her sleep. Project management felt like a perfect fit for her, and her identity as an individual was never far from her identity as a project manager. When Ken Schwaber, creator of Scrum, showed up in 2003 to promote Scrum in her company, Stacia was vehemently opposed to this new, lightweight, not-sponsored-by-any-formal-governing-body methodology. To make matters worse, when her boss failed to show up to ScrumMaster training, Stacia was immediately anointed as his replacement and had to go through training with Ken in order to take over the management of three Scrum teams. She writes:

As I drifted in and out of the two days of ScrumMaster training, the line that caught most of my attention was “You have no power.” Ken meant it in the sense that the product owner and delivery team roles would be collaborative in nature, and that a project manager wasn't the decision maker in Scrum. Like a mantra, I repeated this line to see if I could get used to it. I kept thinking, “How could you possibly manage a project or people without power?”

So now I had to *lead* people. I had never led people before. I had certainly managed them and collected the status of their tasks and quizzed them on how much time was remaining on those tasks. And, of course, I questioned their estimates.

What I didn't realize at the time was that I really had no power to begin with. You see, I had always man-

aged a group of knowledge workers—folks who grew up crunching numbers, writing complex code, creatively banging out products that at their roots consisted of only 1s and 0s. I truly believe that up until learning to lead, these knowledge workers merely tolerated me. I had never really managed them. They managed me by deciding to make me happy by filling out their timesheets. They put up with me when I asked to be walked through the testing phase of the project plan again. They humored me and provided me with percent-complete numbers that didn't really mean anything. They certainly knew way more about how stuff really worked than I did.

Another of Ken's comments compounded her confusion: "Scrum teams are self-organizing."

I thought "Wow, if teams are self-managing, they'll no longer need me. I don't have a place in this organization now that we're using Scrum." I had no idea how to act within this new realm. I had a very real struggle with getting past the "me" and focusing on the team. I was also troubled with ownership issues. I routinely struggled with not owning the administrative task of updating the product backlog; for me, this represented scope, and not having it within my charge was very frightening. I felt powerless and as if I had no role.

Somewhere around the third sprint, I started to get it. Once the teams started delivering real value that could be seen and touched, the light bulb went on. What were once yelling product owners were now engaged, energized product owners who actually worked with the teams to talk about the user experience, helping developers deliver valuable product increments. Observing collocated team members who were often heard laughing, working closely together, and enjoying their personal lives again touched me in a way that no perfectly calculated project Gantt chart or nested work breakdown structure ever could.

Stacia learned that having such a strong identity with her project manager title made it hard to let go. But once she was able to let go of that first item—control over the product backlog—the rest fell with much more ease.

If Stacia had not shared her story, I never would have suspected that it was so difficult for her in the beginning. She's one of the best coaches I've ever had the pleasure of working with, and it was an honor to coauthor a book with her. How many other stories are out there, these great success stories of personal transformation that we so rarely get the opportunity to share or hear about?



Letting Go of Control

This next story is Skip Angel's. He's a Scrum Coach, another amazing individual so at ease with his job that I was surprised yet again to hear that he struggled with the transition. An excerpt from his

story illustrates how he found a different way to deal with the discomfort of letting go of control.

I felt ignored. I felt rejected. My confidence was shaken, and my lack of results over time was showing that others thought I lacked confidence. My initial reactions were human nature. First, be more assertive or demanding: "YOU MUST DO THIS!" When that didn't work, my next reaction was "I am a failure, perhaps I am not cut out for this." However, deep (and I mean really deep) inside of me I knew that I was meant to be in a coaching role. I had moments of success along the way. It just wasn't consistent and I didn't know why. Then, the moment of truth came:

They are not ready for what I have to say, so I will figure out when and how to say it differently another time.

Instead of feeling I had no control and should probably give up, this gave me something that I did have control over. I flipped the bit from being a victim to being in control. All of a sudden, I had plenty of questions to ask myself and others. Why are they ignoring what I have to say? Is now the best time to provide advice? Am I providing too much information? Are they having a bad day? Do they not see the problems I do? Do they need to experience something painful to learn from it? I had several options available to me on how to approach the problem. As a result, I started to become more effective in working as a coach. I learned to listen more and discovered how important the timing of information is. I became more patient in my approach and learned that it takes several times (and different ways) to communicate ideas before they are received. I put myself in their shoes and looked at things from their perspective instead of only my own.



Life Changing, Even for CEOs

Stacia and Skip both alluded to servant leadership [1], or the concept of focusing on meeting the needs of those they lead. Stacia realized that she needed to let go of the backlog so that she could focus on the team. Skip realized that he needed to find ways to communicate that were best for those on his team—what they needed to hear when they needed to hear it.

And, they both talked about the quality of life, in terms of working at a sustainable pace, and how their personal transformations improved their ability to be better coaches. Beyond these coaching improvements can be found changes at the executive level that affect not only the organization but, once again, the individual's personal perspective, as illustrated in this story from Kevin Torf, CEO of Inventtrex:

Agile has many values, but by far the most important to me was learning about people. At first, I struggled to introduce the disciplines and mechanics involved, but over time I learned that the secret is in the commitment of each individual ... Agile's openness and delegation of

individual responsibility and accountability was a powerful tool in driving productivity, once all parties had bought into the methodology.

Agile has changed my personal perspective on not only how I manage my employer responsibilities but on how I handle and manage other things in my life, allowing me to be more effective and efficient in everything I do.



Customer Happiness

Thanks to the above contributions, I now had enough stories to open up a website where others could read and post their agile light bulb moments. When the next set of stories started coming in, they all seemed to have the same theme: customer happiness. Not that old standard phrase “customer satisfaction,” but “our customer was *happy*!” Not only were customers happy, they actually gleefully expressed their appreciation. Bob Fischer of Enagility tells a story of his first agile experience where everything seemed to be going wrong, yet it was still a great experience for their business partner.

Our first agile project had a lot of things wrong with it (code pulled from production after launch, formal QA not done until the end of development), yet our business partner said it was one of the best experiences he’d ever had working with IT. He really appreciated the level of communication and collaboration. I realized if we could be that bad and still get that kind of feedback, agile was really worth our time.

This is a great example of how just trying to do the right thing, starting with building a relationship of trust with the customer, can reap big dividends.

The last story is from an anonymous poster, who shares how involving the customer early helped the team catch a big problem right at the beginning of the project, which impressed and excited the customer:

Our team began our first agile project last fall. The team was rather apprehensive as to whether this approach could work for the type of projects that we do and was also very skeptical about the value of doing so. We insisted that a few of the key customers attend our sprint demo/retrospective meetings. It was during this meeting at the end of sprint No. 2 that our “light bulb” went on!

During the demo portion, one of the customers noticed that patient demographics were in the data string, which could potentially be a regulatory violation. He also pointed out some formatting changes that he’d like to see on a report. Both of these were added as stories to our backlog and were easily corrected within the next sprint. To me, this is really what agile is about. The problems were found early in the process because we actually showed the customer what they would be receiving, then the problems were quickly resolved.

During the next sprint demo meeting, the customer was excited to see the revised formats. He also made the comment that he had never had so much communication during a project and that he appreciated the visual evidence of what we are working on. In addition, he mentioned that he is very impressed with how much work the team is putting out in the short sprints. Since that point, we have been much more successful in engaging our key customers in the process as they are starting to see the value of agile.

Customer happiness and appreciation—that’s a wonderful moment of clarity and a great way to realize the benefits of your hard work.



Now It’s Your Turn

As a result of looking for and gathering these stories, I’ve noticed these themes repeating: letting go of power and control, emergence of servant leadership, happy customers, and personal transformations that improve the quality of life. You can read more of these stories at www.agilelightbulb.com.

I have also learned that just as agile is not a magic bullet, there is no magic universal light switch that I can flip to make the adoption a snap for those who struggle. As American novelist and poet Don Williams, Jr. stated, “The road of life twists and turns and no two directions are ever the same. Yet, our lessons come from the journey, not the destination.”

Indeed, each individual has a personal journey to undertake to learn how to cope with the loss of the old way of doing things—the old way you used to define yourself and your role—and how to cope with the change that agile adoption brings. Be it a moment of clarity, a series of events that leads to slow realization, or a simple acceptance of what seems like the most natural way to develop software, each journey, and each story, will be unique, yet many will share the same understanding.

I believe we can help each other by sharing our stories in a virtual peer group with others who are also experiencing the joys and pains of agile. It is in this space that I hope you will share your light bulb moment and your techniques of how to become comfortable with being uncomfortable.

It doesn’t have to be a pivotal point in your agile adoption (although that does make for terrific reading!). It could be like my story and others’ stories, where we already had light in the room but didn’t realize its name was agile. Or, it can be like Stacia’s or Skip’s stories, which are more of a journey. It can be a few sentences or a few pages; it doesn’t matter. What does matter is the sharing. Someone somewhere will no doubt find your story helpful or, at the very least, of interest.

Please, tell me—what’s your story? **{end}**

michele@sligerconsulting.com



For more on the following topic go to
www.StickyMinds.com/bettersoftware.

■ References



5 Myths Testing Regulated Software

ISTOCKPHOTO

“We can’t do that, we’ll never pass the audit!” If you’ve worked in regulated software, chances are you’ve heard that statement. The threat of an audit can cause even the most enthusiastic proponents of new ideas to cower in fear of the dreaded “finding,” an auditor’s report on something noncompliant in your process. The truth is, many of the most cited “can’ts” about regulated software testing are nothing more than myths.

Myth 1: We Can Only Test with Prewritten Test Cases

Scripted manual tests have been used for so long that many organizations assume that they are the only option, but that doesn’t mean this is the only acceptable approach. When someone says, “We can’t do exploratory testing,” what he is usually afraid of is not being able to present objective evidence of testing, such as the kind called for in FDA regulatory guidelines [1]. Those regulations require that someone looking at your documentation can tell what you tested without taking your word for it. There is nothing about prescribed tests that make them better for this than exploratory tests. Some problems with test evidence, like putting a checkmark in the Pass box or just typing “as expected” for a result, are actually unique to prescribed test cases. In this regard, exploratory testing can be more suited to showing objective evidence

because notes on observation make up the bulk of the documentation. An exploratory test session is designed so that the tester has objective evidence about what he saw.

Screen and video capture tools can make gathering objective evidence even easier. One tool, SiriusQA’s Test Explorer, is specifically designed to support the recording of objective evidence during exploratory testing [2]. Other tools I’ve used that feature video capture are BlueBerry Test Assistant and HP Quality Center.

Myth 2: Test Automation Is Too Difficult

Test automation is a tricky road to walk in any context, but in the regulated world, misunderstood rules often remove it from consideration completely. This is unfortunate because automation can supplement a regulated testing process just as much or more than a nonregulated one. An automated test can add control to your evidence because it logs results the same way every time, but some auditors may ask you to validate each tool for your specific process to make sure it meets their standard of objective evidence. Types of tool validation that I have seen are *general* and *specific*. General validation is usually a certification for a certain industry. Some companies will tout this certification about a particular tool if it has passed an audit by a regulatory body in that industry. Second, the tool may need to be validated for your specific

S of regulated ware

by John McConda

environment. In my experience, our team had to prove that each automated test's results were identical to the same test run manually. If your regulations require a high level of validation before use, you may have a difficult choice between the amount of time automation will save you in the long run against the additional cost you will incur for tool and test validation. On the other hand, if you are spending a substantial amount of time running a large suite of regression tests manually, you may still come out ahead.

Automation is more than just regression testing, though. It can also be used to run high volumes of test data through thousands of combinations that would take months to do manually. In his article for StickyMinds.com, "The ROI of Test Automation," [3] Mike Kelly lists even more types of test automation, many of which, if done in addition to a validated manual test process, may not even need to be a part of your officially validated process (see Myth 5).

Myth 3: Regulated and Agile Don't Mix

Over the past ten years, agile software development has taken the world by storm, but regulated software has been one of the last holdouts. Some of this can be attributed to one of the Agile Manifesto's stated core values of "Working software over comprehensive documentation [4]." As agile processes have evolved, so have ideas about where they can be

used. Today, FDA-regulated companies like Abbott Labs are presenting case studies that describe how agile processes have been used in developing their applications [5], and Department of Defense-regulated companies like Lockheed Martin are also publishing case studies of their results with agile [6]. The common thread among many of these case studies is that software teams were able to cut through perceived notions and get to the real purposes and constraints of the regulation. Once those were understood, then it was possible to tailor agile processes to the regulatory requirements.

Rich Sheridan, CEO of Menlo Innovations, a biotech software firm, believes agile processes are essential to software that is developed for medical devices. He talks about many agile ideas his teams implement like "weekly show and tells where the customer runs the software in front of our team," automated unit testing, and paired programming. He adds: "In our environment, no production code can be produced by anything other than a pair of developers, thus each line of code is reviewed by two sets of eyes. Regardless of what the FDA may think about such a system, I know I wouldn't want to have a device hooked to me that wasn't developed in a test-driven, paired-programming environment [7]."

Myth 4: The Auditor Is Your Enemy

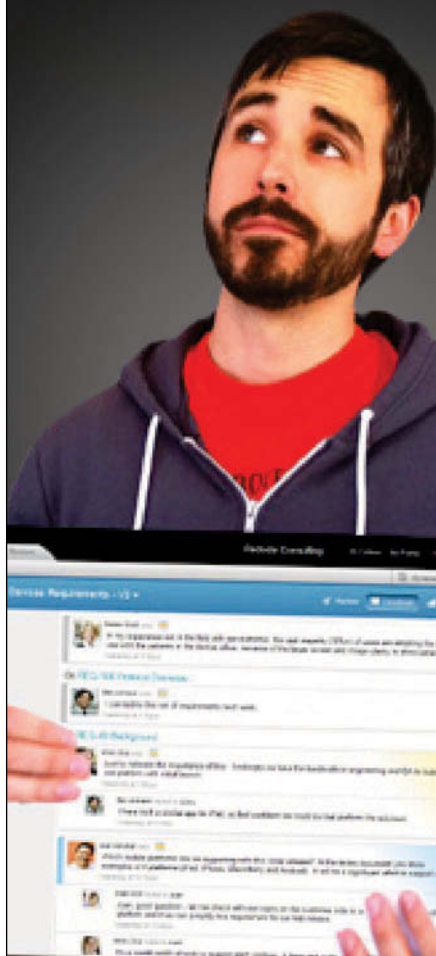
When most of us hear the word auditor, we think of a



jamasoftware.com

TIRED OF SCOPE ISSUES?

try **Contour.**



grumpy IRS agent with horn-rimmed glasses and a thin black tie. In my experience, the auditors I've encountered were quite willing to work with our team to make the process as smooth as possible. The most effective way to avoid significant findings is to know what you're being judged on as you conduct your process and to make sure everyone on the team does, too. At one of my former companies, we put together an agenda for the auditor to approve and then put all of our evidence into binders in the conference room so she could peruse at her convenience. We made every effort to be transparent, so she had no reason to suspect we were covering up a flaw in our process. If we did have a known flaw, we would document it for corrective action up front. Practitioner and author Matt Heusser, who has tested systems subject to HIPPA and SAS 70 audits, believes the perception of an adversarial relationship with auditors is counterproductive. "There is the perception that you never volunteer information to an auditor, so we were taught to answer questions and say nothing more. There was even a joke that, if an auditor asks if you know what time it is, the correct answer is "Yes." But even the word "audit" itself, comes from the Latin word for hearing. An audit should be a software team's invitation to "Come hear what we do [8]."

Myth 5: Regulated Testing Is Boring

A few years ago, when I was working on my master's degree, I talked after class one evening to a few fellow students who mentioned they were testers. Excited about having classmates in the same field as mine, I asked them how they liked their jobs. All three of them agreed, "I'm getting this degree so I can get out of testing. It's so boring!"

Bored often means you're not using your brain. Regulated testing is subjected to this stigma even more often, mostly due to Myth 1. Doing nothing but executing the same manual scripts over and over is enough to make any tester consider taking apart his cubicle with a screwdriver. Despite the ideas I described in Myth 1, your company may not be ready to move away from test scripts for their officially validated testing. If that's your situation, then there's always what I like to call Black Ops testing. The first time I brought up the idea of using exploratory testing for our regulated software, I encountered some resistance to making it part of our standard operating procedures. What we agreed to was a clause in our documentation stating that we "reserve the right to do other forms of testing as needed." This gave our testing team carte blanche to do as much additional testing as we needed to feel good about our coverage, without the burden of test cases. To succeed at this, though, you need to have time for your extra testing built into the project plan. If your project manager balks at this, make a case for just a few extra hours the first time. If you can show how much more efficient your testing is becoming, you'll have leverage to get the time you need.

Certain software is regulated for a reason. These applications are often life critical or they process sensitive information. Stakeholders deserve the best testing possible, performed by the best testers around. Unfortunately, the myths around regulations can keep good testers away and handcuff those who do test them. Regulations are intended to provide accountability, but it is up to those of us who create and maintain these applications to make ourselves accountable for testing with the best tools and techniques available—and not to be intimidated by myths.

{end}

jmcconda@gmail.com

**Sticky
Notes**

For more on the following topic go to
www.StickyMinds.com/bettersoftware.

■ References

BETTER SOFTWARE CONFERENCE

www.sqe.com/bsc

**Mark Your
Calendar Now!**

June 5-10, 2011

Las Vegas, Nevada

Caesars Palace



**Conference
Bonus:**

*Your conference registration includes
full access to the Agile Development
Practices West Conference!*

An Introduction to Scala

by Daniel Wellman



Scala is a programming language designed to be concise, safe, and compatible. Programs written in Scala run on the Java Virtual Machine and can reuse existing Java libraries and code.

Scala is already used in production for business-critical systems; it powers several services at Twitter [1] including message queues, the streaming API, and people search function. Foursquare's main and mobile websites are written in Scala [2]. In addition, Scala is used in projects at companies such as LinkedIn, Novell, Sony Pictures Imageworks, and Siemens. Martin Odersky, Scala's creator, has a strong language design background: He wrote Sun's Java 1.3 compiler and cocreated Generics for Java.

Scala refines the object-oriented features of Java and borrows functional programming techniques from languages including ML and Haskell. This yields code sizes that are typically reduced by a factor of two to three when compared to equivalent Java code [3] and draws aesthetic comparisons to other dynamic languages [4] like Ruby. But Scala code is fast— with performance on par with Java [5] and faster than Ruby.

One of Scala's key strengths is its excellent compatibility with Java. Scala's compiler compiles Scala source files to Java class files (bytecode), and its runtime libraries are straight Java JARs. Scala code can call any existing Java code you have, so you can reuse all that Java code you've written. This means you can write code in Scala and run it anywhere you would run Java code—from desktop Swing applications to web applications powered by servers such as Jetty or Glassfish.

Functional Programming

The functional programming style is an alternative to the imperative style often used in object-oriented languages. Functional programming languages model their computations in terms of side effect-free functions, like mathematical functions—the output of a function depends only upon its inputs. If you call a function with some input value, it will yield the same output no matter how many times you call the function. Compare this to an object-oriented language, where calling a method on an object may give you a different result on every invocation if an object's state may change (for example, a GUI window object's `getPosition()` method will return different values as the window is moved around the screen).

Functional languages are written in terms of immutable values; once a variable is assigned, it can never change. Think of final variables in Java—those are immutable values. Here's a sample assignment in Scala:

```
val theAnswer = 42
```

To declare an immutable value in Scala, you use the `val` keyword. Any attempt to reassign the value will result in a compiler error.

Furthermore, functions are first-class values just like numbers and strings; they can be assigned to variables and passed as arguments to functions. Here is how to find the even numbers between one and eight:

```
val numbers = List(1, 2, 3, 4, 5, 6, 7, 8)
numbers.filter { x => ( x % 2 ) == 0 }
// Result is List(2, 4, 6, 8)
```

This code creates a list containing the values one through eight and assigns it to the variable “numbers.” The “filter” function on List is invoked, which evaluates the list and returns a new list containing only the items for which the specified function evaluates to true. In this case, we specify a function (between the curly brackets) that takes a variable “x” and returns a Boolean indicating if x modulo 2 is zero (another way of asking “Is x even?”)

Functional programs can be simpler to understand than programs with mutable state, since a function's result depends only upon its inputs. If you've tried to understand how a method on an object works when the result depends on what happened to the object before the call, you've found a problem that depends upon mutable state.

This simplicity is helpful when applied to multithreaded concurrent programming. In Java, multithreaded programs are written using memory locks and synchronization, a technique that is extremely easy to get wrong and that can lead to race conditions requiring hair-pulling debugging sessions. Immutable state makes concurrent processing much simpler since you avoid many concurrency problems altogether (see the book *Java Concurrency in Practice* [6]).

Of course, programs often need mutable state at some point, and Scala supports this programming model as well. Scala enables you to design programs that have mutable, encapsulated state in objects and rely on the functional parts of the language for your computations and parallel processing.

Scala—a Better Java

Scala is also an object-oriented language and improves upon many features of Java. In Scala, everything is an object—even numbers, characters, and Booleans—which makes for simpler code with fewer special cases. (Note the compiler replaces these objects with primitives, which gives you the best of both worlds: objects for simpler code, yet primitives for faster performance.) Scala has a powerful type system that enables type-safe code in many cases where Java would have required casting. And just like Java, Scala will catch type mismatch errors at compile time.

Scala code is less verbose than Java. Consider the code in Java to create a map of some HTTP result codes to their status messages:

```
// Java
Map<Int, String> statusMsgs =
    new HashMap<Int, String>();
myMap.put(200, "OK");
myMap.put(404, "Not Found");
```

In Scala, it's simply:

```
// Scala
val statusMsgs = Map(200 -> "OK",
                    404 -> "Not Found")
```


While Java requires us to specify the type of the map's keys and values on both the left and right sides of the expression, Scala looks at the creation on the right side and determines the variable's type. This is known as type inference, and it means that many type declarations can be omitted, which reduces the amount of code and clutter per line. Additionally, note how Scala lets you initialize the map's contents in one line, which is a nice touch. In Scala, Map is a library class like Java's HashMap.

Another feature that enhances Scala's object-oriented power is support for traits. Traits are similar to Java interfaces, which define a set of method signatures that a class will implement. However, traits may also contain method implementations, which is not possible with Java interfaces. And like Java interfaces, classes may implement many traits. Multiple traits may be stacked together in one class to assemble rich behavior with minimal code, but Java is limited to extending at most one base class to reuse any default behavior and requires that the programmer implement the remainder of the interfaces.

For example, the Scala library provides the Ordered trait, which is used to compare objects using semantics like greater than or less than. If a class implements the Ordered trait, the implementer must define how objects are compared, just like Java's Comparable interface. However, the Ordered trait provides default implementations of the methods `>`, `>=`, `<`, and `<=`, which the implementing class now gains "for free." (Note that Scala methods may use symbols in their names, unlike Java.)

Why Learn Scala Now?

Scala is gaining momentum. The availability of books is a good indicator of market interest in the language. Presentations on Scala are becoming frequent at conferences. In April, Scala passed 1,000 questions available on the popular question site Stack Overflow.

And improvements are in the works. As of this writing, the latest stable version of Scala is 2.8.0, released in July 2010. Version 2.8 offers several improvements including: an improved collections library; support for nested Java annotations, which is especially useful if you want to use annotations for Java Persistence Applications API; new language features, including named and default function arguments; and hooks added to the compiler to make it easier to build Scala IDE plug-ins, which means better tool support.

The Good and the Bad

Scala is an advanced language, but it's not without weaknesses. There have been some annoying problems with point releases of Scala that require you to recompile your code and use libraries recompiled with the latest version. However, the development team has stated that it will be focusing on binary compatible releases starting with version 2.8, which may help. Scala is a flexible, powerful language, which means that not only does it enable you to create beautiful, sophisticated code but it also won't prevent you from creating a snarled, unreadable mess.

Scala is developed at a university—not funded commercially by a big company like Java and Sun/Oracle—so its resources are more limited. There have been occasional compiler bugs, and the API documentation can be a bit thin compared to its Java counterpart.

That said, the Scala community is friendly and helpful, whether it's the active Scala mailing lists [7] or an online IRC chat channel. In fact, Scala's creator, Martin Odersky, actively participates in the mailing list, which reflects the close-knit nature of the community.

How Can I Get Started?

Download Scala from the Scala website at (www.scala-lang.org.) Once you have installed Scala, you can get an interactive command prompt (also known as a read-eval-print loop or REPL) by typing "scala" at your command prompt. Several books about Scala are now available, plus many blogs on the subject (see the StickyNotes for links).

An IDE can help when learning a new language, and Scala has plug-ins for all the major IDEs (Eclipse, NetBeans, and IDEA). These plug-ins are still in their early development phases so, while you can expect syntax highlighting and some code completion, there aren't many refactoring features, and you can expect to find some problems such as the editor not marking all errors in the code (though the compiler will catch them).

One useful way to start practicing Scala is to write tests in Scala for your existing Java code. You can write tests just as you would in Java using libraries like JUnit or TestNG but in Scala's succinct syntax. Or to try out some of the more expressive features of Scala, you could use a Scala testing library like ScalaTest [8] or Specs [9].

Let me suggest one technique to avoid: Don't try to learn Scala by first learning Lift, a popular Scala web application framework. Lift takes a unique approach to web applications, which can be quite daunting to understand if you're new to Scala or functional programming. As an alternative starter project, consider the "99 Problems in Scala" [10] web page that features several exercises that demonstrate the functional programming style. **{end}**

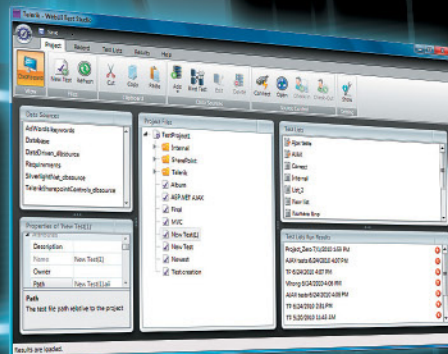
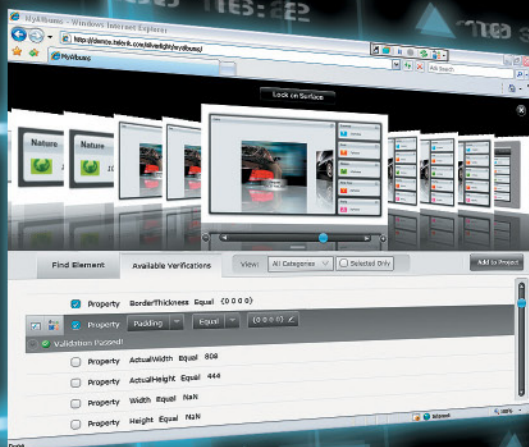
dan@danielwellman.com

**Sticky
Notes**

For more on the following topics go to www.StickyMinds.com/bettersoftware.

- References
- Further reading

Testing is Now Radically Easier



Telerik WebUI Test Studio

Easy Test Creation

- Test any web app – HTML/AJAX/Silverlight
- Intuitive point-and-click recording
- Record once, run against multiple browsers

Easy Test Maintenance and Management

- Element abstraction and reuse
- Scheduling, execution and results reporting
- Seamless QA – Developer collaboration

Download your free 30-day trial at www.telerik.com/RadicallyEasier

 **telerik**
deliver more than expected

Attend Live, Instructor-led Classes Via Your Computer

*New Live
Virtual Courses
Now Available!*



NEW Live Virtual Courses:

- » Mastering Test Automation
- » Essential Test Management and Planning
- » Finding Ambiguities in Requirements
- » Getting Requirements Right the First Time
- » Testing Under Pressure

Live, instructor-led classes are now available right from your computer! SQE Training uses Cisco's WebEx technology to provide you with all the benefits and personal contact of classroom instruction right from your desktop. You get the same valuable content and instructor interaction as you would in the classroom but with the convenience and cost effectiveness of being online.

Live Virtual Package Includes:

- **Easy course access:** You attend training right from your computer, and communication is handled by a phone conference bridge utilizing Cisco's WebEx technology. That means you can access your training course quickly and easily and participate freely.
- **Live, expert instruction:** See and hear your instructor presenting the course materials and answering your questions in real-time.
- **Valuable course materials:** Our live virtual training uses the same valuable course materials as our classroom training. Students will have direct access to the course materials.
- **Hands-on exercises:** An essential component to any learning experience is applying what you have learned. Using the latest technology, your instructor can provide students with hands-on exercises, group activities, and breakout sessions.
- **Real-time communication:** Communicate real-time directly with the instructor. Ask questions, provide comments, and participate in the class discussions.
- **Peer interaction:** Networking with peers has always been a valuable part of any classroom training. Live virtual training gives you the opportunity to interact with and learn from the other attendees during breakout sessions, course lecture, and Q&A.
- **Convenient schedule:** Course instruction is divided into modules no longer than three hours per day. This schedule makes it easy for you to get the training you need without taking days out of the office and setting aside projects.
- **Small class size:** Live virtual courses are limited to the same small class sizes as our instructor-led training. This provides you with the opportunity for personal interaction with the instructor.



Go

SAN FRANCISCO, CA and BANGALORE, INDIA—ThoughtWorks Studios has announced Go, a new agile release management platform that allows organizations to incrementally automate the entire build, test, and deployment process, and release software faster and more reliably.

Go enables continuous delivery, making it possible for enterprises to move from a world where software releases are governed by operational constraints to a world where they can release according to their business needs. Pioneered by ThoughtWorks as a key element of large-scale agile adoption, continuous delivery combines automation and engineering rigor with a heavy emphasis on collaboration to break down the prevalent organizational silos that segregate development, QA, and IT operations.

Features include:

- **Environments**—This first-of-its-kind capability lets organizations see what version of each application is in each environment, audit back to the source, and perform one-click deployments of any version of software.
- **Parallel Test Intelligence**—Speeds up tests by running them in parallel on a grid. Go's unique test intelligence lets organizations see exactly which tests broke, which check-in broke them, and who was responsible, even when tests are run across a grid.
- **New UI for the DevOps world**—Enables easy, centralized management of large numbers of projects and machines for both developers and IT operations.
- **Templates**—Allow management of large numbers of projects within an agile ALM environment; allow organizations to manage branches and define reusable workflow.

For additional information, visit thoughtworks-studios.com/go-agile-release-management.

Jama Contour 2.9.8

PORTLAND, OR—Jama Software has announced the availability of the Review Center module along with its new version, Jama Contour 2.9.8. The Review Center is a module that provides teams a fast and agile way to review requirements.

Teams can:

- Get requirements right the first time
- Manage scope changes faster
- Be agile during product planning
- Provide greater visibility into planning decisions
- Increase accountability and participation by everyone
- Document approvals at key milestones

To see the Review Center in action, watch the video and download the play-book at jamasoftware.com/review.

QA Wizard Pro 2010.2

CINCINNATI, OH—Seapine Software, Inc. has delivered new capabilities to testers with the general availability of QA Wizard Pro 2010.2. QA Wizard Pro 2010.2 now includes web load testing, which enables QA teams to ensure that applications can meet performance and scalability requirements.

QA Wizard Pro provides comprehensive test automation features for testing teams, enabling them to accelerate functional and regression testing in support of agile development practices. The new release adds the ability to combine automated user actions with multiple simulated users to provide the ability to identify issues that occur during normal application use.

Copyright © 2010, Varad Corporation. All rights reserved.

S-C-A-L-A-B-L-E

Ská-lə-bəl adjective.
capable of being easily expanded
or upgraded on demand

Individual

Startup

Small group

Large Group

Big Business

Enterprise

**UPGRADE OR
DOWNGRADE
AT ANY TIME**

**ISN'T IT NICE TO KNOW
THAT BUGHOST SCALES
WHEN YOUR
BUSINESS NEEDS
CHANGE?**

- ✓ **BUG TRACKING**
- ✓ **REQUIREMENTS**
- ✓ **TEST CASES**
- ✓ **TASK TRACKING**
- ✓ **FREE TECH. SUPPORT**

**THE RIGHT TOOL.
THE RIGHT PRICE.
THE ORIGINAL**



BugHost™

www.BugHost.com

BugHost, the BugHost logo and "Seymour" the bug are trademarks of Varad Corporation.

1. Publication title: Better Software
2. Publication number: 0019-578
3. Filing date: October 05, 2010
4. Issue frequency: January/February, March/April, May/June, July/August, September/October, November/December
5. Number of issues published annually: 6
6. Annual subscription price: \$40.00
7. Complete mailing address of known office of publication: Software Quality Engineering, 330 Corporate Way, Ste. 300, Orange Park, FL 32073-6214, Clay county
8. Complete mailing address of headquarters or general business office of publisher: Software Quality Engineering, 330 Corporate Way, Ste. 300, Orange Park, FL 32073-6214, Clay county
9. Full names and complete address of Publisher, Editor, and Editor in Chief: Publisher: Wayne Middleton, Software Quality Engineering, 330 Corporate Way, Ste. 300, Orange Park, FL 32073-6214, Clay county. Publishing Director: Holly Bourquin, Software Quality Engineering, 330 Corporate Way, Ste. 300, Orange Park, FL 32073-6214, Clay county. Editor in Chief: Heather Shanholtzer, Software Quality Engineering, 330 Corporate Way, Ste. 300, Orange Park, FL 32073-6214, Clay county.
10. Owner: C. Wayne Middleton, Software Quality Engineering, 330 Corporate Way Ste. 300, Orange Park, FL 32073-6214, Clay county.
11. Known bondholders, mortgagees, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages, or other securities: None.
12. NOT NON PROFIT; DO NOT NEED TO INCLUDE IN STATEMENT
13. Publication title: Better Software
14. Issue date for circulation data below: September/October 2010
15. Extent and Nature of Circulation:
 - a. Total number of copies (Net press run). Average Number Copies Each Issue During Preceding 12 Months: 8,315. Actual number of copies of single issue published nearest the filing date: 4,796.
 - b. Legitimate Paid and/or Requested Distribution (By Mail and Outside Mail) (1b) Outside County Paid/Requested Mail Subscriptions Stated on PS Form 3541: Average number of copies each issue during preceding 12 months: 6,229. Actual number of copies of single issue published nearest the filing date: 2,687. (2b) In-County Paid/Requested Mail Subscriptions stated on PS Form 3541: Average number of copies each issue during preceding 12 months: 0. Actual number of copies of single issue published nearest the filing date: 0 (3b) Sales through dealers and carriers, street vendors, counter sales, and other non-USPS paid distribution as stated on PS Form 3541: Average number of copies each issue during preceding 12 months: 289. Actual number of copies of single issue published nearest the filing date: 285 (4b) Requested Copies Distributed by other mail classes through the USPS (e.g. First-Class Mail). Average number of copies each issue during preceding 12 months: 0. Actual number of copies of single issue published nearest the filing date: 0
 - c. Total paid and/or requested circulation [Sum of 15b. (1), (2), (3), and (4)]. Average number of copies each issue during preceding 12 months: 6,518. Actual number of copies of single issue published nearest the filing date: 2,972.
 - d. Nonrequested Distribution (By Mail and Outside Mail). (1d.) Outside-county Nonrequested Copies Stated on PS Form 3541. Average number of copies each issue during preceding 12 months: 419. Actual number of copies of single issue published nearest the filing date: 212. (2d.) In-county Nonrequested as stated on form 3541. Average number of copies during preceding 12 months: 0. Actual number of copies of single issue published nearest the filing date: 0. (3d.) Nonrequested copies distributed through the USPS by other classes of mail (e.g. First class, non-requestor Other classes mailed through the USPS. Average number of copies each issue during preceding 12 months: 0. Actual number of copies of single issue published nearest the filing date: 0. (4d.) Non requested Copies distributed outside the mail (include pickup stands, trade shows, showrooms, and other sources) Outside the Mail (other carriers). Average number of copies each issue during preceding 12 months: 1,295. Actual number of copies of single issue published nearest the filing date: 1,400.
 - e. Total Nonrequested Distribution. Average number of copies each issue during preceding 12 months: 1,714. Actual number of copies of single issue published nearest the filing date: 1,612.
 - f. Total distribution (Sum of 15c. and 15e.). Average number of copies each issue during preceding 12 months: 8,232. Actual number of copies of single issue published nearest the filing date: 4,584.
 - g. Copies not distributed. Average number of copies each issue during preceding 12 months: 83. Actual number of copies of single issue published nearest the filing date: 212.
 - h. Total (Sum of 15f. and g.). Average number of copies each issue during preceding 12 months: 8,315. Actual number of copies of single issue published nearest the filing date: 4,796.
 - i. Percent paid and/or requested circulation (15c. divided by 15f. times 100). Average number of copies each issue during preceding 12 months: 79%. Actual number of copies of single issue published nearest the filing date: 64.8%.
16. Publication of Statement of Ownership. Revised Statement of Ownership is printed in the November/December 2010 (12-6) issue.

I certify that the statements made by me above are correct and complete.

Wayne Middleton, Publisher

Product Announcements

QA Wizard Pro 2010.2 includes five evaluation virtual user licenses, enabling testers to immediately begin writing and testing load tests. Non-evaluation virtual user licenses can be purchased in blocks of fifty to simulate real loads.

For additional information, visit seapine.com.

CloudStart

HP has announced HP CloudStart, the industry's first all-in-one solution for deploying an open, flexible, private cloud environment. Built on Converged Infrastructure, HP CloudStart simplifies and speeds private cloud deployments for any size environment. Consisting of hardware, software, and services, it empowers businesses to deliver pay-per-use services reliably and securely from a common portal while providing the ability to automatically scale and deploy new services. Access to consumption and chargeback reports in real-time allows clients to operate their private cloud in the same fashion as a public cloud.

Benefits include:

- 75 percent reduction in compliance management time through advanced automation and governance functionality
- Improved business response time to changing market demands by delivering technology services on an as-needed basis rather than a dedicated system
- Simplified addition and management of pools of network, storage, and server resources including enhanced data services like unified provisioning and disaster recovery on HP Storage
- Best-practice guidance from HP Professional Services provides expertise in deployment, customizing and executing on the long term vision for the private cloud tuned specifically for the client's environment
- Clients can broaden and customize their cloud service options as well as integrate legacy business applications including database, storage, messaging, and collaboration with the HP Private Cloud Platform

For more information, visit hp.com

Qualify

WESTMONT, IL—Original Software has announced the launch of Qualify, a brand new addition to its solution suite. Designed from the ground up to answer the growing need for QA to be less siloed, Qualify has brought effective quality management right into the heart of the software delivery process. This launch comes off the back of a new survey, which highlighted discontent in the market, with 84 percent of users stating that current products do not meet their functional requirements.

Built with the purpose of directly answering these needs, Qualify unites all aspects of the software development lifecycle from requirements to release. It is also fully integrated with Original Software's manual and automated test execution solutions for the user interface and database layers. Qualify encompasses requirements, planning, scheduling, resource management, test execution, defect management, and reporting in one platform, ensuring collaboration and communication throughout. The methodology-agnostic, entirely code-free Quality Management solution even comes complete with a smartphone interface, allowing instant access to cross-project information from just about anywhere.

For more information, visit origsoft.com/qualify.



www.sqe.com/adpwest

**Mark Your
Calendar Now!**

June 5-10, 2011
Las Vegas, Nevada
Caesars Palace



**Conference
Bonus:**

*Your conference registration
includes full access to the Better
Software Conference!*

FAQ

expert answers to
frequently asked
questions

by Dale Perry
dperry@sqe.com

Why are many performance tests considered failures?

A performance test is one of the most complex tests to plan and design properly. I have found that there are several key areas that affect the usefulness of the test and its results.

Starting the testing too late. The common perception is performance testing can only occur at the end of the development process when all the features have been thoroughly tested for functional validity. The main problem with this approach is that issues discovered late in the process may require changes to the application or system that will require retesting the affected areas and regression testing before the performance test can be rerun. It is important to note that changing more than one feature at a time can be difficult to coordinate. It also is possible that making multiple changes at once could obscure the solution to the problem.

Incremental and iterative styles of application development offer a great opportunity to implement performance testing early in a project. Using simple review techniques with a focus on performance testing rather than engineering the functionality, we can include features with potential performance impacts in the earlier builds and releases. Once the first build is functionally tested, the performance test group can start testing while the functional team continues to develop the next increment.

The advantage to this approach is twofold: The infrastructure gets an early test and, as there is a limit to completed features, changes to the system architecture or design are far less costly.

Architecture issues. There are two key elements that must be looked at to ensure the test has at least a reasonable chance of providing useful information: the physical architecture and the platforms on which the software will operate.

If the physical layout of the test system is substantially different from the production or target system, the test may not provide useful performance data. I find that a scalability factor (ratio of equipment used compared to actual production) greater than 5:1 tends to yield useful but nonpredictive results. Even at a lower scalability factor the results are no guarantee of production system behavior as many elements are nonlinear in nature.

We also have to examine the internal characteristics of each platform that will comprise the test. Internal differences in architecture can create or hide problems in the test lab that may not occur in the production system. If you adjust the application or system for the problems discovered in the test lab, you may be making a fatal mistake when the system goes into the production architecture.

Confusing business goals with performance goals. To be a performance goal, there has to be some element of the system or application that we can measure during the test. Poor performance typically occurs when some resource is under stress or reaching its maximum capacity. If you cannot design a test for something, it is probably a good idea not to agree to accomplish the task. Gathering and defining performance goals can be one of the least enjoyable aspects of planning a performance test—it tends to get political at times. However, the time to tell someone you cannot do something is before—not after—you spend his time and money.

These basic issues cause many performance tests to be of little value and often lead to tests that are not worth the effort to run.

I have addressed these topics in more depth in a four-part series posted on StickyMinds.com: StickyMinds.com/12-6Perry.

Which Obstacle Should You Tackle Today?

As the lead or manager, you must develop the skills needed to remove the obstacles that slow, impede, or halt your team's project work.

by **Johanna Rothman** | jr@jrothman.com

Imagine that you're a technical lead or a manager responsible for the work of at least one team. You also have a ton of technical work you're supposed to do yourself. Your to-do list is a mile long. What's the most important thing you can do today?

As the lead or manager, your most important job is to clear the way for other people so they can do their work. In other words, you remove the obstacles that slow, impede, or halt project work.

Remove the Physical Obstacles

The first thing to do is to look around your team's workspace. What do you see? Are people crooked in their chairs? Have they cobbled together a pairing space? Do they have enough lab space? Do they have the whiteboards, desks, index cards, stickies, monitors, or network connections they need?

Many years ago, when I was a program manager, the architect came to me and said, "I need some time at the chiropractor's. I'll be out all afternoon."

"Ok," I said, "What's wrong?"

"My back hurts." He paused for a few seconds. "That darn desk is too high."

Now, under normal circumstances, I'm the one complaining about too-high surfaces, but this architect was close to six feet tall, so a too-high desk was a surprise to me. I asked if he could show me what was wrong. We discussed it for a while, and concluded that he needed a different chair and a new desk. I told him I would work on the problem.

I spent almost six hours cajoling, pleading, and fighting with the facilities people. The desk and chair this fellow needed were not justified by his pay grade. But they were what he needed. I had to use all my influence and persuasive powers, and, with the promise of brownies, the desk and chair were delivered the next day.

Now you might not think that this physical obstacle was the most important thing I could have done all day, but we needed the insight this architect could provide. If he was unable to work, my program would have been in worse shape, not better shape.

The architect returned the next day, and was delighted.

"JR, how did you do this?"

"I need to make a bunch of brownies over the next few weeks."

"I'll buy them at this bakery on my way in on Mondays."

We both were happy with that situation!

If you observe people working or ask about physical obstacles, your team will tell you what they need. Determine what you can do and what you can't. I've run afoul of the furniture police when it came to whiteboards and encountered finance people who refused to buy index cards or stickies. I was able to buy the cards and stickies out of petty cash, yet I had to wait until the furniture police had a new manager before I could deal with the whiteboard problem. But if you don't look and you don't ask, you can't remove the physical obstacles people need you to remove.

**"If you don't look and
you don't ask, you can't
remove the physical
obstacles people need you
to remove."**

Look for Systemic Obstacles

If you're like many of the managers I know, you want to know why it takes so long for your team to finish work. The team doesn't think it takes so long,

but it sure feels that way to you. Look at the flow of work (value stream) to see where the systemic obstacles are.

During an engagement, a manager asked me how long I thought a project should take. I admitted I had no idea but thought it was relatively small, say, a couple of people for not more than two weeks. He told me it took six people three months, assuming all the planets aligned and Murphy didn't visit the project. We investigated.

We learned that the build for this small system took more than six hours. That meant every time a developer wanted to try something, the developer could only build once a day. So, the developers did not integrate continuously, and all the tests were manual, through the GUI. The testers had to restart their tests every time the developers changed something.

There was nothing small about their project, and it was painful to work on. The team decided to take a two-week timebox and decrease build time and automate some of the manual tests.

As a result of that timebox, the build time decreased to

less than ten minutes. And they had some system tests that could run automatically along with automated smoke tests. Now the testers had a choice about whether to restart testing or continue where they were with a new build. The six people finished that project in two months, including the two-week timebox where they removed obstacles.

Systemic obstacles make people's lives miserable. Discover them and fix them.

Watch for "People" Obstacles

"People" obstacles often occur when people have not learned to discuss the issue in a constructive way and are stuck on position rather than discussing principles.

One manager wanted to provide a tool to a loyal customer for free. The tool would allow the customer to prepare data for the company to transform and would reduce to almost zero the custom work the company normally provided for the customer. Without that tool, the company had to spend a minimum of six person-weeks on the data before it could be transformed. With the tool, the company could spend about two person-days, the same as for other customers. The salesperson wanted to charge \$2,000.

The manager and the salesperson got stuck on their positions—free versus \$2000—until a senior manager, Ted, arrived. Ted asked why the manager wanted to give the tool away.

"Because it's cheaper and easier for us to give away that tool than it is for us to prepare the data."

Ted asked why the salesperson wanted to charge for the tool. "Because it sets a bad precedent to give away our tools. What do we do about support? What if another customer wants this tool? The tool wasn't free to develop. We use it all the time. I'm concerned about a precedent."

Each person had the best interests of the organization at heart. Ted asked if either person had another option. They hadn't yet considered other options, so Ted helped them develop some.

The salesperson decided it was okay to provide the tool for free, as long as the customer realized there was no support with the tool. And, the manager agreed to let the salesperson know if the customer wanted support.

Helping people develop options when they are stuck helps with people obstacles. Most people get stuck when they are convinced there is only One Right Way. If they can see alternatives, they remove the obstacles themselves.

So, think about it.

What's obstacles can you remove today? Have at it. {end}

This article originally appeared on StickyMinds.com: www.stickyminds.com/12-Grothman

index to advertisers

ADP West 2011	www.sqe.com/adpwest	33
Better Software Conference and Expo	www.sqe.com/bsc	25
BugHost	www.BugHost.com	31
Hewlett-Packard	www.hp.com/go/agile	Back Cover
Jama Software	www.jamasoftware.com/go	24
Microsoft	www.microsoft.com/visualstudio/test	6
Microsoft	www.microsoft.com/visualstudio/test	11
Neotys	www.neotys.com	13
Rally Software	www.rallydev.com/bsm	Inside Front Cover
Ranorex	www.ranorex.com	5
Seapine	www.seapine.com	1
SQE Training—Training Week	www.sqetraining.com/Testing	16
SQE Training—Live Virtual Training	www.sqetraining.com/Virtual Training	30
STAREAST 2011	www.sqe.com/STAREAST	10
TechExcel	www.techexcel.com	2
Telerik	www.telerik.com/automated-testing-tools	29
Userlytics	www.userlytics.com	15

Display Advertising advertisingsales@sqe.com

All Other Inquiries info@bettersoftware.com

Better Software (USPS: 019-578, ISSN: 1553-1929) is published six times per year January/February, March/April, May/June, July/August, September/October, November/December. Subscription rate is US \$40.00 per year. A US \$35 shipping charge is incurred for all non-US addresses. Payments to Software Quality Engineering must be made in US funds drawn from a US bank. For more information, contact info@bettersoftware.com or call 800.450.7854. Back issues may be purchased for \$15 per issue (plus shipping). Volume discounts available. Entire contents © 2010 by Software Quality Engineering (330 Corporate Way, Suite 300, Orange Park, FL 32073), unless otherwise noted on specific articles. The opinions expressed within the articles and contents herein do not necessarily express those of the publisher (Software Quality Engineering). All rights reserved. No material in this publication may be reproduced in any form without permission. Reprints of individual articles available. Call for details. Periodicals Postage paid in Orange Park, FL, and other mailing offices. POSTMASTER: Send address changes to Better Software, 330 Corporate Way, Suite 300, Orange Park, FL 32073, info@bettersoftware.com.

BETTER SOFTWARE MAGAZINE

Save with an

Early Subscription Renewal!

Get another year of *Better Software* magazine for **Only \$14.95!***

Digital subscribers, now is the time to upgrade to print and take advantage of this special pricing!



To subscribe:

www.bettersoftware.com/11magad

*Offer for U.S. residents only

HARNES

the power of Agile.

HP Application Lifecycle Management for Agile initiatives

Looking to improve the visibility and
consistency of your Agile initiatives?

The HP Application Lifecycle Management
solution goes beyond traditional
development management. It's the Agile
solution that delivers quality *and* value,
speed *and* control, for the whole team
and the whole lifecycle.

Outcomes that matter.

Get more information at
hp.com/go/agile



Better Software magazine
StickyMinds.com

2010

SALARY SURVEY

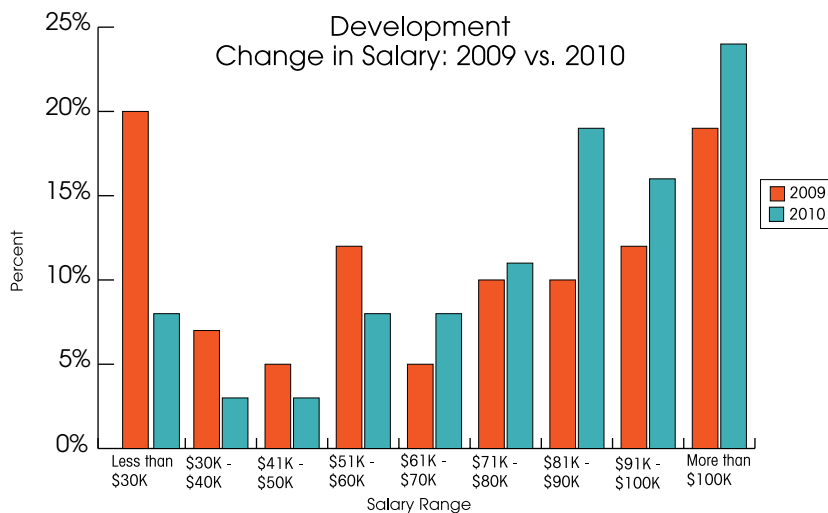
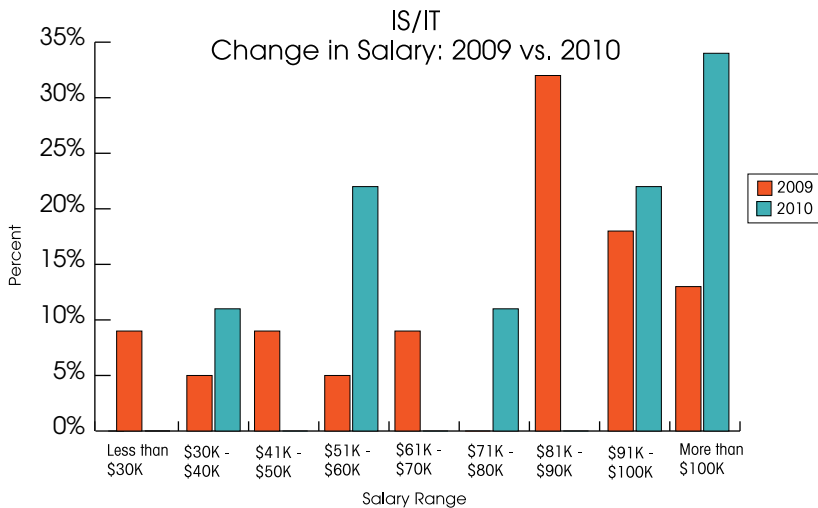
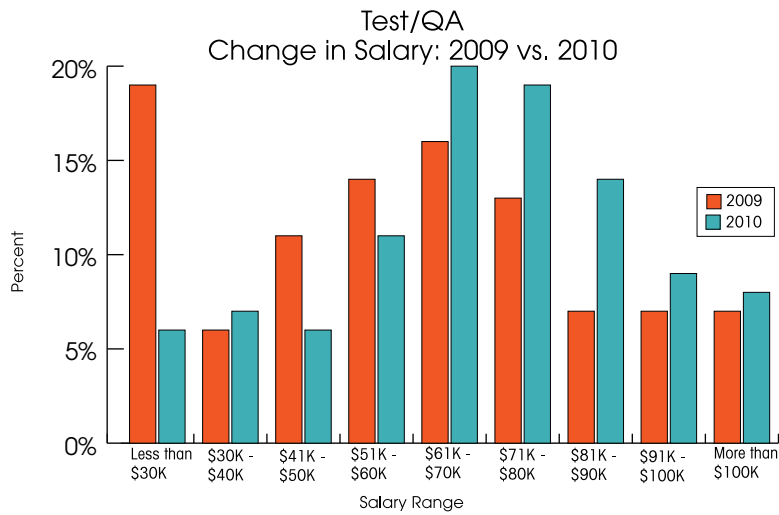
BY HEATHER SHANHOLTZER

The results of our annual look at industry employment trends are in. This year's salary survey reports salary figures for director-, management-, and staff-level professionals across a wide variety of fields including test/QA, development, project management, and business analysis. Our survey included questions not only about salary but also demographics, education, and experience.



ISTOCKPHOTO

STAFF LEVEL



GENDER

Male	52%
Female	48%

AGE

40 or under	39%
Over 40	61%

EDUCATION

Some college	29%
Bachelors	50%
Masters or higher	21%

DEGREE

CS/IS	38%
Engineering	16%
Business	14%
Liberal arts	7%
Other	25%

GEOGRAPHIC DISTRIBUTION

Northeast	18%
Southeast	9%
Midwest	24%
Northwest	7%
Southwest	18%
Canada	7%
India	2%
UK/Ireland	6%
Other	9%

YEARS IN SOFTWARE INDUSTRY

5 years or fewer	13%
6 to 10 years	23%
More than 10 years	64%

YEARS AT PRESENT COMPANY

5 years or fewer	51%
6 to 10 years	21%
More than 10 years	28%

YEARS IN CURRENT POSITION

5 years or fewer	58%
6 to 10 years	23%
More than 10 years	8%

JOB FUNCTION (MORE THAN ONE FUNCTION MAY APPLY)

Test/QA	81%
Development	14%
Project Management	7%
Business Analysis	11%
Other	6%

CERTIFICATION STATUS

Yes	46%
No	54%

MANAGEMENT LEVEL

GENDER

Male	57%
Female	43%

AGE

40 or under	39%
Over 40	61%

EDUCATION

Some college	20%
Bachelors	51%
Masters or higher	29%

DEGREE

CS/IS	26%
Engineering	27%
Business	14%
Liberal arts	10%
Other	23%

GEOGRAPHIC DISTRIBUTION

Northeast	19%
Southeast	12%
Midwest	19%
Northwest	9%
Southwest	11%
Canada	9%
India	3%
UK/Ireland	5%
Other	13%

YEARS IN SOFTWARE INDUSTRY

5 years or fewer	5%
6 to 10 years	12%
More than 10 years	83%

YEARS AT PRESENT COMPANY

5 years or fewer	49%
6 to 10 years	19%
More than 10 years	32%

YEARS IN CURRENT POSITION

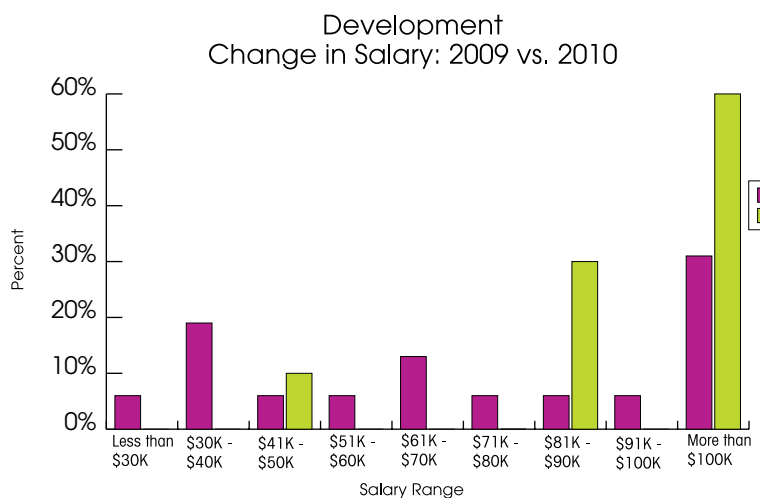
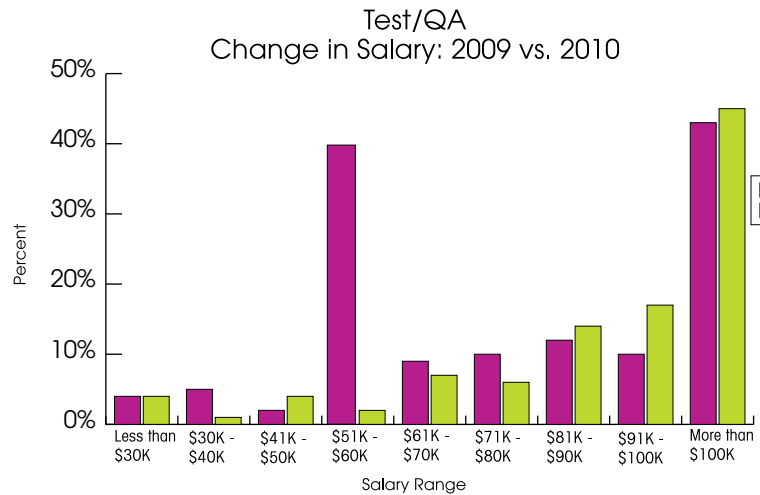
5 years or fewer	33%
6 to 10 years	23%
More than 10 years	44%

JOB FUNCTION (MORE THAN ONE FUNCTION MAY APPLY)

Test/QA	76%
Development	7%
Project Management	29%
Business Analysis	3%
Other	11%

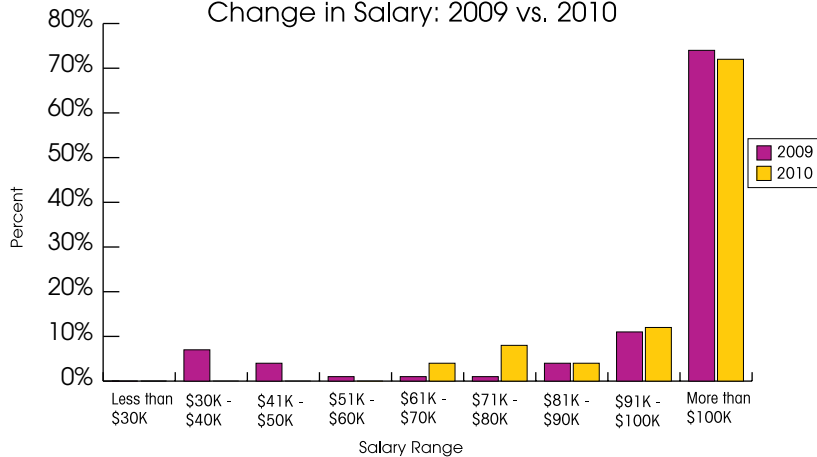
CERTIFICATION STATUS

Yes	40%
No	60%

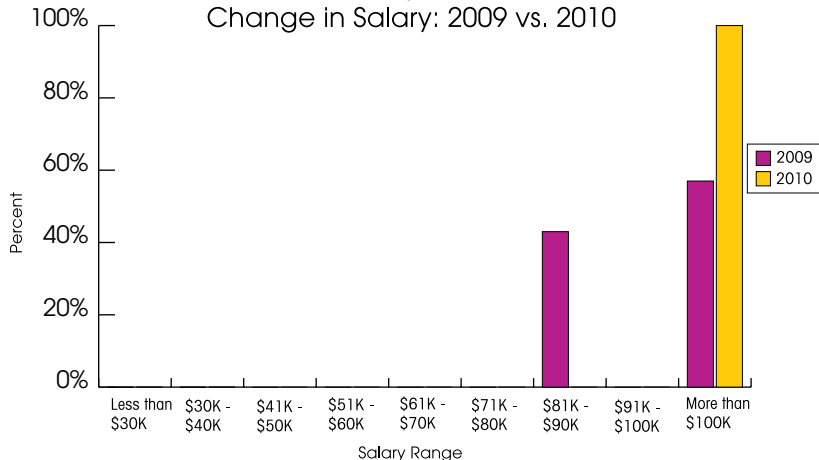


DIRECTOR LEVEL

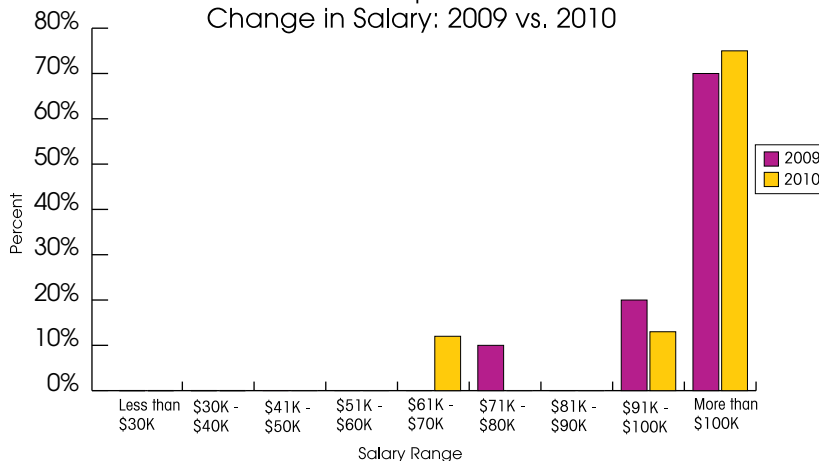
Test/QA
Change in Salary: 2009 vs. 2010



IS/IT
Change in Salary: 2009 vs. 2010



Development
Change in Salary: 2009 vs. 2010



GENDER

Male	73%
Female	27%

AGE

40 or under	30%
Over 40	70%

EDUCATION

Some college	16%
Bachelors	52%
Masters or higher	32%

DEGREE

CS/IS	40%
Engineering	26%
Business	5%
Liberal arts	13%
Other	16%

GEOGRAPHIC DISTRIBUTION

Northeast	38%
Southeast	9%
Midwest	13%
Northwest	7%
Southwest	9%
Canada	11%
India	2%
UK/Ireland	4%
Other	7%

YEARS IN SOFTWARE INDUSTRY

5 years or fewer	0%
6 to 10 years	4%
More than 10 years	96%

YEARS AT PRESENT COMPANY

5 years or fewer	46%
6 to 10 years	27%
More than 10 years	27%

YEARS IN CURRENT POSITION

5 years or fewer	16%
6 to 10 years	23%
More than 10 years	61%

JOB FUNCTION (MORE THAN ONE FUNCTION MAY APPLY)

Test/QA	56%
Development	18%
Project Management	31%
Business Analysis	0%
Other	27%

CERTIFICATION STATUS

Yes	37%
No	63%