DEDICATED TO SHOWCASING NEW TECHNOLOGY AND WORLD LEADERS IN SOFTWARE TESTING

LogiGear MAGAZINE

MOBILE APPLICATION TESTING

FEATURE ARTICLE

Extending Test Automation To Mobile Testing

By Gal Tunik, Hewlett-Packard Company

RELATED ARTICLE

What Matters in Mobile Testing?

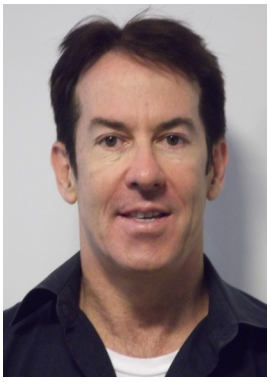
By John Roets, ITX Corporation

SPOTLIGHT INTERVIEW

Robert V. Binder
CEO, mVerify Corporation

November 2011 | Volume V | Issue 7

EDITOR'S LETTER



Everything is mobile. What else can we say? Everything. If your product or service is currently not, it will be very soon. As Apple says: "There's an app for that." There is an app for everything. The race for mobile apps has consumed the software development world.

I did a few projects at Palm Computing in the late 90s. Their Palm Pilot became the first commercially successful handheld computer. It was an exciting time. And still, to have so many people around the world buying these devices seemed a bit odd — why would so many people want to carry around a little computer with them? At the same time, I did a few projects at Unwired Planet (which became Openwave) as well. Openwave is historically significant in its introduction of the Mobile Internet. With its implementation of WAP (wireless application protocol) mobile phones with a browser were an odd new invention where you could have the internet on your phone — anywhere!

Now, these two work experiences seem funny themselves; it seems like ancient technology despite its introduction less than a dozen years ago! The world has truly changed. It now seems odd to not have a smart phone. This new technology revolution is not about phones. The 2010 US Census Bureau canvassing of over 300 million Americans was completed using hand held computers. It saved millions of dollars in paper, data correlation and salary costs. This success was not without problems. There were data transmission problems in rural areas and there were update and patch problems that were said to cause "dramatic dysfunction." But still, this hand held computed was a big success, saved millions of dollars and revolutionized the census process.

With these "first of their kind devices," testing needed new and innovative thinking. Testing devices, operating systems, applications, performance, connectivity, download speeds, security, carriers, and more, were new to most testers but very often carried out by previously skilled black box, desktop testers. The testing effort required quickly learning new technologies, trying to find tools on new platforms, and leveraging previous testing experience to rapidly test, improve quality and minimize risk. Now those same test jobs seem slow and simple. The rate of change has exponentially increased. The speed of code and product delivery has increased. Competition has significantly increased. Our test strategies and the testing itself must be more dynamic, agile, responsive, automated and effective. This will not be easy.

We want to get you started on the right path! In this issue, Gal Tunik addresses the importance of test automation in mobile applications, Robert Binder answers questions from testers currently working on mobile projects, Blogger of the Month Harshal Kharod offers a checklist for testers on what they should look for while testing and our 2010 Global Survey series continues focusing on respondents and their common challenges in testing management politics.

We will continue providing information in this budding area of testing in the near-future. We are sure to have another issue on mobile testing in 2012!

Michael Hackett
Senior Vice President
Editor in Chief

IN THIS ISSUE

Gal Tunik

Departments

5 BLOGGER OF THE MONTH

Harshal Kharod

The Mobile Guru

21 VIET NAM SCOPE

HCMC: Motorbike City

Ingenuity reigns as the Vietnamese prove that cars are second best when it comes to transporting heavy loads and entire families.

6 FEATURE ARTICLE

Extending Test Automation to Mobile Applications

Gal Tunik, Hewlett-Packard Company

Tunik explains the necessity of test automation in mobile applications, citing the unique challenges that platform raises.

9 RELATED ARTICLE

What Matters in Mobile Testing?

John Roets, ITX Corporation

Testers today must deal with a whole new bag of tricks testing mobile devices. Roets highlights key challenges testers should be aware of in preparation for testing.

10 SPOTLIGHT INTERVIEW

Robert V. Binder

As CEO and founder of mVerify Corporation, Binder shares his strategic considerations in mobile testing, from dealing with single stack apps versus globalized enterprise mobile apps, to methods and tools used in the field.

14 2010 GLOBAL TESTING SURVEYS

Politics in Testing

LogiGear VP Michael Hackett discusses the results of the seventh installment of the Global Surveys, focusing on common management challenges in software testing.

BLOGGER OF THE MONTH

Harshal Kharod

Blog: The Mobile Guru

The phenomenal growth of smartphones has opened up avenues for organizations to integrate the devices into the mainstream computing environment. Today's mobile applications manage to deliver complex functionality on platforms that, by themselves, have limited resources for computing.

Unlike the PC-based universe, the mobile environment is comprised of a number of devices with diverse hardware and software configurations and communication intricacies. This diversity in mobile computing environments presents unique challenges in application development, quality assurance, and deployment, necessitating unique testing strategies. Many enterprise applications that were originally deployed as desktop/web applications are now being ported to mobile devices.

These applications are empowering workforces in varying fields, especially those in sales, the supply chain, field support, and on the shop floor. Mobile applications, despite the limited computing resources at their disposal, are often built to be as agile and reliable as PC-based applications.

To meet the challenge, mobile application testing has evolved as a separate stream of independent testing.

The goal of mobile application testing is not to find errors. Perhaps your developer has actually done a great job and made no mistakes. Instead, your goal in mobile application testing should be to understand the quality of your offering. Does it work? Does it function as you thought it would? Will it meet the needs of end users so that they are delighted and come back to your app again and again?

But when it comes to mobile application testing, there are unique challenges, like device compatibility, browser compatibility (if it's a mobile web app), OS compatibility, UI compatibility and what-not. You can't just ignore various devices and platforms and launch your mobile app casually. You need to succeed. And to succeed, you need to have a robust mobile application that works across the various platforms and devices you intend it to run on. So how should you move forward?

Here's a checklist to help you in your approach:

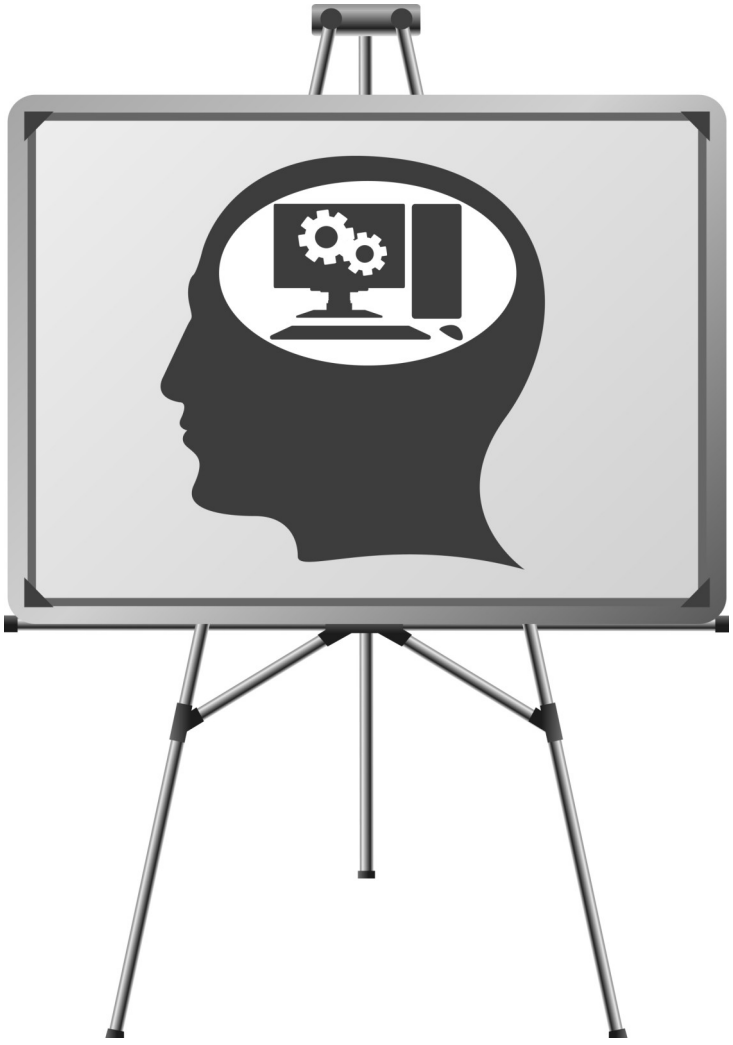
1. Understand the network landscape and device landscape before testing.
2. Conduct mobile application testing in an uncontrolled real-world environment.
3. Select the right automated testing tool for the success of your mobile application testing program. Some important considerations: a), one tool should support all desired platforms; b), the tool should support testing for various screen types, resolutions and input mechanisms; and c), the tool should be connected to the external system to carry out end-to-end testing.
4. Use the Weighted Device Platform Matrix method to identify the most critical hardware/platform combinations to test. This will be especially useful when the number of hardware/platform combinations is high and time to test is low.
5. Check the end-to-end functional flow in all possible platforms at least once.
6. Conduct performance testing, UI testing and compatibility testing using actual devices. Even though these tests can be done using emulators, testing with actual devices is recommended.

Performance should be measured only under realistic conditions of wireless traffic and maximum user load. Keep in mind the mischievous customer who will always try to find bugs in your app!

Companies intending to deploy mobile applications must plan their testing strategy across manual and automated testing approaches for efficient and error-free delivery. In addition to actual device-based testing, emulators should be included as an integral part of the mobile application testing program. Enterprise applications require special testing techniques. Partnering with a third-party vendor who operates an independent testing practice can be a viable option to manage the expertise, scalability, and quality assurance requirements of mobile application delivery. ■

FEATURE ARTICLE

Extending Test Automation to Mobile Applications



Gal Tunik *explains the necessity of test automation in mobile applications, citing the unique challenges that that platform raises, including the intricate differences between mobile and desktop interfaces, and performance test results for applications as well as mobile networks.*

This article was originally published in Software Test & Quality Assurance Magazine in the March/April 2011 issue.

It is no surprise in this day and age of instant-on gratification that mobile devices are gaining momentum. People have become accustomed to accessing information when they want it, wherever they are, and the mobile device is one way to stay connected. Across all industries, companies are realizing that to remain competitive, they must provide customers with ways to make everyday connectivity to the world easier. However, in order to create a successful mobile application, it is important that companies conduct the necessary tests to ensure that it is market-ready.

Accelerating this momentum is the hype surrounding such new technologies as the superfast 4G network, bar codes, Near Field Communication (NFC) and augmented reality. These all provide companies with attractive business opportunities and spur innovation. More and more, companies are increasing efforts and adding resources to develop mobile applications. Twenty-four percent of executives in charge of their companies' mobile strategies reported that they would more than double their spending in the upcoming year, according to a recent independent Forrester Research survey[1] of mobile executives. Additionally, 46 percent stated they would increase their budgets.

Mobile applications have revolutionized how people conduct business. For example, using a smartphone, a retail store manager can check the exact time product inventory is due in her store. Nurses can quickly access patients' health records from the examining room and sales representatives are able to multi-task and report on deal statuses while on their way to the airport.

However, as with any emerging technology, a number of unique challenges can arise when developing and implementing mobile applications. Mobile applications don't simply imitate the desktop environment—they have their own user interface (UI) requirements, business process flows, and infrastructure dependencies.

In addition, there are a growing number of mobile devices—smartphones, PDAs, tablets, and specialized devices such as a delivery clerk's scanning and tracking handheld. In recent years, there have also been a number of different mobile operating systems. To add even more complexity, there is a choice of implementation options—such as thick vs. thin client. All of this increases the strain on IT to build, port and maintain mobile applications. It also heightens the risk of quality and performance problems.

Functional Validation

One way to curtail glitches is to make certain mobile applications go through functional validation. This requires testing applications for mobile devices, similar to how desktop-based applications are tested, to ensure usability. Functional validation is based on the concepts of collaboration, test asset management, reusability, accurate reporting and analysis. Similar to the desktop application world, the bulk of testing on mobile devices is currently done manually.

There are some who are skeptical about the accuracy of manual testing. Can a test engineer simply drive to a remote location and log into the credit union's website to check his account balance? Or, can he see whether he still gets a good network connection while delivering a package to a far-away golf course? The problem with manually testing mobile applications is the lack of coverage and scalability, as well as the associated costs, especially if an application needs to be re-tested multiple times to ensure development and quality. While a test engineer can manually key in a handful of transactions, he cannot continuously test and monitor the connection for application availability or test for all possible scenario permutations. Additionally, a tester is not able to quickly identify problems, fix them, and re-run all the tests required to validate the fix.

Another popular debate is whether testing should be done on an application's physical device versus using emulators. However, the question should really be: how should we test both native applications and mobile web applications to achieve the required quality?

Many industry experts argue that using handsets for testing, as opposed to testing the application on a normal browser or emulator, requires an overhead of logistics. However, in order to ensure that they function and perform properly, it is imperative that applications be tested on mobile handsets. There are several reasons for this, ranging from the theoretical to the practical:

- For one, usability testing on emulators and browsers with any extensions do not represent what will be seen on the actual device. With Android specifically, emulated "vanilla" environments almost never represent the actual behavior of the phone, as vendors tend to modify

the "source code" of the handset and add their own customization layers.

- Environment-related testing is critical as mobile applications rely on the location, accelerometer and network. Simply put, the test cannot be done in a simulated environment.
- Another factor is interoperability. A mobile handset "lives" in a communication environment and behaves accordingly. For example, an incoming phone call always receives priority over any application that is in use. Testing scenarios that see what happens to an application when a phone call is received is critical, as this is an everyday occurrence.
- As many mobile users travel internationally, the proper network-related testing must be conducted. For instance, an application developed for Vodafone in the UK does not necessarily work on the Verizon network in the US. Network neutrality has not been integrated in the cellular world, which means traffic manipulation and modification is still very common in the cellular environment.
- A common concern amongst mobile users is security. People are sensitive to data such as bank account numbers that remain on handsets, or passwords that are displayed on the screen. Testing for these types of security concerns in a simulated environment is not a good use of time, because what is required to be tested is the behavior of the actual handset. Adding to the challenge, not all handsets have the same security designs, so each device must be individually tested.

Performance Validation

A second aspect of the proliferation of mobile applications is the need for performance validation. Applications must be validated to ensure performance expectations are met. The user expectations from these applications may even be higher than those of their desktop counter-parts. Performance validation for mobile applications is crucial and, at times, more complex than that of desktop applications.

Mobile applications, by their very nature, answer a desire as well as a need for access to information "24/7". Quite often they are built to provide access to a critical subset of desktop applications. These micro-business processes must always be readily available, to ensure businesses continue to function even when no one is in the office.

Another important performance factor is that mobile applications are affected by mobile network conditions, such as bandwidth limitations, which are more pronounced than those of land-line networks. These restrictions may adversely affect the communication between a handset and the back-end servers. Automating performance validation is needed to reach the large number of simulated users that will be accessing the tested application at any given moment. This can be achieved through simulated users running on emulators and browsers. Without this kind of automation, it is virtually impossible to create a realistic load on the application servers and infrastructure.

When replicating the appropriate network conditions, it must be ensured that a simulated mobile user is experiencing the same bandwidth throttling and network delays that a real mobile user would. In order to complete the test, real handsets executing the relevant business processes should be running parallel to the automated performance test. This allows the end user experience to be measured, while also testing the behavior of the application's servers and infrastructure under load.

To appropriately test mobile applications, it is best to bring together a large number of browsers and emulator-based virtual users, simulations of mobile network conditions, as well as actual handsets to test business processes. This not only ensures performance validation, but provides comprehensive and accurate end-to-end results for mobile applications.

Conclusion

Automated functional and performance testing has a proven track-record in desktop-based applications. It helps companies deploy higher-quality software applications, reduce business risk, and accelerate problem-to-resolution. This ultimately helps organizations avoid costly interruptions and errors. The growing demand for instant information, especially through mobile applications, should encourage organizations to adopt automated testing strategies and solutions that are designed specifically for their mobile application needs. ■

1. Forrester Research, Inc. "2011 Mobile Trends," January 24, 2011

Gal Tunik *is a senior product manager of Unified Functional Testing and Sprinter Software at the Hewlett-Packard Company. Previously, Gal served as an associate strategy consultant and key member of the IT Strategy Group for Booz & Company, where he designed market entry strategies and defined governance models for customers' IT organizations. Prior, Gal worked at SAP, as a senior consultant and later as a solutions architect. Gal holds an MBA from the Kellogg School of Management, Northwestern University, and a BA in Computer Science from the Interdisciplinary Center in Israel.*

RELATED ARTICLE

What Matters In Mobile Testing?

By John Roets

Devices matter. We don't yet trust the mobile devices like we trust desktops and laptops. In the course of testing traditional web applications, rarely do you have to think about the model of the actual machine. In mobile, however, the behavior of an application can vary from device to device. You can no longer just think about browser/OS combinations.

Mobile emulators aren't good enough. They get you part way there, but you still have to physically test on many devices.

The version of the OS can be quite important. Just as you think about the physical device, you also must consider how nuances of each OS version can affect application behavior: iOS 3, 4 or 5; Android 2.1, 2.2 , or 2.3; Blackberry 5, 6, or 7; Windows Mobile 7.0 or 7.5; etc. Add that to your test matrix.

Mobile web apps can become native apps. Tools like PhoneGap and Appcelerator enable web applications to be compiled into native applications. Now you might need to test a version of the application that runs in browsers and a version that runs natively.

Native phone features must now be considered. With the ability to turn web applications into native applications, this opens up access to native phone features, like the camera, accelerometer, address book, etc. These are not things we are all that familiar with.

Mobile web applications tend to be heavy on the JavaScript. Many factors lead to a thick-client architectural style for mobile applications. Business logic is pushed to the client-side — i.e. JavaScript — and the server-side becomes a set of services. Suffice to say that you now may find yourself much more concerned about how to test JavaScript.

Offline capability is important, and may become a first class concern, such as Airplane mode and driving into a tunnel. There are several reasons why a phone can lose connectivity, and mobile applications must be designed to deal with this fact — yet, another thing to add to your testing list.

Connection quality is unpredictable. Remember when you had to worry about dial-up users? It's like that. Even if you assume your target user base will have at least 3G, connectivity can degrade or be lost at any time. One minute you're on Wi-Fi, the next on 3G, and the next on Edge.

Data plans might be metered. Many users do not have unlimited data usage. Now the application architecture and design has to take into account how it will need to respect the user's data plan. And you have to test it.

Instrumentation and test tool support is lagging on mobile devices. Any time there is emerging technology, the development of supporting tools (like testing tools) lags. For example, there are not currently a great many mature tool options available to automate tests of mobile applications across the spectrum of devices, OSes, browsers, etc.

The network provider might matter. Perhaps a particular network provider throttles connection speeds. Perhaps a particular network provider configures its phones in a particular way that might be important to know.

You have a lot to consider building your test matrix for testing mobile applications. The test strategies are still evolving. We are all still figuring out what matters in mobile application testing. Good luck! ■

John Roets *has been providing software engineering leadership, specifically to application development for the web, since 1998. John is currently a Senior Architect at ITX Corporation in Rochester, NY; providing solutions to a multitude of clients, focusing his attention (at the moment) on mobile applications. He is just as likely to engage you in discussion about design patterns, databases, and the latest web technologies as he is about business strategy, software development methodologies and organizational structure. John can be contacted through* http://jroets.blogspot.com/

SPOTLIGHT INTERVIEW

Robert V. Binder

CEO and founder of mVerify Corporation, Robert V. Binder tackles questions from field testers regarding such issues as strategic considerations when dealing with single stack apps versus globalized enterprise mobile apps, and methods and tools that developers and testers should be aware of. He also offers his own advice from lessons learned from experience.

1. What are the popular mobile platforms out there?

The handheld platform installed base is rapidly evolving, according to Comscore. From May to August of 2011, Android's share of US smartphones was up 6% to 43.7%. Apple's iOS increased slightly to 27.3% in the same period, while RIM's Blackberry OS slipped 5 points to a 19.7% share. Windows Mobile was nearly unchanged at 5.7%, with Symbian shrinking to 1.8%, down from 2.1%.

Global shipments of tablet devices for Q3 2011 were up 280% year-over-year to 16.7 million, according to Strategy Analytics. Sixty-seven percent of this was iOS (iPad only), 27% Android (seven versions; shipped on Motorola Xoom, Samsung Galaxy and others), 2% Windows (desktop and mobile OS), and RIM's BlackBerry Tablet OS at 1%. Also contending are Kindle, MeeGo, and Chromebook. WebOS was killed, but could reappear.

Mobile applications are not limited to handheld devices. Telematics and other systems that rely on wireless data links also use mobile platforms. Windows Mobile has the dominant share of the installed base for embedded and industrial applications which do not use a hard real-time OS. For example, Ford's Synch system and ATT's Uverse set-top boxes use variants of the Windows CE stack and run on many millions of endpoints.

A good survey of this rather complex space can be found at http://en.wikipedia.org/wiki/Mobile_operating_system

2a. With Agile development, Continuous Integration strategy, along with the wide variations of mobile platform, what are the challenges for testing and test automation for mobile apps?

First, let's define "mobile apps" broadly – not just as a simple download from an app store. Let's also assume that the risks of releasing a buggy mobile app can be much worse than losing a star or two from your ratings.

The typical practices of Agile development and continuous integration can be followed for mobile applications – these strategies are platform-agnostic. However, mobile stack tooling is limited compared with desktop or server environments. This means that lower productivity can be expected for mobile app development. For example, the features of automated testing tools that run on an actual handheld device are similar to early 1990's tooling for client/server development. See Déjà vu All Over Again – The Mobile Testing Nightmare for more about that.

In addition to limited tooling, the unique dimensions of mobile apps can significantly increase project scope:

- Which platform stacks/versions will be supported? If you plan to release on multiple stacks, you'll need to specialize the code and UI, then test not only on each stack, but also on all supported versions of that stack, even if functionality is identical.
- Do you plan to seek certification for each stack? For example, will you be seeking Windows Logo certification or acceptance in the Apple App store?
- Do you plan to seek carrier certification for that data link? Major cellular carriers (ATT, Sprint, Verizon) have certification requirements for certain kinds of applications.
- Which locales will be supported? Do you need internationalization? Even if your app relies on platform localization capabilities, you'll need to design, develop, and test for each locale. Business policy and regulatory requirements vary considerably. For example, the Eurozone has stricter privacy regulations than the US.
- Which wireless data links (air links) will be supported? High speed? Low speed? Legacy? Cellular? CMDA or GSM? 802.x? To what extent is your app dependent on variation in bandwidth, response time, and air link quality/QOS? For example, if you use a real-time streaming capability, what happens when the signal cuts out, or is switched to a different air link stack?
- What about security? Security hazards for mobile systems are the single greatest business risk mobile apps present. You'll need to understand the appropriate security profile, design and develop for that, and then provide some evidence that its goals are met.
- If the mobile endpoint is a server's client, to what extent can you test round trip transactions? How can you generate workloads that represent typical variations, locales, and modes for peak and off-peak use?

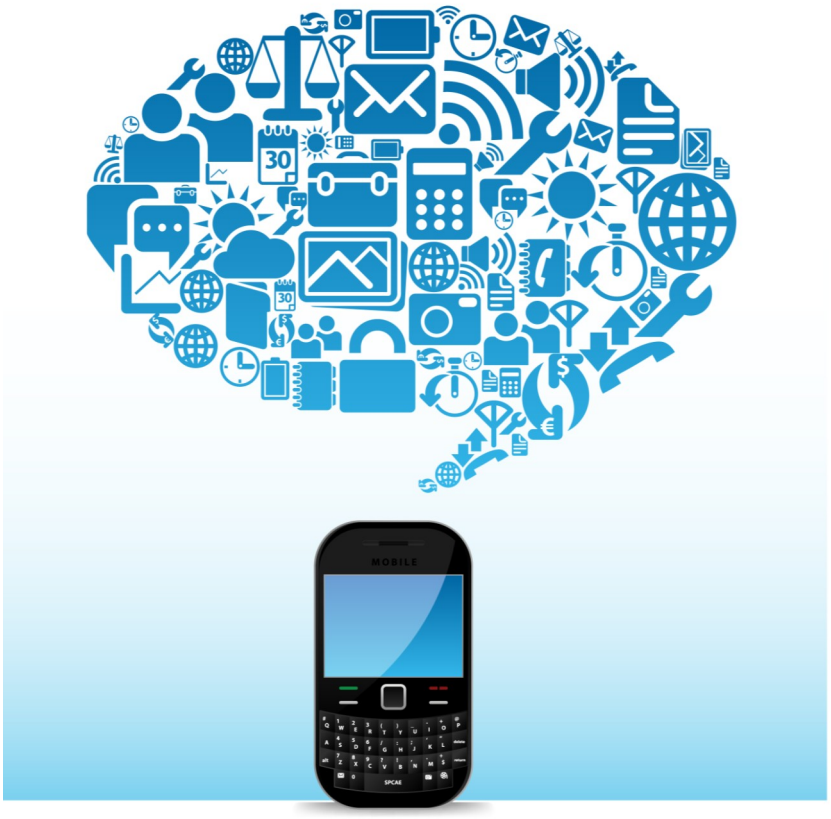
Developers of mobile candy – simple, single stack apps for convenience or amusement – probably can ignore most of these concerns. Developers of globalized enterprise mobile

apps cannot. I'd target one slice of all these aspects first , then develop and stabilize that configuration. Next, taking into account your business, operational, and technical situation, layout and follow a roadmap for additional slices, possibly ramping up with parallel teams.

2b. What resources are available that developers and testers must know?

Clearly, the team needs sufficient skills in the development environments for the targeted stacks. There are two relatively new services that address some mobile app tooling limitations.

- Crowdsourcing of alpha and beta testing can provide coverage of handheld configuration variations that would be otherwise very difficult to achieve. Leaders in this space include Mob4hire and uTest. Developers should be aware that the testing done in this way tends to be uncontrolled and subjective.



- Recently offered cloud-based services can also provide a low cost alternative to covering device and stack combinations, as well as emulating certain aspects of load for the server side. Leaders in this space include Micro-Focus, Compuware, and Keynote. This too has limitations, which should be taken into account in setting a test plan.

3. What are the various types and/or strategies of testing mobile applications one should consider?

I think the well-established patterns for testing functionality and performance are applicable to mobile apps, including the test design patterns in my *Testing Object-Oriented Systems: Models, Patterns, and Tools*. The new challenge is in recognizing and managing the hazards arising from the configuration combination explosion, also known as the "mobile testing nightmare."

4. What benefits do you see Cloud Computing and virtualization bringing to mobile application testing?

The cloud modality has enabled crowdsourcing, shared emulation, and on-demand performance testing. The testing benefits of virtualization are less clear. Virtualization may allow us to run more than one mobile emulator or stack on a single computer. However, even a virtualized stack still supports only a single developer or tester. Although reducing overall hardware costs is useful, I don't see that it will have a significant individual productivity effect. In a large lab, there is a possibility of significant reduction of setup and configuration work.

5. What are the issues with performance testing for mobile apps one should consider, and do you have recommendations for testing solutions?

The ideal configuration for performance and load testing is the deployed target environment. With mobile apps supporting millions of diverse endpoints, it is rarely feasible to replicate this in a dedicated test lab. So, the crowdsourcing, emulation, and load-generation services mentioned in 2b are worth considering, but with their limitations, there remains a risk of uncovered failure modes and bugs. I think testers are between a rock and hard place with this. A step-by-step approach can help to mitigate these risks.

First, devise a configuration roadmap. Then, for each configuration, do alpha testing in your own lab and/or a virtual lab, and follow that with crowd sourced beta testing. Then make a general release to a configuration subset of the user population, timed so that cyclical peak loads can be closely monitored. When that configuration subset is stabilized, repeat for the next roadmap milestone. Subject to budget and other operational considerations, some of these steps can be done in parallel to minimize time to market.

6. Do you have thoughts on security testing for mobile apps? Any practices, methods and tools the developers and testers should apply?

Why are banks robbed? The (apocryphal) answer is "That's where the money is." The huge and growing amount of personal and financial data stored on handhelds and transmitted over the air is a lucrative target for criminals, spies, and vandals. At the same time, handheld ubiquity and personal attachment to our devices ("crack-berry", etc.) can induce a false sense of security. So, it isn't surprising that mobile exploits are surging.

For designers and developers, I think this means security should be a first class aspect of a mobile system. Technical strategies to minimize the attack surface are not as well evolved for mobile stacks as they are for desktops or servers, so pay attention to emerging exploits and be paranoid. For testers, this means identifying abuse cases and aggressively probing for security flaws, as well as for typical bugs and failure modes.

7. What are some of the top lessons learned testing mobile apps you might have and want to share?

First, be sure to weave the unique characteristics of mobile endpoints into your test plan. For example, try critical updates while powering off or as the battery fades out. Interrupt usage with incoming calls, text messages, Bluetooth conversations, etc. Power cycle and check state. Install/uninstall updates. Set up a streaming video upload/download and then attempt to enter a transaction that should update a remote server. Establish a WiFi connection and repeat. Repeat on the smallest/worst quality display handheld and the biggest/best. For position-sensing devices, find a shaker and rock. Identify usage patterns that will drain the battery quickly – turn on all the radios, upload/download high data rate content, and run an app that saturates the CPU.

Second, systematically vary air link quality as part of your test plan. When I started developing mobile test automation ten years ago, I met a tester in New York City who would take a box full of handhelds into the subway. He'd found a spot on a subway platform that got zero bars from any carrier, but only a few feet away, got weak reception. He'd walk from the zero signal to the weak signal area to test how robust the apps were to drops, repeating this until he'd worked through all the devices.

Third, do not ignore the well-established patterns and lessons of testing: don't limit your testing to sunny-day scenarios. Define a complete and realistic usage profile, then exercise each use case in proportion to its real-world use. Start your test planning as soon as possible. Work with developers to evaluate evolving designs/implementations for testability.

Finally, plan for and make the best use of fingers and eyeballs. Automate where it makes sense, but with today's tooling limitations, plan on a lot of manual testing. This doesn't mean testing by poking around – think through, document, and follow a strategy that best covers the unique aspects and risks of your app and its configurations. ■

Robert V. Binder is a business leader, serial entrepreneur, and software technologist with extensive systems engineering experience.

As President of System Verification Associates, he has provided solutions for clients facing existential regulatory challenges. As CEO and founder of mVerify Corporation, he took a unique solution for mobile app testing to market. He led RBSC Corporation's consulting practice in software process and advanced software testing, delivering expertise and solutions globally.

Binder has developed hundreds of application systems and advanced automated testing solutions, including two projects released as open source. He was awarded a U.S. Patent for model-based testing of mobile systems.

He is internationally recognized as the author of the definitive Testing Object-Oriented Systems: Models, Patterns, and Tools *and two other books.*

2010 GLOBAL TESTING SURVEY RESULTS

Politics of Testing

*LogiGear Vice President **Michael Hackett** discusses the results of the seventh installment of the Global Surveys focusing on common management challenges in software testing.*

Few people like to admit team dynamics and project politics will interfere with successful completion of a software development project. But the more projects you work on, the more you realize it's very rare that technology problems get in the way. It's always the people, project planning, and communications issues that hurt development teams the most.

Historically, test teams bore the brunt of ill-will on projects in traditional development models. Since their work was last, comments often included, "They are slowing us down," and "They are preventing us from hitting our dates." And, when the test team misses the seemingly obvious bug, their entire work is suspect.

Even in Agile, issues of being "cross-functional," inclusion in the estimation process, and extremely *lean* {read: none} documentation can exacerbate some political problems for people who test.

It is no accident that the first *value* of the Agile manifesto is "Individuals and interactions over processes and tools." People, and how we get along with each other, consistently have a larger impact on project success than any process or tool. Communications problems are most commonly cited as the biggest problems with software development.

I had hoped we have moved away from the worst work environments since we now have so much information and experience that politics and difficult work situations badly impact productivity and work quality. We should resolve these problems more quickly when they arise.

As I review the results of this survey, it seems we still have far to go to fix team political problems. A better understanding and awareness of common political problems will help you recognize them more quickly and give you an opportunity to remedy situations before they get worse.

PART 1

This survey was quite large. The first half of the survey is located in this magazine, at the end of this article you can click to Part 2.

1. What is the biggest pain area in your job?

Duplicated responses were deleted.

- "Poorly Written or insufficient requirements leading to scope creep."
- "Delayed release to QA creates immense testing pressure in terms of meeting deadline."
- "Not enough resources for development or testing."
- "Timetables to testing."
- "Thinking alone with no help."
- "Making teams understand the importance of QA participation early in the life cycle."
- "Time pressures and lack of understanding in regards to testing by external parties."
- "Estimating time to automate on clients' perceptions."
- "Educating people on the value of testing, tester skills, domain knowledge - for example, helping managers understand that quantifiable measures related to testing are generally weak and sometimes risky measures, especially when used independently (like number of bugs, or number of tests run/passed/failed, etc.). Demonstrating how to use qualitative measures more effectively and advise managers/stakeholders on trusting the intuitive assessments of senior testers to make decisions."
- "Keeping track of schedules and keeping test cases up to date."
- "Complete lack of any process at all - all projects done uniquely = chaos."
- "Domain knowledge."
- "Production and testing environments are not 100% synced."

2010 Global Testing Survey Results Available Online

Overview

Agile

Automation

Test Process & SDLC

Methods

Tools

Metrics

Offshore Survey

- "Dealing with some executives who do not understand a thing about testing."
- "Having other team members value my opinion when it comes to process improvement, as well as scheduling for releases correctly - giving unrealistic expectations and time frames."
- "Explaining the GAP between theory and practical."
- "Testing rare scenarios."
- "Developer says, 'This is not reproducible on my machine' and marks the status back to 'Can't Reproduce'" I call him back and say 'It's 100 % reproducible on my machine. Come here, look and fix the problem.' Sometimes have to reproduce the issue many times for better debugging and this eats up lot of time."
- "Managing resources and data for testing. Such as Database servers, Different environments, etc."
- "Underfunding for new projects."
- "People's unwillingness to choose testing as a career path."
- "No clear career path."
- "I don't have access to the code base."
- "Creation of reports."
- "Balancing on-the-job learning with providing results."
- "Over reliance on testing quality into a product, application or service."
- "The need to repeatedly prove that independent testing adds value."
- "Writing good test cases and having to retest the application due to late changes in requirements."
- "Developers are not doing proper unit testing and give QA build late for testing. This results into - QA Team stays late."
- "Not being trusted to know what I'm doing."
- "Delivery dates, testing work is not measured or estimated."
- "Deliver documentation from regulatory and quality assurance perspective and not tester perspective."

- "Not giving much time to do regression testing and also not giving time to developers to fix the defects."
- "HR issues with certain personnel."
- "Finding good pragmatic QA lead, who understands solutions architecture, solutions design and the value solutions bring to customers."
- "Deadlines driving releases, not schedules or quality."
- "Having to do the same things day in day out."
- "Getting the project organized before code is produced and sent for testing."
- "Switching in multiple projects daily."
- "Changing requirements, unavailability of business experts to inform requirements gathering process."
- "Designing the effective testcase."
- "Repeated releases due to very small changes in requirements."
- "Receiving timely information."

Analysis: The universality of problems in software development continue to astound me. On one hand, most people think their team and company problems are unique — they rarely are. Secondly, the problems are so universal and well known — why haven't organizations moved to fix such common issues that would have a big impact on productivity and quality?

Also, Testing and QA political problems are sometimes confined to testing, but are much more often a result of problematic corporate culture.

2. Does your development team value your work?

	Response percent	Response count
Yes	57.9%	73
Somewhat	38.1%	48
No	4%	5

Analysis: Most teams are valued. The response of 58% is actually a low number here — the number should be closer to 100%!

3. Who is ultimately accountable for the high quality or poor quality of the product?(QA- quality assurance/ guaranteeing quality)?

Everyone	53.8%	64
Project manager	16%	19
Testers	14.3%	17
Product team (marketing, business analyst)	7.6%	9
Upper corporate management	5%	6
Developers	3.4%	4

Analysis: The only correct answer here is "Everyone." I often ask this question to clients and it is usually a much higher percentage. Everyone on the team has unique and important contributions to the level of quality. Any weak link will break the chain. We all share responsibility.

4. Who enforces unit testing?

Development Manager	37.8%	45
Developers	32.8%	39
Testers	16%	19
Project Manager	10.9%	13
Product Management	2.5%	3

Analysis: Over 70% respond that the "Development Manager" or "Developers enforce unit testing" is the correct answer. The 16% responding Testers is wrong.
Just in the past few years as Agile—particularly XP practices such as TDD— are finally becoming more common unit testing practices are becoming much more common.

5. What is the most important criteria used to determine if the product is shippable?

Risk assessment	43.7%	52
Completion of test plan/sets of test cases	22.7%	27
Date	17.6%	21
Bug counts	14.3%	17
Code turmoil metrics	1.7%	2

Analysis: Interesting to see how the ship or release criteria differ greatly from company to company.

6. What is your perception of the overall quality of the product you release?

Good	58.1%	68
Average	21.4%	25
Excellent	17.1%	20
Poor	3.4%	4
Bad	0%	0

7. What pain does your product team have regarding quality? (You can select multiple answers.)

Insufficient schedule time	63.8%	74
Lack of upstream quality assurance (requirements analysis and review, code inspection and review, unit testing, code-level coverage analysis)	47.4%	55
Feature creep	38.8%	45
Lack of effective or successful automation	36.2%	42
Poor project planning and management	32.8%	38
Project politics	31%	36
Poor project communication	26.7%	31
Inadequate test support tools	25.9%	30
No effective development process (SDLC) with enforced milestones (entrance/exit criteria)	25.9%	30
Missing team skills	23.3%	27
Low morale	15.5%	18
Poor development platform	7.8%	9

Analysis: These responses speak for themselves. Insufficient schedule for testing, lack of upstream quality practices, feature creep, lack of effective automation; anyone familiar with software development could have predicted this. A larger problem here is that we have had these same problems since 1985!

8. What is the most important value of quality?

Ease customer pain/deliver customer needs, customer satisfaction	52.1%	61
Better product (better usability, performance, fewer important bugs released, lower support, etc.)	40.2%	47
Save money during development (cut development and testing time to release faster)	7.7%	9

9. Who ultimately decides when there is enough testing?

Test team	35.6%	42
Project manager	31.4%	37
Upper Management	14.4%	17
Product (marketing, product, customer, biz analysts)	14.4%	17
Developers	4.2%	5

Analysis: Surprising answer here. Remember this survey is primarily made up of testers. I would have expected "Product decides" or "Project Management decides" as the number one answer.

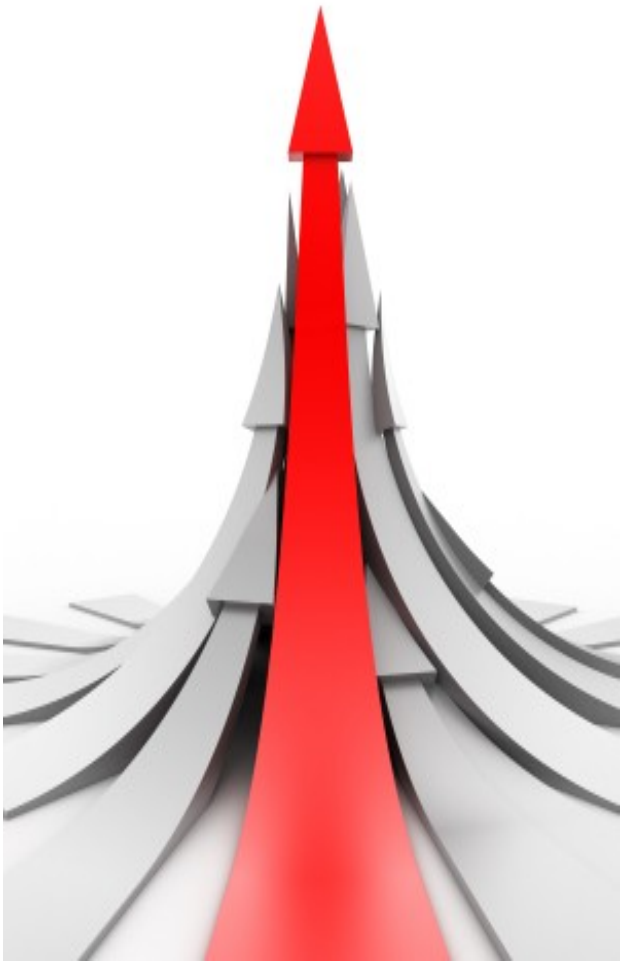
10. How often does your testing schedule not happen according to plans?

Some/few	38.1%	45
Often	26.3%	31
Most	25.4%	30
Every project	6.8%	8
Never	3.4%	4

Analysis: From these responses, clearly projects rarely happen according to plan. Over 57% responded "Every, Most or Often" projects do not happen as planned. Have project planners not learned much in the past 20 years?

11. What are the main reasons test schedules do not happen according to plan?
Duplicated responses were deleted.

- "Dev issues or dependencies on hardware."
- "Scope Creep"
- "Requirements changes, greater than average defect count."
- "Change of release content."
- "Resource conflicts - our testing resources are not dedicated to testing, it is a function they perform along with their normal job functions."
- "The deadlines for the delivery of the projects are short."
- "Unstable or late deliverables, changing requirements or scope." Jun 23, 2009 10:23 18. "Usually dev slip the date or too many problems (bugs)."
- "Emergency releases."
- "Unexpected events, additional workload, new directives, rework ..."



in the production release date, the days for QA are shortened."

- "Inconsistency in development process."
- "1. Delay in releases from development team.
  2. Scope creep during testing phase.
  3. Rework of fixes due to lack of unit testing and poor quality of the component released to QC."
- "Lack of quality Development and so the repetitive dev cycle."
- "New Feature requests injected in between a sprint."
- "Major bugs that prevents testing of some areas."
- "Lack of understanding of new product/enhancements."
- "Other, equally high-priority deliverables (in products/ features in different ship vehicles)."
- "Unrealistic and well vetted project plans, schedules and estimates."
- "Lack of unit testing by dev team, due to which when build is given to testing team, then test team got blockers etc. This increases test cycles."
- "Never actually included in scheduling. Developers decide what goes into the next release based on development effort. Testing is an afterthought."
- "Unexpected user issues that come up; problems with software that our product integrates with."
- "Late turnover."
- "Environment stability."
- "Analysis/Code not complete on time for test to begin."
- "Multiple developers putting in changes right before deadline."
- "Multiple projects at one time."
- "Lack of funds."
- "Issues with product installation, server problems, unrealistic testing schedules assigned by upper management."

Analysis: Scope creep is by far the number one answer. (I removed duplicate answers.) It comes by many names — all meaning addition of features after agreed upon schedules. Hopefully, as Agile — particularly SCRUM practices such as the Planning Game, Sprint Planning and a strong Product Owner — becomes more common, one day scope creep will become just a bad memory.

12. How do you decide what to test or drop (prioritize coverage) if your testing schedule/resource is cut in half?
Duplicated responses were deleted.

- "The CEO."
- "What feature is critical and test positive cases not negative."
- "Project Team Decision"
- "Focus on high risk areas that would cause customer dissatisfaction."
- "1. Products' basic functionality.
  2. Severity of corrections.
  3. Areas affected by corrections.
  4. Importance to customers.

5. Importance to sales and marketing.
6. Ability to simulate in test lab."

- "We don't drop testing, we delay deployment of features if they cannot be tested."
- "What code has been changed the most."
- "Negotiate with developers and Project Management."
- "Core test case coverage."
- "Based on the historic data and feature change: what has been tested in previous releases and what new features has been introduced/changed."
- "Depends on the requirements prioritization and their impact in de software functionality."
- "Speak with the business to prioritize the most important/critical items to receive testing and/or personal experience of where the pain points are going to be (where do we often see the most issues with previous testing?)."
- "Project management and application owner."
- "Knowledge of customer usage, therefore impact."
- "Technology and business risk assessment. Mostly from the QA/Test team. Not ideal but the best out of a challenging situation."
- "Combination of time required, business criticality and business priority."
- "Discuss the Functionalities with Business Analyst/ Product Manager and decide priorities."
- "I think about how the customers are going to use the product. Anything which will affect all customers is the highest priority. Anything affecting individual customers is lower priority. Anything which requires a reset to the server (affects all customers) has a higher priority. Anything with a work around is a lower priority. The more important the customer, the more important their user stories."
- "Items that need to be delivered for launch will be tested."
- "These decisions are made project by project. We use customer feedback and knowledge of which features are most used by customers to decide."
- "Focus on main functionality; leave out additional features that we don't have time to test."
- "Safety critical vs. non-safety critical."
- "Risk Based Testing, risk categorization."
- "Scrum planning."
- "From bug history, new feature, customer usage."
- "ROI"
- "Any test cases with dependencies to new feature are prioritized."
- "Drop multiple passes on relatively simple changes; drop regression testing for portions of system that scheduled changes should not impact."
- "Test main functionality and skip regression."
- "Test cases are ranked H-M-L and we focus on the High. Usually the most desired feature has the most High ranks.Also, it may be decided to drop a feature and only focus on that feature and defer release of other feature. The business ultimately makes that decision of focus."
- "Test features that are crucial for clients acceptance, and / or features which are complex and broad within the solution."
- "Operational requirements."
- "We usually don't. We will throw more resources on a project."

13. Is testing (the test effort, test strategy, test cases) understood by you project manager, product manager/ marketing and developers?

Yes	75.7%	87
No	24.3%	28

Analysis: I often ask this question during my consulting work and it is always a very similar answer: testing is not well understood by a significant percentage of people in the development team. ■

Part 2 of Politics in Testing is located online. Please go to www.logigearmagazine.com to view entire article.

HCMC: MOTORBIKE CITY

Ingenuity reigns as the Vietnamese prove that cars are second best when it comes to transporting heavy loads and entire families.

The industrious cosmopolitan hub of Ho Chi Minh City is globally known as "Motorbike City." To traverse the streets of Saigon is to navigate through a seemingly endless flow of two-wheeled machines, each street evoking visions of high-octane salmon runs. Any tourist who's ever crossed a major thoroughfare in this city is entitled to a certain sense of triumph, akin to having completed an initiation rite. Wading through a stream of 300-pound salmon is not for the faint of heart!

To be fair, there are cars on the streets here — even a Bentley can be seen in downtown District One. But if your goal is to get from the airport to your hotel in less time than your trans-Pacific flight just took to bring you here, you might want to look at alternatives, such as the ubiquitous motorbike taxi.

Cars are nice, and certainly comfortable, but those ponderous beasts are no match for the nimble two-wheelers that snake their ways around them, almost mocking them. Aside from having to obey traffic lights more conscientiously than their two-wheeled counterparts, automobile drivers often find themselves confined to a single lane of traffic, even on major avenues. And to add insult to injury, unlike the case with motorbikes, the police take a very dim view of four-wheeled vehicles driving on sidewalks.

Of course, it's not just efficiency that accounts for the preponderance of motorbikes. With a 93% import tax, the price of even a mid-sized auto exceeds a lifetime of wages for the average citizen. For the typical Vietnamese, car ownership simply isn't even on the radar screen.

Hence, the Vietnamese make do with what they can afford. For many Vietnamese, it's second nature to maneuver bikes through dirt roads, hauling mattresses, large 100-pound hogs, floor to ceiling mirrors, rolls of carpet — even coffins.

Along with being the heavy-utility-and-everything-in-between means of transport, motorbikes are the preferred method of transportation for the Vietnamese family. It's not at all uncommon to see five family members on an outing with the baby sitting in front atop a tall wicker chair (sans helmet, of course – the law only mandates them for adults!), and two other kids sandwiched between mom and pop. At first sight, one may be tempted to nominate Britney Spears for Mother of the Year, for at least driving in an *enclosed* vehicle with a baby in her lap.

Yes, in Ho Chi Minh City, where practicality outweighs safety, the streets are flooded with a sea of bikes performing any and every task. ■

LOGIGEAR MAGAZINE
NOVEMBER 2011 | VOL V | ISSUE 7

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

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

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