PUBLISHED BY POCKET PROTECTOR PRESS - A DIVISION OF STOUT SYSTEMS
734.663.0877 – WWW.STOUTSYSTEMS.COM

STOUT® SYSTEMS

I/T Sanity Check

12 Questions Top Technical Managers Ask

"A computer lets you make more mistakes faster than any other invention in human history, with the possible exceptions of handguns and tequila." Mitch Ratcliffe, Technology Journalist

Visit us at www.stoutsystems.com.

Printed in the United States of America.

ARE YOU AT RISK?

Top technical managers are engaged in an on-going process of risk analysis. They constantly ask themselves, "What would happen if…?"

There are key questions—key problem areas—that are relevant to many technical endeavors: software or Web development, product engineering and the like. In fact, if any aspect of your business relies on custom systems, you might worry about these, too.

So the question is: are you at risk?

Well, do you or your company develop, use, support or provide proprietary software applications, Web sites or databases?

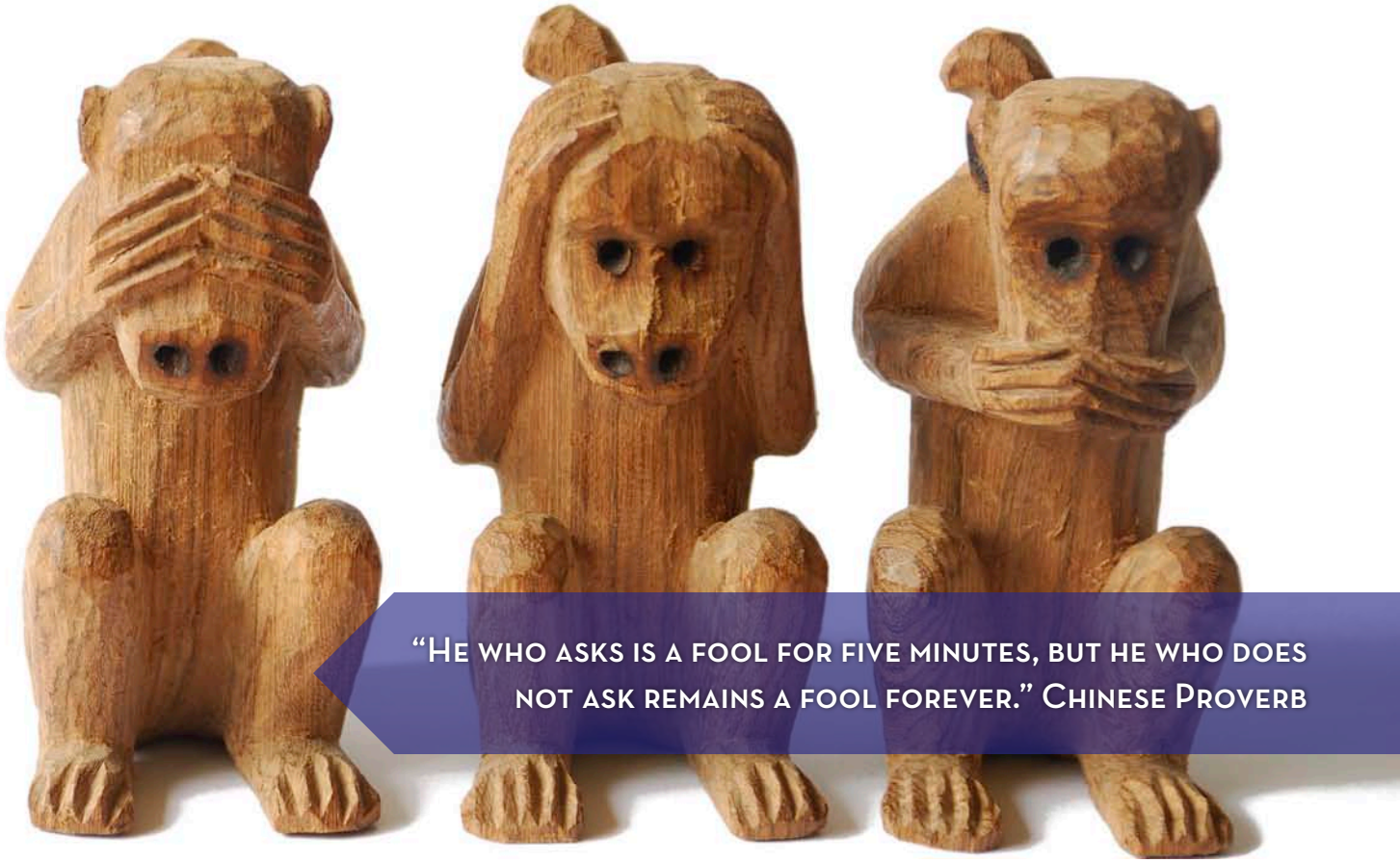
Do you have customers, internal or external, that use or depend on these services?

If the answer is *yes*, it would be beneficial to know the primary problem areas that deserve attention. And then to know what questions to ask about them. And further, to have an idea of the best practices in each.

That, in a nutshell, is the focus of this publication.

There may be other topics that are unique to specific fields. This list is a consensus from a broad range of professionals and their endeavors.

If you seek a panacea, then be aware that handing out answers or making a pitch for a particular solution isn't the purpose here. Instead, the intention is guidance toward asking the right questions, which then makes it possible to find the right answers.



"He who asks is a fool for five minutes, but he who does not ask remains a fool forever." Chinese Proverb

1. IS YOUR ARCHITECTURE APPROPRIATE FOR ITS FUNCTION?

Does your architecture adhere to best practices? How easily do your systems scale to accommodate more users? How well-poised is the system for adding new, previously un-imagined functionality? How secure are the systems? How easily can the code be maintained? How understandable is the code?

WHAT WE'VE SEEN

Spaghetti code. There is no better term to describe how convoluted and complex an application can become when no one is architecting it. You keep piling on functionality without ever fixing the foundation to support it.

Code that doesn't tolerate simple changes is a clear symptom of bad architecture. A developer updates one module, and this breaks another piece of code, often in a seemingly unrelated way.

You can also have an architecture that is overly complex. It manifests itself differently. How long will it take to make a simple change? Two weeks? Really?

WHAT TO DO

Start new development with an intelligent architecture that makes reasonable predictions about the direction the software is going. Then review the architecture periodically to see if it's doing its job properly.

The Wikipedia definition of refactoring is "the process of changing source code without modifying its external functional behavior in order to improve some of the nonfunctional attributes of the software."

This is one of those things that management hates to hear about. But it is supremely important. If you don't refactor code, but continue to pile on, you get something that is so complex that it takes way more time than it should to maintain or to extend functionality.

If you don't have an architect-class developer on your staff, you can have your code reviewed by an outside expert during the design phase to make sure things get started on the right foot.

"With good program architecture debugging is a breeze, because bugs will be where they should be." David May, Computer Scientist

2. ARE YOU USING THE RIGHT TOOLS FOR THE JOB?

Are you on the right platform(s) for what you are doing? Is there some tool that's better suited for the task at hand? Are you adopting new tools and platforms appropriately?

WHAT WE'VE SEEN

One technical manager we encountered wonders why he's having trouble attracting talent. Really? You're still developing Classic ASP, dude. No developer wants to work in a shop where his or her skills are becoming more and more obsolete by the day.

Another company stuck with its database right down to the end-of-life date for the product. The database was the heart and soul of the company's main application—and it was huge. By waiting as long as they did to change to a more contemporary product, they created a painful problem for themselves that was both expensive and risky to solve.

One customer had a technical advisor with a philosophy of "early adoption"—more like "way-too-early adoption." This advisor bought into marketing hype. He did his best to implement every new piece of technology far in advance of it having proven itself. Consequently, the customer ended up with costly rework because it threw away tons of technology that didn't work out.

> "One only needs two tools in life: WD-40 to make things go, and duct tape to make them stop." G. Weilacher

WHAT TO DO

New tools come out every day of the year. Upgrades to existing products come out regularly. The merits of switching to a new platform, adopting new tools or making upgrades need to be evaluated regularly.

There are often compelling arguments in favor of moving on. Companies stop supporting older versions—that's one problem. New operating systems and software won't work with older versions—that's another problem. But making upgrades all the time isn't the right decision either. Sometimes the improvements of the newest whizbang tool or version just aren't worth the trouble or cost involved. There are many issues to consider—not the least of which is developer spin-up time. Just what is there to be gained by moving on? Productivity gains? Stability? Features? You'd better know the answer before pulling the trigger.

3. IS YOUR SOURCE CONTROL ADEQUATE FOR YOUR NEEDS?

How big does a project have to be to use a source control repository? How easy is it to roll back code or find when a bug was introduced?

WHAT WE'VE SEEN

No source control. At all.

Without a repository, you cannot roll back to the last working version. You cannot figure out where a bug was introduced so that you can track it down. You cannot build previous versions of the application at will. You cannot branch and merge.

This is, to our surprise, more common than one would think—especially given the fact that there are a number of free or low cost repositories available.

The felony is compounded when your developers aren't all working at the same location. Code modules are emailed from one person to another. Version stamping is just about impossible, and it then becomes incredibly complicated to determine if everyone is working on the same version, to figure out when a bug was introduced and whether or not the bug is now fixed.

And don't forget the worst of all possible scenarios: hardware failure. Unless your personnel are diligently backing up to some other medium, a day's, a week's a month's worth of work can vanish in an instant.

WHAT TO DO

Get a version control system. Now. There are open source options that are adequate for small to mid-sized operations. If you don't know how to properly use one, get trained—because there is a correct way to use a repository.

It is well worth it to designate one person as the source master. She set the rules and enforces them. So versioning becomes a reality, check-in intervals are regular and branches are easy to follow and merge back in.

"Laughing at our mistakes can lengthen our own life. Laughing at someone else's can shorten it." Cullen Hightower, Writer

4. DOES YOUR MODEL FOR CODE INTEGRATION WORK WELL?

Why not build every time code gets checked into the repository? Nowadays, any software or Web development activity should be done with continuous integration.

WHAT WE'VE SEEN

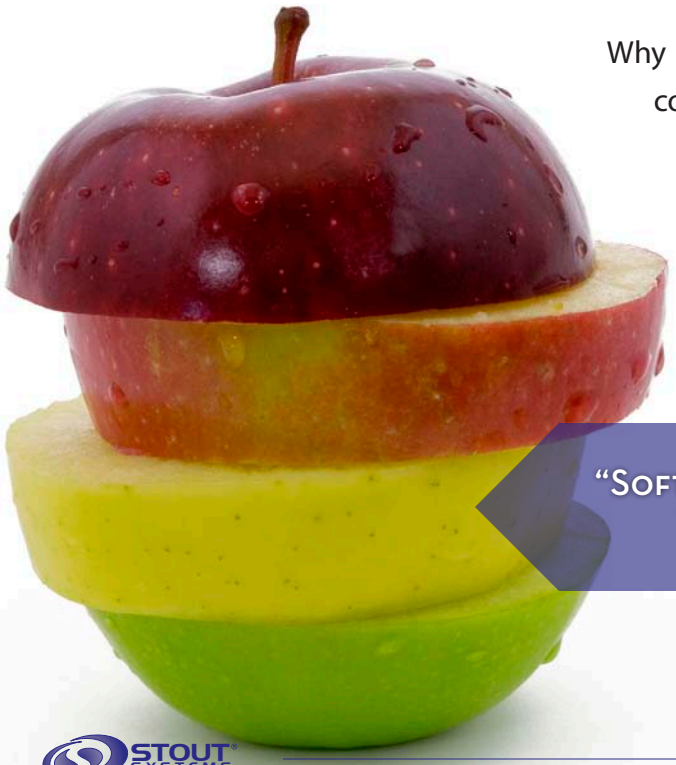
Consider this: the best case scenario would be routine check-ins that send notifications if something is broken. How far from that is it possible to be?

It used to be a grand goal of software teams to build nightly. Your team consists of five people. Five people work eight hours each, and then the code gets checked into the source repository and built. If everything builds properly, hurray! But if not, someone (or several someones) has to back out eight hours of work.

Many companies have developers who check in their code but do not build it with each new check-in. When does it get built? Whenever someone decides to…

And we've seen a worse situation: developers working in a vacuum and trying to integrate when the code is complete.

WHAT TO DO



Why not build every time code gets checked into the repository?

Any software or Web development activity should be done with continuous integration. That means that you have a repository, and when code is checked in it is automatically rebuilt. The new version is then tested against a basic set of tests to ensure nothing was broken.

In the simplest example, the code should at least build—that's the most rudimentary implementation of continuous integration. If it fails to build, that's a problem. Better still, the code would have automated tests to ensure all prior functionality still works (so-called regression testing).

You should have environments for development, testing and production. A pristine environment for testing can expose a host of issues. Perhaps more importantly, a production environment that is insulated from experimentation can save time and heartache.

"Software integration is the prime cause of software aggravation." Peg Bogema, Project Manager

5. DO YOU HAVE AN EFFECTIVE METHOD OF BUG TRACKING?

Is a tracking system in place that records the number of bugs, the areas in which they appear, the developers who created them, how long it takes to remedy them, which applications have the most bugs, which technologies are involved, and also whether the bugs are in third party tools or the systems themselves?

WHAT WE'VE SEEN

No bug tracking at all. Scary. Unless your developers all have minds like steel traps, how will they remember everything that needs to be corrected?

Many companies have lists of bugs in spreadsheets or Word docs or on white boards. Better than nothing. At least there is a way to track what's broken and ensure that it has been fixed. But bug tracking like this doesn't lend itself to capturing and comparing valuable data that puts the entire activity into perspective.

WHAT TO DO

Having a list of bugs is a good start. But such lists often fail to distinguish between bugs and feature requests. Tracking the following can actually improve the development process itself:

Who are bugs assigned to? The developer may need to learn better coding and testing practices. Or he might be working in the most complex area of the code, which needs refactoring.

Are there integration points that cause a lot of bugs? Is there some way to create tests that will detect these issues before the code is integrated (black box testing, test-driven development, etc.)?

How long does it take for bugs to get fixed? If you don't know this, you don't have metrics for the amount of time being spent in development versus defect remediation. If a lot of time is being spent remediating defects, there is something wrong with the development process. This can be remedied as long as you know what to look for.

"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live." Martin Golding, Software Developer

6. DO YOU HAVE METRICS FOR MEASURING PRODUCTIVITY?

Do your software developers use development metrics? Can you measure how each developer is doing? Are the tools and technology impacting speed? Do you know how much work is getting done?

WHAT WE'VE SEEN

Sometimes development teams resist metrics because they believe management will use the metrics against them. But the bottom line is that a software or Web development team has to understand how much work it can get done in a unit of time. And it has to understand where things move along quickly and where things move along slowly. Without this, you cannot handle any of the following:

The blindfold and dartboard method of estimating.

Feeling that an area of code needs refactoring because it takes *forever* to get anything done in it, but no way to justify the time.

Uncertainty about upgrading to the latest version of a tool. Suspicious that the old version is no good. The new version has features that can do things in a single keystroke that now take several hours. But no metrics to justify the added cost.

WHAT TO DO

Metrics should provide more information than is currently available by simple observation. Each metric should provide practical benefit and expose waste or inefficiencies in the team, the tools and development process.

The simplest start is to capture time spent on individual tasks. Knowing a complex development task took two weeks probably isn't that helpful. Use a sensible, detailed level so that it's easy to see how much time is spent on all development activities.

From there, move to metrics that use estimates and actuals. This sharpens the team's grasp of how much work can be done in a unit of time.

> "So who had invented the lines-of-code measuring metric? Either financial or statistical mathematicians who write code, keep it, protect it and refuse to share." Bilal Ahsan

7. IS TECH SUPPORT COSTING YOU MORE THAN IT SHOULD?

If you don't track the details about where your support calls land, you cannot identify which areas of an application have the most bugs, which areas need a redesign to make the user interface more intuitive, and spots that need better training aides.

WHAT WE'VE SEEN

Many companies don't have support personnel. Any issues that arise go straight to the development team. Even when companies do have support personnel, they often don't keep records detailing what they are receiving support calls about.

We've listened to support staff answering the same questions and debugging the same issues over and over. When is it time to make an investment in solving a problem permanently?

WHAT TO DO

All of your support staff need to track and log the area to which the support time can be charged and the duration of each support activity. This is one of the key ways to figure out the cost/benefit of enhancements to various areas.

Imagine if you saw one hour charged to installation issues 400 times. That might give you a compelling argument in favor of allocating 200 hours to improve the installation programming.

Even if your "support staff" is really your development staff, this disciplined approach to support activities needs to be followed.

Tech Support has a wealth of information from trouble tickets that should be mined. It's not just about enhancements. It's about remediation and training, too.

The right orientation and insight will yield the proper return on investment in this area.

"If you make customers unhappy in the physical world, they might each tell 6 friends. If you make customers unhappy on the Internet, they can each tell 6,000 friends." Jeff Bezos, Amazon.com Founder

8. ARE YOU WORKING WITH YOUR USERS TO MAKE IMPROVEMENTS?

Software and Web development have to satisfy an end user. How often do you talk with them? Are they the Rodney Dangerfields of your process?

WHAT WE'VE SEEN

Wow, someone has an idea for a cool bit of functionality. It only took 80 hours to implement. The problem? None of the users are interested in it, despite its proclaimed ultimate coolness.

The responsibility for this problem is sometimes shared amongst management, marketing and development. People get ideas and they implement them without consulting the end users.

Do the majority of users *really* want it? What are they *really* clamoring for?

WHAT TO DO

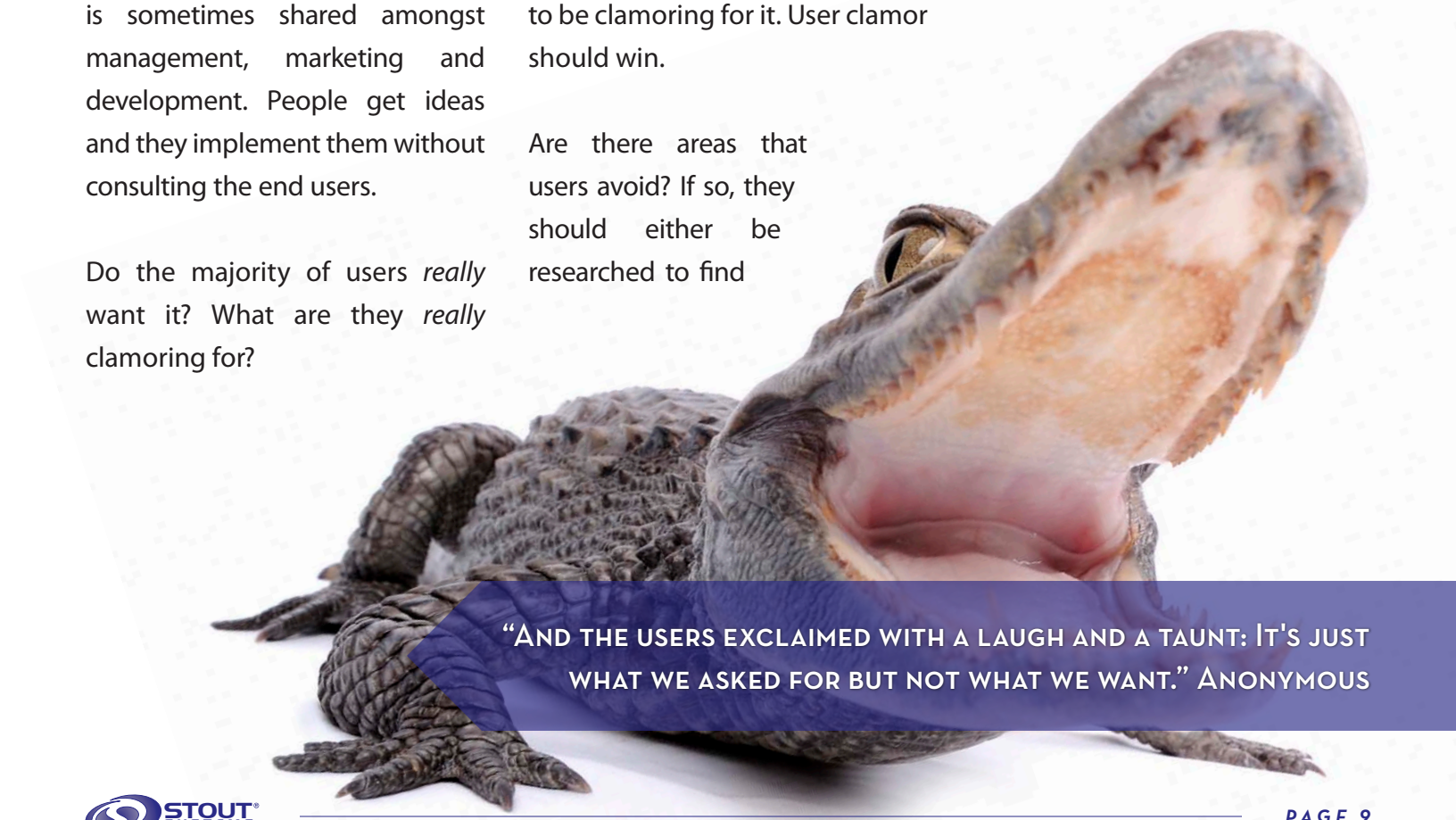
This goes hand in hand with tech support. But gathering feedback also has to stand on its own as a dedicated activity. If a company hasn't been spending time interacting with users, then it doesn't know the answers to the following questions:

What are the features and functionality that most users would like? It's one thing for developers to think of something cool. It's another thing for users to be clamoring for it. User clamor should win.

Are there areas that users avoid? If so, they should either be researched to find out what would make them useful or they should be dumped (or left as they are but not given one more second of development time).

Are there areas that users complain about? If so, what are the complaints? Actions take too many steps? Too much time? Too much redundant data entry? These are key areas to research and improve.

Set up a mechanism for listening to your user base. That will be proclaimed ultimate coolness.



"And the users exclaimed with a laugh and a taunt: It's just what we asked for but not what we want." Anonymous

9. IS THERE TESTING DONE INDEPENDENTLY OF DEVELOPMENT?

Do you have a Quality Assurance department? Does QA keep metrics? Does it create an improvement feedback loop to development so that their bug counts reduce?

WHAT WE'VE SEEN

No QA staff. Using developers to QA their own work. Deadline pressure taking precedence over sufficient QA. Or, "We'll let the users test it."

Some companies have QA departments that are so large you wonder…why? And it comes as no surprise that the software development cycle time is often very long in such enterprises.

And, in either case, the bug tracking mechanisms aren't always designed to help developers understand their code and their issues. Long lists of bugs are fine when it comes to getting the code fixed. But they do nothing to guide software developers into improvement activities until the defects are all concatenated into a big picture.

WHAT TO DO

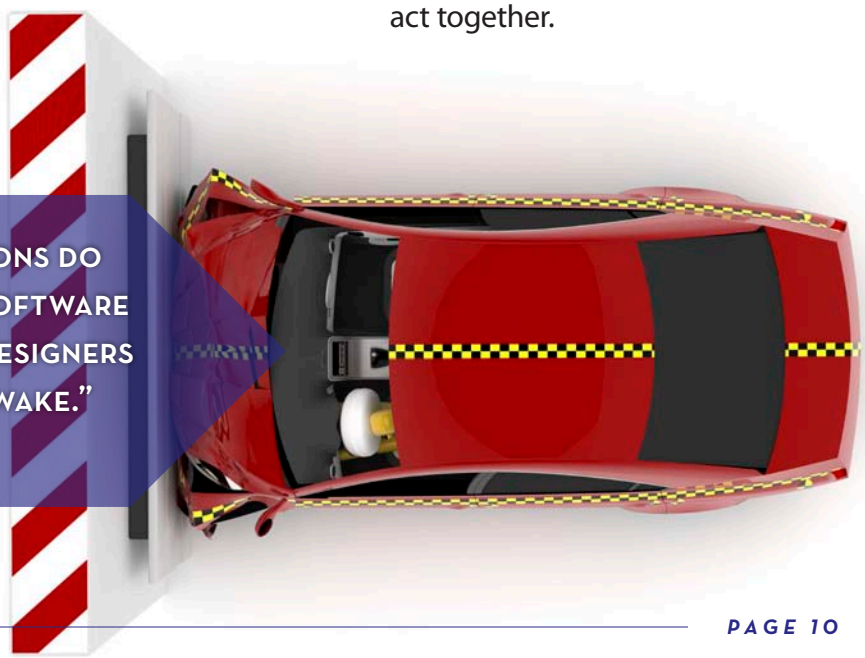
The right balance between development and QA depends upon the specifics of the software or Web development activity.

There should be QA. It should be separate. It should be autonomous, i.e. it can say how long it needs to test a product and how it should go about testing it.

QA should be providing feedback constantly to development and helping development get better and better at providing high quality, bug free code.

When QA becomes an integral part of development, the entire development activity actually speeds up. The company has the metrics it needs to understand its problem areas and its problem developers. Steps can be taken to improve upon both. Suddenly QA isn't the ugly ogre that complains about the code; it's the cute little Yoda that untangles development's problems.

But the QA activity should not be a huge endeavor…at least, not if it is really helping development get its act together.



"What is this talk of 'release?' Klingons do not make software 'releases.' Our software 'escapes,' leaving a bloody trail of designers and quality assurance people in its wake."
Unknown Klingon Programmer

10. DO YOU SUPPORT YOUR DEVELOPERS' CONTINUING EDUCATION?

Software development may be the fastest moving industry in history. The pace is breathtaking. Somehow, developers need to stay current and learn about what the future is going to bring.

WHAT WE'VE SEEN

Some companies bring in in-house training or pay an annual amount for developer training. Some companies have technology luncheons in which developers cross-train each other to stay current or learn something new.

On the other hand, some companies provide nothing. They won't even let developers go to developer conferences or training without taking vacation dates!

And, worst of all, some companies stick on old technologies for eons. Then they fire the poor developers who were stuck in the corporate Edsels because they don't know the new technologies that are being adopted.

WHAT TO DO

Doctors, nurses and teachers all have annual continuing education requirements. Developers are constantly bombarded with new technology. A company needs some way to support its developers learning new technology and tools.

At a minimum, a company should permit a certain number of hours per week of self-education. During that time, developers can experiment with new technologies or take online training classes.

Additionally, companies can provide structured ways to help their developers stay ahead of the curve. Some methods include sending developers to training or product conventions, creating in-house training and funding certification programs.

> "Computer Science is the first engineering discipline in which the complexity of the objects created is limited solely by the skill of the creator, and not by the strength of raw materials." Brian K. Reid, Computer Scientist

11. WHAT IS YOUR PROCESS FOR SOFTWARE DEVELOPMENT?

If your company is using a process, can you tell how closely you are adhering to that process? Is the process appropriate?

WHAT WE'VE SEEN

When software people hear the word "process," they typically think waterfall or agile.

But why have a process when it's not respected by the entire organization? Constant interruptions, changes in priorities and requirements and so forth are the bane of any software development activity.

An organization needs to plan and execute. How it goes about doing that—or what makes planning become successfully realized—is the process. When companies succeed, their process is a good one. When companies fail, their process is a bad one. Simple.

WHAT TO DO

Software and Web development groups need to have a process. And that process must be respected by management. Furthermore, a software or Web development group must be given its requirements and deadlines…and then permitted to actually work.

Imagine a doctor in the middle of surgery being told that he should be working on another patient? Or that the operation he is already in the middle of (appendectomy) should be abandoned in favor of some other operation (liposuction). Come on!

The fact is that there are many, many processes. The right process is the one that works well for the particular situation a company finds itself in.

"Lots of methodologies have come and gone, paradigms have changed but the requirements are always the same; Make it good and make it fast." Anonymous

12. IS YOUR STRATEGY BEING MANAGED PROPERLY?

The art of management is to manage in such a fashion that people know what they are supposed to do, when they are supposed to do it and how they will know that they are done.

WHAT WE'VE SEEN

Competing philosophies amongst the management team are the bane of a software development activity. For instance, some parts of the team think that it's better to make small releases quarterly while the rest of the management team favors large releases annually.

Management disagreeing as to what is important can lead to changing direction and continual interruption.

The GUI is good enough. The GUI is not good enough. The functionality is right. The functionality is wrong.

Development is caught in the middle of a war.

WHAT TO DO

The first decision management must make concerns the length of the software development release cycle. This is true whether releases are internal or commercial.

Then management must agree to content. Prioritizing content gets done at the top. This is then passed along to the development group, which in turn schedules the content into one or more releases. Re-prioritizing content gets done only at the beginning of a release cycle—not mid-stream.

Finally, management must provide a single stakeholder to guide the development activity. His or her voice must be the only one that development has to listen to. So management can argue amongst itself, but the argument stays behind closed doors. Development doesn't get caught in the cross-fire.

Management needs to align itself as to goals and objectives. Without that, the technical department will flounder.

"Poor management can increase software costs more rapidly than any other factor."
Barry Boehm, Software Engineer

Intelligent use of this booklet will help you avoid the most troublesome mistakes encountered within the information technology business. In the end it could help you ***eliminate*** the worst enemies of all: ***wasted time and money.***

"The trouble with the world is that the stupid are sure and the intelligent are full of doubt."
Bertrand Russell, Philosopher

1. SHARE THIS BOOKLET with people you know. The more who know these points, the more likely they are to get used.

2. LIGHTEN UP. Ever notice that some of the best ideas occur during moments of fun? There's a reason for that.

3. IF YOU HAVE QUESTIONS, pick up the phone. We like to talk, but we really like to listen: 734-663-0877.

Soft copy available at StoutSystems.com/ITSanityCheck.

Fueled by the most powerful technology available: Human Intelligence®

STOUT® SYSTEMS