Presentation
Paper
Bio

Return to Main Menu

P R E S E N T A T I O N

WG2

Wednesday, May 12, 1999
10:30AM

BUILDING A RESPONSIVE TESTING TEAM

Jack Cook
Qwest Communications International, Inc.

Building A Responsive Testing Team

Jack S. Cook

e-mail
Jack.Cook@qwest.com
phone (614)-799-7071

ride the light
Qwest

Agenda

- Management Awareness and Support
- Qualifications Of Testers
- Team Building
- Challenges Of Keeping Testers
- Conclusions

Management Awareness/Support

- What are corporate managers objectives regarding QC (Quality Control)?
- Who do corporate managers consider responsible for Quality?
- Who do corporate managers expect to perform final testing?

Who Performs Final Testing?

- Developers testing their own code?
- Developers testing other developers code?
- Tester(s) within the development group?
- A separate testing group within the same development organization?
- A testing organization outside of the development organization?
- Where is the common decision point?

ride the light℠

Qwest®

Qualifications Of Testers

- Variety of experience
- Testers must be methodical and detail oriented
- Technical competency is a must
- Experienced users of your product(s) can be of great value

ride the light℠

Qwest®

Qualifications Of Testers (cont.)

- Testing has become a profession. It is not be the place to train new people
- Development experience is highly recommended
- Good inter-personal skills is necessary
- Recruiting individuals with these skill sets

ride the light℠

Qwest®

Development Experience - why?

- Credibility with developers
- Able to understand developers terminology
- Know when code is ready or has been changed
- Evaluation of problem impact(s)
- Assist in expediting corrections

ride the light℠
Qwest®

Development Experience (cont.)

- Avoiding false indications of problems
- Depending on code structure, test suite size may vary
- Automation of Testing
- Leadership

ride the light℠

Qwest®

Personal Skills

- Must be assertive
- Must be able to handle aggressive behavior
- Should not be easily offended
- Verbal and written communications must be above average
- Bring situations to a win-win closure
- Willing to present oneself as the one *not understanding* a problem

Recruiting

- Encourage IT developers to join the team
- Bonus for bringing new people into the organization
- Use newspapers with well defined adds
- Attend local Career Expos Have on-site corporate Career Expos
- Attend college campus Recruiting Expos

ride the light℠

Qwest®

Recruiting (cont.)

- Have testers take part in the hiring process
- Have two or more testers take part in the interviewing process
- Assign interviewers responsibility
- Each interviewer should discuss their recommendation and have a vote

ride the light℠

Qwest®

Team Building

- Every team member's responsibility
- Document the testing methodology
- Define the testing terminology
- Testers must have an environment in which they work
- Testers must have an environment where they execute tests

Team Building (cont.)

- Expectations and assignments must be clear
- Consistency in the processes/procedures
- Good communications within the team are a must
- Make improvements based on team feedback sessions
- White board brainstorming

Challenges of Keeping Testers

- Management support for tough calls
- Financial parity between development and testing organizations
- Mobility within and outside organization
- Reward systems
- Career advancement opportunities
- Training and technical assignments

ride the light℠

Qwest®

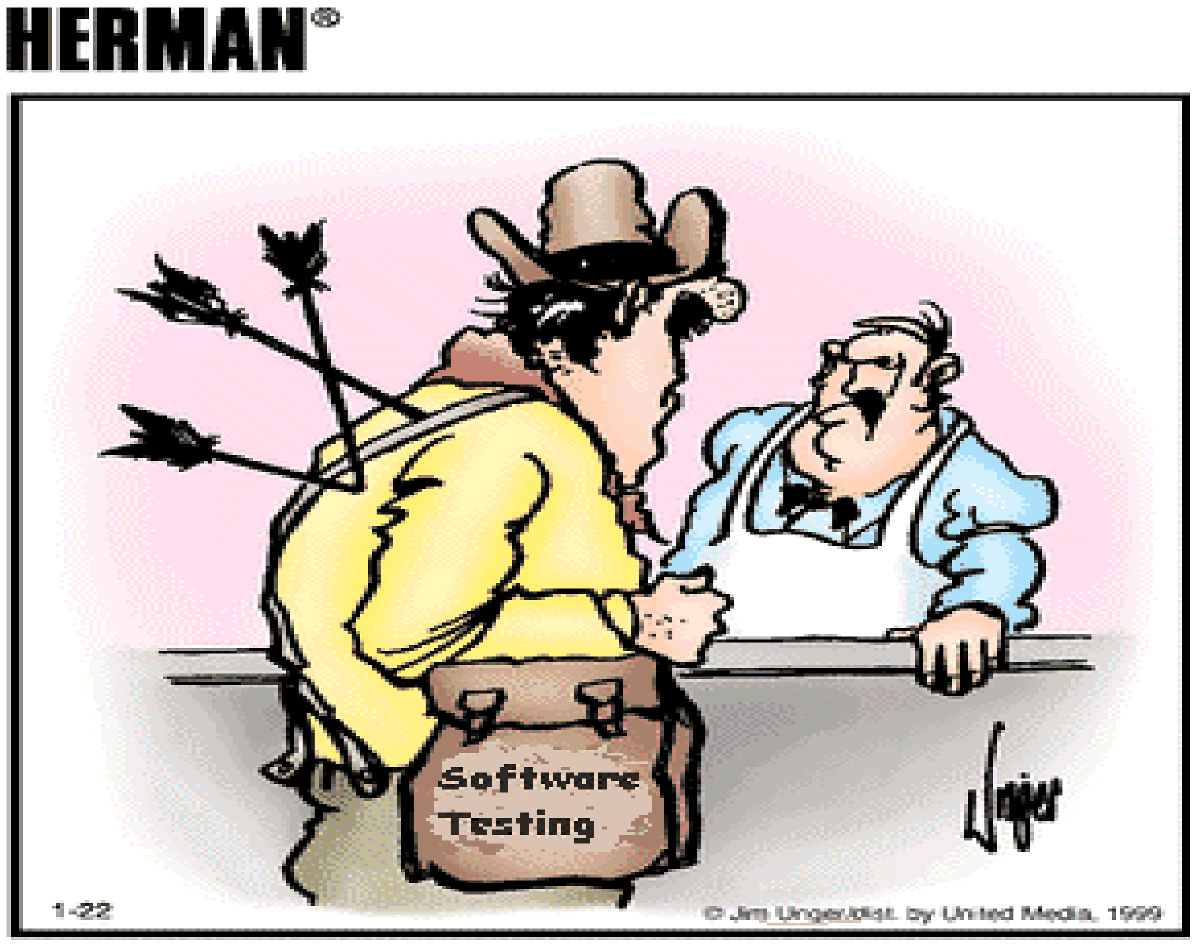
Training

- Testing conferences
- Test tool seminars and training
- Programming training
- Database training
- Certification programs and testing

Conclusions

- Management support is needed from all levels
- Build a team with a variety of skills
- Having a responsive team requires contributions from everyone
- Keeping good testers is a challenge

Questions?



"No one said the job was going to be easy!"

Building A Responsive Testing Team

Jack S. Cook

e-mail
Jack.Cook@qwest.com
phone (614)-799-7071

Agenda

- Management Awareness and Support
- Qualifications Of Testers
- Team Building
- Challenges Of Keeping Testers
- Conclusions

ride the light℠

Qwest®

Management Awareness/Support

- What are corporate managers objectives regarding QC (Quality Control)?
- Who do corporate managers consider responsible for Quality?
- Who do corporate managers expect to perform final testing?

Who Performs Final Testing?

- Developers testing their own code?
- Developers testing other developers code?
- Tester(s) within the development group?
- A separate testing group within the same development organization?
- A testing organization outside of the development organization?
- Where is the common decision point?

ride the light℠

Qwest®

Qualifications Of Testers

- Variety of experience
- Testers must be methodical and detail oriented
- Technical competency is a must
- Experienced users of your product(s) can be of great value

ride the light℠

Qwest®

Qualifications Of Testers (cont.)

- Testing has become a profession. It is not be the place to train new people
- Development experience is highly recommended
- Good inter-personal skills is necessary
- Recruiting individuals with these skill sets

ride the light℠
Qwest®

Development Experience - why?

- Credibility with developers
- Able to understand developers terminology
- Know when code is ready or has been changed
- Evaluation of problem impact(s)
- Assist in expediting corrections

ride the light℠
Qwest®

Development Experience (cont.)

- Avoiding false indications of problems
- Depending on code structure, test suite size may vary
- Automation of Testing
- Leadership

ride the light℠

Qwest®

Personal Skills

- Must be assertive
- Must be able to handle aggressive behavior
- Should not be easily offended
- Verbal and written communications must be above average
- Bring situations to a win-win closure
- Willing to present oneself as the one *not understanding* a problem

Recruiting

- Encourage IT developers to join the team
- Bonus for bringing new people into the organization
- Use newspapers with well defined adds
- Attend local Career Expos Have on-site corporate Career Expos
- Attend college campus Recruiting Expos

ride the light℠

Qwest®

Recruiting (cont.)

- Have testers take part in the hiring process
- Have two or more testers take part in the interviewing process
- Assign interviewers responsibility
- Each interviewer should discuss their recommendation and have a vote

ride the light℠

Qwest®

Team Building

- Every team member’s responsibility
- Document the testing methodology
- Define the testing terminology
- Testers must have an environment in which they work
- Testers must have an environment where they execute tests

Team Building (cont.)

- Expectations and assignments must be clear
- Consistency in the processes/procedures
- Good communications within the team are a must
- Make improvements based on team feedback sessions
- White board brainstorming

ride the light℠
Qwest®

Challenges of Keeping Testers

- Management support for tough calls
- Financial parity between development and testing organizations
- Mobility within and outside organization
- Reward systems
- Career advancement opportunities
- Training and technical assignments

ride the light℠

Qwest®

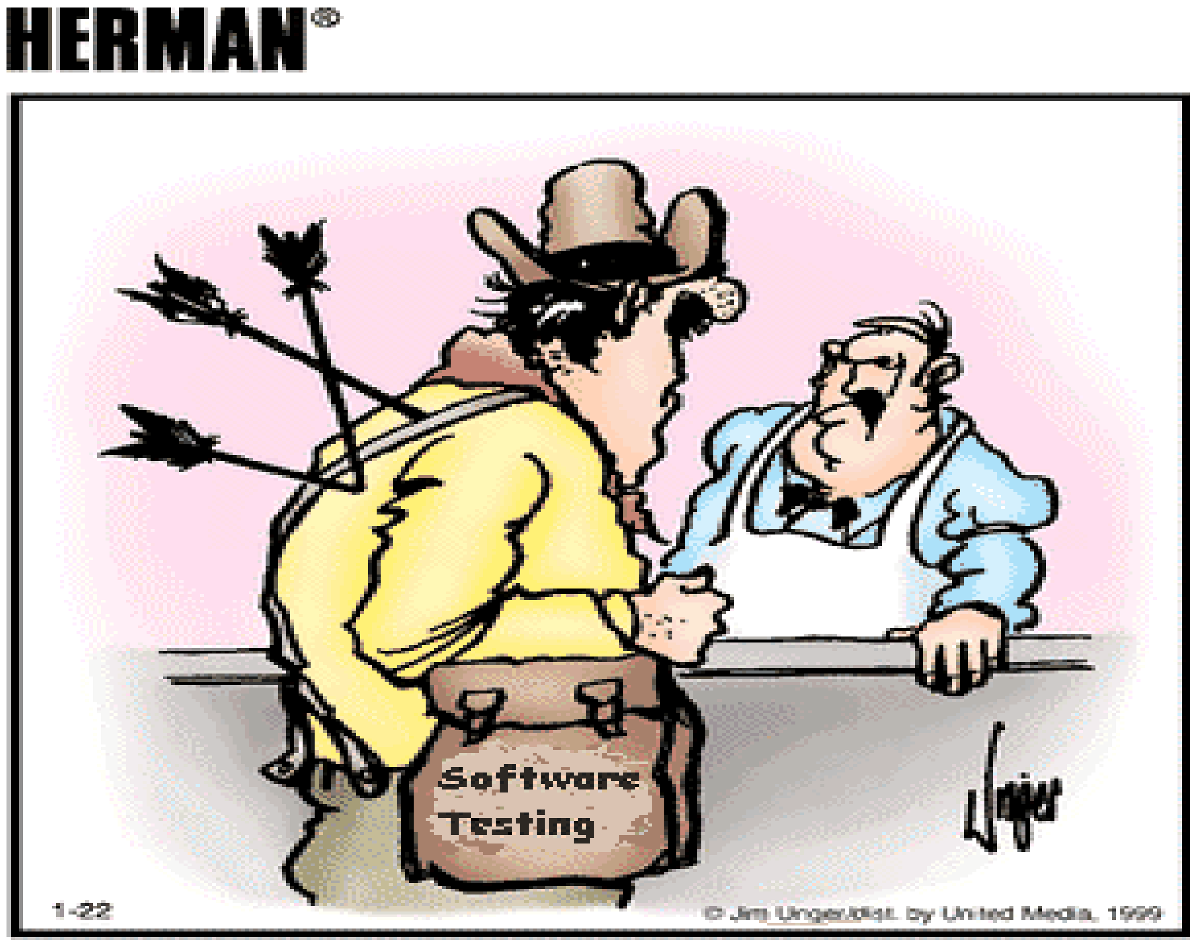
Training

- Testing conferences
- Test tool seminars and training
- Programming training
- Database training
- Certification programs and testing

Conclusions

- Management support is needed from all levels
- Build a team with a variety of skills
- Having a responsive team requires contributions from everyone
- Keeping good testers is a challenge

Questions?



"No one said the job was going to be easy!"

Building A Responsive Testing Team

STAR'99 EAST

Jack S. Cook

March 1999

Building A Responsive Testing Team

STAR'99 EAST

Jack S. Cook

March 1999

Introduction

This paper discusses issues that have proven to be important to successfully assembling responsive testing teams. If done properly, the team becomes a value added extension to the quality process. The issues discussed are not all the issues by any means. Nor are the proposed methods of dealing with these issues the only way to do so.

I am sharing from my experiences spanning several years of assembling testing teams. I've been part of very successful and not so successful testing teams. There are no silver bullets! Every team and its needs are unique.

If you are assigned to building a new test team, expecting to build one soon, or are currently managing/coordinating a team, I expect you to learn something from this paper that will lead towards success. If you are a tester, you may discover areas where you can improve. What you gain may be as simple as confirming that you are on a path towards success.

This paper discusses four key factors to building responsive test teams.

1. Understanding and working with corporate management's perspectives regarding testing support and quality.
2. Some important qualifications to look for when building your team.
3. Team building is an important ingredient to success.
4. Dealing with the challenges of keeping testers

I share *Nuggets* throughout the document that represent personal experiences. You may find them helpful. They can be skipped without losing the context of the paper.

Management Awareness and Support – "Be Prepared"

Prior to going head long into an effort to building a responsive test team, there needs to be some fundamental understandings and agreements between the testing team management and all other management. Upper management support is imperative to having success in a testing organization.

Ask a few simple questions to understand where you may need to work with managers to bring awareness for the need to test and have everyone involved with creating quality. Four good questions to start with are:

- "*What are your objectives regarding Quality Control (QC)*?"
- "*How supportive will you or can you be*?"
- "*From your perspective, who is responsible for quality*?"
- *"Who should perform final testing?* I define *final testing* as the testing a team performs just prior to release of the product. For some teams this testing goes as low as unit level and other teams don't go below end-to-end type tests that observe multiple sub-systems/systems and their behavior.

The responses will be varied but generally upper management wants good quality. However, they don't always understand what it takes to get there. On the flip side of that, sometimes we don't understand the demands of the market and the ramifications if our company misses a window of opportunity. To put together a testing team, it is imperative that managers at all levels work with the testing manager to establish the "working relationship." Education from both sides can bring this effort together. The objective is to understand where you may need to place effort in an awareness program to gain support.

Nugget:

> Tracking metrics is superb for making managers aware of concerns. They tend to take the human element out of the picture. Gut feelings and emotions typically, become less effective when compared to metrics. However, the metrics better represent the correct facts! I have used metrics many times when convincing management to delay a software release. I have used them as reflective data when customers identify software problems postpartum.

Who Performs Final Testing? - "Aim for Authoritative Parity"

The way this question is answered varies depending on the size of the development team(s), development organization(s), and possibly company size. Following are a few approaches to testing that I've seen used.

Developers Testing Their Own Code For Release

This is easiest for the developers but there can be risks depending on the maturity of the development team. A primary risk is TUNNEL vision. Developers typically focus on the end objective. They know how it should work and often focus their testing on making it do what is intended. This is not bad. It is a different mindset. Testing for failure is sometimes missed in the development approach. Also, we all have written documents that contained grammatical even though we had read it several times. Another pair of eyes sees the error right away. The same can be said of testing. Therefore, this method of testing is considered dangerous and not recommended by *most* quality circles.

Developers Testing Each Other's Code

I have seen this technique work fairly well in a few small projects. I believe it worked because the development teams were very mature in both technical skills and their understanding of the deliverables and how the user incorporated them. Even with those attributes, it is risky business once the size of the team goes above four or five. As project size grows in software functionality and people, this method becomes more dangerous.

Dedicated Testers Within A Development Group

I have seen this technique work well a couple of times. Remember the testing team must have parity in importance for this to work. It takes a disciplined manager to make it happen. Most of the times I've seen this happen is when the organization wanted some form of "organized testing" but, it shouldn't get in the way of meeting dates/deliveries. Another danger of this structure is there may be a tendency for developers to view the testers as the ones who should do their low level testing. That should not be allowed. The success of a program like this really depends on the discipline of the manager.

A Separate Testing Group Within The Development Organization

This is the approach I've seen used most often. It allows for some accountability between developers and testers that must go through separate managers. The tendency to move questionable software through to release lessens as more levels of management become involved. Be careful how many levels of management must be involved though. It can be bad too if getting approvals for changes takes too long.

Testing As A Separate Organization Outside of Development

There are pros and cons to this method but depending on the size of the organization(s); this approach can be very effective. One caution though, as the testing team is moved further from the development team, typically, communications must be drastically improved. If the testing team will be testing across multiple organizations (departments), this method is probably best. The final decision will be with a manger that does not necessarily have the same vested interest as the manager pushing to release.

A Key Issue Is The Common Point Of Management

This is a critical factor for the testing team. This is typically the point where decisions will finally be made on questionable software. As the level goes higher the less people want to get those individuals involved and tend to resolve problems at lower levels. If the common point of contact is not fully supportive of the testing organization, I have experienced that the testing team will have a tough time and better put a plan in place to change that situation. After a few times of going around the testing team, I have seen testers typically do one of two things: one, leave the organization; two, their expectations and drive are reduced possibly resulting in lower quality testing.

Authoritative parity is essential for testing teams to be effective. I'm not saying that testers have more authority than developers but tester's comments, discoveries, and recommendations must be considered on an equal level with developers. This is a primary reason to have the testing team report to a manager a level above or at least on a peer level with the development manager(s).

Nugget:

> My recommendation is this: whenever decisions are made for software releases to go against the recommendations of the testing team, and that will happen because of business needs, that the highest level manager in the decision making process communicate the reasons to the testing team. This is something I request and have not been denied so far. Often times, the mangers don't realize the impact their decisions may have on the testing team morale.

Qualifications Of Testers - "Credentials & Balance"

When bringing testers into an organization, there are some fundamental qualifications that can make significant differences in how your team is accepted and operates. The team should be well balanced with people who have a variety of skills. There should be a good cross section of experience and technical abilities. To be successful at testing, one must be methodical and interested in details. They must be technically competent to test the software for which they are responsible.

If possible, having people from your user community who know your products can be very helpful. On my last testing team, I had two testers who had over a decade of experience using the products IT had produced. They were extremely valuable in reviews of requirements and tests. They knew how the various systems inter-related and contributed significantly to improving our testing scenarios. They also understood the types of data that made things happen.

Nugget:

> One company I worked for serviced several companies across the country. Our software releases were typically focused as specific capabilities to sub-sets of companies I would contract with the companies that would be most affected by the release to loan us one or two people to assist us in our final testing. Having users assist in our testing drastically improve the acceptance of our new releases.

Testing **is not** the training ground for new people. Testing is a profession just as development and we need to understand the same rules apply. Most development organizations don't put a new person on a critical piece of development. We must apply the same philosophy for testers. I try to keep a 1:6 ratio or lower.

Nugget:

> Years ago, I was convinced that bringing new people into the testing team was a good idea. They hadn't seen the product, they could test the user documentation for errors, if the code wasn't defensive, the chances were they would break it, and so on. The philosophy sounded good. After a few releases of our software, I discovered we weren't doing a very

good job of finding some serious problems. Underlying things like data being put in the wrong place, incorrect conversions, errors not being reported, performance issues, stress issues, etc. It became obvious we weren't testing in the right places. The new people didn't know how to drive an autonomous process by writing underlying software. Entry level people did add value to the testing effort. However, I only had funding for so many people and I wasn't hitting my objectives in defect removal. I generally have few entry level people on my teams.

Development Experience – why? "Some good reasons:"

Following are some reasons I believe are credible for hiring/bringing in testers with past development experience:

Credibility

A primary reason for having development experience is to gain credibility. Ever have someone come up to you and tell you how to do something and when you asked about their experience, they didn't have any… how credible were their ideas from that point on? I mean you may listen but are you really going to give their ideas much time? Credibility is critical to your team success!

Speak The Same Language

Developers typically have their own terminology. To communicate with them we must speak their terms or if they want to show us a piece of code, we should be able to read along with them. It should not look like something foreign to us. We need to know terms like object, class, inheritance, polymorphism, extended classes, variables, structures, IDL (Interface Definition Language), base class, relationships, and so on. At one company I was surprised that even developers didn't understand that a stored procedure in Oracle was "code". If our eyes glaze over when a developer goes into his/her world, we may miss important clues or indicators that can help us in our testing.

Know When Code Is Ready/Changed

By knowing how to look at source through the source code control tools (if you have them), a tester can determine what has actually changed. Often defect records don't explain all the impacts of the change(s). Sometimes, more changes than expected and observing the deltas associated with the change documents, a tester can better understand what he/she must test. By following this process, the likelihood of changes slipping by is greatly reduced.

Evaluation Of Problem Impacts

Where is the problem? What is the impact of the fix? By looking into the code to evaluate a fix, you may find it impacts a library, base class, or common code. Sometimes developers only look at the impact of the code change on their feature/function. Changing "common code" may result in MAJOR re-test efforts. Testers must stay on top of fixes and their impacts to testing especially in the latter part of the testing cycle. There will not be time to do major re-test. I recommend testers be involved in these decisions when the developer is going to make the change(s).

Assisting In Locating Problems

When a tester experiences a defect, if they are familiar with the code, they may be able to identify the area where the problem exists. Developers appreciate this information. It saves them time because they don't have to attempt to duplicate the problem(s). I recommend the tester work closely with the developer and be ready to rerun the test when the fix is incorporated in the testing area. It can save time for both parties. It definitely helps the relationships between testers and developers.

Avoiding False Indications Of Problems

Having development skills may save the tester some embarrassment. By understanding the code, a tester may discover a problem that appears to be caused by one developer when in fact it may be someone else's problem. The following nugget is prime example of wrong first impressions.

Nugget:

> John has great credentials, he has experience with development, design, architecture, of hardware and software. He was testing an application ABC (for simplicity). He initiated his test suite and found a problem. His confidence with the developer's capabilities for this application was high. He decided to check the code before going to the developer with the problem. He found the code to be correct. The next step was to talk with a DBA (Database Administrator) because he saw the code was expecting something returned from a stored procedure. He discovered the DBA had not updated the testing database with the modified stored procedure. He requested the update to be done and everything worked. Now think about it, one, he didn't even bother the developer (time saving), it took about 2 hours of John's time, we estimated it may have taken the developer 4 times that to discover it wasn't his problem. John's credibility went up considerably. In fact, one development organization requested John test their code! That's a relationship we want.

Test Suite Size May Be Changed

Testers can benefit from observing how code is assembled. Often, we have requirements that give ranges of values. A typical method for testing this is to identify equivalence classing to reduce the size of the test suite.

Nugget:

> For example, something can have a range of 1 to 20. Test cases would involve <0, 0, 1, <mid value>, 19, 20, 20<, a total of 7 tests. If the code is constructed where a "for" loop is performed to evaluate each value, these seven tests may even be too many. You may get by with <0, 0, <mid>, 20, 20>, a reduction to 5 tests. If however, the construct is a series of "if" "else's" and each value is handled specifically, you need to run 22 tests! This happened to a radio company that manufactured a 32-channel scanner. They did equivalence class testing and everything passed. The scanners went to market and it was discovered channel 27 didn't work! This was a very costly mistake for the company.

Automation Of Tests

Another reason for having development experience is when we desire to automate tests. There are several great tools for automating GUI type testing or executing test programs from a standard ASCII window or command line input. However, there are numerous software systems containing autonomous processes with very little human intervention. When testing these types of applications, it almost always requires writing programs to determine if the processes are functioning properly. Even automating tests via a vendor tool, requires learning the "language" the tool uses. Most systems will require some type of programming to automate. Of course, it depends on the depth of testing you need to exercise.

Nugget:

> I am an advocate for automation but with moderation and by design. To have automation work for your team, it will require someone, or several people depending on the size of your testing organization, to make testers scripts generic. That is a topic that could be discussed as much as this entire paper.

Leadership

Structuring your team can take on many variations. One that I have found to be very beneficial was to have the people with more technical depth to take a lead in the designs for testing. They work with all the team members through reviewing test cases, designs, and even code reviews for tests that vendor products can't handle. They are responsible for making recommendations on improving the overall processes. As the less technical folks become more experienced, maybe through training, they step into these positions. This serves a couple of purposes. One, it seems reasonable the more experienced people will have a broader inventory to select from for leading and two, it gives the less technical folks a target to go for if they want to.

Personal Skills - “Inter-personal skills are important”

Assertiveness

Being assertive is much different that being aggressive. Assertiveness typically consists of being persistent, using multiple methods of reflective listening to determine exactly what is being communicated. Your team may need some training in this area.

Handling Aggressive Behavior

Testers will eventually encounter some aggressive verbal behavior. I recommend training for handling aggressive behavior, typically, you get this when you have assertiveness training. There are several workshops for interpersonal skills. However, aggressive behavior should not go unnoticed. The test manager should be made aware when this is taking place and talk with the person’s manager.

Don't Be Offended Easily

Along with handling aggressive behavior, testers must do the same in their dealings with others. We must present a professional image by handling comments in a professional way, controlling our emotions and not take things personally. If someone makes personal accusations, the testing manager must deal that with quickly.

Communication Skills

Having the skills to communicate effectively is a must. This is especially true when testers test software for multiple development organizations. Because of their exposure to multiple managers and development organizations, testers must have excellent written and verbal skills.

Reaching Win-Win Closures

Bringing issues to a win-win closure requires discipline, control of emotional situations, and assertiveness. Testers can often be viewed as "bearers of bad news" and if the only time you show up is when there is a problem, the perception is reasonable. I encourage testers to keep in contact with the developers when they aren't testing. Build rapport with them.

Humility

A good method of keeping people calm is to approach the problem from the "I" perspective instead of "you". This approach is a very effective way of doing business and often results in a win-win closure.

Nugget:

> Another positive exercise is that of using I in place of you. You tend to be perceived as accusatory. When using I, the focus is on the person asking the question(s). For instance, when I find a problem, a tact that I will take is that I'm not sure if it is a problem or if I'm just not doing something correctly. I do this sometimes even when I know it is a problem. It just tends to keep the shields down.

Recruiting - "A Multifaceted Approach"

Creativity in recruitment is a must. Pursue and use a variety of avenues for bringing good people into your team. I have brought people onto my teams by recruiting them from teams in other organizations. However, I typically don't do this method unless a person approaches me first. However, an objective I recommend is to bring developers into the testing organization. This means your testing team must provide something developers can't receive by being developer. That is the challenge for the manager. Some things that have worked very well are:

- ♦ Bonus incentives to employees that bring in people from their networks.
- ♦ Use of newspapers, be sure your add is specific to what you really want.
- ♦ Attending "Career Expos" if they are available in your area.
- ♦ College recruiting can be very beneficial. Remember that they are typically entry level.

- Have a “Career Expos” at your company.

Nugget:

I have found that involving your team members in the hiring process is very beneficial. One, they have a vested interest in bringing the person in and making them successful; two, the person coming in gets to see how it might be to work on the team; three, I give each team member an assigned task. One of three: technical evaluation; testing knowledge; or “how we work here”. My job is to provide an overview of my team’s objectives, how the company works, and determine the candidates’ areas of interest. After the interviews, we vote and that decides who is hired. Does it work? Yes, most of the time. Out of 24 people I hired over the past 14 months, we only have one person who has not met expectations. I believe in being proactive when this situation occurs. I had a meeting with the other two people who interviewed him. We have determined his weak areas and tried to strengthen them through mentoring, training, and job assignments. Even with all this activity, he was eventually let go for other reasons. I believe a 4% error rate is respectable. It would be nice to get our software to that error level.

Team Building - “Creating Solidarity and Support”

Having a responsive testing team requires supporting one another and being willing to go the distance as a team. Individual recognition is important but so is team recognition. Having solidarity within a team is the responsibility of every member.

Mentoring

Team members must be educated regarding the methodology being used. A good way to bring a new person on-board is through a mentoring program. A mentor’s responsibility begins the day the new person arrives for work. I recommend a mentor at a minimum the following responsibilities:

- Familiarizing the new employee with the facilities, where things are and how to get to them.
- Showing how to fill out corporate paperwork to get things done.
- Introduce the person to the team and other teams.
- Handle questions the person may have regarding normal everyday operations.
- Train/show the person the methodology being used and why.
- Have discussions on how to test, what are good tests, when/how/why tools should be used, and so on.

Standardization

It is imperative to have a standard vocabulary of terms so everyone understands what is being said. For instance, if you’re in a room with 10 people and ask the question “what is a test case?” You can almost be assured of 10 different definitions. Standardizing your terminology will not only help your team but, if it is shared with other organizations, the terms will begin showing up in requirements, and other documents. Everyone will understand what you mean when you refer to a test case.

Testing Environments

It would be reasonable to ask, "what do testing environments have to do with team building?" Having environments that can be shared will greatly enhance your testing efforts. However, if you have a single environment in which to test, teamwork is critical to being responsive. There should be two testing environments for any testing group. One environment is structured for where your work (i.e., test documents, code, metrics/results, etc.) is stored. The other environment is where your tests are run or run against (i.e., systems, databases, applications, etc.). Having good testing environments builds team cohesiveness. Sharing, educating, maximizing efficiency by avoiding redundancy can all take place.

Expectations

Every tester should know what their assignments are and the expectations that come with the assignments. Their managers must provide this information. I recommend spending some time with each person each week just talking about things. Do not confuse this with micro managing. I'm talking about getting to know your people and their interests beyond work. Informal discussions and sharing can go a long way in understanding their needs and how you can help.

Consistency

Consistency is typically something testers look for in their testing. Having it in the testing process is just as beneficial. The process should not change all the time. Stability is essential for successful testing of software and the same holds true for the process people are expected to follow. Try to be consistent with the entrance and exit criteria for testing. When inconsistency occurs, be sure to identify the reasons to everyone and indicate the need to correct the direction.

Information Sharing

Sharing information is a must to being successful. Instilling good communications is probably one of the most difficult efforts to overcome in this industry. Methods I have found beneficial include e-mail, white boards discussions, centralized documentation repositories so everyone can get access, and regularity with (daily/weekly) meetings. However you can get folks talking and/or sharing will generally be beneficial to your team.

Brainstorming

This is also a great method of information sharing. Take notes and publish minutes. Require your team members to attend. At these sessions, all ideas are welcome, no analysis during the meeting. Brainstorming sessions are generally fun and educational. Often times, the most ridiculous sounding idea becomes THE IDEA in the future.

Challenges Of Keeping Testers - "Retention Requires Awareness"

Keeping testers is definitely a challenge, especially in today's market. Following are some key factors to being successful in retention of good people:

Management Support

As already mentioned, testers must know they have support from various levels of management. Encourage managers to recognize the testing efforts as much as they do the developers efforts.

Financial Parity

If there is not financial parity between testing and development, guess who is going to win? It's not rocket science. Pay isn't everything but it is definitely an issue that must be addressed. I recommend tester pay scales be above developers (technical skills must be there too!). A primary objective is to bring developers in to testing and then train them to be testers.

Reward System

Rewards are a great way to encourage people to continue improving. Often, there is little money in the budget for financial rewards so your resourcefulness will be required. Merit increases, pay range adjustments, stock options, office environment, etc., all are things that can make the difference.

> *Nugget:*
> I have made small certificates to give to people for special efforts. The certificates allow the bearer to spend $100. They spend the hundred dollars and turns in the voucher and I sign it. Another nice reward idea is to purchase gift certificates from a very nice restaurant and give them as rewards.

Career Path

Testing cannot be a "dead end" career path. There must be opportunities for testers to move into other areas of challenge (i.e., system engineering, project leads, managers). Testing may not be where some folks want to stay. The manager must work with their teams to ensure career advancement. This is true of any organization. It's no different for testing.

Training

Testing technology is changing almost as fast as development. Therefore, training is necessary. Encourage it and then provide the means for them to take it… maybe even require some types of training.

Training - “Keep Up Or Fall Beyond”

There is no middle ground regarding training. Either you keep up with the technology changes or you fall behind. I don’t believe there is any such thing as status quo in this technology.
Testing conferences are a great place to network, compare where you are and where you want to be. If you are in the lead, consider sharing your knowledge. If you are behind, network and begin the process of catching up.

Testing tools are changing with the technology. It is no longer cost effective to build in house automation tools when there are corporations focused on just that effort. Evaluate tools, know what they can and can’t do for you and get your folks trained.

Nugget:

> We have incorporated several tools in our company. For the testing team, we have a process we go through before accepting a tool. You may find it helpful. We first decide what our requirements are and document them so everyone is in agreement. We bring in the vendors that may have the tool we need. We try to have them within a day of each other for comparison purposes. When we narrow them down to two (maybe three), we ask for a “proof of concept” demo. This is a demo where they must bring in their tool, put it on our system(s), execute against our applications, and successfully pass the requirements statement. It works great. We have had very good success with this approach. I’m sure it’s not a new idea.

If you are going to get into the automation business, then I highly recommend you have a small team of developer types to convert automated scripts into “generic” scripts. Typically, testers will not have time to do make their scripts generic. This requires a significant amount of design and planning.

If you have testers who have not done development, I encourage you to have them attend some introduction classes. They may become very good developers with some training. Remember “career paths”.

If you or your people are considering making testing and/or quality process a career, I strongly encourage becoming certified. This is the direction the business is going.

Conclusion

In conclusion, I have discussed four key points to building a responsive testing team. There are many issues I have not covered. Some of these ideas will not work in every company or environment but the concepts should be adaptable for most situations. The four points are:

Point 1: "Management Education and Support" - Prior to going head long into an effort to *build a responsive test team*, be sure to understand the educational and/or awareness programs you have ahead of you to succeed in gaining the necessary support.

Point 2: "Credentials And A Balanced Cross Section of Skills" - Qualifications of testers is very important. Technical skills are top on the list but, almost equally important is having people with "user" skills for different reasons. A primary objective of the team should be to gain credibility with both, developers and users. Having expertise in both areas is most beneficial.

Point 3: "Team Building & Training Will Make A Difference" - Having a responsive testing team requires supporting one another and being willing to go the distance as a team. Having consistency, training and buy-in to the team objectives and responsibilities will bring success.

Point 4: "Retention Requires Awareness" - Keeping testers is definitely a challenge, especially in today's market. To be successful, you must use good recruiting practices. Once you have them, they must know they have a positive career path. Training should be provided whenever it will benefit your efforts.

Hopefully, through reading this paper, you have discovered useful information that can make your workplace better.

Jack Cook

Jack Cook comes to us from Qwest Communications International, Inc. where he is currently the senior manager of the Product Assurance Testing Team in Dublin, Ohio. He joined Qwest two years ago. Prior to joining Qwest, he was employed by AT&T for 28 years. His career began in hardware design and development for Western Electric in 1968. In 1975 he joined Bell Laboratories in the software development arena. For the next 21 years his primary focus was on software testing with a few developmental responsibilities interspersed. His interest is to assist in making software testing more of an engineering or scientific discipline, and to move away from the view that testing is *an art*.

He has developed test tracking tools, record/playback tools, and automated test report generators. He believes in using testing metrics to improve both development and testing processes. He has been involved in researching methods for improving software reliability and fault density predictions using test metrics.

He is a Certified Software Quality Engineer through the American Society for Quality. He is also a Certified Quality Analyst and Certified Software Test Engineer through the Quality Assurance Institute.