Survival Skills for Today's Test Managers and Test Teams

By
William E. J. Ginn
Director of Quality Assurance
Onvia
Seattle, WA

8/11/2003

Summary

It has been my experience through my career, from reviewing translated training materials in Japan, to localizing such products as IDE toolkits for Windows CE for Microsoft, CAD solutions for Autodesk, VPN solutions for Nortel Networks, all the way to trying desperately to keep a growing B2B and then B2G ecommerce sites from blowing up daily, that this industry is still young. By young, I mean immature, underdeveloped, rife with instability and ill advised behavior driven by that great driver of humans, MONEY.

Early on in my career, I was put into totally hopeless situations by upper management. Before I understood that upper management is perfectly happy to throw dedicated, smart people into hopeless situations I would work like mad to try to overcome the barriers to success. In the first of these, I was put in charge of testing Allaire's Cold Fusion Server for a localization company in Seattle, and was told that it was up to me to keep this first big project, and thus the company, afloat. I would work until 2 AM, run outside and throw up from the anxiety, and come back in and work till 4. After a number of these experiences, I realized that upper management, while being paid eight times as much as I was, didn't necessarily have eight times the brains of those who work for them.

I have developed tools over the last 10 years to keep myself from having to work past 2 AM unless I want to, and to keep the tensions down to the point that I don't need to run outside and throw up anymore. I hope to share these tools and techniques with anyone in similar situations. This paper is specifically targeted at IT professionals who find themselves and their projects in constant, or even occasional turmoil. I have been able to create a relatively stable environment with the help of many good people. These tools have worked well and so I document them here. If you work in IT and do not find yourself fighting chaos and with an ordered, stable environment, then I salute you, and you have no need of these suggestions, but please read on, because I provide what I hope are interesting vignettes of the craziness that has built my experience as I and my comrades combated the madness.

Briefly, these are the areas I will discuss in looking at ways to avoid aneurysms and keep your job in today's market.

- Intro, what is your environment? Are you trying to launch a shuttle or keep the doors of the store open? Does your corporate culture pat you on the back or pound you into the ground?
- If you are in chaos, or at least in constant turmoil without the ability to produce quality consistently, then investigate your options, and make a plan to affect positive change for yourself and your team.
- Once you have a plan, prepare to execute it, mentally, physically, and emotionally. Get a stomach for confrontation!
- Part of your plan must be to expose what you do to those who employ you. Be ready to produce and publish results: Get a bookshelf. Get lots of binders. Get a web server dedicated to publishing results. Produce, Publish, and Document!

- Everyone remembers a good catch for three days, everybody remembers a huge egregious miss for thee weeks. Be ready to anticipate the politics of a corporation and use your publishable, consistent results to show value and fight resistance.

I hope here to share some of the experience and tools that have helped me to keep QA in a position of respect and to maintain a strong team in difficult environments. We hope that some or all of the information here will help you affect positive change in your organization and your career.

What is Your Environment?

I spent 4 years localizing American software products for the Japanese market, and it could not have been better exposure. We had huge to small software vendors with a wide variety of products. It was a great broad spectrum education in different applications, and exposure to a wide variety of corporate cultures. While a number of the shops we worked with had fine, stable processes and the projects went smoothly, there were just as many in which a structured approach was impossible. The common theme in these projects was the perceived need to rush to market. We would ask for the US product test cases, and a good number of larger organizations sent back asking us to provide our Japanese test cases (in English). They would compare this to their US cases and tell us where we had missed coverage. We had full NDAs and as we persisted, it became apparent that they had no test cases and were hoping to backfill with our work.

Now, there are situations in which it is appropriate to move forward with all haste, and let process slip if adhering to process would allow a competitor to beat you to market. If you have a website selling widgets, and you know that 5 other sites are being developed to sell widgets, simply get a caffeine drip line going and get the site presentable. I call this just getting the store doors open. If you are in medical or aviation work, then processes must be observed, as the legal ramifications of failure mandate accountability.

So evaluate whether you are trying to keep the store doors open or if you are trying to launch a shuttle. It is not difficult to match the appropriate level of care and structure in your work to what you are trying to do.

The larger issue creating havoc today is the immaturity of the industry. This situation is not unique to IT. Looking back through history, one can see a pattern in immature industries and technologies, in which there are wild fluctuations in investment and salary, intense efforts to capture market share and dramatic successes and failures in those efforts.

The automobile industry as it emerged in the 1950s, the railway races of the mid 1800s, and more recently the race to capture the personal computer operating system market follow the same patterns of dramatic over-inflation of stock prices, huge efforts to build the infrastructure to meet demand, and overnight fortunes and ruin for many of the participants. Microsoft's early

development efforts are reported to have been fueled by caffeine and cocaine as workers stayed up days at a time to complete product launches. Each industry as it developed had to cope with the huge errors in design and execution that are inherent in such high paced efforts, and E-commerce follows the same basic pattern.

This of course creates a dynamic, adrenalyzed working environment many of us live in today. It is Quality Assurance's job to try to ameliorate the chaos with a structured approach, to develop a process to calm the turmoil of rushing to market that allows consistent results. This paper hopes to provide techniques for making the value of this approach visible and keep it in full public view.

It is important to keep in mind as we investigate the immature aspects of our industry that we are on the backside of the latest great economic revolution, and while there are many frustrations and anxieties that come with the work, it is still a very interesting time to be in this particular industry.

The greatest issue facing us, to speak bluntly, is incompetence. An analogy for IT in it's formative years would be dredging for gold with siphon pumps. Divers go into rivers and use powerful pumps to suck up all the silt and rocks off the river bottom. In the deep crevices are found the heavier flakes of gold. As companies built IT organizations, in some cases, they dredged along without regard or understanding of what they are trying to build, pulling up all kinds of silt and gravel and getting a couple of pieces of golden talent by pure chance.

The industry is maturing certainly, and I have been involved with projects in which each individual of the team was aware of how to make a reliable product, but I have also struggled with stunning indifference from burned out staff, egomaniacal Developers who threatened physical harm, and Vice Presidents and Directors who spell MS SQL as "ms sequel", who suggested that maybe rebooting production servers every night to keep them from crashing during the day actually is good for the servers, and who didn't even know what a code freeze is.

It is remarkably easy to get on a browser and discover a documented, structured approach to use in software engineering. You can train yourself in less than 3 months to get a good basic understanding of current processes and methodologies that are proven. The trick is to develop an approach and expose the stability and insurance that a simple defined process brings to your organization. The fight is to overcome the conflicting interests that are the traditional bane of QA existence, without burning bridges or losing your job in the process.

So we have your business environment, which as I stated earlier, is relatively easy to evaluate for approach, but we also have the corporate culture, which must be more carefully examined to develop a plan, if your organization is driven to the point of releasing unstable product, or is at risk for burning out or losing staff.

Corporate Culture

The other aspect that you must evaluate as you develop your approach is the amount of conflict that you experience in your IT world. Within IT and from the business side, I'm sure that many of you experience the frustration of being pulled, pushed and prodded by the various interests that make up any corporation. Some of this is intrinsic to our work and simply cannot be avoided, but as you evaluate your circumstances, remember the above warning about incompetence. If within IT you are not supported by those above you, or are being led by someone not interested in process or with no understanding of quality assurance, you owe it to yourself, your company, and to those you work with to confront this problem as politically as you can.

You can devise a plan and do groundwork to affect positive change, but if you are not supported your chances of success will be severely limited and it could nothing more than a learning experience for you. It is our goal and purpose to bring order and structure out of disorganization, so I ask that anyone struggling with their organization to exhaust every means to bring understanding to those who lead them. If after you have explained and cajoled and begged and pleaded and threatened, you find that you have made no progress, don't stop. Affecting positive change in an organization requires force of will and the attempt must be made, but if those who lead you are not completely willing to learn what good process can bring then you will probably want to look for a situation in which you will be sponsored and mentored, but do not stop trying while you do.

I have found it helpful in training new staff unfamiliar with the corporate world to characterize the various personalities that make up an organization. These generalizations are not completely flattering, and of course will not be true for each of the members I will describe, but we find that these gross generalizations do help anticipate the kind of interaction you will have with these people.

The **Product/Marketing group** resides on the business side of the company, they dream up ideas for products that they want us to build because they will have a positive impact on the ROI (return on investment, aka, the idea will make us a ton of cash) They historically do NOT have a complete understanding of what the technology is capable of, and are hired because they are driven to succeed, and this combines to create a dynamic that is less than conducive to clear communication with those in Engineering who will carry out their projects. I term these resources "driven, well dressed, good looking people".

The intermediary between Product and the rest of Engineering is the **Project Manager**, who writes up the project in a Functional Specification, and who maintains the schedule for development. PMs tend to be realist, and have the communication skills to convey the requirements for a project to engineering in a way that can be understood

Developers take that specification and build the code to implement the idea.

Combine the need to be able to fathom the different coding platforms with extreme creativity, and you will sometimes get an Artist, with all the generalizations that come with that breed. Someone who is not to be constrained by common rules, occasionally with large egos, and with an artist's inability to be realistic or care about schedules, earthly requirements, and where their work will affect other parts of the system.

QA staff are tasked with being the gatekeepers of the production system, and thus tend to be relatively inflexible, but at the same time, must communicate between the creative forces of development and the truly inflexible Operations staff who own the production systems. An ideal QA resource would be very politic, able to communicate, but with very high expectations and a kindly but firm intolerance for anything that disrupts stability and workflow.

Operations staff traditionally have tremendous technical expertise in systems, but less experience with Development practices, and thus quite often are mystified by the less than black and white world of creating the products they are required to maintain on the production system. They are hired explicitly for their inflexibility as their job is to protect the stability and security of production systems with extremely valuable data, including customer's credit cards, etc. Their experience tells them that they have a group of developers who believe themselves to be geniuses giving them code that breaks over and over, and so are hesitant to believe any promise made by developers, and rightfully so. QA and PMs end up running interference between Devs and Ops, as communications can often break down with such diametrically opposed mindsets

These would appear to be some very derogatory generalizations, but that is not the intent here. In each of these positions you want and need just these kinds of people. They each fill their roll in the business world, and are required to make a dynamic organization. But it is still true that these personalities are in some ways diametrically opposed in motivation and expectations, so it is here that we begin to discuss the tools that can be used to display for them the value of your team to the organization. But first one last gross generalization of the most important group of all.

Upper Management. We can sum up management by saying simply that they are bigger, meaner Program Managers. They may be charismatic and great leaders, or they may be Dilbert-esque, but upper management pays your salary. It is important to realize that for many organizations, the IT group salary is the single largest expense the company incurs. A corporation, or any place of employment for that matter, is very seldom a democracy, and management can unilaterally decide your future if sales are down. Especially in these economic times, they are super critical in every budget cycle of where the money goes. Combine this with the lack of understanding of what IT, and especially what QA does in some organizations and it is easy to see why management considers outsourcing overseas and eliminating staff. This also makes it clear that these

are the people that need exposure to what Quality Assurance is providing for the company. You must, if your team is threatened by cuts or political sabotage, make your presence understood and appreciated by those who control the budget. I include both subtle and less than subtle techniques below. Use whatever you feel comfortable with, and good luck.

The Tools

1. Process and Process Documents

You live in chaos, or at least some level of anxiety in your workplace. It's time to formulate a plan. I list a number of references in the appendix that can help you understand some of the current methodologies, but the basics are the same. To achieve structure, you need all of the basic steps in a methodology that allow you to control your environment and ensure stability. Ideally, you would get all of the following:

- Product or Marketing providing a rough sketch of what they want in a charter document or a product outline
- A ROI analysis that shows that this idea will indeed make X amount of money
- A preliminary Engineering estimate of the costs to build the idea, with QA and Operations input
- A kickoff meeting between Product/Marketing to discuss the idea, suggest technical solutions that Product may not be aware of, to get details about what is required and is NOT required, to get buy in from all parties so there is consensus for the effort, and to air out any dissention before the work begins.
- A functional specification with requirements, screenshots, and a modeling diagram to show each possible branch of logic
- A schedule with milestones and the ramifications defined for missing milestones
- A technical approach meeting or document, with Operations, QA, Project management, etc to have the developer/architect explain how this project will be implemented, so that Operations can plan on buying the required equipment, Project Management can plan the schedule, and QA can begin planning approach
- A test plan written by QA
- Test cases written by QA
- A technical review of the code once code complete is reached
- A test plan review with the business, developers, operations, and project manager.
- Launch plans or Build plans, including scripts to diff between different environments to verify builds
- Automation scripts to speed smoke tests of new builds
- A test report to show how far you have to go at any point in the testing cycle

- A triage meeting to review the bugs and make decisions about what will and will not be fixed.
- A final check off form with signatures from business owner, development, project manager, QA, and operations to show consensus.
- Release notes indicating what was released and what issues you are aware of.
- A post mortem to evaluate the performance of the entire team and make a list of mistakes to improve future projects.

Choose form this list the processes that you think you could sell to the organization, develop or find templates, and make your presentation to the decision makers, showing how these processes will help your orgainization. Provide documentation including the templates, training materials for the new process, and an ROI analysis that shows how this will save the enterprise money. The importance of documentation cannot be stressed enough. The best manager I ever had used to walk around yelling at everyone in the team, "ARE YOU PRODUCING?" and "SPEC! Show me SPEC!" Besides providing a foundation of structure to help an organization achieve reliable results, QA's primary function is to remove all assumptions. To do this, you must document exactly what is required and requested. The initial battle that must be fought in chaotic organizations is to display that there are issues, explain that something can be done about some of these issues, and finally negotiate what processes will be instituted to help. You want the craziness to stop, the business wants to continue producing very quickly, and so you will not sell your complete plan in one week. It is a continual process that begins with agreeing on what you will do in the first of many baby steps. I am unable to provide the actual documents that we use to define our processes, but I can provide their titles and a description of what each is:

1. All of the steps defined in the list above need to be documented and signed off on by the stakeholders, templates will need to be developed and versioned, and all of these should be versioned in a version management system. It is assumed that there are tools for SCR tracking and version management in your organization. If there are not, start by acquiring those in the first step of your plan, and then implementing as many of the process steps described above as you can as the next step.
2. Roles and Responsibilities, defines who owns what part of the new processes.
3. Project Workflow Diagram, shows what defined steps will be observed in a project.
4. SCR Workflow Diagram, shows what defined steps will be observed in resolving a SCR (a bug, a "System Change Request")
5. Escalation Criteria Spreadsheet, lists the criteria for escalating work priority by priority, what defines the priority, and who is responsible for making decisions along the way.
6. Standards and best practices documents.

7. Tool and process training materials. Someone will have to train IT staff in whatever new processes are instituted, and if QA is sponsoring these changes, then QA will own the materials and the training.

2. Email

I have used email as a means to place QA in the view of upper management. There are two basic types of email, official sanctioned, periodic reports, and then more personal, direct, unofficial emails. Official emails can be used to create a situation in which QA became the interface between IT and the business side.

- When new functionality is introduced to production systems, whether they be a web application, site, or enterprise client server applications, QA, being able to articulate in layman's terms what new functionality exists and what known issues there may be, becomes the logical choice for publishing Release Notes". In the last round of layoffs at my company, I was told that while Management really didn't understand what QA did, the question of preserving the QA team never came up, mostly because they saw release notes every cycle that told them in simple terms what new functionality was introduced to our production systems and what issues could be expected the next day.
- If QA is involved with production maintenance or monitoring, and you are in firefighting mode, with production systems down or functionality compromised, it is critical that clear communication if given of what the plan of attack is and who is responsible for doing what. In my current job, it simply fell into QA's lap to document what was decided at emergency meetings, and sending the plan out afterwards became another way to expose the value of QA to all of IT and to management. Now I am not suggesting that you send directly to the President/CEO with what horrible mishap needs to be fixed in IT, but including the appropriate members in IT will keep all responsible members on task and stabilize the process.
- Informal unscheduled emails are another way to inject good process into an IT organization. Once Standards and Best Practices have been defined in a group, it is invariable that someone will begin trying to erode these standards. I was at a QA conference in California in 2002, and heard from many of the testers that I met that they could not control the developers, could not get unit testing, could not get them to do peer review of each others code, and could not get technical specification. As mentioned before, if you do not have support to enforce these basic best practices, you must try to get it, and I have been very lucky in that when I have suggested these practices, the ideas have been embraced. Once agreement is made though, a strong voice is needed to maintain these standards. A sense of humor is essential here or you will burn bridges and simply build resentment. Here is a brief sample, pasted from an actual rant in 2001:

All,

I'd like to go into a little bit of detail here on what the process is when we are finishing up the development of a new system or updates to an application, so that everyone is aware of what is involved, and why I am writing this flamemail to attack any changes attempted in the copy in the new system at this point:
UI Frozen is where we say that the UI has been finished and will not change. This is HUGELY important for lots of reasons:

We use very expensive tools to write performance tests and automation scripts, and when the UI changes, even just a single transparent pixel sometimes added to a page, our automation and performance scripts fail and we waste time fixing them.
Every time someone goes into the code to make a copy change, they check the file out of the code repository. They can accidentally check out the wrong version, with old bugs in it and put it back into the QA system. they can accidentally add copy in the wrong place and break something. Or miss-direct a link and break the program flow. Lots of opportunities for big problems.
Also, we have testers write bugs on the new changes because they aren't aware that the new look is the intended look. And then we have to go look and check to see what it's supposed to be. And there is no record of what it's supposed to be, so instead of testing the functionality, we run around trying to see where the changes to the UI come from and what it's supposed to be.
Heaven forbid we get two groups, like Marketing and Sales, making suggestions at the same time. OhmyGod.
The copy changes to important keywords need to be corrected EVERYWHERE or we end up looking inconsistent, counter-intuitive, and badly organized to the user. Help files need to be edited, the help index needs to be fixed or people will not find the information they are looking for, etc.

That's why in Seattle, we try real hard to make sure that when we say such-and-such a date needs to be a UI freeze it's clearly understood that we can't make ANY changes afterwards, and that's why we make it a hard rule that no copy changes occur unless the Engineer uses a specification document before a freeze that has been reviewed by QA, or a bug report. That way we have all changes going thru one central documentation area.

So when a QA resource finds that he/she has finally gotten a good grasp on what the product is supposed to look like, has tested and discussed, and consulted, and struggled 97 hours in a week, and makes it clear that we really need a frozen UI to be able to test, I'd expect and hope that everyone would understand that he/she, groggy, tired, and frustrated, would be REALLY PISSED OFF to find out that the heaven forbid situation described above had come true.

So the solution here is:

- *use this email as a final declaration, if there has been any vagueness, of how we in Engineering and QA REALLY feel about this*
- *for everyone to make sure anyone who might ever want to change copy after a freeze has their awareness raised by forwarding this email.*
- *understand that any change requests that do not go thru a bug report will lead in the copy changes being backed out unilaterally by QA,*
- *and raise awareness in the dev group that copy changes cannot be made after a freeze unless a bug is written and triaged by the mgmt team.*

Pedantic, condescending, and strongly worded with enough humor to keep form burning bridges, (in my organization at least). There are specific process points explained in no uncertain terms and expectations set at the bottom.

At the conference, when I told people that our devs have code review, and write unit test cases, and write technical spec, many asked how this was possible. I told them that: 1. I am very lucky to currently have a boss who supports my efforts, but even before my current boss, in extreme chaos, I was able to introduce standards and maintain expectations, by NOT being a nice person. It does take force of will to achieve these changes, and after the initial battle is fought, it takes constant vigilance on the borders to insure that the standards are not eroded. Email is a fine vehicle for educating about the value, providing status on current efforts, and preserving QA's ability to enforce standards.

Now while I do not encourage sending directly to upper management, I should mention that I have sent 3 page rants directly to the founder of a company I worked for, when I felt there was no other recourse to the absolute chaos that existed in the entire organization. I could have been fired for the email, it was very strongly worded, and the President/Founder really didn't understand the points I made about process, but it did

firmly cement QA as a force in the organization and helped enhance our ability to affect positive change in IT, as there was understanding in management that we were well intentioned, impassioned, and had a plan to help the company achieve what it was trying to do without serious failure, which was understood to be a requirement for our IPO.

- One final use for email is for legal documentation. I encourage everyone who works with anxiety and uncertainty to keep every email they every get outside of spam. I have helped my companies avoid 3 separate lawsuits with email that was over a year old in all three instances. If you struggle to get standards in place and are pressured to produce without enough infrastructure in place, use email to protect yourself. In one instance, where we asked for the English test cases for the Visual Basic Toolkit for building CE applications, I stated, without thinking really) at the end of the email: "we cannot test adequately if the US test cases are not provided" The cases were never provided, and I am certain that they didn't exist. We shipped the Japanese version of the product with a serious flaw, and were threatened with lawsuit. This single email avoided the entire affair. If there are expectations that you will pull a miracle and get a project out without enough time or structure in place, send an email requesting delivery receipt, stating your concerns and state definitively what will prevent success. Use personal folders to keep emails on you local machine or a save location, and periodically burn all your sent emails and received emails to CD or tape and keep these archives very safe.

3. Metrics

Bug Metrics is an area even established code factories have a hard time getting to, but I believe it well worth the effort to develop a small set of reports that help you display in simple graphic form information that will help the rest of the organization see what is happening in IT.

The very simple reports are current open, closed this week, defects vs. enhancements, resolution rates and discovery rates. It is outside of the scope of this paper to describe these, but again, it is very easy to investigate how to create these. What I am suggesting here is that you try hard to find ways to have these reports viewed by those who pay you. A few other reports that may be useful are evaluative reports, which help you control what areas of the IT process to try to improve on by displaying what areas cause the most cost to the organization.

The first is called a Root Cause report, and through this you are able to track what kind of mistakes are made that cause defects to be introduced to the system. We use these values to categorize what CAUSES issues to be introduced into our systems currently:

- Changed Requirement
- Missed Requirement
- Missed Dependency
- Configuration Management – Human error
- Configuration Management – Launch Plan error
- Configuration Management – Versioning error
- External

- Legacy
- Field Validation error
- Etc.

If you assign the values for each bug entered, you can at the end of the month or quarter or year tell how many of each defect are introduced. The causes with high numbers point you directly at where you can work to reduce those numbers and thus cost. You can group cause by individual developer, but I warn you to never do so without the approval of the development manager, and make absolutely sure that these reports are shared only with that single person. This information in invaluable in helping a Dev manager improve his or her staff, but it could be disastrous to have such sensitive information go outside.

The second report is a simple accounting of Re-opened SCRs. Through examining all the issues re-opened in a year, you may be able to address the most inefficient areas of your processes. We found that over 90% of our reopened issues in 2001 were configuration issues. By focusing on CM, we were able to improve the reputation of IT in the company and save money by reducing the time spent investigating issues that were not defects in the code.

4. Budget

While you may not have control over your budget, you can still affect the money allocated to your team. Say you have no input into your team's budget at all. This means you have a fixed amount of dollars that you have to work with. Most tools used by QA Teams have support costs, and the annual fees can be substantial.

The reps that contact you from these vendors are always anxious to have you renew, and you may notice that they begin to call in the middle of a quarter, and become really insistent and anxious as the end of the quarter approaches. The reps have sales quotas that they must meet at the end of the quarter and year, and as salespersons, they understand negotiations very well and are used to playing hardball. They are paid commission on the money you give them usually, so it is in their interest to NOT discount support costs.

You can negotiate the support costs or initial purchase cost of your tools. I watched a beautiful display of this by a CFO. I had negotiated 15% off the price of a complete suite of tools, getting the cost down under $45,000 to the company for 40 seats. I asked the CFO to try to get the price down one more time, as the rep was being very difficult, and he agreed to try. I sat and listened to the call, and our CFO was direct, honest, and absolutely ruthless in his insistence that we could not pay more than $15,000. I smiled thinking that he was about to run into a wall, and was shocked to hear the rep eventually break down and agree. Not satisfied with this, the CFO kept pressuring for more discount, and finally was able to win another $1000 savings. Negotiating existing and new tool support allows you to use those savings for your team. I have taken savings from recent negotiations, and started an internship program at my current company. I find that within a month, I can have interns, who are quite often recent college graduates in Computer Science or with IT AA degrees, and a good grasp of current technology, working as Jr. STEs at minimum wage and eager for the chance to learn. Of the 6 interns who have

come through, all have felt the program a huge benefit to them, and 3 have extended their terms, and so after 6 months I have people who are domain experts in areas of our systems and able to greatly increase throughput. Through this, I have been able to preserve training budgets and existing full time staff. A complete win-win for my team, IT, and the company.

5. Scheduling

Traditionally, scheduling has been the domain of project managers, but if your world doesn't include set milestones and an automatic extension of time when you receive code late, then you might as well take matters into your own hands.

There are many scheduling programs out there, but by far my favorite is , MS Project. While there are many who believe that it is counter-intuitive and unstable, I had the pleasure of working on the Japanese version for one year, and I learned that each function in Project is driven by established project management concept written into the program logic. Keep it very simple and you will hopefully get comfortable with this powerful product.

In that first large project in localization, testing Cold Fusion Server Japanese, I was put on the test team on my first day on the job. The director told me to report to the lead, and so I set up my systems, and went to see him. I said, "Hello, I'm Bill, and I'm supposed to work on the new project. He told me to get to work, so I asked for the test plan. He again asked me to get to work, so I asked to see the test cases. He became agitated, asked why I was being so uncooperative, and again requested that I get to work immediately. I asked, ".. is there a specific area of Cold Fusion you'd like me to work on?" and he threatened to talk to the director if I didn't do as he asked and GET TO WORK!. I stayed after hours that night, got out my Beta of MS Project Japanese, and opened the CF Server manual. I broke out the various areas of functionality into tasks, guessed at the time it would take to do them, and arbitrarily assigned those tasks to the other team members I had met that first day, depending on what they had said and guesses about their technical ability. There was a Japanese American there who spoke not a word of Japanese, but looked Japanese, but was technically very able, I assigned the functional logic to him and the Adminitration dashboard translation review to a Caucasian who was nearly fluent in Japanese. The next morning I asked for a meeting with the Director, and explained to him that the project was doomed, and presented my Gantt chart. The Lead was suddenly called away to an external assignment, and I was made test manager. It was not that I was capable of managing a large test team at that point, but I was the closest they had.

Another use for project is milestone tracking. If your organization creates estimates for projects, you can take the original copy of the estimate, and even if you don't have ownership of that file, make a personal copy to your local. Three weeks later when the schedule shows that the milestone for "Handoff to QA" is a week away, you can pull out your copy and show that originally, it had been

scheduled for a week ago, or you can use the Tracking Gantt feature to show how QA cycles are truncated.

Even if no one in your organization tracks hours spent on projects and maintenance efforts, you can institute time tracking for your group. Our group was constantly having our QA cycle truncated (still are actually, the battle never ends) but you can build a "Bucket" spreadsheet that shows your staff, and for each staff member, a list of the tasks that they will be assigned, including administrative tasks like status reports and email. We figure that we can count on 6 hours a day of productive work. Take your estimates of the time that will be taken by your staff for these tasks for this week, and when the "bucket" is full, reaches 6 hours a day, you can refuse requests for more work. You now have the information to show that you will either need to pay overtime, reduce the scope of testing, or extend the schedule.

Conclusion

I hope that some of these ideas can help professionals who work in less than professional settings. Last year as I attended the Software Test Automation conference, I talked with many people who were experiencing the same exhausting, anxiety provoking situations I had as I built a career. I do see the industry maturing, but believe that there is no single thin line between well and poorly run shops. Every organization can improve their processes in some ways, and I hope this provides some insight into how force of will, persistence, and a solid plan with ideas can help you improve your situation and affect positive change.

Resources:

http://www.standishgroup.com/

http://www.sei.cmu.edu/

http://www.mpug.org/

http://www.stickyminds.com/

http://www.di.ufpe.br/~java/graduacao/referencias/SciAmSept1994.html