

For the 2008 Game Developers Conference, I was invited with colleague Dennis Mooney to co-host a roundtable session on the topic of QA, which we titled "What do you need to become a great QA team?" Three one-hour sessions were scheduled for the main conference days: Wednesday, Thursday, and Friday.

What follows are our feedback about the logistics of the sessions and the conclusions that we (including the attendees) drew from the discussions.

Logistics

We had excellent turn out each day with many repeat attendees for each session. The rough numbers per day was as follows: 70, 40, and 30. No direct feedback from the attendees that the time slots were problematic.

For our particular session we found it extremely helpful and useful to have two moderators. One moderator could focus on maintaining the flow of the discussion and general moderation while the other was taking notes and keeping the conversation on topic.

Many attendees participated in both QA round tables. Based on that we believe there is ample room for multiple QA specific sessions and attendees found value in the information.

The easel and pad of paper was quite helpful, but some audience members had difficulty reading from the back of the room. Recommendation is to consider providing an overhead projector and screen. This would also allow the presenters to create materials beforehand to help further spark better discussions.

Consider having the capability to record the audio of the round table. We took this upon ourselves and found it very valuable in helping write this document and for our own edification.

We had to request that a QA forum be created for the GDC website. This struck us as a bit odd since there were QA focused sessions at GDC this year and previous years.

Daily Sessions

Day 1

Wednesday, February 20th, 2008

There were four major themes to the discussion on day one:

- The roles of testers during the game development cycle
- The evolution of testers' skill set
- The impact of testing on quality
- The challenges of communication between test and the rest of the product team

The roles of testers during the game development cycle

One attendee posed the question of when should test be involved in the product cycle. Answers generally pointed to "as early as possible", with attendees describing their challenges and successes in this area. The most often-voiced problems focused around the technical teams (programming, audio, art, etc.) either not trusting the test team or not knowing what to do with them in the early stages of the project.

To address the trust issue, the test team needs to build and maintain a reputation for technical competence and subject matter expertise so that its members can be regarded as equals when key decisions are made before and during production.

To address the uninvited-guest issue, it was recommended that management not just buy in to the idea of having test at the table early on in the project, but should also clearly communicate to all team members the reasons why it is valuable to have the test team represented at each phase of the development cycle.

The question of how the test team should be represented revealed a number of different approaches:

- The traditional [waterfall](#) approach puts the test team near the bottom of the waterfall, at or after the point when feature development has been completed and only content and bug fixes are being introduced.
- The [Agile](#) approach, which is gaining popularity, recommends integrating a representative of the test team into each development team, putting their finger on the pulse of each feature from the beginning. Advice around this area includes the use of experienced testers in the role of scrum master.
- An approach that doesn't require either waterfall or Agile involves embedding testers into feature teams that correspond to their aptitude or relevant expertise. For example, one company benefited from assigning a tester with a background in music (e.g. enthusiast, performer, teacher, sound engineer, etc.) a sound-proof room with high-end audio equipment and mixers that allowed the tester to easily listen for quality issues in the original audio recordings against the output from each of the platforms that the game was built for. The tester was also provided with a specially designed level of the game where s/he could easily test each sound effect. This tester's passion and management's recognition of this tester's potential contribution to the quality of the game were a successful match of skills and resources.

The evolution of testers' skill set

How and when to add test representatives to the production effort brought up the question of what skills testers should bring to the table. Obviously, bringing unskilled labor into a game design meeting isn't a recipe for success. However, opinions on hiring, training, and applying skilled testers varied, showing that the skill sets desired of a test team are evolving away from unskilled playtesting.

- One attendee was very clear about how he prevented the test team from having the reputation of [a room full of monkeys](#): he didn't hire them. The implication being that he brought skilled testers onto the team by hiring skilled testers. This would obviously be the ideal way to build a great test team, but not all companies allocate a high-enough budget for the higher incomes such talent would ask for or are located near such a talented job pool.

- The attendee who brought us the audio tester success story gave the impression (apologies if I'm mistaken) that the team took a more serendipitous approach to matching testers' aptitudes with their testing responsibilities. Another example cited was of a tester who was described as "possibly being an English teacher in a previous life", who was assigned to review all of the in-game text and the documentation for mistakes of spelling and grammar. The lesson here is that the test team did the best it could with the resources that were available to it.
- Another attendee described a middle ground where people hired onto the test team were given training in key skills before they were assigned test responsibilities on an active project. Granted, much of that training is in the tools and processes unique to that company (e.g. e-mail systems, bug-reporting systems, debugging tools, cafeteria hours, location of the toilets), but the investment in training for testers doesn't have to be limited to the minimum skills necessary to be productive. Continuously investing in the test team's skill set makes them more capable in the long run.
- Still others cautioned against going too far to the extreme of hiring only technical testers: playtesters were still considered a valuable resource because they most closely resembled the end-user, and could focus on the breadth of the game experience rather than the depth of a particular technology area. However, the test teams with a technical focus found outsourcing these playtesters to be effective.

Some mentioned that these changing expectations for technical skill sets among testers can lead to confusion in the terminology applied to testers. For example, when someone identifies himself as a "tester", should you automatically assume that s/he has or doesn't have technical skills? Some companies distinguish between technical and non-technical testers by job title. Similarly, the distinction between QA and testing is unclear to some.

The impact of testing on quality

In response to complaints that testing is brought on board too late to have more than a superficial impact upon the quality of a product, many people recommended having the test team be represented at earlier and earlier stages of the project, where issues in the design of the game or features within the game could be addressed earlier and fixed less expensively. Key reasons for testing to be involved early include:

- Bugs found earlier can be fixed earlier and more cheaply than after the game has shipped. For example, correcting problems with game design helps reduce costs by not allowing those problems to be written into the code in the first place.
- Defects in tools, pipelines, and processes can also lead to bugs in the product if not addressed. Making sure that the tools work properly and that each step in the build process is working properly helps prevent people's hard work from being corrupted somewhere between their desk and the final build.
- Testability can be designed into the game to make it easier for the test team to create test automation or easier to ask the game for data that might reveal unexpected behavior.

In response to suspicions that overly technical game test teams could end up with functionally sound--but dull--games, several attendees supported keeping playtesters in mind as a valuable tool for evaluating less objective characteristics such as how *fun* a game is.

The challenges of communication between test and the rest of the product team

During each of the three days, in both our QA roundtable and in Chuck McFadden's, the recurring theme was about communication problems between the test team and the rest of the team. Some solutions for improving the working relationship between development and test were:

- Understand the root causes of the lack of trust, then address each of those.
- Make sure that the communication problem is not on *your* side: make sure that your testers are giving the developers the correct information in a consistent, timely, easy to understand, objective, and actionable manner. If there is any room for ambiguity, either remedy it or call it out explicitly so that both parties understand the limitations to the defect report (e.g. the defect only appears once in ten attempts, but it might take another 8 hours of investigation to improve this to once in every two attempts).
- Make sure that the priority and severity of issues are well understood. In other words, don't give development the excuse to stop paying attention to the test team by allowing them to [cry wolf](#) (exaggerate the importance of bugs).
- Gain support for the test effort from management. Once management understands the value of testing, have them use their authority to communicate this value to the whole production team.

Day 2

Thursday, February 21st, 2008

Today's themes included:

- Documentation
- Tools
- Staffing
- Quality and Methodology

Documentation

Writing ideas down makes them easy to communicate and share with others. Design documents shared with the test team enable them to understand the product better and gives them a chance to give feedback and prepare the necessary test infrastructure. Test documents shared with the production team keeps everyone informed about what the test team needs to be effective, and gives them a heads up about the criteria that their features will be measured against. Sharing these documents with the whole team helps build camaraderie by helping everyone understand how they are working towards the same goal: an on-time release of a high-quality game.

The benefits of documentation:

- Sharing (not everyone has to be in the meeting room at the same time to receive information)
- Reference (people don't have to rely only on their faulty memory of what was said)
- Updates (the team doesn't have to live with its mistakes--it can fix them)

The problems with documentation:

- Insufficient documentation leads to ambiguity and inconsistency.
- Writing too much documentation steals time away from other efforts.
- Documents can grow stale if not actively kept up to date.
- Copies of a document can cause misunderstandings if changes are made to one copy and not propagated to all copies.

Tools for storing documents:

- File shares
- [Microsoft SharePoint](#)
- [Google Docs](#)
- Mercury TestDirector (now [HP TestDirector](#))

Suggested solutions to some documentation problems:

- Use the Agile development model instead of the waterfall model to keep testers directly in communication with feature development teams. Test requirements can be generated on the first day of each two-week sprint, allowing testing to be productive in parallel with development.
- Make documentation a requirement. If you don't allow design documents to go missing then the test team won't have to guess how a feature works when planning how to test it.
- Whenever a developer is assigned a task, assign a tester to that developer to figure out how to test that task. The tester can then give feedback directly to the developer to make sure that the feature is written in a testable and usable manner.

Tools

Flexibility was heard over and over again as a desirable feature for a test tool.

In particular, test management tools were found to be lacking in this respect. Many teams used [Microsoft Excel](#) for everything from simple tables and checklists to custom test case management, budgeting, and scheduling applications. These features are also available from ready-made test management products such as [DevTest](#), TestDirector, or [QuickTest Pro](#) (the last two from HP). What made some prefer Excel was its programmability. Even though the off-the-shelf tools represented a full-featured test management solution, some companies did not want to--or could not--change how they worked to fit into the model that the tool was designed for. Although some companies found it worthwhile to adapt their processes, many would have preferred being able to customize the tool. Of course, customizability comes with the price of having to have someone on hand who can write the code necessary to adapt the tool to your needs.

Features attendees wanted to see in a test management solution:

- Ease of use, so that even junior testers can quickly learn how to use the tool
- Customizability, so that any differences between what a tool provided and what the test team needed from the tool could be easily added, instead of making the test team bend over backwards to take advantage of the tool's features
- Test matrices, so that a large number of test cases can be represented with a small amount of writing; see [PICT](#) for a relevant tool for reducing the number of test cases without significantly reducing [code coverage](#))
- Test cases

- Prioritization of test cases, so that more-important test cases can be scheduled before less-important test cases
- Time estimates per test case, so that test leads can predict how long test passes will take to complete
- The ability to choose test cases automatically based on feature area, priority, and time requirements, so that very targeted test passes can be quickly scheduled
- The ability to schedule test passes at various levels of detail
- The ability to generate status reports from the pass/fail results
- The ability to associate test cases with specific status reports, so that stakeholders could receive reports that only contained the information they cared about; examples of reports:
 - Summaries for producers
 - Daily summaries
 - Deeper status
- The ability to print out pass/fail checklists of a given subset of test cases, so that test leads can give these printouts to testers who didn't have easy access to a PC to access the test management tool directly (e.g. a locked-down playtest lab)

Other than for test management, some companies found tools useful for finding defects. Off-the-shelf tools provided as part of software development kits were often cited as useful for both technical and non-technical testers (with a little help). Windows and Xbox 360 development tools such as Netmon, PIX, and XSim were mentioned, but no Playstation- or Nintendo-specific tools were named. Custom tools were also developed in-house to aid in the discovery of a variety of defects such as memory leaks, framerate drops, and unexpected network traffic.

Worthy of note was that not all test teams or not all members of the test team have access to these development tools. How are such test teams effective without those tools? Do they make their own or do they live without?

With all of this talk about tools and the requisite programming skills necessary to adapt or write tools from scratch, someone brought us back to the title of the roundtable, "What do you need to become a great QA team?" He asked about ways to objectively measure a test team, by analogy with the [Capability Maturity Model](#). He proposed the following five levels:

1. Seat-of-the-pants testing (ad-hoc testing) with no test documentation.
2. You have documented test cases.
3. You also have metrics.
4. You are more involved up-front with developers, you have a standard of excellence for QA.
5. ?

He hoped that the audience would have an idea of what level five should be, but other than a joke about "omnipotent perception", there were few takers. The point was well taken, though: it would be helpful for test managers to have a scale to measure their teams against, and a clear path from one level to the next.

Staffing

Key points in staffing:

- Identify people with good analytical skills who are self-directed. See [Anibal Sousa's poster](#) for a hierarchy of characteristics that make a great software tester.

- Hire people first and then find out where to best apply their skills, or hire people for specific skills? The dragnet approach has the risk of pulling up deadweight while the targeted approach has the risk of coming up empty.
- Value diversity in background and skill sets. Don't just hire [left-brain thinkers](#). Think about how people with a background in languages, music, art--and yes, gameplay--can add value to your test team by finding bugs that a homogenous team couldn't.
- Take advantage of subject matter experts. If someone likes doing something that your other testers don't, chances are good that s/he'll be happier and more productive if assigned to test related features.
- Centralize your test team, or embed it into development teams? Some companies have had success with embedding testers in their development teams for better communication, work flow, and rapid iteration. Others warn that embedding runs a (small) risk of [Stockholm Syndrome](#). Of course, for some companies, such as publishers, it is impossible to co-locate the test and development teams due to geographical separation.

Quality and Methodology

Key points:

- Drive quality upstream. It was strongly and repeatedly recommended that defects be found as early as possible, up to the design phase of the current project. Some even suggested taking the next step by driving quality into the culture of the company so that instead of just addressing the quality of one product, a test organization would champion the changes necessary to improve quality for current and future projects at the company.
- Set standards. Agree upon the role and responsibilities of the test team relative to the other functional groups. Agree upon the terminology that you will use to describe every part of the product, so that there is no confusion when describing what's wrong in a bug report. Agree upon the levels of priority and severity for issues, so that everyone can correctly evaluate the relative importance of one bug over another. Agree upon a standard of quality for your bug reports, so that developers don't feel that their time is being wasted by trivial issues or poor communication skills.
- Keep lines of communication open. Make sure that everyone is working with the information that they need to get their jobs done correctly and on schedule. Information that is delayed or withheld can have a negative impact upon productivity and morale.
- Test early, test often. Driving quality upstream doesn't only mean changing when you involve testing during the project cycle, it can mean changing when you involve testing during the build cycle. Some developers find value in giving private builds to a tester that they respect to get early feedback about changes that might have a significant effect upon the quality of the game. In the ten to fifteen minutes that a tester can explore a new feature, some design, implementation, and integration issues might be identified before the developer has a chance to affect the next build, potentially preventing many hours of lost time and effort by the rest of the team. The developer can even make quick adjustments with the tester present to give quick feedback. Test teams that are not co-located with the development team--as is the case with most publisher-developer relationships--can't take advantage of this, but will ask the development company to perform its own sanity testing prior to submitting a build to the publisher's test team, so that the publisher's time isn't wasted with a build that is dead on arrival.
- [Black box](#), [white box](#), or both? Some test organizations still engaged only black-box testing practices. This has the benefit of not requiring highly skilled and technical test staff, but the drawback is that there are entire classes of defects that cannot be detected or accurately reported without a technical background. A growing number of test

organizations engage in white-box testing, but although there are relatively fewer people who meet the skill requirements, these testers can find a greater variety of defects and can communicate them more effectively to development. Some organizations value both kinds of testers (see [The evolution of testers' skill set](#), above), and apply each to their particular strengths: the technical testers to the left-brain problems and the non-technical testers to the right-brain problems.

Day 3

Friday, February 22nd, 2008

Today's discussion was directed around successes. After two days of talking about problems, we wanted people to talk about their solutions. To guide the discussion, we organized them under the stages of production.

Pre-production successes:

- Add QA to the team early in production (QA management at least)
- QA review of design specifications
- Prepare for/anticipating potential problems
- Review TRCs and TCRs to catch design issues early
- Review competitors' products for a given genre that you might not have shipped a title for before
- Use mock-ups, storyboards, usability studies, and UI flow to communicate ideas more effectively
- Establish a common vocabulary for terms
- Write test documents and test plans
 - Test plans for identifying resources, staffing needs, risks, and procedures
 - Test specs for describing the testing of individual features or groups of similar features
 - Test matrices for describing large numbers of test combinations
 - Test cases for describing individual tests
 - One company found that test documents were also helpful for identifying feature ownership, which gave members of the team a sense of inclusion and worth
- Review how bug reports are written to ensure concise/accurate repro steps, descriptions, prioritization, etc. Do this on an ongoing basis throughout the project.

Production:

- Inspire shared responsibility for quality across the entire team--all disciplines
- Encourage effective communication
- Have all disciplines meet with a prototype/mock-up to come to an understanding about the product under test
 - Revise expectations and create feature lists, driven by QA
 - Iterate in a feedback loop to keep design docs, specs, tests, and other relevant documentation updated and in synch
- Walk through all of the features with the entire team
- Create BVTs: build verification tests to quickly detect blocking problems
- Bring tests only quickly as new features are added

- Train your staff continuously; consider training your external partners and contractors, too, to make sure everybody understands how to work with you effectively
- Invest in tools, even if they are specific to a given title

Post-production:

- Teach your customers your best practices and give them the tools and documentation to help you work better together
- Improve the reliability of your tests by automating them
- Track changes to tools and features to keep everyone informed
- Keep customer support in mind and involved with the project

To wrap up during the final minutes of the final session, we asked everyone to name one thing that they learned about QA during GDC. We didn't get all the way around the room because we didn't give ourselves enough time, but many people indicated that others had already spoken for them, so the exercise was a reasonable sampling.

So, in no particular order:

- Allow new hires a training period to get up to speed and be more effective
- Allow a ramp up / training period when switching to a new role / area
- Learn new things, stay current
- Don't be in denial about the value of QA
- Unit testing
- Evangelize testing, quality at all levels
- Learn from peers and others on the product team
- Track metrics
- Measure, analyze, improve
- Levels of automation
- Look for more skilled testers - invest in training testers with potential
- Review hiring practices
- Don't undervalue build engineering
- Keep everyone accountable for quality
- Think globally, be sensitive [to localization issues]
- Think outside the box, consider issues beyond your features, market, customer
- Demonstrate QA value to the team
- Pair with dev to test their changes before they check in their code
- Build productive relationships with dev [other teams], be polite, respectful
- Be proactive
- Get to know your team - give QA a face attached to that name on the bug report
- Face-to-face meetings can be productive, humanizing, build camaraderie
- Beware Stockholm syndrome - going native. This can occur when you embed test with dev, especially when in a different city/country
- COMMUNICATION!!!!!!!!!!

In Closing

Dennis and I were very pleased with our experience co-hosting the GDC 2008 roundtable sessions "What do you need to become a great QA team?" We learned where we could

improve our moderation from the example Chuck McFadden set in his "Improving QA" roundtables, and from our own mistakes.

We see opportunities for game testing to follow the lead of other professions in setting standards by which test organizations can measure themselves, compare their efforts against others', and make informed choices about how to measurably improve their testing efforts.

We see that test organizations at all levels of maturity are interested in improving the quality of their efforts, and that there is no One True Way to enlightenment, yet.

We hope that GDC and similar conferences will continue to provide QA a venue for discussing our concerns and sharing our solutions, so that we can help each other find our own path to shipping great games with great quality.