An Expert Discussion on App Quality

BBC and AQuA Round Table

Introduction

BBC Future Media (the team involved in the production of mobile apps for the BBC) recently approached AQuA to request that we bring together a group of like-minded experts to discuss the fast moving and at times complex area of app quality.

As a result, on February 4th 2014 we convened 25 leaders in app development and testing to exchange views, debate issues and share best practices in a one-day closed session round table.

> *"A truly open forum to discuss common issues with like-minded peers that you don't normally get to share a platform with"*
> **Marc Morrison, Penrillian**

With organisations such as BBC, BluFocus, Epicentre, Keynote Systems, VMC, Kotikan, MX Data, Penrillian and Webresint represented, the quality of discussion was impressive and we gained many valuable insights.

This report shares the general discussion, insights and conclusions from the round table in order to share valuable expertise on app quality with the industry as a whole, which was divided into areas, agreed upon in advance by our group of experts.

> *"The round table was both insightful and re-assuring in discussing common industry challenges within a safe open environment. Our mobile app industry is relatively young and we will continue to evolve best practice together for the greater benefit"*
> **Michael Cliffe, MX Data**

Discussion, insights and conclusions

The areas covered

The day was divided into eight sessions, agreed upon in advance by participants, each of which commenced with an overview from a relevant expert within the group and was then opened up to discussion. These sessions covered the following areas:

1. Privacy Issues
2. Test Automation
3. Continuous Delivery
4. Beta Release Testing
5. Managing Device Coverage and Device & Platform Updates
6. Framework and Tools
7. Stress Testing and Stability Testing

1. Privacy Issues

The discussion focused on the key topics of control, protection, transparency, permissions and operating within the law.

In the UK, legal implications of privacy issues are handled by the Information Commissioner's Office. They have one person responsible for the whole of ICT, so obviously mobile apps are only a small part of that person's workload.

Participants expressed interest in exploring how to build privacy & permissions guidelines into the development process and considered how the process might have to adapt to handle this.

It was noted that while there have been moves within the regulatory community to create a form of certification, a 'privacy seal', the idea as currently proposed it is likely to be very expensive (perhaps up to €20,000 per app which is clearly untenable). Privacy should not be a barrier to entry for mobile app development, so dialogue with regulators is needed to ensure that there are proposals suitable for all sizes of developer.

In discussions on what might be suitable for different categories of apps one issue identified was that it is possible for an app to be categorised as a lightweight "lifestyle" app - such as a fitness monitor – yet be handling exactly the same personal data as a "serious" medical app, giving rise to information security implications for developer and user. There may be a need for developer education in this area, especially with the possibility of more stringent regulation of data privacy in the future.

It was concluded that there is a real need for dialogue between developers and regulators (because the voice of the small developer is not currently heard at all in that area), that developers need to take part in shaping the privacy debate, and also that consumers want honesty and clarity from developers about information handling and permissions requirements.

2. Test Automation

The initial concepts opened up for discussion were:

- "Automated validation that your app behaves as expected, and which documents that behaviour".
- As many stakeholders as possible should write the tests – this allows buy-in from customers right at the beginning of the process.
- No one is monitoring the tests as they run, so you need to have verified a high level of trust in both the tool and the results.

One productive method of developing tests was referred to as the "Three Amigos" approach – teaming a product person, a developer and a test person, to work out the usage patterns and develop scenarios. From this they can work out which tests can be automated. After any automated testing session, there will always be some issues needing investigation, so manual testing is then used to drill down to the reasons for failures, and evaluate their significance.

It was agreed that automated testing is valuable but has limitations and therefore could never completely replace manual testing. To illustrate this point, one participant explained that he had spent some time investigating the costs of automated testing, particularly with regard to setup, scripting and proving the validity of results. His conclusion was that it was only worth investing in test automation where a particular test was going to be performed 11 or more times – for less than 11 iterations it was consistently cheaper to test manually.

There was also discussion of the terminology surrounding testing and test results, with agreement that this was an area that could cause communication difficulties, particularly when using external testing providers, as each tended to use a differing terminology.

3. Continuous Delivery

Some of the concepts that emerged from discussion were:

- Continuous means agile: continuous feedback and testing improves quality.
- It gets software into people's hands quickly and allows assessment of high risk items early in the project.
- It can be complementary to automated testing.

Two aspects were regarded as important:

a) Technical infrastructure and working practices need to be defined and in place, otherwise Agile can be used by some people as an excuse for corner-cutting and poor working practices.
b) Customer buy-in is essential: it's necessary to get the devices and product deliveries into the hands of the customer quickly, and to get their commitment to giving the continuous feedback that is essential for this technique to provide benefits.

One challenge identified was the need to speed up test phases, so as to get deliveries through the pipeline as quickly as possible and avoid significant issues in a delivery being uncovered after the next delivery cycle was already under way.

Continuous delivery was viewed as useful in enabling multiple deliveries of individual components by moving responsibility to the development and test teams rather than a central admin or ops function. By making the development team responsible for the success of each delivery under test, it made a major contribution to improving app quality.

One downside that was that it was not always as effective as staged deliveries in a waterfall model, in that unless they were tightly managed, iterative builds could be time consuming and a distraction from the objective of creating a single build of identifiable quality that would be suitable for release to the customer. However, continuous delivery was still seen as a positive influence, in that features needed to be complete and testable to get through the process, effectively weeding out incomplete implementation of features.

4. Beta Testing

Beta testing was identified as a valuable tool for obtaining feedback. However, the management of feedback – analysing and grading the importance of issues raised – was not a trivial task.

Managing the distribution of beta versions for testing was also a potential issue – there were differences between the different OS platforms in terms of how easy it was to manage beta versions across larger groups of testers. Sometimes tools and commercial solutions were found helpful – Apphance was mentioned for on-device reporting, and virtualised testing labs had also been useful where access to the app or device needed to be tightly controlled.

Google Circles were also found useful for creating a community of beta testers and managing their feedback, and Hockeyapp and the open source equivalent Hockeykit were also singled out as tools that encouraged user feedback.

Feedback from beta testing was generally brought back into a product review where the final decision on any action to take would be made by the product manager or owner.

It was agreed that to get the best results out of beta testing, feedback needed to be tagged as originating from a particular tester, and they needed to be kept aware of the decisions made as a result of their specific input, so that they would feel their work was of value and contributed to the make-up of the final product. This meant that a certain amount of time and effort had to be invested in managing communications with beta testers properly.

5. Managing Device Coverage and Device / Platform Updates

Lots of difficulties were identified in obtaining devices and ensuring they had the correct level of firmware / software. The best choice for many seemed to be to partner with a testing organisation rather than attempting to maintain their own stock of devices.

Participants from testing organisations shared that testing services do their own research to identify the most popular devices, and those that are representative of the largest segments of the device population then purchase and maintain their pool of testing devices accordingly. Their experience has shown that typically devices were relevant in the testing pool until their popularity dropped to the point where they had little resale value, and at that point would be regarded as end-of-life, going for recycling rather than resale, and therefore no longer needed in the testing pool.

6. Frameworks and Tools

Most participants had experimented with frameworks, however many had eventually discontinued their use. Problems cited were updates to frameworks sometimes lagging behind OS releases, which could lead to apps that suddenly stopped working, and difficulty in recruiting developers who were willing to use a framework – most were only interested in working on native apps.

A number of participants also said they found that apps which were developed as a hybrid to work across multiple platforms were more difficult to manage and to maintain their quality, compared to developing multiple native versions of an app for different platforms, each focused on getting the best out of the specific platform.

One developer reported particular success with following the native route, but getting iOS and Android developers to work together in pairs, as they both understood the app well enough to be able to identify potential issues or errors in their partner's code, even though that was not their own development environment. The result was a kind of automatic code review, and their identification with both versions as a single output from their teamwork led to an improvement in quality.

7. Stress Testing and Stability Testing

There was discussion around how stress and stability testing could be defined and differentiated. A couple of definitions that were offered:

- Stress testing is generally short-term testing covering abnormal conditions of usage, trying to displace the device or app out of its normal operating conditions.
- Stability testing is generally longer-term testing, using the device and app under normal conditions, but with extended run time and in the presence of other running apps or services.

One thing that deters against significant use of either type of testing is that the majority of testing cycles need to be quick to deliver worthwhile benefits within an Agile development process.

Generally the group saw more value in stability testing, in that users often accepted a certain degree of instability in mobile apps under conditions of stress. It was also felt that stress testing was not something to be used in a continuous development cycle, rather a double-check to make sure that new issues had not crept in to a release which were missed by the normal testing process.

It was key that any tool or test environment should capture the inputs required to reproduce a particular failure, to aid investigation and fix. Anything that required complex scripting was seen as counterproductive.

AQuA's suggestion of automated tests which could run unattended for periods of time that fitted in with normal working practices was generally well received.

Stress and stability testing needed to have both automated and manual elements to be of value – for some concerns, such as battery usage, a human tester would take context into account in a way that would be difficult to frame as an automated test case.

Conclusion

> *"An enjoyable and useful day filled with relevant and open conversations. Our industry progresses most when we collaborate and days like that add oil to the machine"*
> **Charles Harley, Kotikan**

To follow up on the success of this event, and the value of the discussion to all participants, AQuA will seek to convene further expert round tables. Some of the key findings we will look to further explore include:

- Privacy – how can we help the regulators?
- Test Automation – a need to define common areas of testing that would lend themselves to automation. A need for a common glossary of terms that could be adopted and used by all.
- Fragmentation – not improving, and likely to be greater in future with the increasing presence of devices like smart watches and TVs.
- Frameworks and tools – often very capable, but practical experience underlined that for the best results there was still little alternative to developing apps natively.

> *"The BBC & AQuA roundtable session was an excellent opportunity to meet a wide range of people in all areas of the app development ecosystem, willing to share insights across a carefully selected set of topics on app quality. Thanks to AQuA and the BBC for putting the session together. I look forward to the next one"*
> **Breffni Horgan, Webresint**

You will find details of all our events on our website www.appqualityalliance.org and on Twitter @AppQuality.

If you have any suggestions for additional topics for debate around app quality, please get in touch.