Shake 'n' Send: Enabling feedback submission directly from mobile applications

Billy Landowski
willand@microsoft.com

Sira Rao
sirarao@microsoft.com

Abstract

As the mobile application market continues to grow, product teams need effective ways to gather user feedback about their mobile applications in order to improve software quality. Although extensive application testing is performed by dedicated product test teams, it is extremely valuable to enable beta users outside of the product team to try the application and provide feedback on their experience. This provides the product team with real input from real users that closely align with target customers.

To solve this problem for mobile applications, the Microsoft Lync test team developed a tool called *Shake 'n' Send* to enable feedback submission directly from the Microsoft Lync mobile applications. This tool allowed beta users to provide feedback by simply shaking the phone and providing information about their experience using the application. The simple and easy-to-use interface enabled users to quickly report on product bugs and other quality issues. It also allowed them to "send a smile" about what they liked.

For Microsoft Lync mobile application beta testing, *Shake 'n' Send* has proven to be very effective. Using this tool, beta users submitted several thousand feedback reports and that information helped the Microsoft Lync Mobile team increase software quality by fixing an additional 30% more bugs in the product.

Shake 'n' Send has proven to be a fun and efficient way for beta users to submit feedback directly from mobile applications. Because of this success, the feedback tool is currently being embraced and extended to support other mobile product teams within Microsoft.

Biography

Billy Landowski is a Software Development Engineer in Test at Microsoft. For the past two years, he has worked in the Microsoft Lync group testing mobile applications. His passion is to create new ways of testing to have business impact, and have fun. Billy has degrees in Computer Engineering and Mathematics from Washington State University.

Sira Rao is a Test lead at Microsoft. He has worked on the Unified Communications products at Microsoft for over 8 years. He is passionate about building high quality products that excite customers.

1 Introduction

As the mobile application market continues to grow, product teams are finding an increasing need for methods to elicit feedback from users about their experiences with mobile applications during the product development cycle in order to improve software quality. For desktop applications, obtaining user feedback can easily be done through the use of additional buttons, websites, or completely separate applications because of the abundance of screen real estate, multitasking, and reliable network connectivity. However, mobile devices have a number of unique features that require a different approach to developing feedback mechanisms, including: constraints imposed by the OS, unreliable network connectivity, and significantly reduced screen real estate.

In this paper, we discuss the challenges of soliciting and collecting feedback from mobile application users, previously implemented solutions and their drawbacks, and a new tool that the Microsoft Lync test team designed, developed, and deployed to overcome these challenges that we called *Shake 'n' Send*. We also present results from using *Shake 'n' Send* with Microsoft Lync mobile applications and discuss how it has improved the quality of the software.

The following terms are used throughout the paper:

- **Dogfooding**: This term refers to the practice whereby product team members and others in a company actively use and identify issues in a product prior to release. This practice has been in use within Microsoft since 1988.
- **Microsoft Lync**: The next generation of Microsoft's unified communications software that enables people to connect in new ways, anytime, anywhere.
- ***Shake 'n' Send***: The tool that is central to this paper, helping to enable sending feedback from directly within a mobile application. The tool name is coined after the actions to initiate ('*Shake*') and submit the feedback ('*Send*').

2 Background

In this section, we discuss general challenges that software developers must face while developing on mobile platforms and how these challenges affect the "send feedback" process for mobile application product teams.

2.1 General challenges of mobile applications

Mobile applications by their nature are encumbered by a variety of challenges that desktop applications may take for granted. One of the most obvious differences between the mobile and desktop platforms is the screen size, most of which are smaller than five inches. This limitation imposes new design challenges for product teams. With so little real estate, every pixel must be carefully planned for in the user interface design, making it very costly to change the design once it is finalized.

For security purposes, most mobile platforms enforce strict "sandboxing" rules which limit applications to their own data, and most applications cannot communicate with each other. Similarly, the traditional concept of "multitasking" on the desktop has different implications on the mobile platform. Since platform resources such as computing power and shared memory are scarcer than desktop operating systems, most mobile platforms only allow one application to actively run in the foreground at a time. Although some applications can register with the operating system to run in the background, there is no guarantee that the application will be given sufficient resources to operate.

Network connectivity can also pose many problems for mobile application developers. Access to the internet is not always guaranteed, and developers must accommodate for this by planning for more failure

scenarios and adding additional error checks. Furthermore, developers cannot rely on access to internal corporate networks because mobile devices are designed to work independently of them.

2.2 Challenges of applying desktop feedback mechanisms on mobile applications

Many different feedback mechanisms exist for traditional desktop applications, but as we discuss in this section, applying these methods on mobile applications is no easy task.

2.2.1 "Send Feedback" Button

One standard method for submitting feedback from desktop applications is to add a static "Send Feedback" button to the application's main user interface. For example, in Figure 1 below, the "Collect Logs" button on Microsoft Lync 2010 has been added to the released product for users to submit their feedback directly to Microsoft. This method is quite easy and inexpensive for desktop applications, but due to the reduced screen size on mobile devices, the real estate for such a feedback button is extremely expensive.

Figure 1 - Microsoft Lync 2010 screenshots of the send feedback process. When users press the "Collect Logs" button (left), they are presented with a new dialog to give their feedback (right).

2.2.2 Separate application

Another option for sending feedback on desktop applications that developers can use is implementing a completely separate application for sending feedback. By doing so, developers can use the same "Send Feedback" application for a variety of products that the company may produce, and the end user will have a similar and familiar "Send Feedback" experience for all of that company's products. One may be able to imagine this as a solution for the mobile platform; however, if the "Send Feedback" application needs any sort of diagnostics or additional data from the mobile application that is being critiqued, there is no way to gather this information due to the "sandboxing" nature of the mobile platform.

2.2.3 Link to website

Many desktop applications have simply added a link to a website where users can submit their feedback on the product. This is a nice option because it requires minimal changes to the desktop application, plus

it can rely on access to the internet. This option is much more complicated for mobile devices, however, because internet access may not be available for capturing the user's feedback. Perhaps more importantly, using a web link would require navigating away from the application in order to submit the feedback, disrupting the user's interaction with the application.

3 *Shake 'n' Send* Solution

So far, we have only discussed the limitations of mobile applications when compared to desktop applications, but there are also many unique capabilities that mobile platforms provide as well. Some examples include: GPS, built-in cameras and microphones, and accelerometers.

In order to solve the "Send Feedback" problem for mobile applications, we decided that instead of looking at just the limitations of the mobile platform, we should also look at the benefits and see if we could utilize any of these unique features. Hence, after experimenting with the built-in accelerometer, we decided to pursue a solution with physical gestures, and more specifically, the "shake" gesture.

3.1 Using the "shake" gesture

In order to solve the small screen real estate issue on mobile devices, we discovered that we could remove the dependency on additional user interface elements by using the mobile device's built-in accelerometer. Instead of tapping on a "Send Feedback" button, the user can simply shake their device to initiate the feedback process. With this idea, we developed a tool called *Shake 'n' Send* that is built directly into the mobile application. This enables the user to access the tool from anywhere within the mobile application without the need to navigate to a website or separate application.

When the user wants to send feedback, they simply shake their mobile device (like shaking a can of spray paint). When the shake gesture is recognized, a new user interface is presented to the user on top of the existing application where the user can type and send their comments to the product team directly from the mobile application. Screenshots of the *Shake 'n' Send* user interface can be seen in Figure 2 below. Additional information such as diagnostic data and traces as well as application screenshots can be captured at the time of the shake and sent to the product team with the user's comments. After the information is sent, the user is returned to the mobile application right where they left off, providing a complete closed-loop experience for sending feedback from the device.

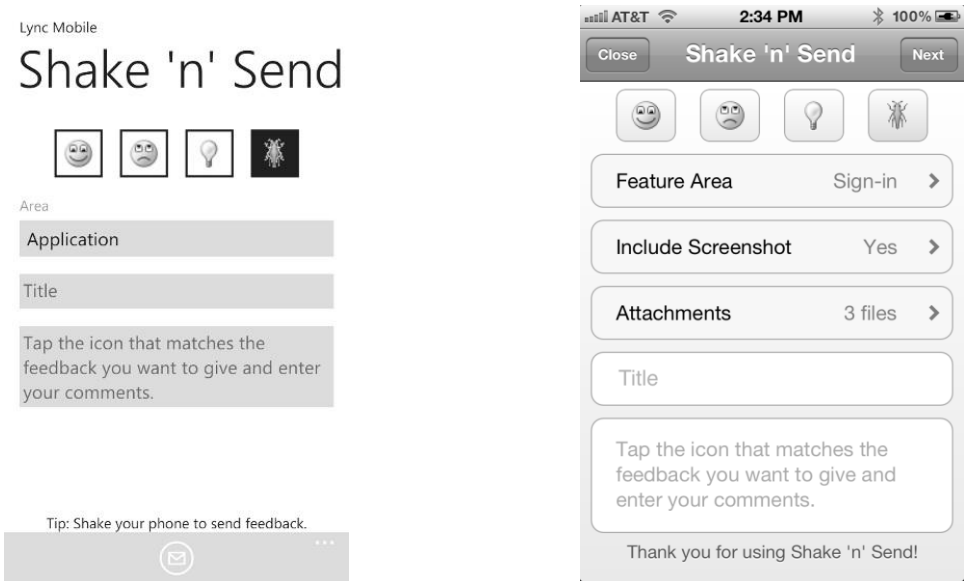

Figure 2 - Screenshots of "Shake 'n' Send" for Windows Phone 7 (left) and iPhone (right)

3.2 Using email to send feedback

The next step we took in building *Shake 'n' Send* was to determine how to send the user feedback to the product team. Since mobile applications must accommodate for unreliable network connectivity, methods are required for caching the feedback data and retrying the upload until it can be successfully sent. Depending on the size of the upload, the mobile device may take a significant amount of time and resources to successfully upload the feedback data, which may also be interrupted if the mobile user becomes impatient and terminates the application prematurely. This would require the feedback mechanism to be able to run in the background without any guarantee of sufficient resources from the operating system. Additionally, we would have to implement both the server and the client for the web service which, depending on the client platform, may not be a trivial process.

After looking at all of the challenges that a web service imposed, we took another look at the existing infrastructure of the mobile platform. We discovered that every major mobile operating system supported a native email client and that certain APIs were exposed to third party applications that could be used for sending email directly from the application. Hence, we decided to experiment with email as the transport mechanism for sending feedback. In doing so, we were able to save significant development costs by utilizing the existing email client on the device with very little additional code. The email infrastructure on the mobile platform also automatically handled all of the retry logic and background processing that a web service would have required us to implement. By using email, we could guarantee 100% reliability on delivery of the feedback data, a feat which would require significant investment by using a web service.

The only potential drawback with using email to send feedback is that it requires the mobile user to have set up an email account on the device. However, since most users use mobile devices for email purposes already, we felt that this was not a significant problem, and this has proven to be an accurate assumption in practice.

3.3 Backend service

For some product teams, simply receiving the feedback email may be sufficient. For a small amount of feedback, the team may be able to afford to manually process the data. But if the amount of feedback

increases, this manual process may become overwhelming. Hence, we decided to develop a backend service that automatically processed the feedback emails as they arrived.

First, in order to automatically process the feedback, all of the emails had to be sent to the same email account, so we created a new Microsoft Exchange email account and configured our mobile client applications to send the emails to this account. Next, we created an Exchange rule that would automatically filter the incoming emails into different Exchange folders based on the email's subject. This allowed us to filter the emails based on the client platform type (e.g. Windows Phone, iPhone, iPad, etc.) as well as to eliminate any additional noise or junk emails that may have been sent to the email account.

Once the email account was properly configured, we created a service using Microsoft's .NET framework and Exchange Web Service subscriptions to connect to the individual Exchange folders and to determine when a new feedback email arrives. Then, using the data that was sent within the email by the user, the backend service was able to act upon that data. If the feedback that was sent was marked as a bug, the backend service automatically filed a bug into our product team's bug database. After the bug was filed, a bug summary email was sent to the original sender as well as to each of the product team members to notify them of the new bug. Additionally, the feedback data was stored into a SQL database for easier reporting and analysis. Other feedback data such as positive comments and feature requests were also stored in the SQL database. Figure 3 below shows the entire end-to-end *Shake 'n' Send* process.

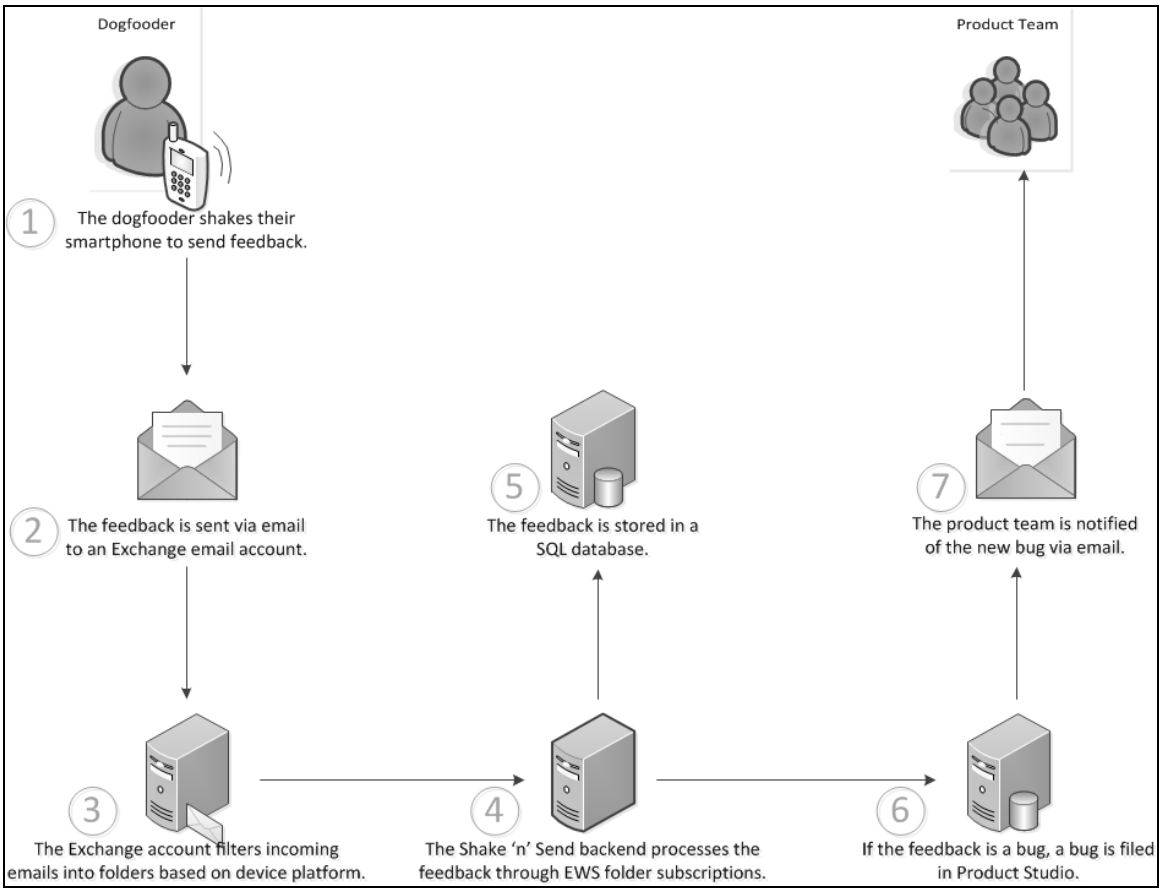

Figure 3 - The end-to-end "Shake 'n' Send" process

3.4 Additional features and benefits

Shake 'n' Send provides many benefits for sending feedback directly from mobile applications:

- **Automatic screenshot capture** – When the user shakes the device, a screenshot can be captured instantaneously, and the user can be given the option to include the screenshot with their feedback. If the user is submitting a bug, the product team can immediately use the screenshot to help understand the issue, thus saving time for both the end user and the product team.
- **Feature area detection** – The current state of the application can be detected at the time of the shake, and the feature area for which the user is sending feedback can be predicted and pre-populated in the *Shake 'n' Send* user interface, again saving time for the end user. Then, when the feedback is sent, the correct feature area owners can be notified directly instead of the entire product team.
- **Easy to integrate/remove** – Traditionally, the "Send Feedback" button approach requires tight integration between the feedback and product code. Placing a button on every view is very costly in both integration (and removal at time of release) and real estate on the small device screen. By using the shake gesture, there is no need for button real estate, and the *Shake 'n' Send* code is almost completely separated from the product code. This allows for much easier integration and removal of the *Shake 'n' Send* code within the product.
- **Familiarity** – *Shake 'n' Send* can be used by many different product teams for any mobile application. If a user has previously used *Shake 'n' Send* while dogfooding one of Microsoft's mobile applications, they will already by familiar with the feedback process and will be much more inclined to use it.
- **Fun to use** – The shake gesture is fun to use! This encourages users to send more feedback which product teams can use to improve the quality of their mobile application.

4 Results

During the beta timeframe of Microsoft Lync 2010 for Windows Phone, iPhone, and iPad product cycles, the Microsoft Lync test team deployed *Shake 'n' Send* to its internal customers to obtain feedback about the mobile applications under development. As a result of using *Shake 'n' Send*, non-product team members (other internal Microsoft users) submitted several thousand feedback reports of which over 10% were found to be unique and legitimate product code bugs that were eventually fixed before the products were released.

Unfortunately, we do not have any previous dogfooding data for mobile applications before the introduction of *Shake 'n' Send*, so we cannot analyze the impact that *Shake 'n' Send* has had on the overall feedback from dogfooders.

With several thousand feedback reports, we see that *Shake 'n' Send* was an effective way for dogfooders to send feedback to the product team. With this vast amount of feedback, the product team was able to utilize it to improve the quality of the software. Without *Shake 'n' Send*, we estimate that up to several hundred defects would not have been discovered prior to the customer release which could have drastically affected software quality. Instead, the product team was able to fix an additional 30% more bugs in the product before it shipped.

Shake 'n' Send has received a lot of positive feedback across a variety of different teams. The product team has commented that the feedback received by dogfooders has been extremely helpful in understanding the end user's perspective on the product, and this has helped the product team spend their resources to address the top issues that have been reported by dogfooders. Dogfooders themselves have also commented that they enjoyed how fun and easy *Shake 'n' Send* was to use. Because of this, many dogfooders were more inclined to send feedback more often which increased the overall amount of feedback that the product team was able to use to improve the quality of the product. The impact and effectiveness of *Shake 'n' Send* was also seen when the product team removed the tool before the product was released because the overall amount of feedback that the product team received from dogfooders dropped off significantly.

In the past year, *Shake 'n' Send* has continued to gain popularity, and other product teams at Microsoft have begun to use the tool internally for receiving feedback from dogfooders on their mobile applications.

5 Conclusions

As we have seen, gathering user feedback on mobile applications offered quite a challenge due to the nature of the mobile platform. By analyzing the benefits of mobile development as well as its drawbacks, we were able to develop a successful tool that enabled dogfooders to send feedback directly to the product team from within the mobile application. With the help of *Shake 'n' Send*, the Microsoft Lync product team was able to gather much more feedback on its mobile products than it could have otherwise. With the amount of feedback obtained, combined with the automated process of collecting, submitting and organizing defects, we were able to make significant improvements in product quality by fixing an additional 30% more bugs in the product. As the tool gets embraced by other Microsoft mobile product teams, we look forward to analyzing *Shake 'n' Send*'s abilities to facilitate improving software quality.

References

Wikipedia. "Eating your own dog food," http://en.wikipedia.org/wiki/Dogfooding (accessed August 2012)
Wikipedia. "Sandbox (computer security)," http://en.wikipedia.org/wiki/Sandbox_(computer_security) (accessed August 2012)
Microsoft. "Lync 2010," http://office.microsoft.com/en-us/lync (accessed August 2012)