Case Study of Customer Input For a Successful Product

Lynn Miller
Director of User Interface Development
Alias, Toronto, Canada
lmiller@alias.com

Abstract

Both agile development and User Centered Design stress collaboration between customers and product teams, but getting these methodologies to work well together is not easy. This paper describes one company's efforts to merge these processes by creating interconnected parallel design and development tracks. The benefits of this approach are demonstrated by showing how, when and why customer input was incorporated during the release of a successful software product.

1. Introduction

Studies in Human Computer Interaction (HCI), User Centered Design (UCD) [1] and User Experience Design (UED) [2] have found that accurate and frequent customer input is essential for a successful software product. Knowing who your customers are, what their environment is like, and what their needs are gives you the information required to plan and design a product. But to be successful, you cannot stop there. It is necessary to frequently contact your customers while making your plans and designs a reality.

At Alias, the Usability Engineering team has been gathering customer input for our products for many years. When product development started looking into adopting agile development, Usability Engineering was pleased to see that customer collaboration was part of the Agile Manifesto [3]. We could see that the merging of usability and agile would be mutually beneficial. Usability could incorporate our existing skills to better effect, and development could save time and effort while producing a better end product.

We found that our methods for collecting customer data did not need to change much, but the frequency and timing of collection changed considerably.

2. Background

Alias is the world's leading provider of 3D software for design, game creation, and graphical special effects for film and television. Our flagship products are highly specialized software like AutoStudio, which is used to design cars, and Maya, which is an animation package used in film and games. Alias software has been used in almost every film nominated by the Academy of Motion Picture Arts and Sciences in the categories of Best Visual Effects and Best Animated Feature Films, since their inception.

2.1 The usability team

Alias has a Usability Engineering team that is part of the product development group. The team consists of four interaction designers, two graphic designers, one intern developer, and one manager (me) who also works as an interaction designer. We have been gathering customer input at Alias for over 12 years.

The Usability Engineering team has a non-standard organization compared to other companies that we have talked to. There are two aspects that specifically affect how we interact with the development team.

The first relates to project assignment. Many companies assign their usability staff to all products that are being developed. We have chosen a different route. At Alias, each interaction designer is assigned to a single product at a time. Interaction designers start on a product at the market validation stage and then work as full-time members of the product team throughout the entire development process.

It also seems common for companies to assign the various UCD duties to separate groups, one group doing the market research and gathering user requirements, another doing interface design, and a third doing usability testing. Our interaction designers are responsible for all of these. We find that having the

same person work on all aspects of UCD over the life of a product means that there is no loss of information when data is transferred from one group to another.

2.2 The product

Three years ago Alias started developing a new product, called Alias® SketchBook™ Pro.

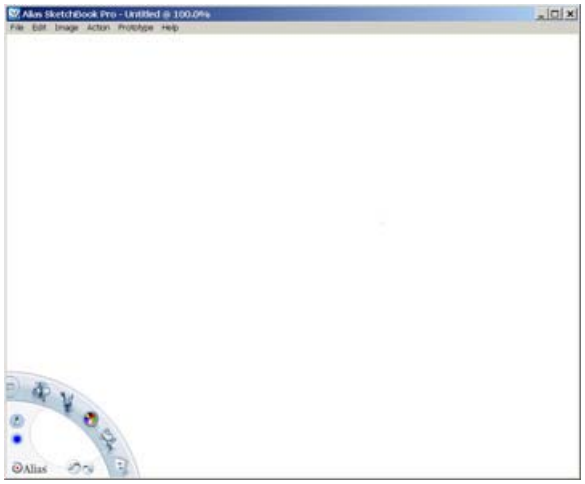

Figure 1. Alias SketchBook Pro

This product is used for 2D sketching and was designed to run on a Tablet PC, or a workstation with a Wacom® tablet. It was initially released during the Tablet PC launch in 2002 and was one of the first products that was designed specifically for the Tablet PC (that did not come with the operating system).

As shown in Figure 1, most of the window is a white canvas that the user sketches on. The arc in the corner is where the user accesses the functionality of the product. A press and hold with a stylus on the icons along the arc displays Marking Menus [4], as depicted in Figure 2.

Figure 2. Brush selection Marking Menu™

2.3 Agile at Alias

At the same time that SketchBook Pro was being envisioned, Alias was learning about agile development. We brought in Jim Highsmith to teach agile principles and methodologies. The product development group chose to adopt the Adaptive Software Development [5] process along with Scrum meetings [6] and many elements of Extreme Programming [7]. The SketchBook Pro team decided to develop their first release using agile development practices. The usability team members took this as an opportunity to modify our customer input process so it fit better with this new development model. Both the usability and development changes were highly successful, and subsequent releases of SketchBook Pro have been developed the same way.

Although agile approaches have been used from the start on SketchBook Pro, this paper will focus on the V2.0 release, which had a much smaller team than previous releases. The V2.0 team consisted of two developers, a technical team leader, a technical writer, a product manager, a quality assurance tester, two graphic designers (not full time), two interaction designers and a prototype developer (intern). I worked on the product as an interaction designer. The duration of the release effort was one year.

3. Avoiding customer input at all costs

Customer input is arguably unavoidable. Software teams that build products without talking to customers do not successfully avoid input. They just receive the input after the release of the product – in the form of bad reviews, lost sales, broken contracts and rewrites. These things cost companies (and governments) their reputation, revenue, and development time. A recent, and very expensive, example is the fiasco over the failed UkeU e-university project [8]. The House of Commons Select Committee cited that not doing market research and being technology-driven instead of user-driven were the main reasons for the failure [9].

Since customer input cannot be avoided, it has been our strategy to try to manage it. Managing customer input means making sure that we are getting the right type of input, at the right time, and from the right people. We have found that all three of these concepts are of equal importance for successful customer input on a product.

4. Who are these people?

Software is built for a particular audience, ranging from the more general (e.g., internet users) to the more specific (e.g., heart surgeons). Numerous HCI studies show that an indispensable aspect of getting relevant customer input is identifying your customers so that you know who (and who not) to talk to.

For SketchBook Pro, we identified our target audience as "creative professionals who do freehand sketching, and need high quality results." A few examples of potential SketchBook Pro users would be character designers (for games and film), industrial designers and fine artists.

4.1 But not these people…

The identified target audience needs to be as specific as possible. A typical mistake is specifying too broad an audience (e.g., Everyone!), which means there are no boundaries to help make user interface and product decisions.

By being specific (i.e., not just saying "artists" or "anyone who draws") we knew that we would get applicable data from the users mentioned previously but not from photo manipulators or CAD package users, even if they bought our software. This is because the latter customers do not sketch so they do not really want our product. Their input could change our software into a totally different product - one that they do want, but not the one that we want to create.

As an example, users of Adobe® Illustrator® are artists, so you would think that they match our target audience, but many of them cannot sketch. This is because Illustrator allows people to make good-looking images without freehand drawing skills. Illustrator users request Illustrator-like assisted drawing features for SketchBook Pro. Having identified our target audience specifically (not just "artists" but those who sketch) we are able to recognize that our product is not aimed at all Illustrator users. This means that we can concentrate on implementing features that truly affect our target audience. Although important for all products, this was particularly relevant for SketchBook Pro V2.0 since we had such a small development team, and one of our guiding principles for the product was *elegant simplicity*. We defined this to mean that features in the product are used by most users, most of the time. Features are not added just because they can be.

We have found that investing time to make sure that the entire team understands, and agrees on, who the target audience is, makes collecting and using customer input easier throughout the entire development process. It allowed us to make decisions on feature sets and design trajectories, and stay true to our vision.

5. Can't we just talk?

Effectively gathering customer input requires the use of an arsenal of different elicitation methods, most of which require training to collect untainted data.

The types of methods used, how they were conducted, and the skills required to facilitate them did not need to change for customer collaboration on an agile development project – only the timing of these activities changed. This meant that our interaction designers could capitalize on their years of experience.

In the next section I will briefly discuss some of the most commonly used customer input methods, and state the pros and cons for each since the methods are not interchangeable. I have included some useful books in the Resources section at the end of the paper to provide more information on these.

5.1 Contextual inquiry

Contextual inquiry is a structured field research technique used for ethnographic study. It is investigative in nature and focuses on the context of the user's work and environment, rather than (for example) on specific features of a product.

Contextual inquiry is used to discover what the user's work environment is like, how users do their work now, what their work goals are, what problems they are trying to solve, what steps and data are needed to solve their problems, what data is produced, and how their work is evaluated.

Contextual inquiry cannot determine how well a product will work in an environment or how easy to use or easy to learn a product will be.

5.2 Interviews

Interviews are structured one-on-one question and answer sessions. They are investigative in nature so the intent is to gain understanding in areas that are unclear.

The majority of interview questions are "open" and conversational, instead of "closed" and quantitative. For example, an interviewer might ask, "What does our competitor do better than we do?" to initiate a discussion, but not "Please rank on a scale of 1 to 5 which of these competitor's features are better than ours."

Interviews are good for learning the customer's pet peeves, the problems they are running into, their likes and dislikes about the software, and what they want to see in future versions.

Interviews cannot determine if software is easy to learn, or easy to use. Interviews will not help identify new market opportunities.

5.3 Usability tests

Usability tests are used to evaluate a product design by watching the intended users of the product try it (or a prototype of it) for its proposed use, and seeing what problems are found.

Usability tests are good for discovering issues with learning, discoverability, error rates, and speed of use. They also uncover issues with incorrect or omitted feedback. Usability tests can uncover missing features that are needed to complete a workflow.

Usability tests cannot discover whether a product will fit into the users' work environment since they are normally not conducted at the users' work place, on their hardware and using their files. They cannot verify that a product is solving the right problems for specific users, or if people will actually buy it.

5.4 Focus groups

A focus group is a moderated, exploratory discussion with a prepared focus. The participants are users or potential users.

Focus groups are good for getting user's opinions, goals, priorities, and seeing how these compare to others in the group.

Focus groups cannot determine if users would actually use proposed new software, or what features should be put into a release. They also cannot be used to determine whether software is learnable or usable.

5.5 Surveys

Surveys are strict sets of questions that are delivered to a large number of people in order to gather quantitative data.

Surveys are good for getting simple factual data, priorities, and confirmation of information that was gathered in an exploratory session.

Surveys cannot give an understanding of the “why” behind the facts that are gathered since there is no ability to have a discussion with the participants. Surveys also cannot tell if people will buy a product.

5.6 Beta tests

Beta tests are when almost-complete software is sent to customers, and they are asked to report problems.

Beta tests are good at finding bugs. They are also good at determining if the software actually solves the customer's problems and if it works in their specific environment.

Beta tests cannot convey if the software is easy to learn or easy to use since beta customers will not typically give this feedback. If the software is hard to learn or use, Beta customers will blame themselves and will not report the problem because they do not want to appear stupid.

5.7 Demos

A demo is a canned demonstration of new software (or new features) shown to your customers to get their opinions.

Demos are good for learning what users think about a proposed feature, if a feature would make them more likely to buy, and if they think you are going in the right direction to solve their problems (although they cannot tell you if you are actually going in the right direction).

Demos cannot determine if a feature will work in a real production environment, if it is easy to use and learn, or how much people will like the feature after they start using it for real.

6. Timing is everything

For the Usability Engineering team, the big change brought on by agile development was when and how often the interaction designers collected customer input. We had been gathering customer input for many years, so identifying our customers and knowing which methods to use was not new to us, but the timing and incorporation of that data into our existing development method had never been ideal.

Previous product development used a mostly-waterfall method of development. We were supposed to do the typical waterfall stages of requirement gathering and analysis, design, implementation, testing, and deployment. This never really happened. As a company that has to respond quickly to changing market forces and has to maximize the output from the development group, we often ended up going from requirement gathering straight to implementation. Basically, the requirement gathering would be pushed as late as possible to get up-to-the-moment data, so by

the time the "plan" was ready the developers were idle and needed to start something. This worked for getting the best possible initial plan, but meant that immediately after the plan was released the developers would begin coding with no time for proper customer input or user interface designs. Features would be implemented based on guessing what the user would do or would want, with no verification with actual users until after implementation.

Worse still, on some products most large features would start being coded simultaneously since each feature was assigned to a separate developer. This made it impossible for the interaction designers assigned to the product to simultaneously investigate and design that many features at once.

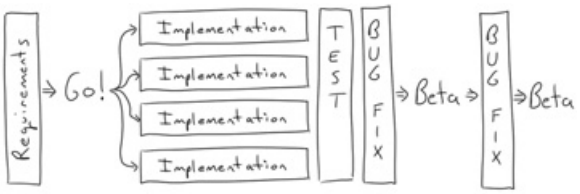

Figure 3. Non-agile development

As interaction designers in this environment, we had various strategies to circumvent this less-than-optimal process and bring customer input and real interface designs into the mix. One technique would be to investigate and design features ahead of time so they would be "ready" when implementation started. Since many of the features would not make it onto the official plan, this resulted in wasted work.

Adopting agile development gave us the opportunity to eliminate this waste. Since one of the Agile Principles is early and continuous delivery of valuable software [10], the development effort had to change from having many developers working simultaneously on separate features, to having the whole team work together to get a smaller set of features implemented in a shorter time so that they could be shown to customers.

This made interface design and customer input an integral part of the development process, as it should be with agile. The SketchBook Pro team organized implementation and design as two equal and highly interrelated tracks, as shown in Figure 4.

This double-track method meant that we got richer customer input and more timely feedback, which resulted in fewer scheduling surprises and a much better piece of software.

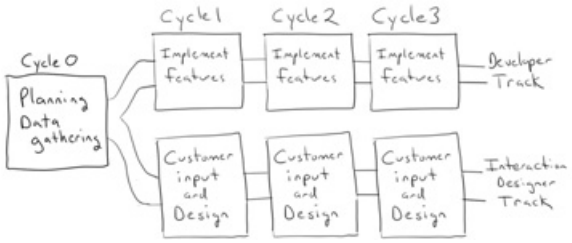

Figure 4. Agile development

7. Customer input in cycle 0

Cycle 0 is the "speculate" phase of the Adaptive Software Development [5] method that we were using. During this cycle the interaction designers gathered customer input to determine the capabilities that would be added and the priority of each.

Potential customers were heavily downloading SketchBook Pro V1.0. The software had a trial period with full functionality for 15 non-consecutive days. This gave users the opportunity to evaluate the software before purchasing. SketchBook Pro was getting excellent product reviews and positive responses at tradeshows, through email, and in our discussion forums. However, only a small number of the people who downloaded and tried the software were actually purchasing the product. To address this, we set the goal for the V2.0 release to be "remove the top obstacles that prevent people who download the product from purchasing it." This goal was established jointly by the product manager, development manager and myself. In order to meet this goal we needed user input to understand what was wrong with the V1.0 release.

7.1 Identifying the critical capabilities

Our first source of input was the usability test information from the last release. Usability tests provide both information that can be used immediately to redefine the feature that we are working on, and information that does not necessarily fit with the goals of the current release but is still worthwhile. It is the latter type that we drew upon during cycle 0.

For example, usability testing had shown that users were having a lot of trouble trying to resize brushes. Fixing this problem did not fit in with the goal of the previous release, but the problem was large enough that it was likely affecting people's purchasing decisions.

Along with the usability test data, we gathered customer input from our discussion forums, and visited

customers to conduct interviews and collect contextual inquiry data.

Amalgamating all of this information gave us an initial list of about 20 things that most likely were causing people not to purchase (contrasted to the total number of feature requests received at this point, which was around 100).

Two developers could not do all 20 features in a single release so we needed to prioritize them. The customer input at this point was investigative and did not have enough data points for us to assign priorities. To get an accurate ranking of the data we needed to contact a larger group of people.

7.2 Prioritizing

The interaction designers, working with the product manager, created and distributed a survey to the people who had downloaded our software but had not purchased it. The survey elicited customer input on the 20 items that we had identified to give us their relative priorities. We also gathered information on what they thought of the product. Reassuringly, even those people who had not purchased still said they loved the product. They just did not think it had all of the features that it needed. This further validated our release goal.

The survey ranking data allowed us to cut the list of 20 items down to 5 that the V2.0 release absolutely had to address to get people to purchase, plus a priority-ranked list of the rest of the features. This information was brought into the initial iteration planning meeting, which had representatives from product management (focusing on company objectives), development (focusing on feasibility and timeframes), and usability (representing the users). All decisions were made by a combination of this group.

In this initial iteration planning meeting the iteration objectives were decided, and the product team roughed out which features would go into which iteration (cycles were each about two weeks long). We used the priority data to make sure that we worked on the most important items first, so if something had to drop, it would be one of the less important features.

8. Cycle 1:

8.1 Getting the time to design

The first cycle was a little different than subsequent cycles because the user interface designers had just been given the feature list and had not had any design time, and yet the development team needed to start coding. This was similar to the problems we had faced before with our old development method. With agile, we were able to solve this problem by filling cycle 1 with features that had a high development cost and a low design cost. For example, one of our new (and crucial) features was the ability to save files in the Adobe® Photoshop® format. This was quite tricky for the developers, but the design was "add a Photoshop line to the Save As dialog". Having a few of these gave the interface designers some breathing room.

8.2 Finding the right users

For customer input on SketchBook Pro V2.0 we worked with three different types of people.

We first sought out Alias employees that matched our target audience. Alias hires users of our software to work as product specialists and application engineers and also to work in quality assurance and support. We were able to find character designers, industrial designers, and other sketchers on staff. These were always the first people that we would talk to, or test with, since they were readily available. However, it was not sufficient just to contact internal people since they knew too much about the company and our technology so did not completely match the people who would be buying our product.

Our second group of people was external target customers who had not seen the software before. One method we used to recruit people in this group was to post physical notices at key schools, and virtual notices on professional discussion forums, inviting people to help us. We would then screen the respondents for those that matched the characteristics that we were looking for. (For example, we would have them send us some of their sketches to ensure they actually could draw freehand.) This group tended to have more students than professionals so we brought them in for usability tests did not use them for contextual inquiry investigations.

The third group was the hardest to recruit, but the most important. They were professionals that agreed to work with us over the course of the release as Design Partners. We would visit them at their work places to gather contextual data and have them participate in a series of usability tests.

To find this last group of people we used the survey data (described in section 7.2). We had asked the respondents if we could contact them for more information and had them provide an email address. For those that said yes, we had to find the ones that were within driving distance of Toronto as we planned to visit them often. Since we had not required the

survey respondents to identify themselves or their location, I traced their IP addresses and pulled the ones that were originating in Ontario and Quebec. Then the other interaction designer on the product emailed them explaining what we wanted to do. She then had to qualify the interested respondents to make sure that they matched our target audience and that they had the time to work with us. Although time consuming, this gave us a small group who provided us with vital data.

8.3 Customer verification of design

In cycle 1 the interface designers worked on features that would be implemented in cycle 2. We would first design the interface and then build a testable prototype. Sometimes these were low-fidelity (paper) prototypes, and sometimes the usability intern would create high-fidelity (coded) prototypes. The latter was used when the design verification relied on real-time interaction. For example, we used paper prototypes for the custom brush dialog, and coded prototypes for interactive brush resizing.

Problems with the designs found during the usability tests were corrected, fixed in the prototypes, and retested. (For example, issues with discoverability, ease of learning, ease of use, and feature completeness.) This cycle continued until the designs had achieved their design goals. We used all three types of customers for the usability tests – first testing and iterating on the design with internal users, then the student group, and finally with our Design Partners.

Because of this customer input at the prototype stage, the design was already known to be correct and complete when development began in the next cycle.

This gave us several advantages. First, we knew the design incorporated all of the necessary functionality that the user needed. This meant that we did not get surprised later on by a crucial missing piece that had to be added to the schedule. Secondly, we knew that the design had achieved its design goals and users could do what we wanted them to be able to do. This allowed us to be able to safely say "no" to incremental feature requests because we understood what was meat and what was gravy.

8.4 Understanding customer needs

Usability test results for design validation were not the only type of customer input during this cycle. The designers also had to gather the information that they would need for the features that would be designed in cycle 2 and implemented in cycle 3. This input came from contextual inquiries at our Design Partner sites.

It is important to note that although this group was called Design Partners, they did not participate in the actual designing of the interface. They provided the upfront information that the interaction designers needed to develop a design, and they verified that the design was working from the prototypes, but they did not actively take part in designing the interface. Having developers, business representatives and users work together to design a solution is covered in a method called Participatory Design. The interaction designers at Alias do not use this method because we find it does not work well with our types of users (creative), interfaces (innovative) and products (shrink-wrapped).

Of course, even if we weren't looking for designs, often our Design Partners would suggest interface solutions to us. The problem was that these suggestions were often based on standard practices from other applications and would not work with SketchBook Pro, which was designed for use with just a stylus (no keyboard, no mouse buttons) and so had to have an innovative interface. For example, customers often suggested adding hotkey support to fix ease-of-access problems, but hotkeys do not work without a keyboard.

Because of this, it was important for us to focus on trying to understand how the customers worked and what their problems were instead of listening to the specific solutions that they proposed. By making sure that we thoroughly understood the problem, we could devise a solution that fit our application better.

8.5 Parallel tracks in cycle 1

The parallel track organization for cycle 1 is shown in Figure 5. The developers worked on features with high development costs and little user interface, while the interaction designers investigated, created and verified designs for the next cycles.

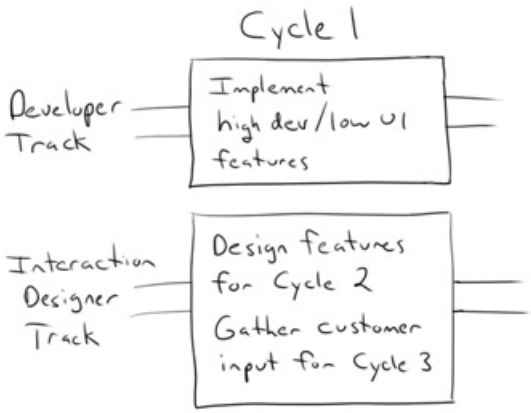

Figure 5. Cycle 1 – two tracks

Developers and interaction designers attended the scrum meeting each day to keep everyone apprised of what was happening in the two groups. When required, after the scrum, the interaction designers would present design concepts to the development group for feedback and feasibility. We would also present usability test results so everyone would know how well the designs were working and could suggest solutions to interface problems.

There were three large wins for the interaction designers with this parallel-track process over our old process. The first was that since we were always designing for the next iteration (or two at times) we did not waste time creating designs that were not used. The second was that we could do both usability testing of features and contextual inquiry for design on the same customer trips, which again saved us time. The final benefit is that we were always getting timely feedback, so if there was a sudden change in the market (like new competing software being released – which happened) we received input on it right away and could act accordingly.

There were two big wins for the developers as well. The first was they were able to maximize coding time since they didn't have to wait for us to complete paper prototypes and usability tests. This was very important for SketchBook Pro V2.0 because we only had two developers. The second was that they didn't waste their efforts coding the various design concepts for the innovative interface pieces. During the research stage for these designs it was common to create multiple diverse prototypes while trying out ideas, and all but one of these would be thrown out.

9. Customer input thereafter

Customer input for cycle 2 and the rest of the cycles were similar to cycle 1 in that the interaction designers would be conducting usability tests to get verification of our prototype designs, and contextual inquiries to get an understanding of the problems being solved by the features that we would be working on next.

Additionally, after cycle 1 we had to get customer acceptance of the features that were implemented in the previous cycle. The prototype usability test data proved that the original designs were discoverable, easy to learn, easy to use and complete. However, most designs had to be tweaked slightly because of technical implementation problems, and the usability tests did not show us how the features would interact with one another. (The prototypes were often of single features in isolation.) We needed to do another round of usability tests, but this time on the actual production software.

Unlike traditional after-the-fact usability tests, problems found in these tests tended to be small and were simply logged as bugs to be fixed in the next cycle.

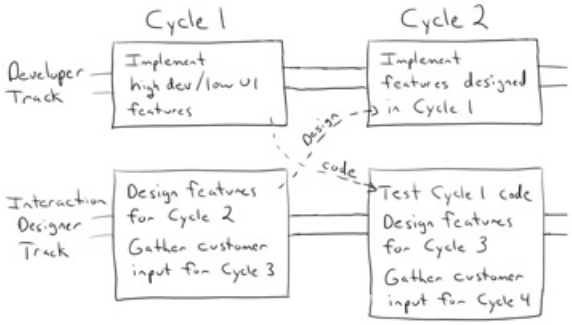

Figure 6. Cycle 2

Figure 6 shows the two tracks for cycles 1 and 2 and the deliverables passed between the groups.

Designs were not just "thrown over the wall" to the developers. Through the daily scrums and interface presentations, the developers had followed the design's progression throughout the last cycle. Once the design was complete there was also a meeting with the developers and interaction designers where the design would be broken up into feature cards and feature points would be assigned. And finally, the interaction designers would work daily with the developers once implementation started to answer questions and solve problems that arose from issues with implementation.

9.1 Customer input after releasing Beta cuts

The last piece of customer input that was needed was whether the features would work in the user's production environment. All of our input so far was on our own hardware with our own test files.

To gather this information, the product team set up a beta testing group of 55 people. After key cycles we distributed software to this group so they could try real work with it. We set up a discussion forum for their feedback.

Customer input from beta tests is one of the most misunderstood types there is, so it was important that we knew how to "read" the input that we received.

The biggest problem is that beta testers try to design user interfaces. When a beta tester proposed a design, we would look for the reasons behind what they were saying so we could understand what they were really

asking for. Very often they were trying to solve a problem that we already had a design for. Even if we did not have a design yet, and the request was valid for the product and the release, we would normally still design our own solution to it. As interface designers we have a lot of experience with what makes a good design and what fits in with our design goals for the product – the beta testers do not normally have this experience.

For example, we constantly had customers asking us to put functionality on the barrel-button (that is on the side of the stylus). Examples include brush resize, zooming, hiding windows, and hotkey input, to name a few. We have watched countless people using a stylus to draw and we knew that it was a bad idea. When people are drawing they hold the stylus like a pen and move it between their fingers, which results in accidental barrel button pushes. If we took the design solution literally from our customers, we would have spent precious time implementing a customizable way of adding functionality to the barrel button, only to have most of our customers turn it off and then ask for a different solution. As it was, we came up with an innovative design solution to the brush resize problem (which was the most requested barrel button functionality) that our customers loved.

Overall, the beta testers gave us a lot of valuable feedback. The biggest benefits were the bugs and performance issues that they found. Beta testers use larger canvases and more layers than we normally can test with. They also have different hardware. They found bugs that would normally have been found only after shipping and helped us track down and verify solutions.

The beta testers also gave us feedback on whether the features we added were desired (they were happy to see them) and whether the features we were about to add were the correct ones (what they asked for next was already in the plan). This validated the customer input that we gathered in cycle 0.

10. Reflection

The Usability Engineering team at Alias has been gathering customer input for many years, but never as effectively as when we work with an agile development team.

For SketchBook Pro, we were able to maximize the quantity and impact of customer input by having the interaction designers work in a parallel and highly connected track alongside of the developers. Daily interaction between the developers and interaction designers was essential to the success of this process.

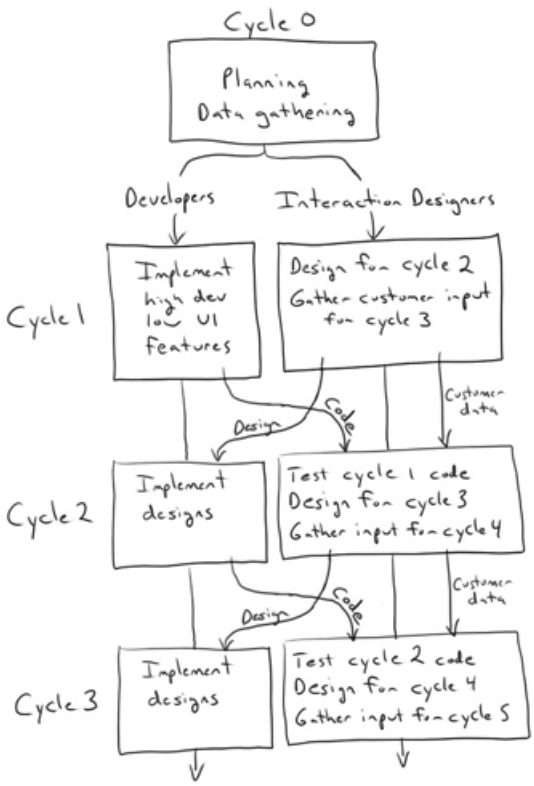

Figure 7. Dual tracks

The two-track organization shown in Figure 7 is what we aimed for, although in reality it was a little more complex. Some designs needed longer than a single cycle to complete. For example, one particularly troublesome feature took us over 5 cycles before the design passed all of its goals. The general rule still held true even for those cases where we had to design more than one cycle in advance – the design is timed so that it is ready just as the cycle that it is needed in starts.

Not all companies organize their usability resources in ways that allow this structure, but if they did they would likely find their usability and development teams could work together more closely, save design and development time and effort, and produce a better product for the end-user.

11. Acknowledgements

I would like to thank the entire SketchBook Pro V2.0 team for their hard work and dedication to the product. Thanks also to the Research team, our co-op students and the other members of the Usability Engineering team for helping out with design, prototyping and testing. Special thanks go to Larry

Philps for supporting the entire Usability Engineering group, and to Jim Hewitt for supporting me.

12. References

[1] Norman, D. A., and S. W. Draper, editors, *User-Centered Design*, Hillsdale, N.J., 1986.

[2] Garrett, Jesse James, *The Elements of User Experience: User-Centered Design for the Web*, New Riders Press, 2002.

[3] http://www.agilemanifesto.org/

[4] Kurtenbach, G., and W. Buxton, "User Learning and Performance with Marking Menus", Conference Companion on Human Factors in Computing Systems, April 1999.

[5] Highsmith, James A. III, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House Publishing Co., Inc., 1999.

[6] Schwaber, Ken, and Mike Beedle, *Agile Software Development with SCRUM*, Prentice Hall, 2001.

[7] Beck, Kent, *Extreme Programming Explained: Embrace Change*, Addison-Wesley Professional, 1999.

[8] Samuels, Mark, “Elearning disaster ignored users' needs”, *Computing*, vnunet.com, March 10, 2005. http://www.vnunet.com/comment/1161820

[9] "e-University failure: MP Committee issues damming report", PublicTechnology.net, March 4, 2005 http://www.publictechnology.net/modules.php?op=modload&name=News&file=article&sid=2545

[10] http://www.agilemanifesto.org/principles.html

14. Resources

Alreck, Pamela L., and Robert B. Settle, *The Survey Research Handbook*, McGraw-Hill, 1994, ISBN: 0786303581

Beyer, Hugh, and Karen Holtzblatt, *Contextual Design: Defining Customer-Centered Systems*, Morgan Kaufmann, 1997, ISBN: 1558604111

Dumas, Joseph F., and Janice C. Redish, *A Practical Guide to Usability Testing*, Ablex Pub, 1993, ISBN: 0893919918

Hackos, Joanne, and Janice C. Redish, *User and Task Analysis for Interface Design*, John Wiley & Sons Canada, Ltd., 1994, ISBN: 0471178314

Greenbaum, Thomas L., *Moderating Focus Groups: A Practical Guide for Group Facilitation*, Sage Publications, Inc., 2000, ISBN: 0761920447

Kuniavsky, Mike, *Observing the User Experience*, Morgan Kaufmann, 2003, ISBN: 1558609237

Wixon, Dennis, and Judith Ramey, editors, *Field Methods Casebook for Software Design*, John Wiley & Sons Canada, Ltd., 1996, ISBN: 0471149675

15. Legal