

# Less, Never More: Launching a Product with Critical Features and Nothing More

Scott A. Gatz and Gabrielle Benefield

Yahoo!

scottg@yahoo-inc.com, gabby@yahoo-inc.com

## Abstract

*Yahoo's Advanced Products team is a small team that develops and incubates new product ideas before formally launching them and integrating them with the Yahoo! network. We conceived, built and launched Mixd, a social mobile product experiment, in a 9 month timeframe. We were extremely focused on a specific problem of our target audience and remained true to that problem throughout. We focused ourselves and selected our product features with our target audience in mind. We took every opportunity to spend time with our users and integrated their feedback through the entire design process. Our process was Agile and enabled us to move quickly and change course based on the feedback we received. We experienced first hand how a focus on our users and how removing complexity and assumptions in our product allowed people to use our product in unintended ways. Those unanticipated uses can end up being more interesting than your original idea.*

## 1. Our Approach

Yahoo!'s Advanced Products team is a small internal team that develops and incubates new product ideas before formally launching them into the world and integrating them with the Yahoo! network. All of Yahoo!'s products go through a very rigorous test cycle before they are announced publicly and promoted to a mass audience.

With this product, we set out to explore college students' usage of mobile phones to communicate and to organize ad hoc events. The aim of the project was to get to market as quickly as possible, while still providing a compelling user experience, and iterate on the product quickly. By launching a simple product quickly, we were able to see if the idea was worth commercializing. During the pilot program, we gained valuable insight about how youth communities socialize via mobile phones. We plan to leverage our learnings to incorporate group text messaging and

multimedia sharing features into future mobile products from Yahoo!

The development team was distributed between the US and India, and the team chose to use an Agile process. Our development cycle was based on iterations ("sprints") where we sought to deliver a complete feature, a chunk of shippable code, or a component in each iteration. At the end of each iteration, and sometimes during, we pushed the code to an internal server where we could use the product. As the product progressed, we let other people throughout the company use it as well.

During this process, we learned a lot about the market, we learned about how to stay focused and move quickly, and gained a lot from using Agile as the communication and development method to drive this type of project. From concept to launch, the product took nine months.

## 1. What is Mixd?

Mixd (<http://mixd.yahoo.com>) was developed as a social mobile product experiment by Yahoo! Mixd was a group mobile messaging and media sharing tool for people who want to organize and remember get-togethers. It was launched in December 2006 (and ended February 2007) to help communities of 18-25 year olds connect both online and offline, share ideas and information, and socialize with each other using their personal cell phones.

With Mixd, you could start a group from wherever you are at the moment and easily send text messages to the group right when the urge hits you to make plans. Mixd created a "reply to all" for SMS making it easy to plan a last minute get together amongst a group of people. Mixd also created an instant group web site with message archives and photo/video sharing from your night out. Mixd provided a solution to users' frustration at getting pictures off the phone and into the hands of their friends and family. The web experience was optional, but made your get-together more memorable.

### 3. A Clear User Need

From the very beginning of the project, we felt that it was extremely important to have a single, clear user need that our product filled.

We started with this simple user problem: “It is 5pm on a Friday night and I want to hang out with my friends, what do I do?”

We filtered most product decisions through this user need. We prioritized features by asking if the feature was absolutely necessary to help the user accomplish their goal of hanging out with friends, and brutally cut features that didn’t address the problem.

We kept this as the driving problem we needed to solve, and we never lost track of this basic requirement (and still haven’t). This was critical to help us meet our goals and stay focused.

### 4. Talking to the Target Audience

During the initial concept phase we conceived a product that was a web-based invitation application (similar to Evite: <http://www.evite.com/>). We thought this would meet the needs of someone planning a Friday night out. We foresaw the users inviting friends, specifying details of their night out and sending out notices to those friends. The idea required email and web access as the primary method of access.

In parallel to the concept phase, we sought to learn as much as we could about the potential users of such a product. We searched the internet for papers, research, and anecdotes relating to how our target audience planned ad-hoc get-togethers. We also spent face-to-face time with potential users of the product. These meetings were casual, but significantly impacted our understanding of how to fill the need.

As the concept grew and morphed, we increased our understanding of how 18-25 year olds used mobile phones. Based on our research, we learned that they consider their cell phones as “a part of their body” – they send and receive texts frequently per day and many have text services in their phone service plans.

Thinking it through, socially active college students were not likely to be on their computers at 5pm on a Friday night. So, we needed another way to reach them and connect them to each other. This led us to make a primarily mobile based service.

We found that an Agile process supported the discovery of new information and allowed us to adjust quickly to take advantage of this information.

### 5. Focus and Less is Better

From here, it was clear that we needed a product focused on lightweight mobile groups: group text messaging (also known as Short Messaging Service or SMS messages). Coding as quickly as possible, we focused only on basic functionality, no UI, no professional looking design; we needed to “feel” the product working, not see it. In less than 5 weeks we had a fully working prototype of the basic functions.

This enabled us to use our own product and get a feel for the way our customers might use it. We were able to see the power in the simplicity of the product as is, but we were tempted to add numerous features to the product’s feature list.

Through the building process we did a fairly good job at keeping the simplicity of the product. When we did give into temptation and added a new feature, we asked ourselves if it solve the user problem in an easier or better way and we asked if adding the feature would make the product harder to understand or use on a daily basis. Using the product daily made us much more sensitive to these issues.

With this in mind, we realized that assumptions and requirements that we had specified early on complicated the product unnecessarily. One requirement—a user having to specify the location and time of the event—made the product unnecessarily complex and was of secondary importance to the core problem: hanging out with my friends on a Friday night. The extra fields would enable additional features like maps and a calendar, but they weren’t critical. A user might “meet up with the crew” easily by texting “how bout Liquid Lounge at 10?” The added “benefit” of the features would be offset by the difficulty of setting up one of these groups.

In the middle of the development process the entire team came to the same conclusion and decided to not complete that part of the functionality and remove it from the product. Agile enabled this course-correction seamlessly. While losing the feature was a major conceptual shift, it was an easy shift to make, and the entire team was on board.

In the middle of the busiest part of development phase, we debated if the users’ event groups should be private or public (like MySpace) to guide adoption. This was a critical juncture for the product. Making them public would inspire usage and provide a communal atmosphere. We wanted it to be easy to discover all the benefits of the product before a user needed to sign up.

We went back to the core problem and asked how making the groups public would help with “it’s 5pm on a Friday night and I want to hang out with my friends”. Ultimately, we determined that making groups public added development complication and

usability complication which didn't further the initial goal. If we had chosen that path, our development would have likely stretched for many more months.

After launch, we received an immense amount of product feedback from our pilot users and almost all of the comments showed us we made the right choice. Most users couldn't conceive of the groups as public, asking, "you want to show my texts to strangers?" We also discovered far more important features and usability improvements we needed to make. By staying focused (and adding fewer features) we were able to get to market quicker and get that feedback quicker so we could iterate with a live product instead of in our heads.

## 6. Remote Team, Stay Focused

We had the added complexity of working with a distributed team with the engineers based in India and the Product and Design team based in the US. We kicked off the project from our two separate locations, which in retrospect was a really bad idea. We underestimated the cultural differences and the importance of getting everyone on the same page as early as possible. The product was targeted at American teenagers and the core team in India was slow to warm up to the concept. It was difficult for them to understand why this was an important product for the market. The US team had to go to extra effort to explain and share that this was incredibly important to the company and the kids who were going to use it. We would have been much more effective in the start-up had we been together and it would have surfaced issues earlier that were harder to deal with when they appeared later in the process.

We ended up taking the US team to India later in the project, and things progressed quickly after that. Once everyone is marching to the same drum, everyone makes better decisions. It was amazing how many communication issues got cleared up over a "chili margarita" (imagine a margarita with incredibly spicy peppers ground up inside).

Once we were up and running, development moved very quickly thanks to frequent, daily communication between the teams and the discipline that Agile provided us. The best part for us was the product backlog itself. A clear to-do list at the beginning of the sprint so that everyone knew what to accomplish in those weeks enabled us to manage the team effectively without us being in the same location. While this worked well, we found that time difference (12.5 hours) and the distance meant that work was slower and less was completed than had we been in the same location. The key to our ultimate success was brutally

culling features to help us reach our dates. This constraint also benefited the product, reducing complexity for our users and forced us to remain focused on the simple problem "its 5pm on a Friday night..."

## 6. Talk to the Audience Again and Again

At each iteration, we aimed to have a new chunk of functionality working without breaking what already worked. For most of the iterations, we pushed the code to an alpha server where we (and other internal users) could use the service on a regular basis. This allowed us to try out new features as they were completed and iterate quickly on things that worked in concept, but not in reality.

It was in one of these early stages that we realized that our product generated too many messages to send to the mobile phone. It was working as we had designed, but we understood quickly the need for a design change. We quickly came up with new designs and changed the message flow in a later iteration.

At this stage, we also casually pulled together a few groups of potential users to show them the product and design explorations of what the final product would look like. With a working prototype, we were able to get them to actually use the product and walk through the flows with us. These sessions gave us an immense amount of information, focused us even more and helped us answer some nagging questions. Afterwards, we often referred to some of the insights we learned in those sessions. The sessions were informal (no fancy focus groups), just a few of us sitting with an equal number of users in a casual setting. We've done this in other settings over the years and find it an important part of the process (both for garnering specific feedback *and* for getting a general feel for the way some people approach the problem you are trying to solve).

Having frequent builds to use on a daily basis was both a blessing and a curse. As the product felt stable, we sought internal users to use the product. We sought people at Yahoo! who were either in our demographic or who were familiar with the problem and the technology space. We wanted to get as much feedback as possible as early as possible. Because internal people were using the product, a data integrity bug or the server crash caused many people to take notice. As people began to rely on the product, we had to apply extra maintenance. This was beneficial and on future projects we would still allow internal people to use the product early, but we would account for time in the schedule for the additional maintenance. The feedback

from the people using the product encouraged us and shaped our decisions.

## 7. Heading for the Home Stretch

We set an internal launch deadline to ensure launch during the college semester and each iteration got us closer and closer to a complete product on time. As the date approached, we were even more brutal about cutting features.

One of the key features we initially found most important was a way to get updates via email, instead of mobile phone. We felt this was a terrific add-on for people who didn't want to get updates or converse on their mobile phone. This feature required a significant amount of effort, but could be completed in time for our launch. We once again bought up the core problem statement and realized that the feature diluted the key focus of the product and that it added extra UI complexity where we didn't need it. We cut the feature and instead focused on strengthening the other features and getting more wireless carriers to support the service. By the time we launched, we were able to get most major carriers to support the service. Users could text to 445566 on Cingular, Verizon, Sprint/Nextel, T-Mobile, Boost, and a bevy of smaller carriers.

## 8. The Final Product vs Our Original Idea

When we started the project, we set out to build a web based invitation system. After all of the twists and turns, user focused design sessions, and thoughtful iterations, we launched a mobile social networking product. Mixd allowed people to create ad-hoc groups, share mobile photos and videos and see it all on a website later. The product was completely different, but still solved the same user problem "It's 5pm on a Friday night and I want to hang out with my friends". Mixd allowed users do just that, and as it turned out, to do a lot more.

We kept the product feature set simple and focused *and* we eliminated features that put unnecessary constraints on the user, like removing the time and place constraints. We reprioritized or removed features that made the product hard to understand, for example, removing email made it solidly a mobile experience, perfect for hanging out on a Friday night.

We rolled this product out on five college campuses across the US and signed people up to message boards to give us feedback. We had over 1500 message board postings responding to questions we had about their use of the product and giving us unsolicited feedback.

Once the product was live, we saw people using it exactly as we had expected, but also in unintended ways. Our core target audience saw immediate value in the product and loved the features we shipped with. Most all of the feedback we got was about making features that we had clearer and easier to use. In the areas where we added more steps or more messaging that users asked us to simplify. Where we thought our extra features or messaging helped, they often cluttered the experience.

One interesting thing to note: they didn't ask for any of the features that we had cut. Features like email notifications or maps or places to invite were never mentioned. We asked these users what they thought about opening their groups publicly, a reference to the earlier debate we had, and their reaction was lukewarm. Overall our focus on cutting features led us down a good path.

One of the key learnings, was how people used the product in unexpected ways. The flexibility of the product allowed them to use it as a group chat/status application like Facebook (<http://www.facebook.com/>) They set up different groups as chat groups for sets of friends, or family back home. We also found that these users were much heavier users of the product than the core audience.

## 9. Summary

Overall, we found that the process of focusing on a key user need was critical to moving quickly, staying focused, coordinating remote teams and communicating the product's benefits to users.

### The biggest lessons learned were:

- Have a core goal that can anchor the team no matter what else is changing around helps a lot with focus and clarity.
- If working on a distributed team, get together in the same location for both the kickoff and final launch – and have chili margaritas.
- Talk to your audience over and over again
- Get the product into the users' hands as quickly as possible, even if they are internal.
- Stay focused on that core goal, and brutally cut any feature that doesn't support it.
- Staying focused on core features makes it easier to communicate benefits to your audience.
- Removing complexity or assumptions in your product allows people to use your product in unintended ways. Those unintended uses may end up being more interesting than your original idea.