experiment mixtape

advancing your *solution* via *prototyping*

prototyping can be magical in how it gets a team to stop (just) discussing, and start building. secondly, prototypes help you engage with users in a different way, to continue your understanding. starting to prototype before you know what the solution will be can seem premature, but in fact taking a stab at a specific concrete implementation is a great way to bring clarity, even if you discard the prototyping concept itself.

use this mixtape to rapidly develop and build your solution concepts and then test your prototypes with users

experiment **mixtape tracklist:**

d.
HASSO PLATTNER
Institute of Design at Stanford

experiment mixtape

liner notes. pt. 1

<u>what to do</u>

1. prep: schedule the day and prepare yourself
read through the six-track mixtape following these instructions. block half a day on your team's schedules. find a space where you can build stuff. gather low-resolution materials (like cardboard boxes, construction paper, foil, pens, tape, etc.) and digital tools (computer, printer, etc.). plan for user testing: who will you invite in, or where will the team go, to get feedback on prototypes.

you should come into the day with a set of solution possibilities. these can be very rough concepts. if you still need to generate concepts, see the ***ideate mixtape***.

2. launch (15 min): warm-up and get to building
do a standing warm-up. do something different than you might do at typical meeting. launch prototyping: remind the team that you are *creating experiences* for users to react to, using low-resolution prototypes. get your team to move from discussion to building quickly. create a fixed end time for prototyping by telling the team when users will arrive or when you will go out into the field. (build in a time buffer break anyhow.)

3. prototype (90 minutes): get building
work in small teams (2-4 people) to create prototypes. when prototyping services and experiences, create the scene and play roles. depending on the stage of your project, either the "user-driven prototyping" approach or "wizard-of-oz prototyping" approach may be more useful.

continued >>

experiment mixtape

liner notes, pt. 2

4. test (90 min): let users to experience your prototypes
your team won't feel done after 90 minutes of prototyping. start testing anyhow to get initial feedback, that you can use in further development. go out in the field or bring users in for testing. remember to capture notes.

5. debrief (45 min): recap the feedback, and plan next steps
review the feedback you received, using the "feedback capture grid" as a scaffold to the conversation. discuss how to take solutions forward, or where you need more insight.

6. going forward:
you should be happy if some or all of your ideas failed to please users during testing. continue in one of three ways:
- keep working to develop your solutions, taking account of the feedback,
- generate new solution concepts or features of a solution (see ***ideate mixtape)***, or
- think about where you need to dig deeper to find insights (see the ***understand mixtape***; now you can do empathy work with or without prototypes in hand).

image pg. 1: flickr/michellecarl

d.

prototyping

WHAT is prototyping?

Prototyping is getting ideas and explorations out of your head and into the physical world. A prototype can be *anything* that takes a physical form – be it a wall of post-it notes, a role-playing activity, a space, an object, an interface, or even a storyboard. The resolution of your prototype should be commensurate with your progress in your project. In early explorations keep your prototypes rough and rapid to allow yourself to learn quickly and investigate a lot of different possibilities.

Prototypes are most successful when people (the design team, the user, and others) can experience and interact with them. What you learn from those interactions can help drive deeper empathy, as well as shape successful solutions.

WHY prototype?

Traditionally prototyping is thought of as a way to test functionality. But prototyping is used for many reasons, including these (non-mutually-exclusive) categories:
Empathy gaining: Prototyping is a tool to deepen your understanding of the design space and your user, even at a pre-solution phase of your project.
Exploration: Build to think. Develop multiple solution options.
Testing: Create prototypes (and develop the context) to test and refine solutions with users.
Inspiration: Inspire others (teammates, clients, customers, investors) by showing your vision.

Many of the goals of prototyping are shared across all four of the above categories.
We prototype to:
Learn. If a picture is worth a thousand words, a prototype is worth a thousand pictures.
Solve disagreements. Prototyping is a powerful tool that can eliminate ambiguity, assist in ideation, and reduce miscommunication.
Start a conversation. A prototype can be a great way to have a different kind of conversation with users.
Fail quickly and cheaply. Creating quick and dirty prototypes allows you to test a number of ideas without investing a lot of time and money up front.
Manage the solution-building process. Identifying a variable to explore encourages you to break a large problem down into smaller, testable chunks.

prototype to test

WHY prototype to test?

Prototyping to test is the iterative generation of low-resolution artifacts that probe different aspects of your design solution or design space. The fundamental way we test our prototypes is by letting users experience them and react to them. In creating prototypes to test with users you have the opportunity to examine your solution decisions as well as your perception of your users and their needs.

HOW to prototype to test?

Think about what you are trying to learn with your prototypes, and create low-resolution objects and scenarios which probe those questions. Staying low-res allows you to pursue many different ideas you generated without committing to a direction too early on. The objective is not simply to create a mock-up or scale model of your solution concept; it is to create experiences to which users can react. Bring resolution to the aspects that are important for what you are trying to test, and save your efforts on other aspects. You also need to think about the context and testing scenario you will create to get meaningful feedback. It is not always the case that you can just hand an object to someone on the street and get real feedback. Test in the context that your solution would actually be used (or approximate the important parts of that context). For example, if you are creating a consumer food storage system, let users test it in their kitchens at home – some of the nuanced but important issues will only emerge there.

Some tips for prototyping to test:
Start building. Even if you aren't sure what you're doing, the act of picking up some materials (paper, tape, and found objects are a good way to start!) will be enough to get you going.
Don't spend too long on one prototype. Move on before you find yourself getting too emotionally attached to any one prototype.
Build with the user in mind. What do you hope to test with the user? What sorts of behavior do you expect? Answering these questions will help focus your prototyping and help you receive meaningful feedback in the testing phase.
ID a variable. Identify what's being tested with each prototype. A prototype should answer a particular question when tested.

track 3

user-driven prototyping

WHY create a user-driven prototype?

Whenever you engage a user with a prototype, you are trying to better understand him and perhaps his reaction to your solution-in-progress. Often with prototypes, we ask the user to experience something we created, and we gain insight by observing their reaction and by talking to them about the experience. The intention with a user-driven prototype is to gain understanding by watching the *user* create something, rather than try something that you developed.

The value of a user-driven prototype is that different assumptions and desires are revealed when the user is asked to create aspects of the design, rather than just evaluate or experience the prototype. The goal is not to take what they made and integrate it into your design, but rather to understand their thinking and perhaps reveal needs and features that you may not have thought of.

User-driven prototypes are often useful in early empathy work, as a way to facilitate a different kind of conversation. User-driven prototypes are also useful after you have determined the context and form-factor of your solution, to help think about some of the features and details of that solution.

HOW to create a user-driven prototype?

The approach to creating a user-driven prototype is to set up a format for your users to create something which leads to your understanding of how they are thinking. As an example, if you were creating a website to allow users to create custom t-shirts, a traditional early-stage prototype might be a mock-up of the webpage with the features and buttons that you think might be appropriate. A *user-driven* prototype could be to give your user a blank piece of paper and ask her to draw what she thinks the features should be. You might provide a light scaffolding to get her going, such as a piece of paper with boxes in the layout of a possible website, and then ask her to create the content for those boxes. Of course, there is an entire spectrum of how much you provide and how much you ask your user to create. You need to find the balance, depending on your project progress, for a prototype that is scaffolded enough that the user feels that she can be generative, but open enough that you learn outside of your own biases and assumptions.

Other examples of user-driven prototypes include: asking a user to draw something (“draw how you think about going to the doctor”), to make an object with simple materials (“make a bag for diapers and baby supplies, using this paper and tape”), or to compile things (“tear out pictures from these magazines that represent your ideal mall shopping experience”).

image: flickr/itv-ntnu

wizard-of-oz prototyping

WHY create a wizard-of-oz prototype?

You use a Wizard-of-Oz prototype to fake functionality that you want to test with users, thus saving you the time and resources of actually creating the functionality before you refine it through testing. Just like the small man behind the curtain faked the power of the wizard of oz, your design team can fake features that you want to test. Wizard-of-Oz prototypes often refer to prototypes of digital systems, in which the user thinks the response is computer-driven, when in fact it is human controlled.

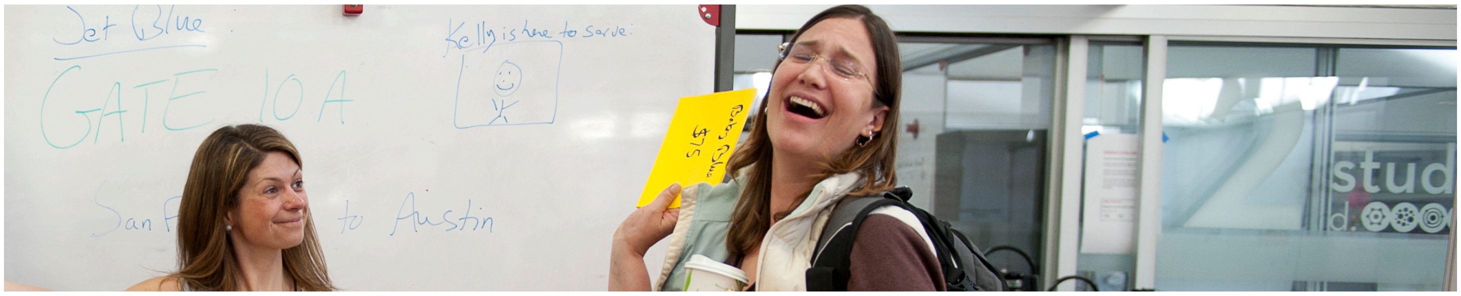
HOW to create a wizard-of-oz prototype?

Creating a Wizard-of-Oz prototype starts with determining what you want to test or explore. It is often the case that you want to test something that requires great effort to create, like coding a digital interface, but you need to learn more before it makes sense to invest that effort. Figure out how to fake the functionality you need to give the user an authentic experience from their viewpoint. Often leveraging existing tools can be very powerful: Twitter, email systems, Skype, instant messengers, Powerpoint to fake a website, projectors, computer screens repurposed in a new skin, etc. Combine tools such as these with your human intervention behind the scenes, and you can create a realistic prototype. The concept can certainly be extended beyond the digital realm, to create physical prototypes. For example, you could prototype a vending machine without creating the mechanics and use a hidden person to deliver the selected purchases.

A good example of a wizard-of-oz prototype is from the company Aardvark. Aardvark connects people with questions with people best-qualified to answer via a digital interface over the internet. To create the network and algorithm to do this would require significant coding, but the team wanted to test user's reaction to the interface well before the coding was completed. They used an instant messaging system and a team of people behind the scenes to physically reroute questions and answers to the right people. The result is they learned a lot and developed their concept notably without investing coding resources.

image: flickr/kaptainkobold

testing with users



WHY test with users?

Testing with users is a fundamental part of a human-centered design approach. You test with users to refine your solution and also to refine your understanding of the people for whom you are designing. When you test prototypes you should consider both their feedback on your solution and use the opportunity to gain more empathy. You are back in a learning and empathy mode when you engage users with a prototype.

HOW to test with users?

There are multiple aspects to be aware of when you test with users. One is your **prototype**, two is the **context and scenario** in which you are testing, three is **how you interact** with the user during testing and four is how you **observe and capture** and consider the feedback.

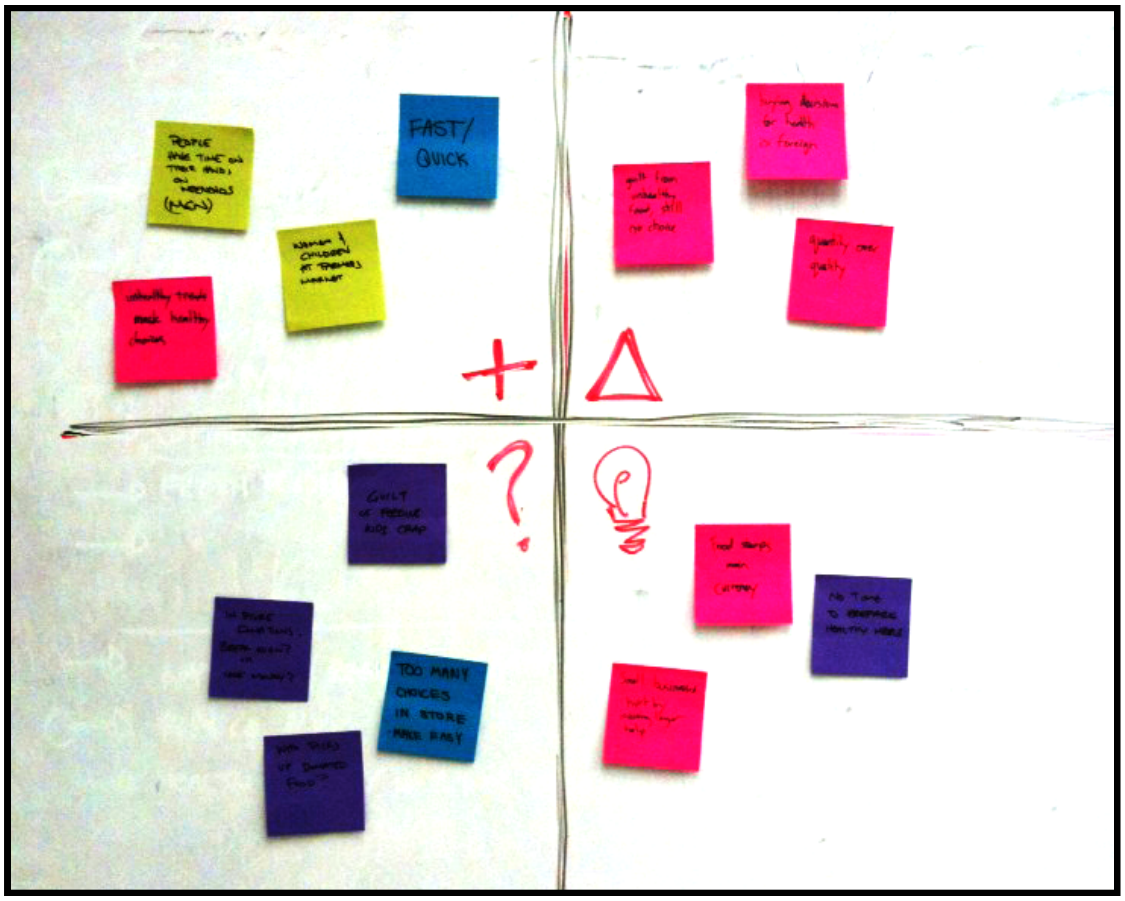
In regard to the first two aspects, you need to test a prototype in a context that give you the best chance for meaningful feedback; think about how the prototype and the testing scenario interact. If the prototype is a scenario, think about how to find the proper people (i.e. users relevant to your point-of-view) and get them in the right mindset so that you get genuine feedback.

Roles
During the testing itself, use intentional team roles, as you would with empathy work:
Host: You help transition the user from reality to your prototype situation and give them the basic context they need to understand the scenario (don't over-explain it, let the user discover through the experience). As the host, you will also likely be the lead questioner when the time comes.
Players: You often need to play certain roles in the scenario to create the prototype experience.
Observers: It is very important to have team members who are solely observers, watching the user experience the prototype. If you don't have enough people to run the prototype and observe, videotape the testing.

Procedure
Use a deliberate procedure when you test.
1. Let your user experience the prototype. Show don't tell. Put your prototype in the user's hands (or your user in the prototype) and give just the minimum context so they understand what to do. Don't explain your thinking or reasoning for your prototype.
2. Have them talk through their experience. For example, when appropriate, as the host, ask "Tell me what you are thinking as you are doing this."
3. Actively observe. Watch how they use (and misuse!) what you have given them. Don't immediately "correct" what your user tester is doing.
4. Follow up with questions. This is important; often this is the most valuable part of testing. "Show me why this would [not] work for you." "Can you tell me more about how this made you feel?" "Why?"
Answer questions with questions (i.e "well, what do *you* think that button does?").

track 6

feedback capture grid



WHY use a feedback capture grid?

Use a feedback capture grid to facilitate real-time capture, or post-mortem unpacking, of feedback on presentations and prototypes – times when presenter-critiquer interaction is anticipated. This can be used either to give feedback on progress within the design team or to capture a user's feedback about a prototype. You use the grid because it helps you be systematic about feedback, and more intentional about capturing thoughts in the four different areas.

HOW to use a feedback capture grid?

1. Section off a blank page or whiteboard into quadrants.
2. Draw a plus in the upper left quadrant, a delta in the upper right quadrant, a question mark in the lower left quadrant, and a light bulb in the lower right quadrant.

It's pretty simple, really. Fill the four quadrants with your or a user's feedback. Things one likes or finds notable, place in the upper left; constructive criticism goes in the upper right; questions that the experience raised go in the lower left; ideas that the experience or presentation spurred go in the lower right. If you are giving feedback yourself, strive to give input in each quadrant (especially the upper two: both "likes" and "wishes").