Iter1. Low Fidelity – Instructions

is211 Interaction Design and Prototyping
2019-20 T1

Due: *3 days (72 hours) before week 6 class*
Weight: *15% of final grade.*
Collaboration: *This is a team assignment. You must work only with your project team.*

Assignment

In this iteration, you will make your first trip through the observe-prototype-evaluate loop. We'll focus heavily on observation. Prototypes and evaluations will be quick and easy.

Phase 1: Observe

- Observation is the groundwork for design ideas that are rooted in real people's needs, goals, and values. **Each team member will lead observation of one person.** When you're done, the team should identify **at least three breakdowns,** i.e. specific situations where participants experience problems, reveal unmet needs, or use clever workarounds. Then you'll use these insights to **brainstorm 15 ideas** that would help participants overcome these breakdowns.

Observing People

Watching how people do things is a great way to learn their goals and values, and come up with design insight. We call this need-finding. Your goal is to uncover users' goals, problems, needs, and opportunities for improvement.

1. Begin by **choosing the type of people you will observe and the types of activities you will observe**. Remember, planning is essential, but your plans may change. Choose specific activities where people tend to experience the problems you have in mind, and plan to look for successes, breakdowns, and latent opportunities that occur when technology is used, not used, or could be used to support those activities. Plan to take notes and photographs (we recommend *not* using a video camera).

2. Next, **each team member must select a person to observe**. Each team member will lead the observation for their participants, and they must be assisted by at least one other team member. Observe people who actually experience the problem you are trying to solve. Also, try to get a diverse set of people, and give preference to people who are not similar to yourself (e.g. people outside SMU). Ask them to participate in this assignment and get permission from them. Coordinate with them to select a time that will be rich for observations. Tell the participants to perform their tasks as realistically as possible, while communicating to you as appropriate. Utilize the strategies we talked about in lecture to help you. You might also find it helpful to interview your participants after you observe them, checking to see if you have interpreted their actions correctly, and asking about breakdowns that often occur at other times (note how often they occur). Also, try to take at least one photo of each participant (if they allow it), but obscure their face for privacy.

Observing someone can take less than an hour if you plan it carefully. It will take longer if you haven't!

Keep in mind that the final application you build for this class will run on a computer or mobile device, but your observation may or may not involve any technology at all. For example, if you are designing a mobile phone interface for a task that can't yet be done on a mobile device, then you'll be observing how people do the task right now. Observing how things are done now will help you find ways to do better. On the other hand, if you are looking for ways to improve on an existing interface (e.g. for calling friends or doing email), then you'll want to observe people in situ, i.e., using their devices to do real tasks in their actual environment.

There are circumstances where you might not be able to observe someone in the real setting or when incidents are too rare to happen, in those cases, Interview may be a better approach. You may also conduct interviews on top of the observation to collect more information as part of the observation exercise.

Identifying Breakdowns and Brainstorming Ideas

Now that you've finished your observations, it's time to choose the problem and solution you will work on for the rest of the semester. Before you did your observations, you probably had only a superficial understanding of the problem, and you probably weren't able to choose the best solution. Here is your opportunity to make them better.

1. Go over your observation notes and **identify at least three moments where participants experienced a breakdown**. These should be situations where one or more participants had to work around something to achieve a goal or could not achieve that goal at all. Make this as specific as you can, listing the participants, the goal, and why it could not be achieved. Breakdowns like these are key for design!

2. Now, using all of your observation notes, **brainstorm at least 15 ideas** that would help participants overcome those breakdowns. Each idea should approach the breakdown from a different perspective, focusing on different causes or unmet needs. For example, you might have this breakdown: “Ben gets to Farrer Park MRT station at 8:30am, but the train is usually too crowded, and he has to wait for two or three trains to go by.” In this case, an idea could be, “Ben needs a way to plan his trip based on how much room is available on each train,” or even, "Ben needs to lose weight so he can squeeze onto crowded trains." (It's helpful to use the phrases "needs to" or "needs a way to" in your ideas.)

   When brainstorming, it’s important to challenge your assumptions about what makes ideas good. Don’t worry too much about what is possible or impossible, and have fun! It’s important to get a lot of ideas, so we’re requiring you to get at least 15 **(that’s 15 total, not 15 per problem)**. When you’re done, you may be surprised at what you find.

3. Finally, choose one or more breakdowns you will address in your project along with one or more ideas for overcoming those breakdowns. Use these to **revise your problem and solution statements**. The problem should clearly express breakdowns that your users experience, and the solution should briefly explain how your system will address those breakdowns. Together, your problem and solution define your high-level design strategy, and you need this before you begin to design a system.

   Note that your problem and solution should be more specific than a simple marketing statement. For example, say you want to improve the check-out experience at the grocery store. If you say that your problem is “shopping” and your solution is “We make shopping fun,” then you haven’t told us much of anything. Decide which part of the experience you will focus on and how you will focus on it. A better problem would be “Waiting in line at the grocery store is annoying” and a solution could be “A mobile application that shows you personalized news so you can make better use of your time.” Alternatively, your solution could be “A queuing system that will reduce waiting times.” Or, you might try a completely different solution like “Make grocery stores more like farmers' markets, where payment is distributed across the stands that have the food.” All of these are valid ways of expressing problems and solutions—they suggest different possibilities and have different implications or entailments for what constitutes a good design. Choose your own and write them down.

Finally, keep in mind that this is a semester-long project. You should be ambitious, but not so ambitious that you will be unable to prototype an interesting solution to the problem you identify. We will help you to get this right, but we need you to make your best effort.

Establish Requirements

Now that your problem and solution are clear, it's time to establish the specific requirements for your project. There are many tools you can use to do this, and we require you to use two of them: **personas** and **scenarios**.

1. **Create at least one persona** that describes a typical user of your system. If your system involves different types of people, then you may need more than one persona, but the first one should be your primary persona. Your persona should mention the following:
    - Name ***(Use invented names, not the names of people you observed.)***
    - Profession, age, location
    - Attitudes
    - Activities
    - Influencers
    - Workflows
    - Pain points and frustrations
    - Goals
    - Relevant technology used (hardware and/or software)

    Your personas should be brief. You can cover all of the important details in a short paragraph. Don't bother making multiple personas unless different types of users would interact with your system in substantially different ways. See the sample persona for an example.

2. After you create your personas, **define at least two scenarios** that illustrate your problem and solution. These scenarios should tell stories about how your personas encounter the problem and how they interact with your system while seeking a solution. The first scenario can be **easy**, where the system does its job in a simple, straightforward way. The second scenario should be **harder**, where the user may have to handle an unexpected or inconvenient situation. You need both, because it's important to think about what happens when things go right and when they go wrong.

    A scenario should give specific details about a situation but should avoid giving specific details about a user interface. Focus on aspects of the environment, the problem, and the solution. In *About Face 3 (Chapter 5 Modeling Users: Personas and Goals)*, Cooper Reimann, and Cronin call this kind of scenario a *context scenario*. Ginsburg gives a concise list of the things you should put in such a scenario:
    - **Motivation**: What prompted the persona to embark on the scenario?

- **Context**: Where is the persona while the scenario is taking place? Does the context change over the course of the scenario? Who else is involved? What other devices are involved?
- **Distractions**: What kinds of distractions or interruptions typically occur in the scenario? How does the persona deal with such distractions?
- **Goal**: What is the persona's goal in the scenario? Is it information, an artifact, an emotion?

Writing these requirements brings you to the end of the observation phase of this iteration. Your personas and scenarios are important tools for designing your prototype. You should refer to them often and update them as your understanding evolves.

Phase 2: Prototype

With your requirements in hand, it's time to begin building a concrete system. Start by looking for some design inspiration, and then start putting your own ideas on paper.

Find Inspiration

Your first step is to find inspiration for designing your own solution by looking at what already exists in the world. Inspiration can be existing applications, artifacts, products, or services that relate to your concept. If you find another system that solves the problem you are trying to solve, then you should certainly include it in this list. (Remember, it's good to be original if you can.) Here, web search is your friend (potentially useful sites include Google Scholar, the ACM Digital Library, TechCrunch, Engadget, and Digg).

Pick out your five favorite pieces of inspiration and note how each one is related to your solution (What did you take away from it? What did you learn from it?...In other words, why did it inspire you?). Keep in mind that it is important to interpret "related" broadly. A carrot-peeler or a measuring cup may be your inspiration for an elegant and ergonomic design of a software interface. You may be inspired to improve upon an existing service, or you may use an existing service as an example of what to avoid. Cast the net wide and find as much as you can. Inspirations that fit our crowded MRT example could be an observation of people using MRT timetables, GoThere.sg, or even a weight loss program.

Build your Prototype

Armed with this inspiration, you will do rapid prototyping to move quickly from you design idea to a first draft of your user interface. Brainstorm some ideas and sketch them out on paper. When you've got a good idea of what your system will look like, then it's time to build a prototype that you can show to others.

1. Start by making a **navigation diagram** that gives an overview of your prototype. Navigation diagrams show how your prototype's **views** are related to each other. a view is a full screen that users will see (e.g., in a web app, a *view* usually corresponds to a *page*). This will help you think through how the views of your prototype connect, and it

will help your evaluators understand how your system works. Your diagram can be one of the following two types:

- A **sitemap** arranges the views of your prototype into a conceptual hierarchy. On web sites, this can also serve as a navigation aid to users.
- A **flow diagram** shows how the user progresses through views in your scenarios using common flowchart symbols.

Make sure that you give every view in your prototype a name, and make sure that your final navigation diagram shows only those views that actually appear in your prototype. Don't worry too much about what you should or should not classify as a view. There are no hard and fast rules for what constitutes a view, but consider the following guidelines:

- Simple alerts and dialog boxes don't need to be counted as separate views.
- Complex dialog boxes could be counted as a separate view.
- A tab that fills most of the screen could be counted as a separate view.

2. Now it's time to **make a low-fidelity prototype that implements your design.** This prototype will show how all the parts of your system are connected, but it will have a rough appearance and no behavior apart from navigation. You **must** make a paper prototype.
    - A **paper prototype** concretely shows all the elements of a user interface, except that it's implemented with pen and paper, as opposed to pixels and code. This helps you focus on the concepts, and saves you from wasting hours twiddling pixels. If you use this method, make sure you use a thick pen with dark ink.

Walk Through your Prototype

After you've built your prototype, walk through your scenarios on it to test out your design, and then try improvising new scenarios. If you've made a paper prototype, all team members should master the skill of operating the prototype, so take turns, with one of you being the user and one being the computer. Walk-throughs like these help you to identify missing steps or missing information in your scenario. They also help you find dead ends in your prototype or parts that you forgot to create. Feel free to fix these problems whenever you find them. (If you made a paper prototype, you can even fix problems during an evaluation!) You may also expand your prototype or add more scenarios if you need them. (Revise your navigation diagram, if necessary.)

Keep doing walk-throughs until you are satisfied with your prototype and your scenarios. Then **take at least one photograph or screenshot of every view** in your prototype and (Optional) **film a video walk-through of each scenario**. Keep these video walk-throughs simple: one person should read the scenario, another should play the user, another should play the computer (if necessary), and another should record video from their mobile phone. Do not speak any words other than those written in the scenario, or you will be penalized. When you're done, upload your video to **YouTube**. See the course web site for tips on making videos.

Phase 3: Evaluate

In the final phase of this iteration, you will conduct a **heuristic evaluation** of your paper prototype. This is an easy way to find usability problems in your paper prototype.

Your heuristic evaluation should follow the How to Conduct a Heuristic Evaluation and Ten Usability Heuristics readings by Jakob Nielsen. You will get 4-5 **expert evaluators** from your own class. Assignment 2 requires each student to serve as an expert evaluator for a team. The team's job is to make sure that the evaluators understand the prototype. As you have made a paper prototype, you will "play computer" for your expert evaluators by operating the paper prototype as your evaluators explore the interface. You will then compile the expert evaluators' results into one document.

Conduct a Heuristic Evaluation

See this list to find out who your expert evaluators are. Try to get all of your own team members to attend all evaluation sessions. To facilitate this, we suggest you try as hard as you can to schedule evaluations at or near the same time. (Make sure they don't talk to each other during the evaluation period!) Schedule your evaluators back-to-back (one at a time) so each evaluator can "operate" the prototype alone.

One person from your team will be the *facilitator*. The facilitator should greet the expert evaluator(s), explain how the session will work, and give a brief introduction to your prototype. For paper prototypes one or more people should be the *computer*. Any team member who is not acting as a facilitator or a computer should observe, taking notes and pictures.

Make things easy for your expert evaluator(s) by printing a sheet of your scenarios, a sheet of Nielsen's heuristics, and a copy of your navigation diagram. If users would normally receive a written manual or instructions, provide these for your experts as well (most projects should not need these). Remind each expert evaluator to bring their computer so they can fill out their As2 submission as they conduct their evaluation. Assign a letter (A, B, C, D, or E) to each expert evaluator to simplify your record keeping. When the evaluation takes place, remember that the evaluator is the expert. Let them explore and evaluate the interface as they choose, but make sure that they go through all of your scenarios at least once.

Compile Results

Your final step is to collect the usability problems from all of your expert evaluators into one document. When two expert evaluators find the same usability problem, you must merge them into one. You must also update severity ratings to whatever **you** think they should be. If you give a usability problem severity 3 or 4, then you **must** propose a fix, and you **must** fix it in the next iteration. If you think that a usability "problem" found by your evaluators is not really a problem at all, then you should give it severity 0 and add a note explaining why you do not think it is a real usability problem. Also, feel free to merge multiple usability problems into one or to change usability problem names, descriptions, and related heuristics whenever you feel it is appropriate. You can even add usability problems that your evaluators missed, if you wish.

When you're done, sort the usability problems you found in order from highest to lowest severity.

To make your life easier, we've prepared templates for you and your evaluators that are nearly identical. You can copy usability problems directly from their submission, but add a "source" at the top. If your evaluators followed the format correctly, there should be a link to each of their usability problems at the top of their evaluation, and you can use this to link to their usability problem. (If they didn't do this, then don't worry about the links.) You should also be able to re-order your usability problems without having to re-number them, as the numbers will update automatically.

This completes your first evaluation! The results you have collected will provide you with feedback you can use to improve your design before you begin implementing a high-fidelity prototype in your next iteration.

Final Notes

You might think that the steps of this iteration should be followed in strict order, but designers rarely work this way. Design usually involves lots of experimentation and backward steps. You could write a problem and solution and later discover that you can't write a convincing scenario for them. When you go looking for design inspiration, you might find a better solution than the one you chose after your observations. Or you might find a serious design bug in the middle of your heuristic evaluation and revise your paper prototype before you see your next expert evaluator. If you feel the need to do something like this, just go with it. Just make sure that the documents you submit for this iteration show your final personas, scenarios, and prototype.

If you really want to go above and beyond for this assignment, then try to explore several design alternatives and pick the best one for heuristic evaluation. As we saw in class, exploring multiple design directions tends to produce the best designs. A relatively easy way to explore alternatives is with **storyboards.** A storyboard is a comic-strip-like set of drawings about what your interface does in a scenario. (If you want some pointers on how to make storyboards, check out Amal Dar Aziz's excellent *Guide to Storyboarding* or see Scott McCloud's *Understanding Comics* on reserve in the SMU Library.) Whatever format you use to document your alternative designs, you should submit *clear* photographs of them in your journal.

Suggested Schedule

You have four weeks to work on this assignment. Here's a plan we suggest for you.

- **Before week 3 class:** Finish conducting observation exercise within this week.
- **Before week 4 class:** Analyze observation notes, identify breakdowns and brainstorm ideas. Then write persona(s) and scenarios and look for inspiration. Start on paper prototype.
- **Before week 5 class:** Update or finish prototype, arrange meeting times with evaluators and run heuristic evaluation.

- **Before week 6 class**: Compile evaluation results, suggest changes and prepare for iteration 1 presentation.

Presenting In Class

Refer to assigned presentation slot online. Please present the following:

- Overview (1 min) : What is your problem and solution? (***Always start with this.***)
- Observation (3 mins): How did you find participants to observe, and what breakdowns did you discover?
- Requirement (3 mins): What personas and scenarios did you write?
- Prototype (3 mins): What does your prototype look like as you walk through your scenarios?
- Heuristic Evaluation (3 mins): How did it feel as your prototype was being evaluated? What usability problems did you find after evaluating your prototype?

If you have a navigation diagram and a prototype, then it's best to start by showing your navigation diagram on one screen while walking through your prototype on the other, highlighting any changes. Choose one speaker, so everything will run smoothly. You may use slides or refer to your submission documents (in whatever state they are in).

Submission Checklist

Submit your assignment by modifying the templates prepared for you on the submissions page. Your submission should contain the following:

1. **Overview**: containing your team's id and name, your project name, concise statements for your problem and solution, walk-through video links, and prototype links (if any).
2. **Observations**:
    - **An introduction** stating how you found people and the activities you observed.
    - **Observation Notes** for five participants (or four, if your team has four people). For each participant, indicate where and when observation took place. A summary at the end is optional. **Do not refer to participants by name.**
    - A list of **breakdowns** you observed. These should indicate the participant(s) who experienced the breakdown and what goal(s) they were perusing. Pictures are helpful. For each breakdown, list **ideas** for overcoming it **(at least 15 total).**
3. **Requirements**:
    - A list of **personas** (at least one)
    - A list of **scenarios** (at least one **easy** one and one **harder** one)
4. **Prototypes**:
    - Five **inspirations** related to your solution. Briefly explain how they relate.
    - A **navigation diagram** that shows how views of your prototype are related.
    - At least one photograph or screenshot of **each view** in your **prototype.**
    - One photograph of your prototype showing all team members' **signatures**.
5. **Heuristic Evaluation:**
    - A list of **evaluators** linked to their **As2 submissions**

    - A compiled list of **usability problems**, linked to their **sources**. **Merge duplicates** and give **fixes** for problems with severity > 2. Use “notes” fields to explain how you modified usability problems or why you gave them severity=0.
6. **Journal**: (Optional) A place where you can write anything else that you want the grader to know about what you did, like alternative design directions that you explored.

Samples

Here is an excellent sample submission that you can use for reference.

Grading Rubric

Make sure you complete everything listed in the **Submit** section. After checking that all required portions are complete, we’ll use the following rubric to mark this assignment. Look here for more information on how to use the rubrics.

- Overview: **20 pts**
    - Problem and Solution are clear and convincing
    - (+) Problem is especially challenging or solution is especially creative.
- Observations: **20 pts**
    - Breakdowns are grounded in observations & show clear design opportunities.
    - (+) Observations are especially insightful or were exceptionally hard to get.
    - (+) Brainstormed ideas are especially creative.
- Requirements: **20 pts**
    - Personas cover all important user categories.
    - Scenarios are clear, give details on tasks, and omit details on the system.
    - (+) Scenarios are especially detailed or cover a wide range of behaviors.
- Prototype: **25 pts**
    - Inspirations are related to your solution.
    - Navigation diagram shows how the views of the prototype are related.
    - (+) There is clear evidence that the team considered a variety of designs.
- Heuristic Evaluation: **15 pts**
    - Compiled usability problems are organized well with no duplicates.
    - Severity ratings are appropriate.
    - Fixes are proposed for usability problems with severity > 2.