Understanding the Design Process

Notes for the UX Design for Developers video course

Design processes vary a lot based on teams, problems, and other circumstances. The purpose of the design process coverage in this class is

- to establish some general philosophical concepts that are associated with good design
- to set out a very general process flow for a design process
- to offer options at each point in the process that a team might want to consider
- to practice various techniques that can be used in a design process
- if the class is given for a team undertaking a specific design project, then the class will discuss the process that will be used for the design project

General Philosophies
In my experience, the most important general philosophies to a successful design process include:

- Understanding the goals of the design process
- Experimental design as the guiding principle
- The "design funnel" that serves as a process metaphor
- The need for collaboration
- Detail before abstraction

Let's look at these goals in detail.

Goals of the design process
During design, you are typically charged with creating a design that has some combination of the following requirements or needs:

- Delivers specific application functionality
- Helps users avoid errors / addresses safety issues
- Helps users be productive
- Is aesthetically pleasing (especially important in software for resale)
- Is intuitive to use and easy to learn

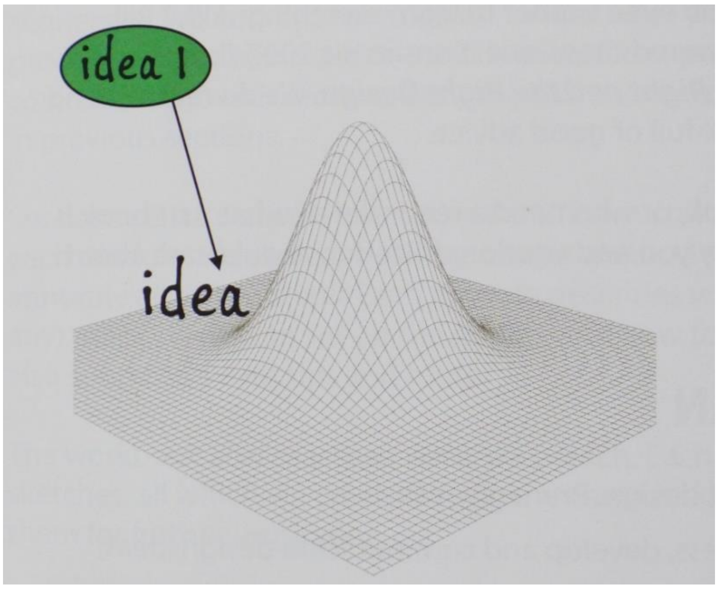
The relative importance of these factors varies with the project. In a healthcare project, for example, error avoidance tends to take a higher profile. In a highly complex process involving specialists, low training time may not be as important.

But usually these factors are all present to some extent as goals in a typical design process. It's useful at various points in the process, especially when reviewing designs, to see if these goals are being met.

The goals and priorities of the design process should be well understood early in the process. We will discuss techniques for achieving that understanding, including a technique called the $100 exercise.

Experimental design

Most development teams, particularly those in the Microsoft space, have a tendency to approach the design process in a highly linear fashion. One alternative is considered at the root, and then the team iterates around that alternative. This can be expressed graphically as exploring a single "design hill" to see how high one can get:



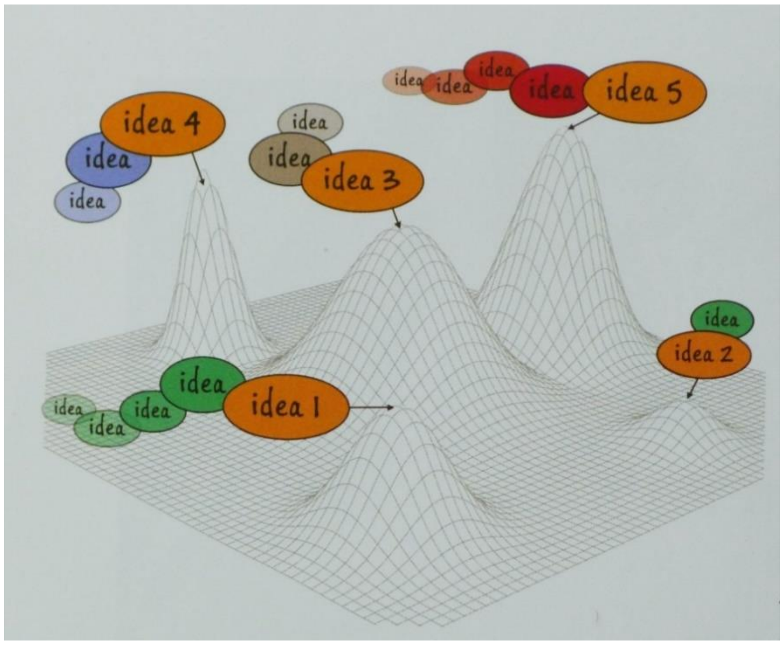
For modern UI stacks, we have enormous flexibility, and new platforms have opened up radically new design possibilities. This linear design process tends to produce traditional designs, and rarely takes advantages of what new platforms and technologies will do. The end result usually just "trims around the edge" of existing designs, adding a few elements and improving the cosmetics.

Instead of focusing on a single design path, I *highly recommend* making your design process experimental. You should being with the intention of exploring multiple designs that are aimed at the same problem. **Design projects that I lead always have phases of experimental design.**

Benefits of experimental design

The clearest benefit of experimental design is that the design team is likely to a better design, faster. It might sound as if experimental design would take more time, but a good storyboarding process will create and assess many designs very quickly.

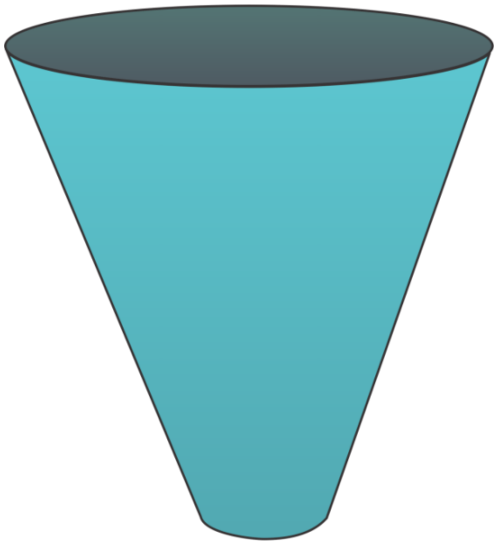
Experimental design is sometimes called exploring a *design space*. Pictorially, this diagram expresses the spirit:

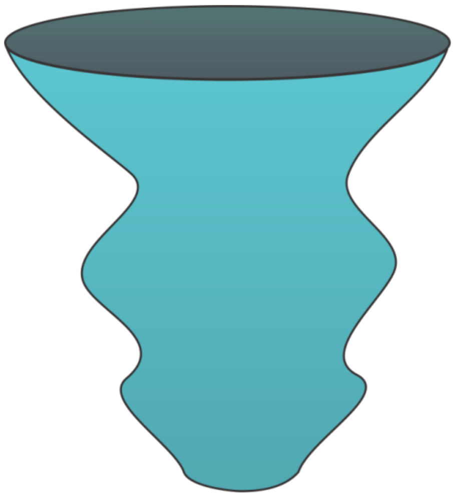


(These diagrams are taken from Sketching User Experiences, the Workbook. A link to the book is in the resources section at the end of this document.)

Design funnel

Such an approach will necessarily generate many ideas, some of which must be discarded. Some designers speak of a "design funnel"





In reality, though, there are phases. Ideas are added, culled, and then new ideas are spawned. So the design funnel tends to look more like this.

The key idea is that you want to contribute lots of ideas at various stages, and then ruthlessly prune so that only the best ideas and designs make it to the end of the funnel.

Collaboration

Some designers abhor collaboration, but in my experience collaboration yields a multiplier effect on the number of ideas generated. Plus, with today's complex systems, it's quite hard for one person to master the entire domain completely, so some members of a collaboration team contribute to a broader group understanding of the domain.

Detail before abstraction

Teams sometimes feel they must get to an abstract, general understanding of their domain as quickly as possible. However, in my experience, such an abstract model of the domain comes naturally once a critical mass of details about the domain have been absorbed.

Abstracting too early will bias later observation, as the observer tries to fit the detail into the already extent abstraction. I believe a good design process starts with an understanding of the problem domain, but then, through user observation, gathers sufficient detail to begin the process of abstracting a solution. I discourage abstracting too much before user observation. During user observation, it's important to have an open mind about what the users really need rather than rely on your pre-determined abstraction of what they need.

Design Teams

Design crosses many areas of expertise. As such, those who are generalists often do well in design. Specialists can also participate in design, depending on the needs of a particular design process. Here are some of the areas of expertise that can be useful during design – and some of them are surprising:

- UI technologies
- Code writing
- Psychology
- Neurobiology
- Evolutionary biology
- Business finance
- Visual arts, including aesthetics, layout, composition, and color
- Drawing / sketching

However, you don't have to an expert in any of these to be a designer. The most important characteristics are open-mindedness and the ability to see things from the point of view of another person (typically the user).

Design teams can be any size, though I discourage design done mostly alone except by those with both solid design experience and an understanding of the problem domain.

It's OK for various members of the design team to work alone at various points – some people get their best ideas when alone. However, an effective design team usually has at least three people working together on various aspects of design, and teams up to ten or twelve can be effective with a good design team leader.

General Design Process

Most effective design processes follow a general order of sub-processes, and a good general model looks like this:

Understanding

- Business needs and UX strategy
- Observation of users in the field
- Analysis of observations to gain insights and form work models
- Listing and prioritizing design tasks based on that analysis

Design

- Visioning
- Ideation and Storyboarding
- Illustration/wireframing/paper mock-up
- Interaction prototyping
- Evaluation of designs

But these steps are rarely completely linear. Construction of work models might uncover gaps in understanding that require additional user observation. Evaluation may suggest going all the way back to storyboarding, if the designs evaluated simply don't fit the needs and requirements of the users.

Business needs

An understanding of business needs lets you draw some parameters around your design possibilities. Generally you want the answer to questions such as:

- Why is a new system required?
    - Where does the old system fall short in business needs?
    - What has changed in the business that must be accommodated in a new system?
- What are the platform and environment requirements?
    - What technologies are available?
    - What older applications must the new system work with?

Business need analysis also helps you understand relative amounts available for investment and general time frames, plus any critical business constraints you must satisfy.

It's important to cover these questions at the very beginning of the design phase.

Getting a baseline from the existing system

If the new design is replacing an existing design, then there is an additional useful task to perform during this stage. Find out how to measure throughput for the current system by finding an appropriate measurement or metric. It's usually something like "orders completed per hour", or "revenue generated per day". It's almost always something reasonably easy to count or calculate.

This information won't usually be needed until much later. However, it's important for assessing the overall impact of your design after implementation. You can usually do a measurement for the new design, and a comparison to the old design will tell you something about the success of the new design.

It's common to see improvements of at least 10%, assuming the old system is a typical, last generation system, and the new design is well done and well suited to the user's tasks. If the new design fares less well on key metrics, that's a sign that something went wrong during the design process, no matter how pretty the new design might be.

Prioritizing investment in design

Business needs also help a design team where to focus time and attention. A team can do design innovation that addresses lots of different areas. Each project has its own "fingerprint" of areas where design is needed.

The most common areas where innovation can be done during design include:

- Changing business processes and task flows
- Adding new capabilities
- Improving performance
- Helping users find the information they need faster and more easily
- Decision support
- Reducing errors
- Improving aesthetics
- Making the system more modular, or combining modules that need a common user experience
- Clean-up (removal of obsolete functionality)

Once a team has completed their understanding of business needs, they should understand (1) which of these areas need design and attention and (2) the relative amounts of attention needed by each.

The “$100 exercise” is a common way to quantify that understanding. Each person decides how he or she would allocate a total investment of $100 in the above areas. The class materials should include a separate sheet that is used for the $100 exercise and a spreadsheet is available to help you analyze results of the exercise.

UX Strategy

The business needs phase may be brief when designing to replace an existing system. The business already understands where it makes money and what kind of technology facilitation it needs in general.

However, sometimes a business is attempting to gain a competitive advantage through UX design. This usually requires more innovation and thus more investment in design. It’s also common in this situation for the software being designed to be used by customers instead of, or in addition to, internal users.

In such a case, UX design often melts into business process re-engineering, sales strategy, and other wider aspects of the business. It becomes essential for those involved in design to understand the *business model* of the organization. This might involve answering some or all of the following questions:

- **Who are the customers?** What are the main groups of customers? Which ones are currently responsible for the company’s profits? Which groups of customers does the business want to grow or add to the customer base? How does the company retain existing customers while fostering change?
- **What value does the company deliver?** What tangible or intangible needs of the customer does the company satisfy?
- **What channels are used to reach customers?** What existing channels does the company use, and what are the possible channels to add?
- **What partners are needed to succeed?** What suppliers or other partners must be integrated into various business processes affected by our design?
- **Who are the main competitors, and how do we ensure that our designs foster a competitive advantage over them?**

User observation

An effective design process usually has some time in which design team members observe users in the field. Afterwards, an interpretation phase is done to understand the observations, and possible additional observation phases to clear up questions or ambiguities from the earlier interpretation/analysis.

If possible, it should happen where actual work is being done. Users can provide running commentary if feasible.

The working environment needs to be understood. A camera is a standard piece of equipment when observing users.

Typical process for observing users

- Choose a small set of representative users for the business function or role involved in your design. Request time to sit with users and watch them do their jobs (or use a web site the way they normally would).

- For each user, allocate about thirty minutes. Begin by telling them to go through their normal work routine. If the environment allows it, ask them to provide a running commentary of what they are doing while they work.
- Observe, make observations, and record questions for later. In general, it's a bad idea to interrupt the user's work flow with questions during the early parts of the observation.
- Try to allocate five to ten minutes at the end to ask questions. Those should include the questions recorded during observation, and possible some of the following types of questions depending on how much you already understand about the user's work roles:
    - Which functions/tasks do you use/do most often each day?
    - What annoys you about the current system? What could be done better?
    - Are you able to quickly find the information you need?
    - How is your job different from the other people in your department?
    - Does the current system have any bugs that interfere with your work?
    - Are there fields you are forced to fill in even though you don't know what should be entered?
    - If you could change or add one thing about the current system, what would it be?

It is also sometimes helpful to draw out users to obtain more detailed answers. A good general purpose question is to ask "Can you tell me more about that?" The single word question "Why?" can be surprisingly helpful.

Video capture

It isn't usually feasible to use a video camera to record the user's actions. Besides logistical difficulties, people tend to act differently when they know the camera is recording them.

However, sometimes an alternative capture mechanism can work even better. Programs such as Camtasia can record the user's screen, along with their running commentary. If it is feasible to do the contextual inquiry on a system with Camtasia loaded, then you may want to use this technique. However, don't forget to budget in time to review the recordings. Such review can be quite time-consuming.

Static vs. Dynamic environments

Most work environments tend towards the static end of the continuum. That is, the environment doesn't change that much or very quickly, and there are few if any external factors that impinge on the environment of the user and the system.

With mobile systems, or with fixed systems in field conditions, environments tend to be more dynamic. This means the need to observe not just users in the field, but users under as wide a variety of field conditions as possible. The designer should understand what kinds of external effects, such as interference, temperature, or whatever might impact the design of the system.

When designing for a mobile experience, it is usually necessary to budget more time to the user observation phase of design. It's typical for a user observation to be an hour or more, instead of the half hour typical when observing desktop users.

Recording User Observations

There are various ways to structure and analyze the information collected during user observation. Many of them start with a relatively unstructured technique. As soon as possible after observing users, design team members should write down observations, which are just simple sentences describing something notable that was observed or concluded while watching users perform work. Here are examples of typical observations:

- "The user can't easily find a record created a couple of hours earlier, but they often need to come back to it."
- "The system won't let the user proceed without filling in this field, but they sometimes don't know that information yet."
- "The user doesn't know why anyone needs data field X, and thinks it might not be needed by anyone."
- "The user gets frustrated trying to find an item by phone number."
- "The user keeps a sticky note to remind them of X."

A typical half hour with a user can generate anywhere from three or four up to a dozen or more observations. Thus, a design team with several members can generate dozens or even hundreds of observations during this phase.

Analyzing observations

On my own teams, I ask team members to email their observations to me, and I place them all in a spreadsheet.

Then, as a team, we discuss and categorize these observations. The spreadsheet includes a column for importance of each observation, since some observations will have a much higher business impact than others.

Some observations will be clearly related to one another, so the team should decide an appropriate category for each. My spreadsheet includes a column for category, and it's simple to fill in categories because Excel will auto-fill categories that have been used previously. Most design projects will have 6-10 categories for the observations. Here are some of the categories noted in a recent design project:

- Basic data entry
- Application data entry
- Data visualization
- Data transfer
- Contact management

Each user observation analysis will typically have its own specific list of categories, so you should not take the above list as a reference – just as an example of the kind of categories you will typically see.

Some observations or combinations of observations will yield a specific insight into a problem to be solved or an opportunity to leverage design to add a lot of value. Accordingly, the spreadsheet I use has a place for insights gained from observations.

Insights should be significant. If you are generating dozens of insights, then it's likely that some of them are too trivial or don't add enough value to be important during design.

Affinity Diagramming

An alternative to the spreadsheet approach above is Affinity Diagramming. This technique also categorizes the observations generated during conceptual inquiry, using whiteboards and sticky notes. The class will discuss this technique briefly, but it is also described in detail in several design books.

Affinity diagramming is often favored by designers, who are not as enthusiastic about analytical tools such as Excel. I find that most developers do not have such a preference, so I generally recommend the Excel-based approach described above for developer teams to analyze their observations. I also find that the spreadsheet approach is better for categorization, capturing insights, and noting priorities.

Other analysis of user observation

In some cases, the precise task flows and activities performed by the users are not known in detail by the design team. In this case, from the information gathered while observing users, the team can create work models which map out major activities and task flows. Many work models are fairly simply flowcharts. Business analysts often make some of their strongest contributions here, as they are accustomed to some of the case analysis that is necessary.

At the end of this work, you should have some general task flow information for the most frequent task flows, especially the ones that have a major effect on revenue or cost. You don't necessarily need to map out all task flows in detail.

These task flows and work models should be validated by an appropriate person. A highly experienced user, or a manager of a group of users, is often a good option for that validation.

Ferreting out important actions and user needs

It's common during analysis of observations to realize that some signification restructuring of user interaction is needed. The most common cause is that the tasks the user must accomplish are all over the application, and need to be made more accessible, often by being brought together.

However, designing any kind of "hub" that handles common user needs means knowing what actions are the most commonly used. While observations may give you a pretty good idea, when that is a central part of your redesign, it's important that you get those actions correct.

Surveys of users are a common way to do that. The survey should be short and to the point – don't draw up something that looks like a test. For example, in one design project, we sent an email to customers asking the single question "What top three tasks do you go to the customer inquiry screen to accomplish or find out?"

Analyzing that summary gave us a list of five things that covered a very high percentage of daily work for the typical user. The Customer Hub was designed around those actions, and was the most successful part of the product redesign.

End results of the Understanding phase

One of the key things to understand by the end of the Understanding phase is **where the application adds business value**.

Every business has core functions where it adds value, and there are usually only two to four of them. Design processes should focus on the modules of the system that match these core areas. In particular designing in flexibility and configurability is particularly important in these modules.

On the soft skills side, the understanding phase isn't really completed until you have a reasonably clear understanding of the users wants and needs, as well as what the business requires. This is sometimes described as feeling *empathy* for the user. You are able, based on your understanding of them, to gauge reasonably well what they will like and what they will not like.

That ability won't be perfect, of course, but it needs to be good enough to serve as a guide during the design process. You need to be able to both reject designs that obviously won't serve them or that they will dislike, and know when you are on the path to a design that is likely to be a good choice for them.

Creating a design task list

One of the most important deliverables for the understanding phase is a list of "design tasks". These are specific areas in the application that need some design focus.

The list is often created by using the categorized list of observations as a foundation.

Here are some design tasks that were the end result of a recent design project:

- Overall navigation – how the user moves from place to place in the application
- Creating a new quote
- Finding a contact
- Work management dashboard
- Account verification
- Data visualization for transport options
- Notes integration

This list should be prioritized to help the design team concentrate on the most important design tasks. Ideally, the design team should develop the list and prioritize it as a group.

Sometimes there are serious differences of opinion about the priority of design tasks. It may be necessary to work with project sponsors or other stakeholders to develop a final priority list. However, such a need should not prevent the design team from going on to work on the some design tasks that are agreed upon by the team as important.

Design phase may begin with Visioning

The design phase often begins with a free and open discussion on the general nature of the new system. This can include some blue sky possibilities, such as how a different platform might fit, or a radically different approach to an underlying business process. For example, the project might have started with the assumption that the application would be browser-based, but the mobile aspects of the application mean that a native app approach has significant advantages.

There are certainly some design processes for which such a phase isn't needed. But it's typically short, so don't dismiss it out of hand. A Visioning phase can be as brief as a half hour meeting to do some brainstorming along the lines of "If we had unlimited time and budget, how would we approach this design?" Sometimes that generates game-changing ideas that turn out not to be so difficult to use or implement.

Another open-ended question that might work well during visioning is "What is the impact on our system of the latest technologies in the field?" As devices multiply in an "Internet of Things" era, it's important not to use approaches via inertia that are no longer optimal because of new capabilities. However, even such capabilities as voice integration are much more practical than ever before, and often not even considered by typical development teams until someone higher in the organization suggests them. A design team should be proactive about investigating the use of such technologies, particularly in the case where the software involved is being written primarily to gain competitive advantage.

Ideation and Storyboarding

Once the design team has a list of tasks, and is reasonably confident that they understand the domain, the constraints, and other business considerations, it is time for the team to generate the first round of design ideas.

This phase is sometimes called "ideation" because the primary purpose is to generated ideas for designs. While ideas are certainly generated at many points in the design process, this phase begins with ideas as the primary focus.

Expression of ideas via sketching

For most teams and most projects, the best way to encourage and develop ideas is in sketch form. Some design professions even consider sketching the heart of all design.

The process of sketching design ideas is usually called storyboarding. For most teams and most projects, storyboarding should be the heart of the design phase. It is usually where most of the innovation happens.

During storyboarding, the team builds rough sketches of various designs that are parts of the application, and figures out how these designs *interact* with each other in additional sketches. The key is *low fidelity*. Ideas during this phase should be expressed as sketches just detailed enough to get the idea across. They should not take a lot of time to draw.

The **low-fidelity, low-investment** approach is absolutely required to effectively build a design funnel of design ideas. Lots of ideas need to be considered at the beginning, and if each one required a lot of time, it would probably not be practical to do a lot of them.

Also, the low investment of time and effort in the ideas is important because it allows people participating in the design process to easily let go of ideas as they descend down the funnel. When someone invests lots of time in an idea, it's very easy to become emotionally involved with it. That can lead them to defend it even when it's clear to the rest of the design team that the idea just doesn't fit the goals of the project. (To be fair, this emotional attachment to ideas is always a danger during design. It's just worse if someone has invested a lot of time in an idea.)

The low-investment, low-fidelity approach is feasible because this phase is about designing interaction, as mentioned above. When a user action results in a different screen or view being presented, then a storyboard can indicate this relationship in various ways. For example, arrows from the visual element on the first view can point to the second view. Some designers like to put all views in the storyboard on different sheets or cards, and then put a note on a view to indicate that, when the action is taken, a different view should be layered on top. When presenting ideas, it's common to simply mention that when the user can take an action, and then slap the resulting view physically on top of the first one on the table.

Storyboarding multiple design options

Remember that experimental design means having multiple design ideas which compete against one another, rather than creating one design and linearly refining it. This turns out to be one of the biggest struggles for those new to the design process. It's not uncommon for a person with a technical background to come up with one idea and fixate on it. They just can't see any other way once they've seen the first way. I call this problem "solution lock".

There are several ways to avoid solution lock and get more ideas during design. One it to play a game with yourself. Pretend that your first design idea is simply not acceptable – perhaps because lawyers rejected it, or whatever. Then ask yourself: if you could not use that design approach, what other approach might work?

If the design team is big enough, it's also helpful to split the team up during ideation/brainstorming/storyboarding sessions. This often results in two different design approaches, and can result in even more if one or more teams find more than one. You can also get this effect by having individuals on the team spend time developing some ideas before a group brainstorming session.

The main thing to remember is this: **don't stop when you have storyboarded just one design idea**. Try hard to find at least one more design approach for each design task. Don't ever be hesitant to just start a brand new storyboard with a new approach, even if you already have multiple approaches. Once you have lots of design options, then you can cull some of them to focus on the best ideas.

By the way, one of the benefits of the multiple design approach is that even when one of the approaches is selected, users may find particular parts of other design approaches that they like, and which might be grafted onto the selected design approach.

Types of storyboards

The four basic types of storyboards are:

- Standalone
- Simple sequential
- Branching storyboard
- Narrative storyboard

The class will show versions of each, and will included exercises for you to draw some storyboards. The beginning of the major storyboard exercise requires you to decide, for the design challenge you have been given, what type of storyboard will best fit.

By the way, there's no rule that only one type can be used for a particular design challenge. Sometimes different aspects of the design are best addressed with different storyboard types.

Storyboarding techniques

Storyboards can be done on whiteboards, sketching on paper, sketching on index cards (which is my favorite), or even with sticky notes. The class will include materials for you to try any of these techniques.

For team based storyboarding, whiteboards and markers can also be quite effective. Don't forget to take photographs of the results for later reference, even the rejected alternatives.

For group storyboarding, I discourage what I call the “group pencil” approach. This is when only one person is actually sketching, and others are trying to get that person to sketch their ideas. All team members should be sketching during storyboarding, even if it’s on paper while someone else is at a whiteboard.

Don’t start storyboarding with a computer-based tool

While computer-based storyboarding tools can have a role in the process, and they are discussed below, I highly recommend that you don’t start with them.

Remember the objective is to generate a lot of ideas, and to do them quickly in a rudimentary form. Computer-based storyboarding tools have a tendency to suck you into investing more time in creating them, and consequently teams that I have seen that rely too heavily on these tools tend to produce fewer ideas.

The ideas that are produced also tend, in my experience, to be more traditional and less innovative. Part of the reason is that when you sit in front of a tool, you unconsciously tend to limit yourself to what you know how to do with the tool. That limitation tends to disappear with paper and pencil or other real-world, manual tools such as whiteboards and markers.

Storyboarding and wireframing tools

Certain tools, such as Balsamiq and UXPin, are designed specifically to do storyboards and early stage prototypes. Visual Studio has a “wire-frame” module which could also be considered in this category.

As I said above, I regard it as a bad idea to depend too heavily on them too early in the process. During the most intense creative/innovative phases, you want the fewest possible barriers between your idea and its tangible expression. Tools tend to put up far more of a barrier than paper and pencil, and thus can dampen the creative process.

However, some design ideas are sufficiently complex that you need to explore details and begin to explore interaction. Storyboarding tools allow some automation of the interaction pattern, so that when a user presses a button, a different part of the storyboard is automatically displayed.

Storyboarding tools are also a benefit for geographically distributed teams. Those teams are likely to move to the electronic storyboarding phase somewhat earlier simply because it’s easier to share storyboards. However, I still think the distributed individuals would be well served to do some pencil-and-paper sketching on their own to get some of their ideas fleshed out before cranking up a computer tool.

Storyboards done with a tool tend to be somewhat more fleshed out than those on paper, and sometimes contain most of the detail of the expected finished product. However, like paper storyboards, they leave out many cosmetic details, and only show the design elements in a bare form. This kind of design is often called a *wireframe* instead of a storyboard. Web designers are especially prone to rely on wireframe designs to build web sites, and many of the tools in this category are optimized to produce web applications.

Evaluating ideas at the storyboarding stage

Once your team has culled some of the designs that seem to be a poor fit, you will hopefully still have several designs that are candidates for the root of your eventual final design. It’s rare for a team to understand the user and the process so well that you can get down to just one without some additional input.

At this point, you may wish to involve users to help you evaluate your designs.

The users should understand that these are just design ideas, and none of them are "promises" about what will eventually be implemented. Fortunately, if you have stuck to good storyboarding principles and not invested too much effort in the designs, the user will see the roughness of the design sketches and intuitively assume that they are "just ideas". This is yet another reason not to put too much detail into storyboard sketches.

The users chosen should be representative, and they should understand that they don't have to narrow down to just one choice at this point. They should help you find a few viable choices.

Sometimes, though, one idea at this stage stands out above the others so much that the users lock onto it as a clear choice. That's OK. Even if that happens, though, the users may see ideas from other designs that they would like to graft onto the chosen design.

Options for developing designs past storyboards

What happens after storyboards depends on circumstances. Here are the major possibilities, and some discussion of when they make sense. More detail on each of these is included after the table.

Wireframing	When detailed layout is important to prove out a design, but precise cosmetics are not as important, then wireframing is often a good next step. Storyboard views are laid out in detail using an electronic tool. Anything from specialized tools to something as straight-forward as PowerPoint can be used.
Illustration	When visual look and feel have a high priority, an alternative to wireframing is illustration. Using a tool such as Photoshop, storyboard views are drawn in detail, and appropriate graphic work is done to let the illustration yield a very good idea of the finished production product.
Paper mockup	If interaction is important or complex, it may be helpful to do paper mockups as the next stage. Paper mockups can use either wireframes or illustrations, but they are typically a bit more crude in layout than either of those techniques. What is important with paper mockups is that various versions of each view are done, and the user can get an idea of what happens to a view as they take actions. To present a paper mockup, the presenter can run through the task flow, showing the paper version of each step. Alternatively, the person learning about the design can indicate the actions they would take, and the design presenter can then overlay the view that would result from that action.
Prototyping	If the storyboarding stage had multiple cycles and yielded detailed views, then it is possible to go straight to prototyping using a development tool. Prototypes are discussed in more detail below. (Prototypes may also be the next step after any of the three options above.)

Wireframing and Illustration

Once you have a smaller number of ideas that have made it significantly down the design funnel, you may wish to explore these ideas in a more high-fidelity fashion. Storyboards are almost always very crude, and don't include a lot of layout, and almost nothing about color/styling/etc.

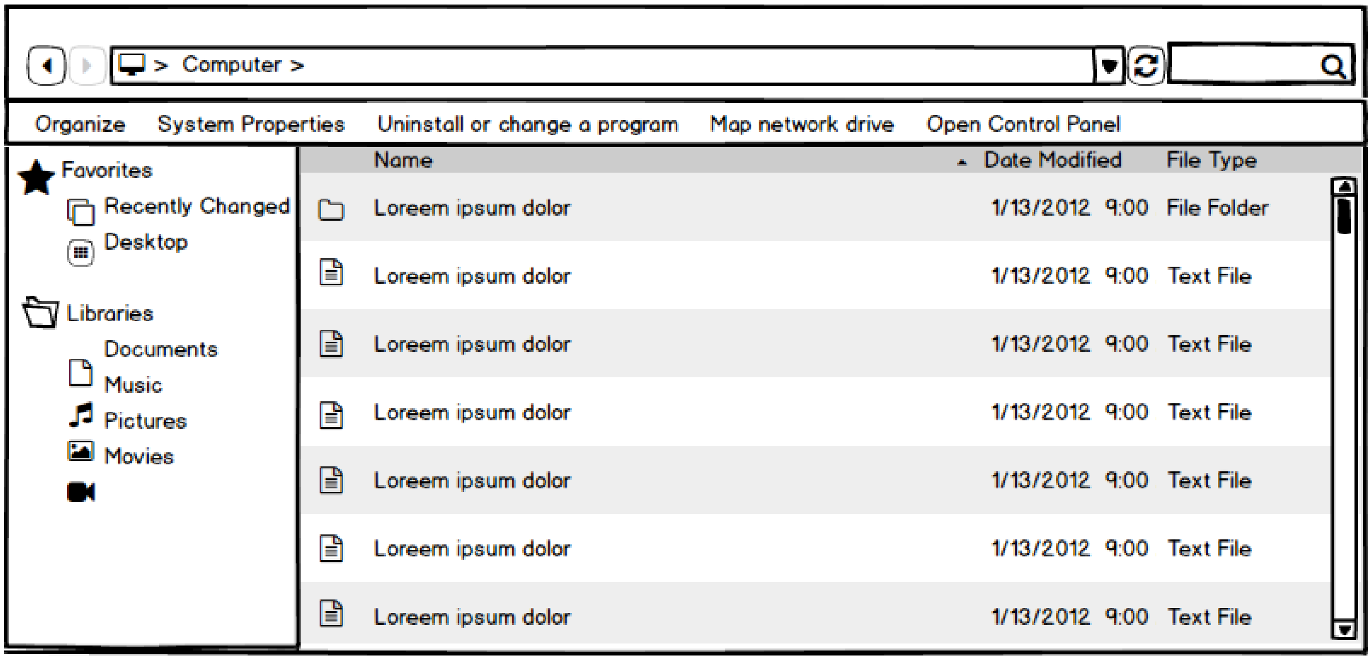
If the aesthetic appeal of the final product is a high priority for your circumstances (usually because you are creating a commercial software product), then this is the place to go into further detail on visual look and feel.

If you are working on a more traditional line-of-business application, in which aesthetics have a lower priority, a wireframe is often used instead of an illustration. Wireframing tools usually have very basic cosmetics.

Both illustrations and wireframes can contribute a lot of value by forcing the designer to accommodate all the information needed in the view. Sometimes a design that seems good in storyboards is clearly less optimal when all the fields needed are included in the design.

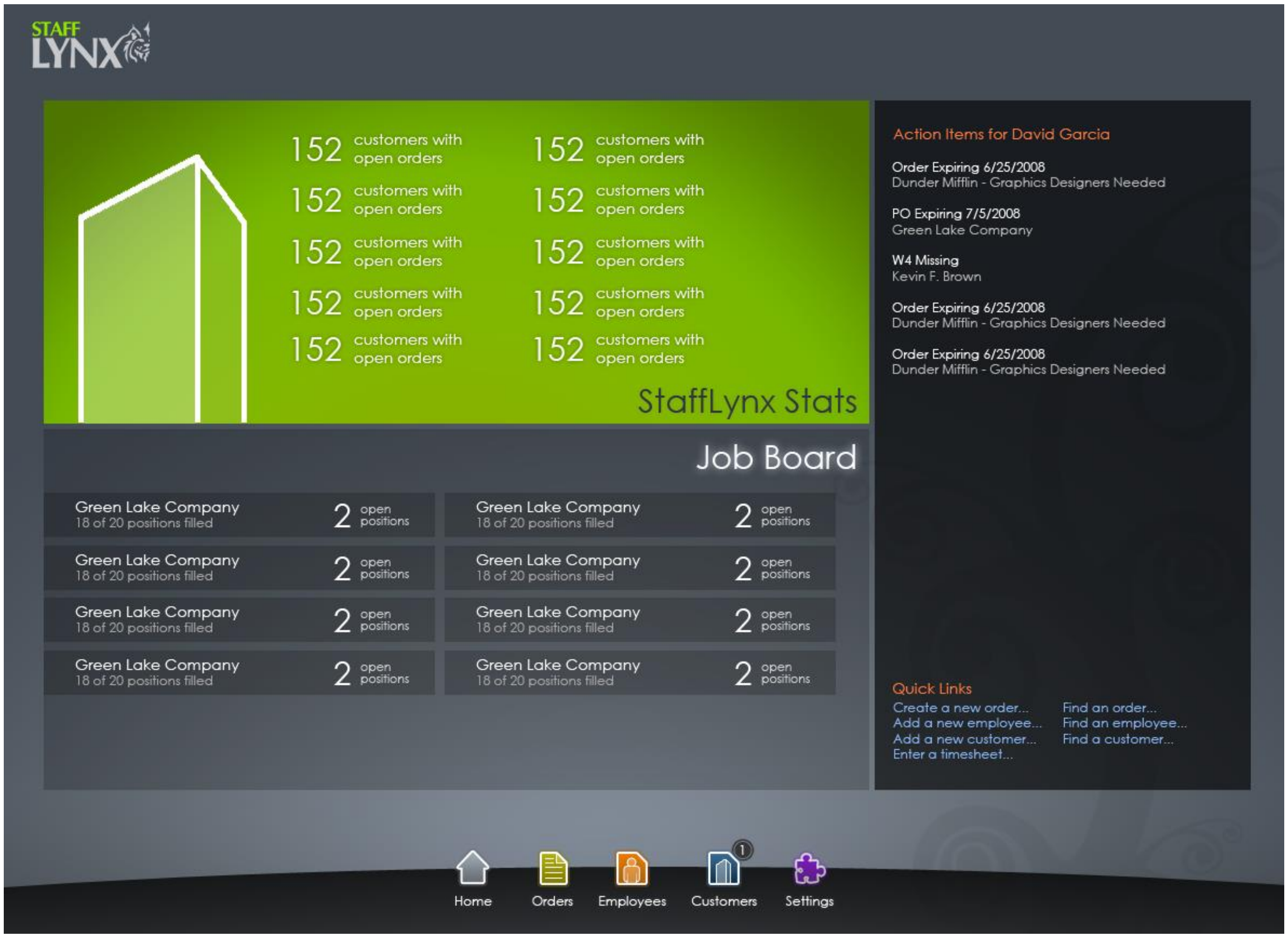
Again, the difference between an illustration and a wireframe is in the overall cosmetic look. An illustration gets much closer to the look of the expected finished product. Here are examples of a wireframe vs. an illustration:

Wireframe



Name	Date Modified	File Type
Loreem ipsum dolor	1/13/2012 9:00	File Folder
Loreem ipsum dolor	1/13/2012 9:00	Text File
Loreem ipsum dolor	1/13/2012 9:00	Text File
Loreem ipsum dolor	1/13/2012 9:00	Text File
Loreem ipsum dolor	1/13/2012 9:00	Text File
Loreem ipsum dolor	1/13/2012 9:00	Text File
Loreem ipsum dolor	1/13/2012 9:00	Text File

Illustration



152 customers with open orders	152 customers with open orders
152 customers with open orders	152 customers with open orders
152 customers with open orders	152 customers with open orders
152 customers with open orders	152 customers with open orders
152 customers with open orders	152 customers with open orders

StaffLynx Stats

Job Board

Green Lake Company 18 of 20 positions filled	2 open positions	Green Lake Company 18 of 20 positions filled	2 open positions
Green Lake Company 18 of 20 positions filled	2 open positions	Green Lake Company 18 of 20 positions filled	2 open positions
Green Lake Company 18 of 20 positions filled	2 open positions	Green Lake Company 18 of 20 positions filled	2 open positions
Green Lake Company 18 of 20 positions filled	2 open positions	Green Lake Company 18 of 20 positions filled	2 open positions

Action Items for David Garcia

Order Expiring 6/25/2008
Dunder Mifflin - Graphics Designers Needed

PO Expiring 7/5/2008
Green Lake Company

W4 Missing
Kevin F. Brown

Order Expiring 6/25/2008
Dunder Mifflin - Graphics Designers Needed

Order Expiring 6/25/2008
Dunder Mifflin - Graphics Designers Needed

Quick Links
Create a new order...
Add a new employee...
Add a new customer...
Enter a timesheet...
Find an order...
Find an employee...
Find a customer...

Wireframing tools include Balsamiq, Axure, PowerPoint, Visio, and several others. Illustration can be done with a variety of drawing/sketching tools such as Photoshop, Illustrator, CorelDraw, and Sketchbook Pro.

Don’t over-invest in wireframing and illustration, but it should be good enough that the user can get a decent visualization of the likely end product.

Prototyping

The ideas that survive through early phases often need to be prototyped to see if the expected interaction patterns actually work well with real users.

The difference between a prototype and a wireframe or illustration is that it includes interactivity. It is therefore done with a tool that allows a simulation of an application, in which the user can click certain things and see the results right on the screen.

Tools such as Sketchflow can be used for prototyping, but I find most developers can produce prototypes using their usual programming tools. I most commonly use Blend/Visual Studio to write XAML for the prototypes I do for clients.

It's OK to prototype multiple, competing designs. In fact, that has many good side effects, mentioned in the section below on evaluation.

Because you are usually producing more than one, prototypes should be hacked for the most part. **They should not be considered as a source of production code.** That leaves you free when prototyping to just throw something out that communicates the interaction idea even when you don't have time to develop a production quality version of the interaction.

For example, when I do prototypes, I rarely connect to a database to get real data. I just fudge up some representative sample data – just enough to give a good idea of how the interaction works.

What goes in a prototype?

If design is being done for a new module or feature that will fit an existing system, then the prototype should normally be done standalone with only new functionality. You can communicate to those evaluating the design that it will be activated by a user through some mechanism added to the existing system.

If you are designing a complete system or web application, often using an entirely new platform, then your prototype will have a lot more elements in it. These could include any combination of the following:

- A home screen (or landing page in a web app), which is the normal starting point for the user entering the application. It may include a lot more than just links to get to other places. It may have dashboard functionality with data visualization, alerts for the user, a list of tasks for the user to carry out, unfinished work from previous sessions, etc.
- Key views that implement the most important new designs – you may have anywhere from one to a half dozen of these, usually with at least minimal ability for the user to enter information
- Search and navigation capability – this may be integrated into other screens, or be standalone, but in most applications this is absolutely essential for judging design effectiveness
- Alternatives for major design choices
- Visualization of warnings, errors, and the like

Creating prototypes for usability testing

In a small percentage of design projects, it is necessary to ensure that the design is usable by certain classes of users. The typical prototype described above isn't usually suitable for this.

A prototype as described earlier may be extended to allow usability testing. Or if it is sufficiently crude, then a new prototype specially created for usability testing may be necessary. I find this is the case more often than not.

This prototype must spoof the functionality of key designs well enough that a user can be assigned a task to accomplish with the software, and attempt to do it. Watching users attempt key tasks helps you identify design elements that are slow or not intuitive.

Such a phase can cost significant money and time, both in creation of the usability prototype and in the actual testing process. Note that more than one round of usability testing may be necessary if the first round uncovers significant usability problems. In fact, it may be necessary to do a mini design phase to generate new designs to address a serious usability problem, and then prototype and test that design.

As such, formal usability testing should only be done for large user populations in which it is essential that the design be a good fit. In normal circumstances, if your design is generally satisfactory, adjustments made during development or in future versions can address usability concerns that arise.

Some teams do some usability testing using production applications that are still under development. This is much cheaper, and can be effective at finding and addressing minor usability concerns.

I wish we could always do a formal usability testing phase after a design process, but experience suggests that time and money considerations make such a phase impractical in most design efforts.

Evaluation

After prototype construction, you will have at least one and hopefully several designs that must be evaluated.

If you just have one surviving design, you still must validate with the users and other stakeholders that you have gotten the overall flow and key details right. However, if you only present the users with one choice, you have severely limited the impact they can have on the final design.

If you have prototypes for multiple designs, you will usually get a much higher level of input from users, and greater buy-in to the chosen design. If you present multiple ideas to the users, they do not tend to get bogged down in trivial details such as colors and minor layout issues. They look at the designs with higher engagement, deciding if each design matches their typical job flow.

After the evaluation stage, your team should have chosen a design that will now be shifted to production work. Only the design normally makes that transition; the prototype itself is likely a hacked project that isn't fit to be folded into a production project. I recommend that you always start from scratch to implement the chosen design in production.

Letting stakeholders choose

If you have more than one design option, you can arrange some kind of ranked voting to gather feedback. This should only be done after all options have gotten a fair hearing and the design evaluators have had an opportunity to ask as many questions as needed.

If you only have one design option to evaluate, you should make it clear that the evaluators have the additional option of rejecting the design as insufficient. If you have multiple designs, you can still offer them the choice of "I don't consider any of the designs acceptable". If only one person takes that choice, it's probably not enough to go back to the drawing board, but if several evaluators make that choice, it's a clear signal that you have more design work to do.

How long does all this take?

The understanding phase varies in length, depending on how much the team already knows about the problem domain and how well they know the users. However, for typical corporate applications at the department level, the understanding phase ought not take more than about two weeks, and will usually take less than a week.

User observation is usually the longest part of the understanding phase. I find that most projects need about three or four days devoted to observing users, and a day or so to evaluate and analyze the observations.

The design phase varies with the complexity of the application, and in my experience typically ranges from a week to a month. If it is taking longer than a month, then it's possible that the project needs to be broken into smaller pieces.

The two time-consuming parts of the design phase are typically storyboarding and prototyping. Storyboarding often takes anywhere from a day to three or four days at the beginning, and then it may be necessary to revisit and do more storyboarding after evaluating the first round. Remember the design funnel has that oscillating nature.

Prototyping should take no more than a few days to a week. The goal is not to produce something slick and polished. It's just to get interaction design ideas in a form that typical users will understand them and won't be surprised by the end product.

The longer design phases are seen in cases where the first rounds of storyboarding or prototyping just didn't get close enough to the needs for a final design. Further rounds of refinement take up the extra time. However, it's still a lot faster to revisit and refine at this stage than to do so after production work has begun.

Resources

The best single book for the design process is the book that contains the "design space" diagrams above. It is called Sketching User Experiences: The Workbook. You can easily find it on Amazon, but if you wish, you can use the following short link: http://bit.ly/SUEWorkbook

Another useful book for browsing interesting processes you can use during design is 101 Design Methods by Vijay Kumar. Given the variation in teams, organizations, users, and applications, having a wide variety of ideas for design processes can help you fine-tune your team's process. This class covers the process my team normally uses and presents some potential variations, but if you are interested in an even wider range of options, this book is the place to start.