An Introduction to UX Design for Non-Designers

A public class in user experience design created and taught by Billy Hollis

Executive summary

As development teams transition to creation of modern apps, exposure to design concepts, principles, and processes can help the team produce results that are more compelling, more productive, and more likely to have a long shelf life in the modern apps era. Most developers have virtually no experience with UX design, and this class was created to be their entry point.

The class is suitable for just about anyone involved in software development: developers, their managers, business analysts, visual designers, and even corporate executives in companies that furnish substantial value through technology.

When going to a new platform, without guidance the typical development team will usually revert to producing the same sort of user experiences as they produced on older technology. This class can help your organization break from the past and do better. A common sentiment among teams who have had this class is "We don't even think about software development the same way now."

Description

This class is for anyone involved in the software development cycle: developers, their managers, business analysts, project managers, and other non-designers. It's even good for executives who don't routinely work in software, because it helps them understand the importance of design and how to foster design thinking in their organization.

It is primarily aimed at helping non-designers learn the core concepts needed to do effective user experience design. The objective isn't to turn them into award-winning designers, or to have them change their careers in any major way. It's simply to inform them about how to create user experiences that are significantly better than what they've likely been producing up to now: faster to use, more intuitive to learn, and with visualization techniques to help users avoid errors and quickly find information they need.

The class includes many hands-on exercises. Some are done individually, and others are done collaboratively as part of a "design team".

The class time is broken out into the following areas:

- **How design fits into software development** – the need for design in modern apps and how various design-related tasks are integrated into a typical software development cycle
- **Design principles** – conceptual principles about what makes design good or bad, based on both historical design understandings plus the science of the human brain and visual system
- **Design process** – the steps needed for a design process, with options for handling each step so that a team can evolve a design process appropriate to their circumstances

This public class for core instruction on UX design is two days. Three and four day versions of the class are also available onsite for organizations who want in-depth instruction for their entire team. Onsite versions include additional modules on interaction and navigation patterns, platform-specific design guidance, and additional hands-on exercise.

Design Principles

The following list shows families of design principles, most of which will be covered in the class. Each principle is illustrated with examples of good and bad application of the principle. Various exercises allow students to apply principles when analyzing good and bad designs. Numerous interactive tests are sprinkled throughout this section to give users an intuitive grasp of significant design principles rather than just an academic definition.

Principles based on human visual system

Inattentional Blindness – The tendency of the human visual system to focus on certain objects in the visual field while ignoring others

Gestalt principles – grouping by proximity, grouping by similarity, filling in gaps, determining the area of focus with figure-ground relationships

Gutenberg diagram – the top-left to bottom-right pattern that the eye scans the typical rectangular surface such as a computer screen.

The human brain's preference for natural effects

Savannah effect – preference for open spaces over crowded ones

Whitespace – providing open spaces in layout to make the layout feel more natural and comfortable

Contour bias – preference for curved objects over sharp ones

Desire line – the tendency to find natural paths to get to a desired goal

Biophilia effect – preference for green, growing things

3D and layering – additional ways to enhance natural appearance or to provide mapping

Aesthetics, first impressions, and general impressions

Attractiveness Bias – human preference for attractive things

Aesthetic-Usability Effect – people rate attractive user interfaces as easier to use

Exposure/habituation effect – How impressions of a design change over time

Horror Vacui – fear of leaving screens empty because "screen real estate is valuable", and the resulting psychological effects.

Expectation effects

Expectation effects and priming – several different principles concerning the psychology of design. Most people are already familiar with the placebo effect, which is in the family. Several other members of the family have more impact on design.

(Hawthorne effect, Halo effect, etc.)

Managing conceptual load

Archetypes – universal, expected forms for certain objects

Hierarchy - understanding complexity via hierarchical grouping

Highlighting – Understanding the different ways a design can draw the user's attention to particular elements

Constraint – Building in limits to a user interface to keep users from making inappropriate choices

Entry points and affordances – giving users obvious starting points and other signals to guide them in first steps or next steps

Forgiveness – allowing the user to easily recover from inadvertent, incorrect actions

Visibility – helping the user keep track of where they are in a process (visual indication of where you are in a wizard, for example)

Wayfinding and context management – helping the user understand their current location in a complex system, and what they can or should do next

Legibility and contrast – ensuring that text, shapes, and other visual elements are capable of being easily understood by the user, without unnecessary strain

General principles for increasing productivity

Fitt's Law – concerns time for a user to locate and use an option

Hick's Law – concerns the slowdown from choosing from a list of many options

Flexibility-Usability Tradeoff

80/20 rule (Pareto Principle)

Performance Load

Progressive Disclosure – Showing users information that is layered to what they need at a given point in a task

Recognition over Recall – Less experienced users are much faster recognizing the option they need rather than needing to memorize it

Mapping – Making an interface resemble real world objects or concepts to facilitate understanding and usage

Meeting business needs

Performance vs. Preference – what the user likes doesn't always map perfectly to what the business needs

Ockham's Razor – with competing designs that accomplish a purpose, the simplest one is often the correct one to choose

Cost-benefit – design costs resources, so it must be invested where the return is greatest

Satisficing (good enough) – the objective of design is not to create a perfect solution

General and meta principles

MAYA – most advanced yet acceptable

Consistency – Designs should be consistent to promote learning and recall, but consistency should not be blindly applied to new areas that are too different to work the same way

Context sensitivity – a good design in one set of circumstances may be a bad design in a different set

Design Process

General Philosophies

A successful design process typical includes the following elements. The class will discuss all of these in depth:

- Explicitly understanding the goals of a particular design project
- Experimental design as the guiding principle, and the "design funnel" that serves as a process metaphor
- The need for collaboration and design as a team

Experimental design

Of these, the one that is most important and least understood by most traditional development teams is experiment 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.

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, the class stresses an experimental design process. You being with the intention of exploring multiple designs that are aimed at the same problem. Design projects that I lead for clients 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. Several class exercises are focused on experimental design.

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. The design process presented in this class assumes collaboration, and many exercises are done by groups of attendees working together.

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 • **User observation** • Work models and task flows • Creation of a prioritized list of design tasks
Design
• Visioning • **Storyboarding** • Illustration, wireframing and paper mock-up • Interaction prototyping • Evaluation of designs

For the understanding phase, user observation is the main activity. Extensive class time and exercise time is devoted to it, including logistics of observation, typical questions to ask, and how to factor user observations into quantifiable guidance for design.

For the design phase, storyboarding is the primary activity. Several exercises are done to give attendees practice in both the mechanics of storyboarding and the creative process behind it.

Designing for effective user interaction

A significant part of the UX design process for business apps includes designing the user's interaction flow with the software. In addition to the patterns for interaction (discussed below), some design processes discussed in the class are oriented around discovering significant work flows, prioritizing user operations, and designing user interactions that raise productivity, lower training, and reduce errors.

Designing for search and data visualization

Another area with high return on investment for UX design is creating good designs to allow users to find and use the data they need. Special content and exercises in the class deal with design search experiences, and data visualization designing to help users interpret information quickly.

Hands-on exercises

Design is a discipline, and learning it requires hands-on work. About one-third of the time in the class is spent in hands-on exercises.

Some exercises are just to help attendees assimilate the techniques of design. However, many exercises are specifically created to help developers and other technical team members break with the past and learn how to be more innovative in their design thinking. Some past attendees have praised these exercises as the most valuable part of the class for them.

Class size

The public class is expected to have 20-25 attendees. This allows considerable individual attention. As mentioned earlier, the class should contain more than developers. Anyone involved in the software development process can benefit.

Instructor

Billy Hollis has been developing software for 40 years, and he has spent much of that time focusing on client-side software and user experiences. He has been teaching developers how to design better user experiences for over ten years. Companies such as Intel, DuPont, C.H. Robinson, and many others have become more design-focused in their software development with this class, and Billy has also done over two dozen workshops on user experience at major technical conferences.

Billy's development team is responsible for applications with a worldwide reputation for innovation in user experience. You can see some of the examples at www.nextver.com.

You can reach Billy on his mobile phone at 615.400.7678.