Design: Knowing what you want to build and why

Cross-posted from http://icanhazdatascience.blogspot.com/.

I recently taught a coding course for international development students (yes guys, I'm still marking your assignments). But teaching coding isn't useful in itself - it's like teaching someone a new language without understanding what they'd want to communicate in it - so woven throughout the course was a design exercise, so students would end the course with a design for a system they wanted to build, and a tangible use for their new skills.

We focussed on user experience (UX): the process of designing a system from the user's point of view, rather than the coder's assumptions about what the user might like. And specifically on a set of UX tools that together define a system:

- personas
- user journeys
- sitemaps
- wireframes
- graphic designs
- user stories

I pointed students at two great resources on this: the beginner's guide to UX and UX Apprentice, but here's a quick run-down on the basics.

Design tools

If you're doing visual designs (wireframes, site maps etc), pen and paper is both free and easy to edit, but not so good at storing and sharing edits between people electronically. Better tools for that are Balsamiq (free for students) and Pencil (less powerful, but free for everyone). And Trello is a good tool for user stories.

Top-level design

The top-level UX elements are personas, user journeys and sitemaps.

Personas are examples of each type of user you expect to interact with your system. They're used to get to know the people who will use your system, understand the problem that they're trying to solve with it, and how people already solve that problem.

User journeys (aka scenarios: they're almost the same thing) describe how the user will solve their problem. A user journey contains elements including

- Context - where?
- Progression - what are the steps in this journey?
- Devices - mobile? laptop? smartphone?
- Functionality - what do they expect?
- Emotion - angry? tired? scared? time-pressured

A **sitemap** lists all the pages on your site, and how they're related to each other. Sitemaps can be generated by a "card sort", where you write all your system functions (or user stories) on post-its, then sort them into groups.

Page design

The UX elements for individual pages on your site are wireframes, mockups and graphics.

A **wireframe** is a deliberately rough sketch of what's on the page. It's deliberately rough to force your stakeholders (users and customers) to concentrate on *what's* on the page, not what colour it is, which images you're using etc. You might see the text "lorum ipsum" on mockups: this is standard 'nonsense' text used by designers, and can be found on sites like http://www.lipsum.com/.

A **mockup** is more detailed, and used as a proxy for the completed page in exercises like talking potential users through the functions of a site. Clickable mockups are good for this (Balsamiq wireframes are clickable, as in you can click a button on one wireframe and the code will display the page that the button goes to), as are printouts of mockups if you don't have a machine available.

Graphics are the look-and-feel of your site, its branding and things that make it seem 'familiar' to your users. This includes the color palette you use, fonts, shapes and images (icons, header images, backgrounds etc). Not everyone who codes can design graphics, so you main choice is either to find someone who can design and build all the graphic elements, or use packages of ready-made graphical elements like Bootstrap or Foundation with tools like DivShot.

When you design your pages, it helps to look at similar websites, and ask yourself what you do and don't like about them, then consider that in your designs.

Back-end design

The above has been all about designing the front-end of a system, e.g. the things that a user can see and the interactions the user has with them. Back-end design (e.g. algorithm design) is a different beastie, and one to be covered later.

Project management design

Design elements used for communicating with a coding team (including back-end coders) include user stories, kanban and minimum viable products. These are the foundation of “agile development”, a process designed around building parts of each system quickly in “sprints” (short cycles of coding and feedback) so you can get user feedback on them (and adapt your design to that user feedback) as you go.

User stories are used for planning code sprints, and look like this: as a I want to in order to

Kanban charts are those cards (or post-its) on the wall that you see in every photo of a modern software company. They usually have cards organized into 5 columns: backlog, ready, coding, testing and done, with a user story or task on each card. An agile project manager (“scrum master”) will pick the most important user stories on their list, and add them to the kanban chart at the start of each code sprint. As programmers write code for each user story, the story will move from left to right across the chart until it ends up in the “done” column.

A **minimum viable product (MVP)** is the smallest system that you can build and still get useful feedback on. You build this, get feedback, adapt your design and repeat.

That’s it: a basic run-through of design tools for coding. Now go out there and design something!