0) TITLE: Designing Device Design Deliverables

Color cover sheet on Elmo.

1) Who is here?

I want this to be interactive, so don't mind me if I wander, and feel free to yell out if I say something confusing. And I'll start with a survey: Why aren't you all learning about iPads? I mean, Apple is gonna take over the world, right?

Seriously, I want to make sure I am talking about stuff relevant to your jobs, so raise your hand if you design devices or device interface directly?

How about if you work with or for an operator/carrier on their home deck, or other customization?

Who else is left? ... why do you care about this?

DISCUSS

2) Designing Device UIs

I think this topic is important for everyone.

Because we are all designing device UIs. And that's not just overdoing the semantics -- Really thinking about interactive design holistically, contextually, means each little app is device design. Your product can change the user's perception of their whole interaction with the device. So your app or widget or site better be mobile (and when possible comply with the device design paradigms) or you are ruining the device UI.

So, you better understand what the device's native UI means.

3) Design, or Modifying Design

And, you should know what “Native” even means. For example, what I call overlays.

Have two RAZRs.

The black one is a world phone, straight from Motorola’s secret volcano factory, exactly as they intended it. The red one is a Sprint phone my dad used for years. Basically the same model, but... three softkeys, etc. Different UI.

And for those that say featurephones don’t matter, there are 2.1 BILLION of them out there. Vs. 620 MM smartphones (and only 80 MM iphones).

3a) Yes, Even for Smartphones

Have Hero, Galaxy and other Samsung, at least.

Same for smartphones. Look at the array of android devices I have here. Not the hardware, but the UI changes. Some with variations as the OS marches on, but some with OEM customization, or with Operator demands.

4) Design and Documentation

But I want to talk more about documentation than devices themselves, or design of the devices.

I have always believed there is a strong relationship between design and documentation. There's no such thing as internal (in your mind) design. A design is only valuable if it can be put on paper.

In a meaningful manner. For specifications, there's another key attribute of this. A design is not valuable unless you can communicate it to other people.

5) Converting Document Specs

About 18 months ago we got a project from a large national operator to, basically, fix their featurephone device specification. Since then we've done the same for a series of smartphones for the same people, and have written device design specs from scratch for a tablet/reader device, and so on.

I turned a lot of my previous documentation best practices into actual codified postulated on that first operator project, and confirmed them since then. Since it's /semi/ recent, as well as easy to disguise, so I'll be using it for most of this discussion.

Original device spec (redacted) with KA&A.

This was the original, historical, specification for a particular state of a handset UI. A requirement or two, and often /pages/ of tables outlining... really not very much. This is a "Key Assignments & Actions" table, covering what each hardware key and what each on-screen key, do in the particular state.

But these tables were /rife/ with errors. And most errors were inconsistencies. Needless extra confusion to the end user, and if executed needless extra work for developers. Early in this project it was obvious these had to go. It took a bit to decide what format to settle on, but eventually I settled on this:

Printout of new KA&A drawing

The vast bulk of these Key Assignments are the same across the whole interface. So, they are defined on this page up front. “Up” moves up, when there’s a list or array, for example.

When you look at a requirement, there’s a little compressed version which refers to it (at the bottom) and the little colored keys are the /exceptions/ to the default states.

What I’ve already started doing is discussing some general principles of design -- you should make up design objectives for each project, but these are general design and documentation principles.

6) First two design principles

Printout of design principles.

CLEAR – Remove ambiguity and the ability to interpret documentation in multiple ways.
CONSISTENT – Terminology and document style should be employed consistently for all requirements and documents.

7) Rest of the Design Principles

Let me explain the rest of these principles, as they apply to this same Key Assignments chart. I could use another example, but this will shorten things, and it's a pretty good example of the process.

EXTENSIBLE - Use systems, such as permanent requirement numbering, that will not expire rapidly, and do not need to change as documents are added, removed, merged, or modified.
It doesn't matter if you add or remove a key, or even if the method moves towards on-screen display. A version of this exists for softkeys. If you don't think that, say, the menu on Android is functionally a softkey, think again.

ACCURATE - Review for accuracy, but also design a process that encourages accurate documentation.

AVOIDS DUPLICATION - Design a system of writing, organizing, and storing requirements that avoids or eliminates stating requirements in multiple locations, eliminating inconsistency due to out-of-synch updates.
Remember how I said most of these tables were just repeating the default state?

REFERENCES - All references to other documents, other requirements and related tables, figures, or examples will be explicitly referenced in a consistent, repeatable, and discoverable manner.
This diagram refers back to the default conditions. And the same chart is used in other documents (the behavior of cameras is in a special document, for example) but the default conditions stay one place, in teh core document.

8) Document Consumers

But how did I get specifically to this style, or even decide these are the key principles of design documentation? By considering the consumer. Not the traditional consumer, not the end user, but the /document/ consumer.

- Here, mostly the developer.
- Now, where are they?
- Do they speak English as a first language?
- Do they work on your products full time?
- Are they new, or experienced?
- Do they use your documents, or have their own requirements management system?
- Where are they located?
- Can you talk to them?
- How do they receive the deliverables?

These are just some of the questions I actually asked. Because over the years I've developed a lot of heuristics. I understand all sorts of user types. But sometimes a new one comes up. And frankly this is a user we hadn't really examined, even though we worked with them. So, we did research. Sat down found out how their process works, at several different manufacturers.

Then just like working with different types of people, made a best-guess median consumable. With the principles above, but also tactical and operational details like:

- Images to side of text.
- Dual coding (pictures and words -- most use both, but some see pictures better, some see words better).
- Flow chart of section front.
- Glanceable documents (Pictures again help, but also diagrams for the introduction, and clear titles on each page for scanning and finding).
- Piecemeal consumption (Do not assume anyone starts at page 1 and reads to the end. Even been in meetings where development mangers literally ripped a document at the staple and gave pages and sections to his team. No one gets the whole document).

9) Componentized Design

Reader IA/Components.

You are probably bored of my key assignment chart. So, let's talk briefly about another example, not just of documentation but where design was influenced by the documentation.

Months and months into the design of the software for this reader/tablet device, we got down to some of the "boring" parts, and I had to go add battery levels, and wifi connection settings and so on. And the first cut done (by the other designers mostly) was the sort of default. A whole bar at the top of the screen with icons for everything. But it bugged me.

Not just from a page design point of view. After a few days work I realized I was being bothered by the documentation of it. The document had been built to support the componentized, re-usable view of the device UI, to encourage developers to build it that way, as well as to encourage all the design teams to keep existing components in mind.

Reader Notification with Some Settings Open.

So I realized that by adding a component we had violated this principle of the design. Even though we had executed as a "common practice," in this case it was not a "best practice." I looked fresh at the whole structure of the design and found it fit seamlessly into this notification system already built for social aspects of the service.

Reader LED settings.

Much the same happened when LED settings were developed. You know, the colored light that keeps you awake at night while your phone is charging. Besides considering context like that, I just considered how the system was already designed, and tried to slot it into an existing document structure, even though it was at first glance well outside the scope of the on-screen IA.

By considering it as part of the overall design, it became part of the overall experience. And this device doesn't have a light that blinks randomly, and you wonder what ever orange means that is different from green. On-screen behaviors are coupled to them, so it's immediately understandable.

10) Conclusion

Scientific discovery has to be reproducible, so anyone else can use the same procedure, and get the same results. John M. Barry wrote that it also has to be extensible, so others can build on it.

I hope that we're now to the point where Interaction and UX design are becoming repeatable, and even scientific. I hope that the sort of work on design, documentation and process I've shown today can be valuable to you as something not just reproducible for your work, but something you can extend, modify and improve upon to make all sorts of products better and better.

11) Questions? Comments?