Creating a User Experience Specification
Phil Oye & John Payne

Creating any system of sufficient complexity requires a diverse team and a dizzying amount of documentation. While these documents do a great job of conveying components of the system, they do not provide an integrated view. This is because each covers different aspects of the system, written by a different author for a different audience. This paper proposes that project teams should create a user experience specification, a document that shows what the system looks like, how it behaves, and how it works. This specification needs to describe the system for all team members, at a useful level of detail, in a form that encourages team members to read it and inviting enough to get them to participate in the design, as well as allow developers to build from.

Introduction

As interface designers, we are charged with creating and defining the user experience. This lofty goal is balanced with the reality that while we may design the experience, we don’t build it. Designing the most beautiful, usable, and powerful interface is useless unless it actually gets realized. Ultimately, the most important thing we produce is documentation that is (hopefully) turned into a working system. The question is how, as designers, can we produce documentation that is effective at communicating the user experience so that it does get built.

Everyone who has designed anything for someone else to build has had the pain of seeing aspects of the design lost somewhere along the path to a completed system. It is inevitable. Some of the reasons are perfectly valid, for example a newly discovered technical constraint might make part of your design impractical. However, all too often the design is simply not built as the designers intended because the entire team didn’t share a common understanding of the design.

The creation of any complex system, be it a simple e-commerce web site or a complex brokerage application, inevitably requires a dizzying amount of documentation: style guides, wireframes, screen comps, scope documents, functional requirements, functional specifications, technical specifications, use cases, executive presentations, prototypes, and more. These documents do a great job of conveying components of the system, but they do not provide an integrated view of the product. This is because each covers different aspects of the system, written by a different author for a different audience.

Enormous attention to detail is brought to bear in creating these multitudes of documents and this results in documents of enormous depth, but often with little breadth. When a technical specification runs to a 500 page Word document that describes field requirements and error conditions, all without a single illustration, it isn’t surprising that a business sponsor ignores it. Yet these business sponsors ignore these documents at their peril. Each document in the development process shares these problems–either too broad or too deep. It shouldn’t be a surprise that IT projects rarely finish on time when usually no one knows exactly what they are building.

What is needed is one document that shows what the product looks like, how it behaves, and how it works, while also being appropriate to the above audiences.

Issues with the traditional specification

The traditional functional specification is the closest thing to what we’re looking for, but in our experience it falls short. Usually, it is a really, really long Word document with a few screen shots and lots of tables and bulleted lists of requirements, everything from the mundane (“passwords must contain letters and numbers, and be at least 8 characters long”) to the essential (“this table contains the list of items that the user has to approve”). When a specification includes all the details of the system, even the obvious ones (“this is the company logo”) it becomes huge and cumbersome. It becomes useful for exactly one person, the one implementing that particular screen. Business sponsors who need to approve the design need to wade through pages and pages of irrelevant content. Inevitably they skim the content, or worse don’t read it at all, thereby missing the essential details.

The other pitfall of the traditional specification is that maintaining it becomes extremely difficult and time consuming. It is a challenge to ensure that the document is updated and prevent inconsistencies due to its size, multiple owners, and the iterative nature of software development.

Goals of the specification

A focus on usability and adoption is critical to developing any system–this shouldn’t be news to anyone. However, this focus is also critical in creating documentation for the development of these systems. Any project to create a system of sufficient complexity will have a diverse team that may include graphic designers, writers, business sponsors, programmers, and project managers. The challenge is to describe the
system for all team members, at a useful level of detail, yet without making anyone’s eyes glaze over.

Given these audiences, what is an effective specification? It should encourage business executives, engineers, and project managers to pick it up and be inviting enough to get them to participate in the design. Participation includes everything from simply understanding the design, to recommending changes, and ultimately to approving the solution. After that, developers should be able to build from it.

Let’s call this a user experience specification, but what, exactly, is it?

It can’t be too long lest your audience be too intimidated, yet it must be extensive enough to cover the entire user experience. It can’t be too boring lest your audience ignore it, yet it must communicate mundane details of the system. In short, it must capture the business rules, explain the user experience, and provide a clear picture of how the system works, all without making a marketing executive’s eyes glaze over.

A good test for effectiveness would be if a design team delivered the document, rode off into the night, and an army of programmers developed it, would the client or the design team be surprised by what was created? Would the client, upon seeing the live system, say, “Are you telling me that I approved that?”

**The keys to a effective specification**

*Be authoritative:* The specification should act as a single source for conveying the user experience. While it can and should reference other documents that support the development process, it should act as the source for defining the user experience.

*Be inviting:* The specification must encourage readers to actually read it and use their trusty red pen to make comments, suggestions, and corrections. Not to mention, actually engage in meaningful discussion about the issues at hand.

*Facilitate consensus:* All documentation created in the development process are necessarily abstractions of the final product. However, some are more abstract than others. The more “real” the depiction of the system, the greater the likelihood that people will react to the design and everyone will have a common understanding of the system.

*Be designed:* Graphic design is crucial in the design of any interface, and the specification should contain screens that incorporate it. Likewise, the document itself should also be carefully designed. Good typography and an effective layout are essential to creating a usable document.

*Embrace evolution:* The design of any system changes over time. A specification should evolve and adapt as the graphic design is created and extended, the requirements are refined, the information architecture is developed, and content is written. In other words, it starts small, but grows to become a robust document with pictures, content, and annotations.

*Provide context:* Showing screens without clearly indicating how they fit together causes people to lose the forest for the trees. Including maps and process flows to provide context of where the screen lives within the system and/or how the user arrives at the screen.

Each screen should have a unique page identifier, a title, and an overview of the page’s purpose.

*Specify most pages, but not all:* Specifying every screen or module in a system can be overkill, look for opportunities to document templates or recurring modules, rather than individual screens. It is important that all of the variation in the system is accounted for, not all the screens. This also encourages a systematic design and has the added bonus that it will make the specification more manageable. Even a system that isn’t implemented as rigid templates, multiple screens can be specified at once using conditional elements.

*Annotate the details, but not too much:* Standalone screens are almost worthless in understanding a system. The reader needs to understand how it works so annotate everything that isn’t clear from just looking at the screen. For example, what is in that dropdown? Where do I go when I click that link? Is that area conditional or static? What are the business rules to validate this form?

Don’t annotate everything. Your goal is to make a usable system, so make the screen itself as clear as possible. Remember that the end-user won’t have the benefit of these annotations. “This is the logo”, “This is the date”, and “This is where the user enters their email address” are all examples of items that the screen should make clear, rather than the would-be annotation writer.

Be sure to have your annotations actually point to the thing being annotated. Often you see a number key used to reference content, this separates the screen from the content and doesn’t encourage people to connect the two.

*Be concise (well, somewhat):* This document must fully express the experience of the system, but be eminently portable. A single document that fits in a briefcase or shoulder bag is critical. A mountain of unread documentation is useless. Keep your writing as tight as possible. As a benchmark most screens should require no more than a single page of the document, including annotations.
Support the division of labor: Many people contribute to the design of any system; the tools used to create this document should recognize this fact. Different authors are responsible for different aspects of the product, so the tools should support this.

Link to, as opposed to embed, as much content as possible. Linking to assets and content allows for distributed content creation and prevents versioning issues. No one has to remember to copy the screens into the spec each time a screen is updated. Different people can be working on the document simultaneously, and a simple “Update links” command can result in an up-to-date document that can be printed or PDF’ed. Version control programs like SourceSafe can be used to track and rollback individual files.

Linking to files allows them to be used for other purposes, such as click-through prototypes and executive presentations. It also allows the screens to be created at 100% for these standalone applications and scaled down in the specification, to make room for annotations.

Be a living document: The document should be started early in the design process. As more is known about each page, more content is added. The screens start as low-fidelity wireframes, but as graphic design is solidified the screens should be updated. Throughout the process more context and explanation is provided for each screen, additional maps and process flows are included, etc.

The form of the specification

What form should this deliverable take? While it is certainly possible to create a web-based document that meets the above criteria, we’ve found that an actual paper document is best, it is extremely portable, has a high resolution, and encourages people to write on it. Adobe InDesign, QuarkXPress, Visio, Word, or any document layout tool, are possible options. Though the more you create these documents, the more you realize that an actual desktop publishing tool works better than diagramming or word processing tools. In our studio, we use InDesign for the specification with Adobe Illustrator to create the linked screens.

Every project is unique and the elements of a user experience specification can be adapted to suit the team. At our studio, we’ve found the following to be a common table of contents for a user experience specification:

- Introduction to the document
- Goals and Requirements for the system
- High-level information architecture
- [System area n]
  - Site map/process flows
  - Screens
- Style guide

Conclusion

We make no claims that the user experience specification is the perfect solution to all of our documentation ills, but we’ve found that it becomes the reference document on projects. Business sponsors, programmers, and project managers use it as the source document because it is accessible, inviting, and always up-to-date. You know you’ve succeeded when you see a busy client executive carrying around your specification with about fifty post-it notes flagging pages. After all, it is always better to find out early that are problems than wait until the first beta.

Phil Oye
Principal
Moment Design, Inc
59 Crosby Street
New York, NY 10012
646.613.9451

Phil Oye is a founder of Moment Design, Inc in New York City and has eight years of experience designing interfaces for Web-based and desktop applications. Phil is an information architect and holds a Bachelor of Sciences degree in Cognitive Science from Carnegie Mellon University. He has worked with a variety of clients including Gucci, Tiffany & Company, Goldman Sachs, LEGO, Morgan Stanley, The New York Times, United Airlines, and the US Army.

John Payne
Principal
Moment Design, Inc
59 Crosby Street
New York, NY 10012
646.613.9451

John Payne has twelve years experience in product design, research and strategy. John has led teams of designers and social scientists to develop application interfaces, consumer products, point-of-purchase systems, retail environments, and workplace interiors. John holds a Master of Design degree from the Institute of Design at the Illinois Institute of Technology in Chicago, and a Bachelor of Industrial Design degree from Auburn University’s College of Architecture. John is also an adjunct faculty member with NYU’s Interactive Telecommunications Program where he teaches courses on Experience Design.
Company snapshot (co_snapshot)

Description: This page displays a "snapshot" or general overview of company information for any given company.

Navigation: A user clicks on a symbol in his watchlist or searches for a symbol or search string in the search field to reach this page.

Example of an annotated screen

user experience specification example prepared by MOMENT January 2004