

*The Art of Building Great User Experience in Software*



Free Sampler

# Effective UI

O'REILLY®

*Jonathan Anderson,  
John McRee, Robb Wilson  
& the EffectiveUI Team*

## O'Reilly Ebooks—Your bookshelf on your devices!



When you buy an ebook through [oreilly.com](http://oreilly.com), you get lifetime access to the book, and whenever possible we provide it to you in four, DRM-free file formats—PDF, .epub, Kindle-compatible .mobi, and Android .apk ebook—that you can use on the devices of your choice. Our ebook files are fully searchable and you can cut-and-paste and print them. We also alert you when we've updated the files with corrections and additions.

Learn more at <http://oreilly.com/ebooks/>

You can also purchase O'Reilly ebooks through [iTunes](#), the [Android Marketplace](#), and [Amazon.com](#).

---

# Effective UI

*Jonathan Anderson, John McRee, Robb Wilson,  
and the EffectiveUI Team*

O'REILLY®  
Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

## **Effective UI**

by Jonathan Anderson, John McRee, Robb Wilson, and the EffectiveUI Team

Copyright © 2010 EffectiveUI. All rights reserved.

Printed in Canada.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://my.safaribooksonline.com>). For more information, contact our corporate/institutional sales department: (800) 998-9938 or [corporate@oreilly.com](mailto:corporate@oreilly.com).

**Editor:** Steve Weiss

**Development Editor:** Jeff Riley

**Production Editor:** Rachel Monaghan

**Copyeditor:** Genevieve d'Entremont

**Proofreader:** Nancy Kotary

**Indexer:** Julie Hawks

**Cover Designer:** Karen Montgomery

**Illustration and Interior Design:**

The EffectiveUI Team

## **Printing History:**

February 2010: First Edition.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Effective UI*, the image of a rainbow lorikeet, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 978-0-596-15478-3

[F]

# Contents

<b>Preface</b> .....	<b>ix</b>	<b>3 Effective Planning and Requirements</b> ..	<b>75</b>
<b>1 Building an Effective UI</b> .....	<b>1</b>	Uncertainty and the Unknown	77
Understanding UX	4	The Humility of Unknowing	78
What Good UX Accomplishes	6	The Weakness of Foresight and Planning	79
Why Engagement and Good UX Matter	10	Friction in a Complex and Peculiar System	81
The Elements of Engaging UX	11	Subjectivity and Change	87
Redefining Two Fundamental Terms	32	Lessons from Uncertainty and the Unknown	89
Design	32	The Further You Are in the Project, the Wiser You Are	89
Development	34	Start Development As Soon As Possible	90
<b>2 Building the Case for Better UX</b> .....	<b>37</b>	Written Functional Requirements and Specifications Are Inherently Flawed	90
Why Now Is the Moment for UX	40	Commitments to Scope Are Untenable	92
Motive	40	Relish and Respect the Unexpected	92
Means	48	Intolerance of Uncertainty Is Intolerable	93
Opportunity	50	Effective Requirements	94
Winning Support for Better UX	53	How Framework Requirements Are Built	97
Stakeholders	53	Reexamining the Three-Legged Stool	99
Education	57	Commitments You Can Live Up To	101
Quantifying the Business Value	67	Effective Process	102
Materializing and Proving the Concept	67	Development Methodology	103
Other Strategies for Building Support	73		

<b>4 Bringing Together a Team</b> .....	<b>113</b>	Who Should Be Involved in the Research	182
The Project Leader	116	Finding Research Participants	184
Relationship to the Product	116	Determining the Research Sample Size	185
Relationship to the Stakeholders	117	Making Recordings	188
Relationship to the Project Team	119	Research Through Speaking with Users	190
Who Should Be the Project Leader	119	User Interviews	190
The Stakeholders	121	Structured Interview Techniques	191
Securing Authority	121	Research Through Direct Observation	193
Collaboration and Decision Making	124	Analyzing the Research Observations	196
The Characteristics of a Successful Project Team	125	Discovering Personas	196
Getting Professional Help	127	Weaving User Stories	198
Insourcing Versus Outsourcing	130	Discovering User Priorities	199
<b>5 Getting the Business Perspective</b> .....	<b>139</b>	Guerilla User Research	200
Defining Success	141	Stakeholder Buy-in Through User Research	202
Creating a Project Mission Statement	142	<b>7 Initial Product Architecture</b> .....	<b>205</b>
Determining Project Success Criteria	144	The Initial Product Architecture Team	208
Exercising Restraint	145	Contextual Scenarios	210
Applying the Pareto Principle	148	Mapping High-Level Workflows	213
What Not to Restrain	148	Sketching Low-Fi Visual Representations of Requirements	215
Refocusing Product Objectives	149	Examining Key Features and Interactions	216
Omissions Aren't Permanent	150	Setting a Style Vision	217
Describing the Product's Users	151	Developing Nomenclature	221
User Attributes	152	Technical Architecture	222
Exercises to Identify Key User Attributes	153	Getting a Lay of the Land	223
Creating Business Requirements	160	Making Platform and Framework Choices	223
Defining "Requirement"	161	Understanding Data Requirements	224
Exercises to Develop Business Requirements	163	Mapping Interactions with Other Systems	225
Maintaining Stakeholder Buy-in	169	Finding Shortcuts Through Third-Party and Open Source Components	228
<b>6 Getting to Know the User</b> .....	<b>171</b>	Discovering Business Logic	229
Valuing User Research	173	Software Architecture in Big Design Up Front (BDUF)	230
Combating Pressure to Skip User Research	175	Project Infrastructure Needs	232
Key Concepts in User Research	177	Code Source Control	232
Empathy	177	Graphic Asset Management	233
User Goals Versus Product Features and Tasks	178	Testing Infrastructure and Environments	234
Qualitative Versus Quantitative Research Methods	180		

<b>8 The Iterative Development Process . . .</b>	<b>235</b>	<b>9 Release and Post-Release . . . . .</b>	<b>263</b>
Regarding “Process”	239	Managing Expectations	265
Iterations and Feedback	239	The Alpha and Beta Releases	266
The Scope of Iterations	243	Receiving Orderly Feedback	268
Prioritizing the Subjects of Iterations	245	Last-Minute Housekeeping	269
Finishing Iterations with Something Complete	246	User Documentation	270
Estimating Iterations	247	And Champagne Corks Fly...	271
Basic Iterative Process	248	Adoption	272
Mapping Progress and Feedback Across Multiple Cycles	252	Post-Release	273
Increasing the Amount of Feedback	254	Review	274
Iteration in Sub-Ideal Project Approaches	256	Measurement and Tracking	277
Strict Waterfall Process	257	<b>Afterword . . . . .</b>	<b>281</b>
Iteration in a Big Design Up Front (BDUF) Process	261	<b>Index . . . . .</b>	<b>287</b>

development; it will also help ensure that stakeholders' expectations remain in line with the project's original goals. This will be essential in ensuring the project remains focused and isn't subject to the big course changes that can occur when stakeholders lose sight of the original problem and goals. It also helps prevent stakeholder expectations from wandering, allowing the project leader to focus on and be accountable to goals that are fixed and definite.

## Defining Success

Everyone involved in a project should be working toward its success—that much is obvious. But what does it mean to succeed? There's often a remarkable lack of clarity and consensus on this most basic of understandings. Left unguided, each participant in the project may have a completely different view of success:

- *Some project managers and stakeholders try to ensure a project is delivered on time and under budget, above any other consideration.*
- *Some stakeholders want to ensure the project succeeds in meeting their department's specific goals, but they aren't focused on how the project will affect the rest of the organization.*
- *UX professionals can focus exclusively on succeeding at meeting user needs without attending to the needs of the business.*
- *Project team members may view success as meeting the isolated demands imposed on them by managers, rather than delivering an exceptional product.*
- *Project leaders can get so focused on pleasing their stakeholders that they lose sight of the overriding quality and business goals for the product.*

If everyone is working toward different goals, it's guaranteed that those goals will come into conflict. And every narrower interest interferes with the greater interest of truly succeeding in the fullest sense. So, again, what does it mean to succeed?

Software projects are born to address business problems, to respond to business opportunities, and ultimately to drive value into the business. In short, software projects are meant to create a return on investment. It's the anticipation of that return that motivates a company to invest money into a project, and determines how much investment is appropriate. Meeting the ROI projections that were used to justify the project is the ultimate standard of success, because it is the truest reflection of the project's reason for being.

If you look at the previous list of the disparate, narrow views of success, you can imagine how each originated from the goal of helping the project meet its ROI goals. Working to ensure that a project comes in on time and on budget is an attempt to make key projections in the ROI model come true; meeting user needs is a stepping stone to meeting the business's needs. If, however, the relationship between these lower-level goals and an overriding ROI-oriented business goal is lost, the lower-level goals will pull the project in differing (and wrong) directions, and each of the narrow interests will come into conflict. Therefore, it is tremendously important to kick off the project with clarity about what success looks like, and strong unity of purpose in meeting the high-level objectives. This is the core purpose of the business planning stage.

As the project progresses, stakeholders won't be able to spend much time working on it. And as the team gets consumed in the details of designing and building the product, the focus of their day-to-day activities will be very narrow. But you cannot allow the team to lose sight of the high-level purpose of the product. And as the project progresses, even the stakeholders can forget the original purpose for the project and divert their attention to narrower goals. You must not let the product deviate from its founding business goals, because those goals were what justified the investment in the project and it is to those goals that you should ultimately be held accountable. This means that practices and mechanisms must be in place to preserve their memory throughout the project, and to ensure that everyone is working toward the same objective all along the way.

## **Creating a Project Mission Statement**

Almost every company has at some point tried to formulate a mission statement for itself. Mission statements are meant to help people keep sight of why the company is doing what it's doing and why each person is doing his work, instead of leaving people to live just in the day-to-day without a view of the big picture. Mission statements are a fixed but flexible point of reference for people to judge whether their efforts are effectively propelling them and the company in the right direction.

A mission can offer the same advantages to software projects. It's easy to get caught up in the features, design, and technology going into a product and stop thinking about why it's being built and how it fits into the larger organization.

Keeping a keen focus on the mission is especially important for projects where that mission is centered on good UX. High-quality UX is a diffuse, general goal that must ultimately be translated into a specific product with a concrete feature set. It's too easy for project team members to focus on narrow details of the implementation while forgetting that better UX is the central priority. A concise, high-level mission statement becomes part of the framework requirements. Members of the project team will refer back to it as they make decisions and judge progress, asking, "Are we being successful in fulfilling this mission?"

Mission statements are best created through direct collaboration among your stakeholders—if you can get their time for it. Mission statements should be just a few sentences or paragraphs, but those can be hard to arrive at. If you can't get your stakeholders together, prepare a first draft and pass it around among the stakeholders for feedback until you arrive at something that meets with general approval. It's again important that every stakeholder approve of the mission statement, since it will be the basis of many future decisions. The project mission will be a point of reference for them to return to as a reminder of the goals as the project progresses.

This is a mission statement from a product we've been working on that is meant to help marketing professionals reach their customers through multiple channels more easily:

*Our mission is to help businesses better communicate with their customers in ways their customers prefer and appreciate, while also helping businesses spend more time on marketing strategy instead of the logistics and tactical details of executing on their marketing plans.*

The company building this product thinks it can find success in expanding the capabilities available to marketing professionals and improve their effectiveness through high-quality UX that makes the new and existing capabilities easier to perform. The mission has a clear user orientation; it's stated in terms of the benefit the product can offer its users. This was possible because the company had drawn a clear connection between its own business success and user needs. Project missions (especially those for internal projects) can focus a bit more on the business's own needs than this one does, but not exclusively. If UX quality is a high priority for the project because it's the means by which you expect to accomplish some business goal, the project's mission should carry the UX focus.

Note also that this mission makes no attempt to define specific aspects of the solution. It doesn't say, "help businesses better communicate with their customers through email," though email channel capabilities will certainly be part of the product. The mission helps the project team decide whether certain ideas and features should be implemented. If an idea furthers the product's mission, it's included; if it offers some benefit that doesn't further the mission, it's omitted. The mission also informs design decisions. There are, for example, many ways to approach email channel marketing capabilities, but only some of them will align with the project's mission.

## Determining Project Success Criteria

Success criteria are the end of the sentence that begins with, "We will have been successful if we..." The best way of settling on success criteria is to return to the financial and business models that you used to build support for the project. The ROI proposition in the models will revolve around certain key variables, such as "percent change in customer retention" or "percent change in call center volume," that are at the heart of how the company proposes to make or save money on the project. So, the degree to which the project meets projections for these key variables will also be the degree to which it succeeds in bringing about the anticipated ROI.

Success criteria are much more specific than the mission statement, but maintain a focus on the high-level, overriding *raison d'être* of the project. They translate the key variables from the financial justification of the project into clear, explicit goals. For example:

- *Reduce call center volume 10–20 percent over a six-month period.*
- *Increase customer retention by 15 percent or more as measured over a one-year period.*
- *Reduce the incidence of data input errors by 50 percent after a six-month period.*

Each of these example goals would have arisen from some key variable in the project's ROI model. For a project to succeed in meeting its ROI objectives, it must meet projections for its key variables. Success criteria are best when they're specific and clear, readily measurable, and "timeboxed"—that is, they have a specific period over which they will be measured.

There are numerous benefits to identifying success criteria:

- *They provide another point of reference for project team members to use in decision making and give them concrete goals to aim for.*
- *They help ensure, when the project is over, that stakeholders judge the success of the product based on its original goals and mandate and not based on personal, subjective misgivings.*
- *They give people outside the project a quick understanding of how the project is meant to affect them and the company without having to understand the product itself.*
- *They amount to a commitment on your part that you can be held accountable to, which can be very helpful for your credibility.*

Once the mission statement and success criteria have been identified, agreed upon, and documented, make sure that they remain present and relevant in the day-to-day progress of the project. They shouldn't just be documents that are produced at the beginning of the project and never looked at again. Every stakeholder and member of the team should receive a printout of the mission and success criteria at their desks, and every significant decision and design concept should be filtered through them. Members of your team should be able to tell you what mission and success criteria they're working toward without having to look at notes, since they should be thinking about the mission and success criteria every day. And each time you engage stakeholders to review progress and offer advice, you need to first reorient them to the project's mission and success criteria, to ensure that their reactions and advice are framed by the project's goals.

## Exercising Restraint

As you examine the business goals, your greatest challenge is likely to be encouraging a discipline of restraint in yourself and your stakeholders. In the early days of a project there's a tremendous amount of valuable thought and enthusiasm. But left unguided and unrestrained, all of the ideas and enthusiasm can run amok. This causes projects to be founded on unreasonable goals and expectations—clearly a setup for failure. At the same time, care must be taken to exercise restraint without quashing any of the enthusiasm or discarding any of the early ideas.

Therefore, much of the work of discovering business goals is a progression of techniques that help you parlay all of the ideas and enthusiasm into business goals that are reasonable and thoughtfully restrained. The process

involves coalescing the information and value brought by your stakeholders, and getting their expectations unified and their progress pointed in the right direction. Getting the various perspectives of stakeholders to align around a common vision can be a challenge, but you should encounter less difficulty than you might expect. At the outset of a project, stakeholders are typically concerned about the risk, uneasy with the scale of and lack of clarity surrounding the problem, and unsure of how to proceed. The techniques and restraint employed during this stage, and the focused objectives that should result, do a lot to allay their uneasiness. Restraint helps eliminate a lot of the complexity and noise, which in turn helps the project seem less difficult and less risky.

Just as perfection is the enemy of the good, over-ambitiousness can set a project up for failure—failure to launch, failure to meet expectations, failure to engage users, and so on. There’s usually a strong impulse at the beginning of a project to throw in every possible feature, and to dream up a product that will appeal to every possible market or demographic. But the more sprawling the initial conception of the product is, the harder it becomes to actually build. And overlarge concepts leave too much room for stakeholders and project team members to have very different mental images of and pre-conceptions about the product.

So, in the interest of restraint, you should be continuously asking the question: “Is this truly necessary for our product to succeed for our business?” This is a useful filter for preventing guesses about aspects of the solution from being confused with actual requirements. A concept is a requirement only if the project would be considered to have failed to some degree if that concept isn’t reflected in the finished product. Surprisingly few things actually pass the test of being truly necessary; many are just ideas for features that are means to a more fundamental business goal.

Though it may seem like we’re asking you to give up hopes and dreams before you even begin, we’re just encouraging you to exercise restraint at the beginning of the project, to ensure that there will be room for those hopes and dreams when their time comes. If you strive for something too ambitious from the outset, you risk overstressing your resources. You typically don’t fully recognize you’ve overreached until the project is too far along to make adjustments and you either run over budget or underdeliver. Broader goals mean

broader susceptibility to change and risk. If you overreach in your original conception of the product, you risk finding yourself in a position of having to make cuts and compromises at the end of the project. This is costly and difficult to do; it forces you to renege on prior commitments, which damages your credibility with stakeholders.

It's much easier to add in features and refinements after you've delivered a successful, thoughtfully restrained first release (which may just be an internal demo release). By virtue of having gained the experience of having built an actual working product, you'll be much better prepared to decide which of the earlier ambitions are actually worth pursuing. You'll also be much more practiced at estimating how far your remaining resources can take you.

A restrained first version of the product also requires less time to develop, which means you can get it in front of actual users sooner. Those users will be much better judges of what's missing or problematic in the first version of the product than you and your stakeholders; you, your stakeholders, and your project team are too intimate with the product to have a clear perspective. If you exercised restraint, you'll be grateful that you have leftover time and money to make changes in response to user feedback. Also, releasing something acceptable and functional (even if it isn't 100 percent of what everyone wanted from the outset) relieves a tremendous burden of pressure and risk. It's far better to have something releasable and imperfect than something that endeavored for perfection yet never got to the point of being releasable. Too many projects die under the weight of overblown expectations and requirements. There will always be more you could have done, but for a product to see the light of day, some sacrifices will always have to be made.

The mindset of restraint, like that of the humility of unknowing, is both difficult and crucial to instill in your stakeholders. They need to understand that restraint is a key discipline of risk reduction, and that "not now" doesn't mean "never." It just means that you're keeping your options as widely open as possible and giving everyone the opportunity to gain experience before making difficult decisions. As a military commander said to the president in the TV show *The West Wing*, "A proportional [restrained] response doesn't empty the options box for the future, the way an all-out assault does."

## Applying the Pareto Principle

The Pareto principle, more widely known as the “80/20 rule,” is a useful cognitive tool for the exercise of restraint. Although some people believe it means that one should focus on the 80 percent and not worry about the other 20 percent, it actually means that the 20 percent portion is quite often the cause of 80 percent of the effect. This suggests that attending to the pivotal 20 percent is the most effective use of resources. Retailers, for example, may recognize that 20 percent of their offerings represent 80 percent of their revenue and focus their inventory and marketing investments accordingly.

With user-focused software, it’s often the case that a product built to do an excellent job of satisfying the needs of a small set of users will also work well for almost every other user. The process of describing target users therefore involves identifying the 20 percent of users who can serve as good ambassadors for the rest.

## What Not to Restrain

The practice of restraint should be a filtration and distillation of ideas that happens in a collaborative environment. It should not be a self-censorship of ideas and enthusiasm before they have a chance to enter the group conversation. Creativity and inspiration shouldn’t be headed off before they’re shared, nor should ideas be discarded before they’ve had a chance to be considered. Restraint is exercised through the process of deciding what ideas to include in the framework requirements, not when initially generating those ideas.

Also, if you’ve identified some way your product is going to significantly differentiate against the competition or drive compelling value and change to your organization, that aspect should be given wide latitude. The proper exercise of restraint should never dilute the core value and anticipated ROI of the product. Restraint is meant to keep all of the other stuff at bay so the core goals have room to breathe.

The nature of the solution you’re trying to create may demand that it be a sprawling and complicated product and greater measures of restraint just aren’t available. In cases like this, it’s useful to compartmentalize the product into smaller concepts and approach them as separate projects, to make the domain of your team’s focus narrow enough to keep them effective.

## Refocusing Product Objectives

It's typical for competing companies to try to retain their competitive edge by matching and then outdoing each other's feature lists. Product objectives are set by what will look the most impressive on a feature matrix in comparison to competitors and what marketing thinks they can message most effectively. But getting into a feature parity war with your competitors puts you on a long road to nowhere. The best you can ever hope for is to be a few months ahead of your competitors with a number of features they're already working to replicate. The intrinsic usefulness of the product suffers as it grows into a Frankenstein of fragmented features that render the product feature-complete but a nightmare to use. And the more that gets bolted onto the product, the harder it is to change, so companies get committed to a trajectory they can't control, because the inertial mass of the product is too great and simply maintaining and supporting it is consuming all their resources.

It's no wonder, then, that scrappy, focused startup products built by college dropouts in their parents' basements pose a serious challenge to previously well-established products made by large companies. Unburdened by the need to tilt against the overwhelming friction of an existing behemoth, these startups are free to build products that differentiate on quality. They focus on the intrinsic value and usefulness of the product, rather than on a long list of mostly irrelevant features. Smaller projects are also easier to design and build, because they can be held in one's head all at once more easily. And the less complexity there is in a project, the more thought and attention is available to each of its details.

If your company needs to offer a comprehensive range of capabilities in some area, consider breaking the efforts into smaller standalone products to be developed separately. Apple took this approach in building the suite of products that includes Mail, iCal, and Address Book. Those products handle many of the capabilities that Microsoft crammed into its incredibly complex Outlook and Entourage products. By breaking up the functionality into separate products, Apple was able to keep the development efforts smaller, more focused, and less prone to risk. Smaller products generally do one thing very well, whereas larger products tend to do a lot of things not as well. Smaller products also make it possible to bring the suite to market in smaller



VS



increments rather than one big one. This lets you get something out sooner, to build a base of customers and receive real-world user feedback. That feedback can be applied to improve the quality of the subsequent products in the suite that you develop.

As you investigate the business needs for your new or remodeled product, don't allow feature parity to control the conversation and product direction. If you commit to matching a competitor's feature from the outset, you've limited the resources and latitude you have to pursue other avenues of differentiation. Matching a competitor's features ultimately may be the right decision, but don't let it pass by as an unchallenged assumption. Do some research and try applying the Pareto principle to your feature list. You may discover 20 percent of your features account for 80 percent of the product's usage, or that 20 percent of your features are the reason why 80 percent of your customers use your product. There are many other ways to differentiate your product than just feature lists—better UX, for example—and those options need to be given due consideration.

## Omissions Aren't Permanent

Rich internet applications (RIAs), software-as-a-service (SaaS) deployment models, and desktop software auto-updaters allow software to be updated much more easily, so product managers can breathe much freer. Decisions to leave certain features out of a given release aren't permanent. If user feedback ends up demonstrating that a decision to omit something was a mistake, that missing feature can be added in an automatic update.

What is permanent, however, is any expenditure of time and resources that occurred before the release. This is another argument for exercising restraint in the product's objectives. Whereas you can add a feature in response to user feedback, you can't undo the expense of building one that wasn't needed or doesn't succeed. If restraint saved you from building an unsuccessful feature, the time and resources you didn't expend will be available to build something else in response to user feedback.

# Want to read more?

You can find this book at [oreilly.com](http://oreilly.com)  
in print or ebook format.

It's also available at your favorite book retailer,  
including [iTunes](#), [the Android Market](#), [Amazon](#),  
and [Barnes & Noble](#).



**O'REILLY**<sup>®</sup>

Spreading the knowledge of innovators

[oreilly.com](http://oreilly.com)