NEWFANGLED WEB FACTORY PRESENTS

THE EPIC AND DRAMATIC BATTLE

CLIENT vs. DEVELOPER WARS

BY ERIC HOLTER
CEO, NEWFANGLED WEB FACTORY

EXAGGERATED EXPECTATIONS

MISSED COMMUNICATIONS

NEGATIVE INTERACTIONS

PRODUCED BY: NON-TECHNICAL CLIENTS AND WEBSITES THAT ARE HIGHLY COMPLEX

Another High Quality Production by Newfangled Web Factory

Client vs. Developer Wars

Eric Holter
CEO, Newfangled Web Factory

Client vs. Developer Wars
Written by Eric Holter
Originally published 2002
Revised and updated 2006

First published as a Lulu Enterprises trade paperback 2006
Manufacturing by Lulu Enterprises, Inc., Morrisville, NC 27560
http://www.lulu.com

ISBN _____
LCCN _____

Printed in the United States of America.

Table of Contents

Part 1 – Communication in the Web Development Process

"The single biggest problem
in communication is the illusion
that it has taken place."

George Bernard Shaw

Foreword 2006

I'm an idea guy. I wish I could say that I'm always a good idea guy, but alas, most of my brilliant notions are better off dead. But one benefit of being an idea guy is that if you have enough of them, once in a while you get a good one. And if you can somehow learn to filter out the bad ones before wasting too much time on them (oh how I wish I could get back wasted time) you may actually come up with an idea that makes a real difference. We stumbled upon grayscreen prototyping back in the year 2000. This book will tell you all about that story and how it came about.

It is refreshing to me that this idea has persisted so long and has had such a positive impact on our web development company. It has dramatically improved our client's experiences in building their sites and also on the ultimate effectiveness of the final websites themselves.

It's also humbling that for all the complex new ideas about the web and intricate systems and applications we've built over the years, none has so improved our overall experience and success more than the simple idea of communicating about a website by using a website.

As we complete our first decade in the web business we're looking back at what we've learned and how we can build upon what we've learned as we expand our tools for the future. Our first and foremost effort reflects how important this "grayscreen" process is to everything we do. The first tool set we're improving is the prototyping system we use to communicate effectively with our clients about the structure, content and functionality of their site before design and development begin.

As boring as refining a process may seem compared to all the very cool Web 2.0 applications being innovated today I have no doubt that this is the most productive and effective area we can spend time in to enable us to serve our clients well, and deliver websites to our clients that they love.

Eric Holter
August 21, 2006

Introduction

Why I wrote this book

As the owner of a web development company since 1995, I've experienced the thrills and terrors of riding the web development roller coaster. I began this ride when my then employer Bruce Leonard (then president of Leonard/Monahan Advertising) asked me to incorporate web design into the agency's capabilities. Most of what I know about computers I learned from reading books. So I went to Barnes and Noble and bought my first book on HTML: Laura Lemay's Learn HTML 1.0 in 7 days. Soon I was building web pages with SimpleText and viewing them in Mosaic. I was hooked. I created my first website for Etonic Athletic, my second for Polaroid, both while working as a freelancer for Leonard/Monahan. I was able to parlay these jobs into many others and soon I had started my own web development company, Newfangled Web Factory. Because of my background in advertising, many of our projects came to us through advertising agencies and design firms. The first few years were fast paced and difficult. Our web projects were fraught with miscommunication, exaggerated expectations, and negative relational dynamics. Early on I assumed that this was simply a consequence of working with a new technology. Later I discovered that the problems ran much deeper than that.

The relationships between web developers and clients can often be turbulent due to miscommunication.

Up until the middle of 2000, Newfangled's development process was much like that of every other web development company. The process started with the "planning/strategy phase," followed by "design," then "programming/testing," and finally "launch/maintenance." Like most web developers, we thought our process was carefully thought out and logi-

cal. However, while the process seemed to make sense, it didn't work. No matter how hard we planned, our projects would slowly degrade and unravel. The downward path of our projects was not due to lack of effort. At one point we were creating generic screen illustrations prior to development that showed how all the proposed content, features, and functionality of a site would fit together. These documents were usually 20-40 pages long even for relatively simple sites. They took a long time to write and even more time to review with clients. In fact, we were investing so much time in planning that clients often grew impatient, wanting to "see something" instead of just discussing the site and reviewing specifications. Yet we knew that if we rushed the planning stages, we would pay for it later. Unfortunately, even when we thoroughly completed the planning stages, problems would still surface, usually in the final stages of the project.

It's been said that necessity is the mother of invention. Well, I definitely needed to do something. It was around this time that I did some analysis of my business and discovered that we regularly invested two to three times the hours budgeted on almost every project. There was no way we could double or triple our budgets (especially since we were in the middle of the dot com crash). I had to figure out a way to effectively communicate about the subtle details of a website before we got into the time intensive design and programming phases.

This book is about a wonderful discovery that transformed our web development process and saved my company. This discovery allowed us to clearly communicate the subtleties of a website to non-technical clients. Our clients "got it" and were able to confidently move through the entire development process. As a result of this simple discovery, many of the advertising agencies and design firms that we work with have become much more comfortable, confident, and profitable in offering web development services to their clients.

Who this book is for

This book is for anyone who has been, or will be, involved in developing a website. There are many parties involved in the development of a site,

some are technical, some creative, some strategic, and some managerial. I am primarily writing for those poor souls (often Marketing Directors) who are tasked with the responsibility of leading a team in getting a website built or redesigned. They will benefit from this discovery because it gives them a means of understanding the subtleties of hypertext, and the technical complexities of the web. Their projects will be greatly improved through identifying and overcoming the common barriers to communicating about the web.

I am also writing for both account executives and creative directors who often get stuck in "no win" situations as they try to meet their clients' needs for web development. These communication professionals are often caught between the exaggerated expectations of their clients and insensitive web development partners who don't grasp the complex relational dynamics and corporate politics that can govern a decision making process. Trying to mitigate these two diverse perspectives can be extremely frustrating for them. In fact, it can be so frustrating that many agencies and design firms have given up on the web entirely. The approach we put forth in this book places a high value on communicating technical complexities to a non-technical audience. Understandable technical communication is the key to resolving these frustrations.

Usually, our clients come to appreciate our process *after* we have proven its effectiveness. By contrast, our agency partners recognize its value simply through an explanation of how it works. I think this is because of two traits common to most agencies and design firms. First, they value effective communication, and second, they have become extremely frustrated with not being able to communicate effectively about their web projects. They quickly grasp the value of our process when they understand how it facilitates effective communication.

Finally, and to a lesser degree, I am writing for other developers. The benefits of this process should be obvious to them since, like us, they contend with the barriers of communicating web development every day. We have deliberately described our process in ways that can be adopted

by any developer, using generic web development tools. The core idea behind the process is technology agnostic.

The main subject of this book addresses how to communicate technical issues non-technically (for the benefit of clients who don't have technical backgrounds). Because this is our aim, the technical developer may find the lack of precise technical language disconcerting. However, this is necessary and deliberate. I would appeal to the technically literate developer to consider how important it is to be able to translate technical issues into non-technical language, especially when it comes to the web. Websites, after all, are custom software applications that are purchased by people who have never bought custom software, making them completely unfamiliar with the language of this endeavor.

How this book is organized

The primary subject of this book is communicating the web development process. I start by using a fictional story (based, sadly, in reality) that describes the common frustrating experiences many have endured when designing and building websites. The story provides a backdrop for analyzing issues of miscommunication, which is the root problem that causes web projects to break down. I then examine how exaggerated expectations and poor relational dynamics, stemming from miscommunication, complicate and compound the breakdown. Finally, I describe the breakthrough discovery of a simple method of communication that addresses the root problem and transforms the web development experience.

I discuss the principles behind this new communication process, highlighting its many benefits. I also offer some specific help and ideas about implementing the process.

The book then transitions to the subject of information design. I have debated whether this book should actually be broken up into two separate books, one on process and the other on information design. However, I believe that these two distinct subjects are so closely tied together in practice, that it is worth keeping them together in one book. Because the communication process is the primary subject, I have

only provided an overview of information and limited it to the context of the web. I believe that without a solid communication process, the skill of information design will continually run aground. Yet even a proven process, without careful attention to the principles of information design, will not result in a well-designed site. Thus the second part of the book discusses information design as a corollary subject to the primary subject of communicating the web development experience. The intent is not to be an end all discussion of website information design. There are so many better books and websites on this subject. Instead we want to cover the basics, but more to give an appreciation for the importance of information design to whomever is involved in the planning stages of website development.

Again, audience has modified my approach to the secondary subject of information design. I've addressed it primarily with non-web developers in mind. Specifically, I considered the needs of designers and art directors at our partner agencies that must exercise their skills as visual designers, within the context of the web. These talented designers are often frustrated when technical and structural issues compromise their designs. Hopefully the principles provided and practical help suggested will help make their web design experience more comfortable, enjoyable, and effective.

To this end I also write a monthly newsletter about website ad internet related topics (www.newfangled.com). The web changes fast and for the non internet oriented marketing firm, keeping up with the latest technologies is a daunting task. Our Web Smart newsletter provides digested information about the most relevant web issues written for the agency that's not daily in the stream of website and internet trends.

A woeful tale…

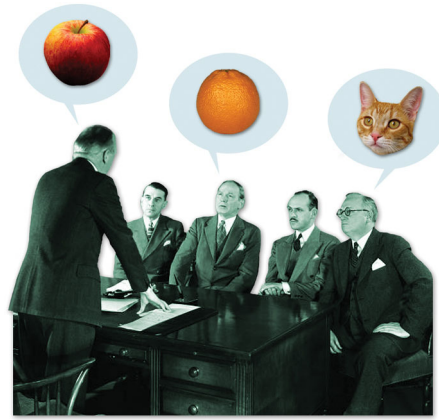
The following narrative is fictional, but just about every detail has been drawn from real life experiences. I've had the opportunity to read this narrative at a few speaking engagements. I usually get several chuckles (and even more groans) from those in the room who have been involved in web development projects. Unfortunately, these experiences, to a greater or lesser degree, are common to almost every web project. The

narrative highlights problems as a means of drawing attention to the underlying factors that cause them. The remainder of the book's first half will discuss the solution.

A Woeful Tale

The following narrative is, unfortunately, typical of many web development projects. Almost every project has at least some aspects of these frustrations. Any developer can share many horror stories about these kinds of projects.

But can the problems that plague website development somehow be avoided?



It was a little difficult to get to the bottom line because each individual from OmniTechCorp had his or her own idea of what the site should include.

Brian slammed the car door a little harder than was necessary as he returned from the client meeting late in the day. Nobody saw the gesture, but it felt good nonetheless. He was fed up with this client, with this project, and maybe this career choice. Being a project manager for Electron Cowboys, a web development company, had its benefits - like only having to wear a tie when he had client meetings. But after this project, he seriously considered throwing in the towel and finding a job with less stress.

The most frustrating part was that the project had started off so well...

Sitting around OmniTechCorp's conference table, they discussed the goals and objectives for the new website. The client group was made up of the Vice President, the Human Resources director, the head of Information Technology, and John, the Director of Marketing and primary lead for the project. Brian and a few others from Electron Cowboys had begun to gather background information, inquiring about the client's goals and

objectives. Everyone at OmniTechCorp agreed that the site needed a "new look," one that would be more "cutting edge" and "interactive." It was difficult to fully define the project because each person had his or her own idea of what the site should include. By the end of the meeting they had nailed down about 80% of their needs. The remaining 20% would be worked out later or added in a second phase of development. Armed with a basic understanding of the client's business, and a fleshed out site map, they returned to the studio to put together a proposal for the project.

Brian had written a great proposal. It was very detailed and reiterated the goals and objectives from the meeting. It established the scope of the project and estimated a budget and schedule. This was a big client and Electron Cowboys desperately wanted the job, so he had kept his estimate as tight as possible. A few days later he delivered the proposal to John.

John was both excited and worried about taking the lead on the web project. He had a lot of ideas about the new site, but since he had not been involved in the previous site's development, he didn't know exactly what to expect. Then again, who did? The web itself was relatively young, and nobody really knew what to do with it anyway. Besides, he had written an extensive request for proposal and was applying due diligence to the vendor selection. He'd worked through many other projects creating brochures, TV and radio commercials, and new brand identities. This was just another marketing tool like all the rest.

John invested a lot of time in meeting with prospective vendors. Some he ruled out as too inexperienced, others not creative enough. All of them seemed to recommend a different technical approach, but he would let IT worry about the technical issues. He didn't know an ASP from an ODBC and he didn't care to.

Electron Cowboys had made an excellent presentation. They had been around for a while and he really liked the sites they had developed. If they came back with a reasonable estimate, they would be the frontrunner for the project.

Brian and John met to review the proposal. The estimate was a little higher than John had anticipated, but he decided that they would spend the additional money, hoping that it would cover some of the "undefined items" in the proposal. The following week, John contacted Brian and told him that that Electron Cowboys had won the job and that he would sign off on the proposal to begin work on the project.

Hanging up after the phone call with John, Brian announced the new project win to the staff and a mini-celebration ensued. Having won the project, he fleshed out a rough schedule that outlined the first deliverables as the initial home page layouts. They were to be presented in two weeks. Sitting down with their designer, Abigail, he laid out the site map that went along with the proposal. She had many questions, such as "what does the client mean by "cutting edge" and "interactive"? Since Brian couldn't answer these questions he had Abigail start her layouts based on the site map and the clients existing logo.

Brian had the unenviable task of telling the development team that the client felt the designs weren't "quite there yet."

It would be easier to have the client respond to a layout then to try and get them to quantify their subjective expectations about the design. "Let's just try and pin down the look and feel at this point," he told her. With the site map in hand she developed three layouts to present to the client. Brian and the development team really liked the layouts, especially the second one, and so they posted them to the client's project page for review. When Brian called John he noted carefully that they should only be considering look and feel right now, not content or functionality. They would get to those details later.

John and the web committee looked at the layouts. They were less than impressed with the work but sort of liked layout three. The HR director liked the navigation bar on layout one, so someone suggested that they take the navigation bar from layout one and use it in layout number three. John called Brian and communicated that they weren't really comfortable with the designs. The committee felt like they were not quite "there yet."

Brian had the unenviable task of telling the development team about the client's lackluster response. "This is definitely going to take the wind out of their sails," he worried. Their disappointment was evident when they were told that the client wasn't impressed (OmniTechCorp did not even consider layout two, everyone's favorite). Abigail, the designer, had an "I told you so" look on her face because she had asked for more detail before she even began to design. Brian still didn't have much direction for Abigail to help her get closer to what the client would like. After a few more rounds of back and forth (and a lot of wasted time), the client finally gave tentative approval to a look and feel, with a few qualifications regarding content and functionality that had not yet been fully defined. Brian, frustrated with the design process, and a little irritated with the client, just counted his blessings for finally getting past the design phase. "The project should proceed much more smoothly from here," he thought. Abigail was burnt out on this client and couldn't wait to get on to something else.

John, however, was still nervous. He didn't expect the design process to take so long and, although they did finally agree on a layout, much of the content and functionality they expected in the new site was not yet reflected in the designs. Brian assured him that these details would be worked out, but John felt uneasy giving approval without a clear definition of these items.

Looking at the budget, Brian was a bit disturbed. They had gone over on the design phase by almost double their original estimate. All those extra meetings, conference calls, and rounds of design took a huge bite into the budget. Should he talk to John about this, perhaps ask for more money?

"No," he decided, "they should be able to make up some of this time in the HTML, programming, and integration phases."

Now that Abigail's layouts were approved, the project was handed off to the studio where the production staff converted the designs to HTML. While doing the conversion, the lead developer needed to talk to Brian. The layouts needed to be adjusted in order to work smoothly in Internet Explorer 5. If the layout had to be maintained in Internet Explorer 5, the coding would be much more complicated and take longer to produce. The developer also had some good suggestions to make the interface a little clearer and coding easier, but it would require a slight change in the design. Brian felt a sinking feeling that, rather than making up some time in HTML, he was about to go over budget here too. Brian nixed the design adjustments. It would be much better to take a little longer coding than to go back to the client to approve more design adjustments, especially after all they had been through to get them approved in the first place.

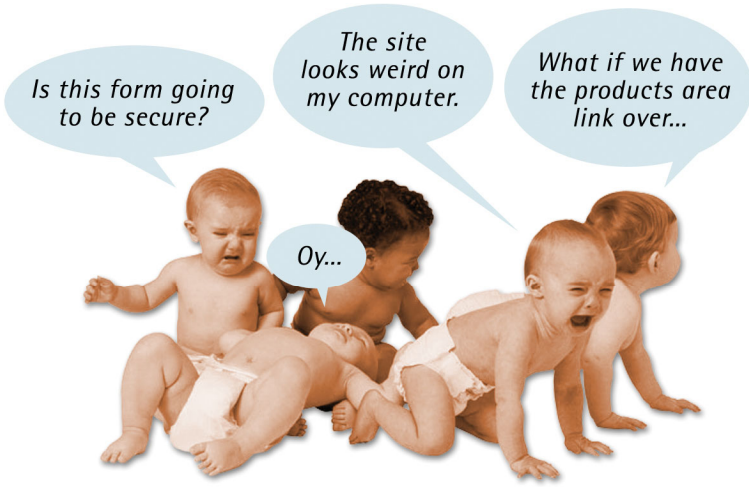
The HTML conversion was complete and Brian asked John and the web committee to look at the templates. The president of the company used an older version of AOL and the site templates "looked weird" on his browser.

Brian explained that this couldn't be fixed without compromising the design and incurring significant costs for re-coding the templates.

Brian and John finally compromised on making a few slight design modifications so that the site would look acceptable in the older AOL browser. Brian finally hinted at the possibility of going over budget based on the many rounds of design changes. Additionally, Brian reminded John that most of the content for the site was overdue and that this would affect the schedule and could also impact the budget. John was confused and angry. He didn't understand how Electron Cowboys could design a site that doesn't even work in AOL. "Doesn't everyone use AOL," he thought. "And now they want more money. We already agreed to pay more than our original budget called for! And what is he talking about 'providing content late,' I'm not even sure what the content is supposed to be!"

Brian's fears were confirmed as he reviewed the budget status. He had gone over on HTML programming by almost as much as he had gone over on design. At this point they were losing money fast on this project. He would have to get the budget increased, or else be extremely strict protecting against any further project creep. He needed to get this job done as quickly as possible to limit their losses.

The client slowly began delivering the content needed to finish the site. But each time they received content, new problems arose. The production and programming staff had many questions and were confused by what they received. Brian had to answer all sorts of questions. "What do we do with all these charts? Is this their 'brief application?' It's three pages long! What section does this stuff belong to? Are we ever going to get the photos for the product section?" Brian had to put a stop to project creep and eliminate any additional or out-of-scope work. It was time for a sit down with John to work these issues out. At the meeting Brian explained to John that they had not budgeted for a three-page form and that the content they had received was much more complex than they thought it would be. "Complex tables take much longer to code than straight-forward text," he explained. The conversation did not go well, but after a few heated exchanges, John agreed to cut back some of their content and pay more for the application form that was longer then Electron Cowboys had anticipated. Brian agreed to allow some of the longer HTML pages and complex tables that John needed in the first release. Brian got back to the shop and gave the final instructions to the development team. They could now complete the integration phase and get the site done. While he and John did come to agreement, he was still frustrated and worried about how far over budget they were - he didn't even want to look. John had to get approval for the extra money and explain to the committee why some content was not going to be on the site. He was deeply embarrassed about having to do this. But if he could just get this project over with, it would be worth it. The committee was disgusted that they were being asked to pay more, to ultimately get less than they had originally expected. The explanation of how complex tables were different from simple text did not seem to satisfy them.



Finally the site was ready for a beta release and the clients were invited to review the site before it "went live." That's when the rumbling began...

Finally the site was ready for a beta release and the client could actually click around a working version of the website. They had been waiting for this and everyone was eagerly anticipating seeing the "real thing." That's when the rumbling began. The rest of the company was invited to review the site before it "went live." John got bombarded with input. "What if we have the products area link across to a form page that will send an e-mail directly to sales, and maybe the form can embed the product numbers in the e-mail." "I thought the employee section was going to be divided up into departments."

"When will the product details be linked up?" "How come I get an error when I try and select this option on the form page?" "Is this form going to be secure?" "The site looks weird on my computer and takes too long to load." John compiled a list of edits, bugs, typos, and problems and sent them to Brian for correction.

Brian couldn't believe it. "How can they expect us to fix typos and edit copy that they provided to us in the first place!" The development team reacted to the list as well. "Where will this new section go? There is no place for it within the existing navigation bar and it adds a third level to the depth of content." "Are we supposed to rework the entire navigation

system and adjust every HTML page we've coded?" Ready to give up, Brian retreated to his office. "They'll go through the roof if I ask for more money," he thought. After thinking through all the possible things he might say to the client he finally decided to just do it and get it over with. "It would take more energy fighting than just getting it done."

Finally, the project was done. Brian reluctantly looked at the budget and was stunned to see that they had more than tripled their original time estimate. The development team was stressed as well. They had come up with some unflattering pseudonyms for the client during the process that continued to surface whenever the client's name was mentioned. Brian took solace in the fact that they had gotten the job done, and that they had a nice piece for their portfolio. Besides, they had extended themselves so far for the client that they should be able to make some of the cost up through maintenance and future projects (even though the development team would rather not work for this client ever again). He hoped that the client would be a good future reference as well.

John was also glad the project was over with. He was still angry that it went over budget and took twice as long as originally scheduled. The committee wasn't thrilled with the end result, and every time someone in the company complained about bugs John took the heat for the whole thing. Talk was already floating around that the site should be redesigned. Of course they would need to find a different developer. They were already planning on taking the maintenance of the site in-house.

Identifying the Root

> "The single biggest problem in communication is the illusion that it has taken place."
>
> – George Bernard Shaw

The root of bad development experiences

Can the problems that plague website development somehow be avoided? The answer is yes, *if the root problem is accurately identified and addressed*. Without going to the root of the problem, attempts at fixing it will be akin to treating symptoms, without healing the disease.

One of the reasons that problems persist in web development is that we often look for a technical solution to them. It can be tempting to believe the marketing slogans used by companies selling the latest web development tools. Desperation causes us to hope that these new products will somehow make developing websites easier. But since the root of the problem is not technical, technical answers won't fix it.

The root of most web development problems is not technical but rather it is the failure to *communicate technical information non-technically*. From this root two braches grow: exaggerated expectations and negative relational dynamics.

The root problem of ineffective communication

In the words of George Bernard Shaw, "The single biggest problem in communication is the illusion that it has taken place." It has been my experience that the underlying skill a web development company must bring to a project is the ability to *communicate technical information non-technically*.

I've heard developers complain about their clients, dismissively saying that, "they just don't get it." However, I believe that of all the parties involved in a web project, it is the primary responsibility of the developer to help their clients to "get it." A website is a technical system and it presents a complex information design puzzle. There are subtleties to web development that clients will not understand without careful explanation. The developer is primarily responsible for communicating the elements of interactivity, dynamic content, hypertext, information architecture, navigation systems, search engine dynamics, and browser compatibility.

These issues are not easy to communicate. It's hard enough for developers to discuss them among themselves. Communicating about the web is a big challenge, but one that must be met if the problem is to be solved at its root. When a developer takes up the challenge of communicating technical information non-technically, a solution to the problem is not far behind.

The first branch: heightened and misaligned expectations

There is no end to the hype about the web. For example, a certain technology company has run television commercials depicting an elderly craftsman checking his current worldwide inventory levels on his PDA. In another ad an e-commerce company is sitting around a computer as their new site goes live. They rejoice when their first order comes in, then they realize they have a shipping problem because orders have begun rushing in faster than the site can count them. All this propaganda builds the expectation that developing a website should be easy, that it should automatically integrate with business systems, and that results will come flooding in.

These kinds of influences cause many businesses to enter into web development projects with misaligned expectations. Such diversions prevent clients from recognizing the real ways that a website can help their business. Beneath all the hype, the Internet remains a true revolution in how people relate to information. There are clear benefits that the Internet

can bring to business. Many of the benefits are obscured by hype, but they are still there.

The weapon needed to combat misaligned expectations is clear and effective communication that replaces unrealistic expectations with appropriate and attainable ones. Unfortunately, it is miscommunication, rather than clear communication, that characterizes most web development projects. Without a way to communicate clearly, exaggerated expectations will not only persist, they will be magnified.

The second branch: negative relational dynamics

It is difficult to manage misaligned expectations. As with any relationship, unmet expectations cause tensions that produce all manner of conflicts and negative experiences. These conflicts naturally cause frustration. When this starts to happen, the relationship begins to spiral downward.

From a developer's perspective, the difficulty in communicating technical information non-technically can be intensely frustrating. Most developers are highly motivated to create websites that are distinctive and effective. Yet miscommunication can result in drastic shifts in direction, changes to features and content, and rejection of designs. Developers often refer to this tendency as "feature creep." Feature creep changes a developer's motive to excel to a motive to survive. A developer in this mode will generally criticize their client and blame them for all misunderstandings.

Feature creep changes a developer's motive to excel into a motive to survive.

Clients become equally frustrated when a project takes longer than they anticipated and goes over budget. I have encountered a number of companies that actually have really nice websites but they speak of them like they are the worst sites on the web. After digging deeper,

I usually discover that it's not so much the site, but the frustrations they experienced in developing it, that left them with such a bad taste in their mouths.

The key to solving web development issues is not in better technology. The key is in the development process. The missing piece is a process that will effectively communicate the complexities of information design between developers and their clients.

Another factor that compounds negative relational dynamics is the different values that developers and clients bring to a project. Not recognizing the gap between what developers value and what a client values will result in further negative relational dynamics. For example, as shocking as it may be to developers, most people don't value a well-coded DHTML drop down menu. They can't appreciate how hard it can be to make a navigation feature work across multiple browsers and platforms. Most people do value attentiveness, listening, and clear communication. They remember the flavor of an interaction more than the subtle coordination of complementary color palettes used in a navigation system. They remember agonizing meetings, not how well a graphic is compressed. The things a developer values, such as the quality of code, are not always things a client can see, understand, or measure. If they were to view the source HTML they could not discern whether the syntax was cross browser compatible or not. Meetings, documents, conference calls, layouts, and the final website are the elements that a client sees, experiences, and measures. Failing to effectively communicate through these things will further damage an already tenuous relationship. Each missed expectation and miscommunication ads to an increasingly sour experience. Nobody wins.

Fixing the problems

Many web development companies have packed it in after one too many bad experiences. What keeps talented, skilled, and technically competent

web development companies from being successful? How can developers create sites that meet the goals set out for them while managing the subtleties of their clients' expectations? How can they keep projects from ballooning in their costs and schedules? How can clients grow to appreciate the difficulties a developer faces in meeting these needs? How can they grasp the subtleties and intricacies of information design, database interactions, and navigation systems?

The key to solving these issues is not in better technology. The key is a better process that effectively communicates information between technical developers and non-technical clients. A solid and effective communication process can help manage client expectations and improve interactions, transforming each point where projects tend to spiral downward into an opportunity to move forward and upward. Without a cohesive communication process that addresses the problems of expectations and interaction, web development projects will continue to spiral downward.

The Discovery of the Grayscreen Prototyping Process

The discovery of grayscreen prototyping

As the owner of a web development company, I've experienced many project horror stories. In the midst of some of these projects, I was tempted to give up. Many times I was at the end of my rope with a project. Nevertheless, I maintained the business philosophy that when we encountered problems, I would not look to the client to fix them with more money, but I would go back to the drawing board and look at our process to see where we could have done a better job in communicating (this philosophy was instilled in me through a wonderful book called Selling the Invisible by Harry Beckwith). We tried many approaches and saw some marginal improvements, but there always seemed to be a threshold of control that we could not reach, making projects tend toward negative experiences rather than positive ones. Fortunately, our clients have usually ended up happy, but only because we absorbed the costs of problems. I looked at this cost as the price of an education in how to communicate effectively about web development. It wasn't until the summer of 2000 that we finally hit on something that worked, and worked well with just about every client.

We stumbled upon grayscreen prototyping.

Synthesis of three disciplines

Web development is fraught with complications because it is the synthesis of three major disciplines: design, technology, and marketing. Because of this, web development projects will often include managers from each of these disciplines. Communication between individuals from these disciplines is often difficult because they each speak their own language. Effective communication must occur for them to work together to create a well designed site.

Finding a process that works

In our various attempts to find ways to communicate with all of the parties involved in a development project, I researched and explored

many methodologies. Having a background in the advertising world, and because web development includes design and marketing, our original process was very similar to approaches used in advertising firms. Web development also shares some commonality with the publishing industry, so we explored some publishing processes as well. Web development also involves software development, introducing processes very different from those in advertising and publishing. Being new to the world of software development methodologies, I had to learn what was out there.

I became familiar with the waterfall model and the spiral method, and tried to integrate pieces of them into our process. One of the key components of any software development model is the documentation of requirements and "use cases." These documents become the technical specifications or "blueprints" that rule and guide a project.

The problem with documenting

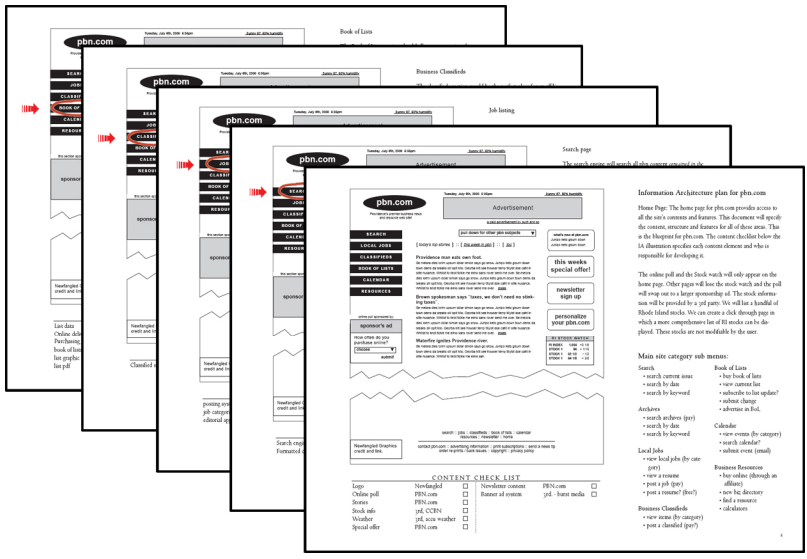
As with software development, web development required clear documentation. I looked at many examples of technical specifications in order to learn how to write them for our projects. I was boggled by the language and terminology of these specifications. I realized that such a document would not be much use to our clients. I knew that we couldn't write website specifications like other technical specifications, because our clients weren't technically trained.

This was true even when I was working for technical clients. This was because technical clients would give non-technical marketing and sales people the primary responsibility for establishing website goals and working through the development process. Their technical knowledge was not deep enough to understand all of the technical issues involved in a project. For the people without technical backgrounds, a written technical specification makes their eyes glaze over. If technical specifications were going to be helpful I had to find a way to translate them so that the non-technical could understand them.

Defining technical specs non-technically

A website is software. It is written in code, and runs on a computer. The more dynamic a site (database driven) and the more functionality it has,

the wider the divide between a clients' knowledge and the product they are buying. Not many of the clients I've worked with have had experience purchasing and developing custom software. This creates a real barrier to smooth project management. Even if a developer takes the time to plan and document the specifications for a project, the client is often unable to understand what the technical specification is saying. The end result is miscommunication, missed expectations, project creep, blown budgets, and missed launch dates.



Information architectures (sometimes called wire frames) held detailed descriptions of each web page. Although these documents were thorough, they failed to communicate how a website would work.

What didn't work

Recognizing the need for documentation written in a non-technical way, I began producing what were referred to as "information architectures" (sometimes called wire frames). These were documents that showed generic page layouts and reflected the overall structure, content and functionality of a site. Each document page would detail the content of one website page. They were visually generic, but would clearly detail the site's navigation, categories, tools, content and features. I would

represent almost every site page in this manner. On each page I wrote a description of its features. These information architectures were lengthy and took a tremendous amount of time to produce and review with the client. They helped, but we constantly found that when we were well into the design and development phase of a given project, we would still find that our clients had many unmet expectations. They had not understood (or had forgotten) what we had documented. I was stuck. I began to think that there was just no way to communicate this stuff clearly enough to avoid these problems.

Then an idea occurred to me...

HTML "grayscreen" prototyping

There is an inherent limitation when trying to communicate a nonlinear, hypertext object (a website), in a linear, non-hypertext way (paper). My idea was to try and recreate the exact same information we put into our information architectures, but create it as a functional HTML model. Would this small change alter the way a client reviews a website plan? Would it make a difference? It did, in more ways than I could have imagined.

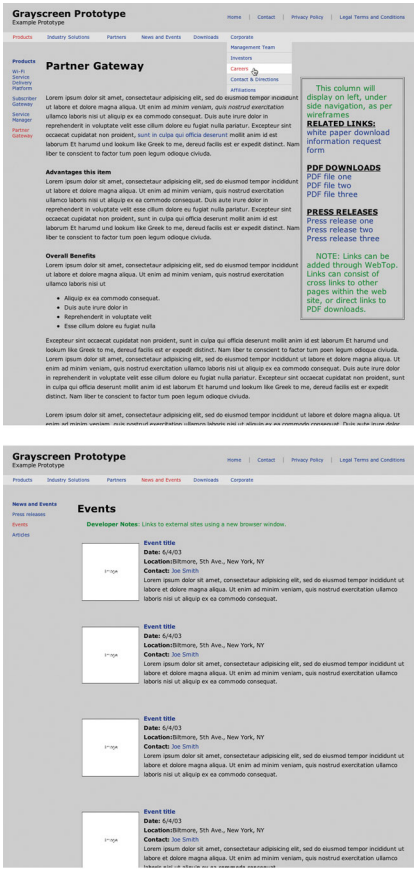
I discovered that viewing a website specification as a working HTML prototype was by far the best way to understand how the site was intended to function. This is not surprising since the web itself represents a true paradigm shift in how we relate to information. The web, unlike all other forms of communication, is non-linear. Other means of communication (books, magazines, television, etc.) are linear. Hypertext is completely non-linear. It allows for the pages in a website to be viewed in any order. A visitor might start from a sub page, perhaps following a link or a search engine query. Even when the home page is their starting point, a visitor may take one of many possible paths through a site. The structural differences between linear and non-linear information (to say nothing of the technical issues involved) can cause significant problems in communicating about the structure of a website.

Creating HTML prototypes allowed us to communicate about a project using hypertext, a medium consistent with the final product. While it was

close to impossible to describe the functions of a website on paper, actually clicking through a site model communicates very well. By building a simple HTML prototype of the site prior to the design and programming phases, we were able to create a technical *specification* and *translate* it into something everyone could understand. This prototype could then be used to *educate* and *communicate* with the client about the many detailed and complex issues of web development in a way that is clear and understandable.

The complex nature of the web, with its integrated disciplines and technical aspects, combined with an inexperienced or non-technical client results in missed expectations, miscommunication and poor interpersonal dynamics.

The grayscreen prototype is a technical specification translated into a structure most people are familiar with. The prototype then becomes an effective means of communication between the developer and the client, overcoming the barriers common to the development process.



An example of two screens from a grayscreen prototype. Prototypes are very simple and generic. They are intended to define categories, sub categories, distinguish between site sections and tools, and represent content.

The generic look helps to focus attention on structure, content, and functionality without the distraction of visual design. In the examples to the right, the main navigation bar is represented at the top of the page. This may or may not find its way into the final design of the site.

We were really on to something

Our first prototype was a desperate attempt to document our projects more effectively. We had already been writing information architectures in order to provide our clients with site details. Using an HTML model of the site instead of a printed document was simply an effort to more effectively communicate the details of a site.

As we started to use this HTML prototyping method, we discovered that it was doing so much more than documenting. It was enhancing our ability to communicate with our clients. When they had questions about how something was supposed to work we reviewed the prototype and discussed the issue in its context. It was during these interactions around the prototype that we began to realize that our clients were able to give me much more detailed feedback about how they wanted a site to work. If they had an expectation about how a particular feature would function, it would generally be discovered at this early stage. We could then deal

with the expectation appropriately. If there were technical limitations or cost issues discovered, we could work them out and provide viable alternatives before they became crises and before they had already been developed differently.

We found that building a comprehensive, simple HTML model of the site helped to solve many of my problems. The prototype also performed the function of a technical specification (we began to add programmer notes to the pages) that accurately described the structure, content, and features of our sites.

Because we translated the specification into a familiar HTML model, it was something our clients could grasp. Additionally, clients spent more time looking at prototypes. They considered them more carefully than paper documents, because they understood what they were looking at. We were able to get the kind of detailed feedback that we used to get only after a site was almost complete. This solved many of the problems that stemmed from our clients' expectations. By using the prototype to define and specify sites, we were able to communicate and educate our clients in an effective proactive manner. This improved the overall flow of information, which resulted in positive relational dynamics with our clients.

With all of the gains we experienced through grayscreen prototyping, we were able to effectively and consistently overcome the barriers that are so common to the web development process.

Benefits of the Grayscreen Development Process

Having uncovered a method of effective communication with our clients, I discovered many other benefits to our new process of grayscreen prototyping.

Enabling a deeper understanding of a site

Not only did our clients understand prototypes better than other forms of documentation, they also paid closer attention to details. The input and feedback they provided was much deeper and far more thorough having "clicked-through" a prototype. We were able to address many of the "what ifs" that inevitably come at the end of projects, before design or programming began. Often, aspects of functionality and usability do not (and cannot) occur to the client until they experience a site directly. It is impossible for a client to define every possible feature or functionality they might want at the beginning of a project. But giving them something to experience and react to will help them to understand and imagine what a site could be, at a time when changes are still painless.

Integrating a broad range of insight into a site

Before grayscreening, our development process was slow and frustrating. Getting a site map finalized and a design approved were the first steps in the process. Once the project was in HTML phase, the production staff inevitably would think of some excellent suggestions. These suggestions would have improved a site, but if they required modifications to a site design or navigation, the ideas would rarely be incorporated into a site. Typically the budget would already have been stretched and the idea of returning to a client with changes they did not initiate was akin to masochism. So this valuable and helpful insight would be wasted.

Programmers likewise would often be brought into a project after most of the site functionality was defined. Their ideas were also often overridden by budget and schedule constraints.

Since grayscreening uses a team-centered approach to web development, each team member is better informed and has a greater sense of project ownership.

The initial grayscreen prototyping process utilizes the entire development team: information designers, project managers, HTML and database programmers, and visual designers. As they review the prototype at various stages, each participant has many opportunities to provide insight and perspective, weighing in on the project before it is finalized. It allows them to ask important questions about the prototype before approval is sought. HTML and database programmers often recognize ways to simplify features. They could not provide this insight if the features were not modeled and fleshed out in a prototype.

A prototype elicits a client's insight as well. In the same way that all the different players within the developer team get to see and contribute to a prototype, a range of individuals from the client's organization get the same opportunity. The more people that review and comment on a site prototype, the more fine-tuned it will be.

Because grayscreening is a team-centered approach, each team member does a better job when it comes time for them to do their part. They are more informed about the project as a whole and they have a greater sense of ownership after having been included in the team from the beginning, and contributing to the overall process.

Effectively translating technical specifications

A prototype communicates the *structure, content* and *functionality* of a site. The functional aspects of a site are often the hardest to communicate. Miscommunication and misunderstanding of site functions can lead to the biggest problems in web development. Functional aspects of a site usually require custom programming. Custom programming is hard

to explain to non-technical people and once custom programming is completed it is usually the least flexible aspect of a site to change.

Programmer notes remind the client and the developer what will happen to specific data when it is incorporated into the website.

Mocking up site features and using programmer notes can simulate a working site in ways that other forms of modeling (site maps, written descriptions, wire frames, drawn schematics, etc.) cannot. For example, how a site will save data to a database, search a database, or create visitor profiles, can be mapped out through a prototype. These series of screens can simulate the functionality of the final site. These screens also contain programmer notes that explain what will happen to data. They might mention if it could be stored, passed on to the next page, or added to a profile. These subtleties are often overlooked and can come back later to cause significant problems. Mock ups and programmer notes are an effective way to translate technical specifications both for programmers and clients.

Encouraging and facilitating multiple iterations

A prototype allows us to rework the structure, content and functionality of a site without doing extensive, time consuming, and expensive redesign and recoding work. In many development projects it's not until a site has already been designed and programmed that a client is able to click through a site. It is only then that they notice things they wanted (or expected). To make changes at this point is not only costly, but time consuming. The end result is delayed launch dates and expanding budgets (or the developer losing money). In any case, the experience is less than satisfying for either party.

The opposite is true of prototyping. The simplicity and flexibility of a prototype allows us to go through many rounds of changes until a prototype is complete. Clients feel free to iterate and request changes to

a prototype because it is so flexible, resulting in deeper specification (that the client understands) and a greater understanding of the project.

Maximizing the skills of the designer

I believe strongly in the maxim "form follows function." Design should always revolve around a predefined function or purpose. Too often sites are designed prior to full project definition.

> "To design is to plan, to order, to relate, and to control. In short, it opposes all means of disorder and accident." – Emil Roder

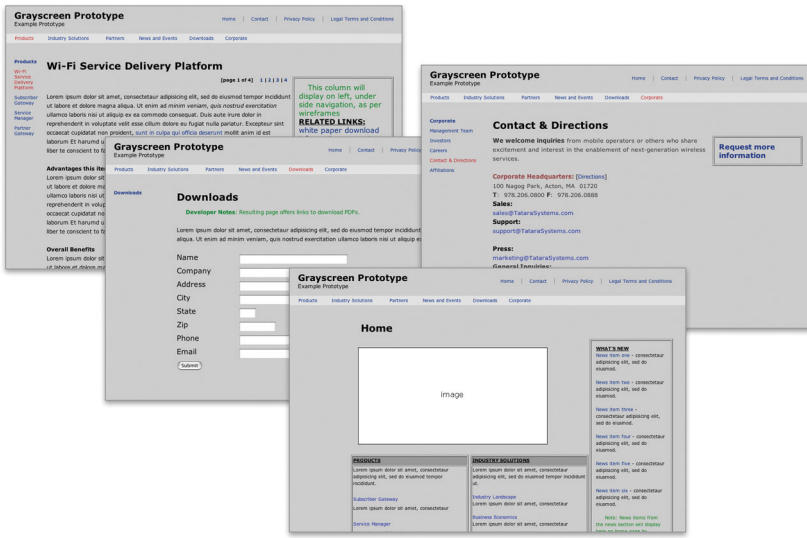
When designs are created without a final structure and function in mind, they will never be as good as they could be. When the design phase follows a fully worked out prototype, it will be more focused and specific to the defined structure, content and functionality of the prototype. Visual design that is based on clear specification will always be more appropriate and compelling than designs created prior to full project specification.

Graphic designers are trained in a visual language whose purpose is to clarify information. When most people think about design, they think about making something look good, making it aesthetically pleasing. They think about things like color, typography, layout, and images. While these are aspects of design, they are far from being the extent of the designer's work. Emil Roder wrote "To design is to plan, to order, to relate, and to control. In short, it opposes all means of disorder and accident." These words more fully describe the heart of design. Design is the opposite of disorder. It suggests intention and deliberation. Good visual designers are concerned with communication, how they can clarify a message and how they can move the eye through information in an appropriate way. Designing without specification is like laying out a brochure without

knowing what it is supposed to say. When a website is fully specified prior to the design phase, the designer can do a better job of clarifying. Without specification, all that is left for a designer to do is decorate.

Emphasizing structure, content and functionality before visual design

In the imaginary (albeit all too real) example of the woeful tale, the client was not thrilled with their layouts when they were first presented. I have found that one of the reasons clients to react negatively to a proposed



The grayscreen prototype allows us to show a client their site's structure quickly. Since the prototype is devoid of design elements, the client feels freer to restructure the site as they see fit.

layout is fear. Their fear stems from a sense that the developer has not yet fully grasped what they have in mind for their site. It is very hard for anyone to divorce their view of a layout from their actual expectations of what a site should be and do. Trying to get the "look and feel" of a layout approved, before working through the content and functionality of a site, requires the client to divorce themselves from their own expectations. It is the developer saying "trust me" before progressing far enough into the project to earn trust. This is asking too much. Any negative reaction from

the client at this point only sets the stage for further mistrust and fear (a downward spiral).

Having recognized this dynamic early on in our own experience, I realized that we needed to spend more time analyzing our clients' needs and planning the overall strategy for the sites. While this was the right thing to do, it created another negative dynamic, waiting too long to "show something" to our clients. Strategy and planning can take a long time, too long for most clients to wait to "see something." This dynamic is often what drives the premature creation of "look and feel" layouts.

I had a dilemma. If we spent an adequate amount of time planning, we risked the negative effect of taking too long to show the client something. If we showed them layouts before clearly understanding the project we would usually miss the mark. It was a real catch 22.

With grayscreen prototyping, we found a way to turn these negative dynamics into positive ones. By entering into the prototyping phase right away we were able to "show them something" very quickly. Clients felt free to change and restructure a prototype. The back and forth at this stage not only facilitated a clear understanding of our clients' needs, it also built our clients' confidence that we understood what they wanted. A prototype "showed them something" while at the same time gathering information needed for design and programming. When layouts are presented *after* the prototyping phase is complete, the client no longer worries that the developer does not understand their needs. When they do see the visual designs, they are not encumbered by concerns about structure, content, and functionality. Instead they simply respond to the designs themselves.

Facilitating content creation and delivery

Grayscreen prototyping takes problematic dynamics in web development and turns them into positive dynamics. This is also true in the area of content creation and delivery. There is a real "chicken and the egg" dilemma regarding the issue of content creation. On the one hand, designers and developers want to see as much content as possible before designing and constructing a site. Knowing exactly how long and

complex a certain form is, how many product sheets there are, or what the charts, graphs and tables look like, can offer good clues for visual design, information design, and page structure. Clients, on the other hand, need to know how long the content they are providing needs to be and how it will be used in the context of the site. They might assume that certain existing brochures or information sheets might simply drop into place on the website. However, content that is reused from other sources rarely fits into a website. At the very least the content needs to be edited so that it is appropriate for the web. This dilemma sets the stage for many small miscommunications that can cause changes and delays in projects.

> The real cost savings in grayscreen prototyping is avoiding the many problems that cause budgets to skyrocket in the first place.

Prototyping allows the client and development team to mock up a site with "dummy content," but structure it in a way that will match the purpose of the final site. This allows designers to do a good job designing appropriately to content while giving enough clues to the content provider to create content that will fit the site.

Creating appropriate content is one area of difficulty and potential confusion; actually delivering the content is also a frequent problem. Clients consistently underestimate the amount of time necessary for content creation and collection. Once a prototype is in place it can become a means of coordinating and delivering content. The specifics of how a prototype can be used in this way will be covered later.

Clarifying scope

Prototyping helps to avoid miscommunication about a project's scope, or

its complexity of content. If the developer, anticipating what a client might need on a particular page, mocks it up as they expect it to be, the client can then react to it. If the client expected more detail than a prototype represents, they can alert the developer and work it out. If there is less detail, a prototype can be adjusted as well. Making adjustments to a prototype is much easier at this early stage of development and can save incredible amounts of time, money and frustration later.

Saving time

Prototyping does add a phase to a project's schedule. However, adding the time to create an HTML prototype does not add to the overall development schedule or budget. Prototyping doesn't take as much time as you might think. By reusing templates and keeping them simple, prototypes can be created very rapidly. Whatever time prototyping does take will ultimately save more time during other phases of a project.

Increasing quality

In addition to saving time, prototyping also increases quality. Time spent working through prototyping generates new and better ideas.

These ideas surface while working through the prototyping process. When using typical development processes, ideas are discovered after it's too late (or costly) to make changes. The real cost savings in grayscreen prototyping is avoiding the many problems that cause budgets to skyrocket in the first place.

Establishing solid relationships

Prototyping not only saves the development team from wasting a lot of time and the client from frustration, it also builds trust between the two parties. While grayscreening does help to get the job done, it also improves communication, which in turn establishes positive relationships.

Hopefully the pattern is clear, that for each of aspect of grayscreen prototyping, there is a corresponding benefit that improves the relationship between a developer and client. Clients that have been educated through the prototyping process will have very few missed expectations. Most clients will appreciate how the developer has taken time to

communicate clearly and comprehensively about a project. It shows them how much the developer cares about meeting their needs and getting the site right.

The prototyping process creates an upward trend in communication that establishes solid relationships. When a developer demonstrates careful attention to the needs and expectations of a client, other inadvertent miscommunications that may occur will only be minor bumps in the road. The trust and confidence built through the communication process will enable the relationship to bear any minor problems that occur. Without the prototype process, anxiety and mistrust can turn some of the same bumps into barriers that destroy a project as well as a client/developer relationship.

Underlying Principles of Grayscreen Prototyping

The goal of grayscreen prototyping is to effectively communicate the structure, content and functionality of a website in a way that allows all of the parties involved to understand. The preceding chapter highlighted a number of benefits to grayscreen prototyping but it is important to remember that its ultimate goal is effective communication.

Creating grayscreen prototypes rapidly

Grayscreen prototypes need to be created and modified rapidly. Working through the *structure, content*, and *functionality* of a prototype will require multiple iterations, so the speed of adaptation is important to implementing the process efficiently. One of the ways to speed things up is to remember that a prototype is a working document. It does not have to be "finished" before the development team or the client can look at it for input and direction. In fact, the earlier and more frequent the input the better.

There will be many changes to a prototype as clients and developers experience a site in its prototyped state and their creative juices get flowing. In order to keep these juices flowing, modifications to the prototype need to be made quickly. The faster a prototype can move from one state to another the more the client and developer will be able to focus. Clients and developers risk losing the continuity of their thoughts as time passes between revisions. The more continuity and focus during prototyping, the better the results will be. To facilitate speed and efficiency, prototypes need to be kept simple.

Keeping them simple

The name grayscreen prototyping comes from the default gray background color. In order to revise prototypes quickly, time should not be spent making them "pretty." This means that time should not be spent formatting the display in a prototype with font styles, colors, sizes and faces. Tables and other more complicated HTML structures should be simplified and minimized. A prototype should be as generic as

possible. Graphics should almost never be used in the overall "design" of a prototype. Simple text based navigation helps keep a prototype as simple as possible.

The physical location of page elements, such as the navigation bar, is not a concern of prototyping. For example, the main navigation may display on the top of the page in a prototype while the final layout may place navigation along the side. How the navigation categories are grouped and named, and what sub-categories live under them are the concerns of a prototype. Separating main navigation from sub navigation should be reflected in a prototype. Where these pieces of navigation are placed and how they'll look on the final site is not relevant. What is important is differentiating main content areas from site utilities. Things like layout, font and color choice are not.

Keeping them non-visual

It is very important to keep prototypes non-visual. Making sure they are organized, clean, and professional is of course important, but I have found that it's far more critical not to suggest what a site might look like prior to the visual design phase. Part of the reason for this is to keep a team and client focused on issues of a site's *structure, content* and *functionality*, not its look and feel. Introducing design elements into the prototype will only confuse people. Adding design elements such as a subtle background color or typographic treatment will only distract from the primary purpose of prototyping. Besides, a designer will ultimately want to make design decisions after a site has been defined, not before. Design suggestions made before full definition are always premature.

This is why I use the term "grayscreen" prototyping. Ideally a prototype will lack any suggestion of visual direction. A prototype should be as generic as possible. At one point I considered "designing" our prototype templates so that they would look more interesting. I quickly realized that this was a bad idea because even if "the look" was just "a prototype look" and not the intended look for a site, the opportunity for the client to confuse visual suggestions in the prototype would cause them to

lose focus on the other more important issues of *structure, content,* and *functionality*.

Keeping prototypes non-visual also contributes to the speed and flexibility of their modification. As soon as graphics, backgrounds, and layouts get mixed in, a prototype becomes less flexible and more difficult to modify. If a prototype itself becomes too complicated or time consuming, its benefits will be reduced. Rather than spending time adding visual style to a prototype, time should be spent making them thorough.

Making them thorough

Any time spent adding detail that fully describes the content and functionality of a site is extremely valuable and will save time in the end of a project. A prototype should be thorough enough that anyone reviewing it will be able to easily understand a site's structure, the nature of its content, and how it will work. If anyone is confused or unclear about any of these aspects, a prototype is not finished. Ultimately the client should understand exactly how a site will be structured, what it will contain, and how it will work.

I recommend spending as much time as is necessary to thoroughly represent a site in prototype form. Taking shortcuts or breezing through the prototyping phase will ultimately add time to a project. Incomplete prototypes will require more work later. Problems that result from miscommunication require significant time to fix. It takes much longer to rework database fields, templates, and designs after the site has been completed than it does to thoroughly work through a prototype in the first place. When a prototype becomes a thorough representation of a site, the entire design and development process is made more efficient.

Another benefit of a thorough prototype is the depth of insight it elicits. The more detail a prototype contains, the more ideas, questions, and clarifications it will invoke. This avoids costly changes at the end of a project, but also improves the final site.

Prototyping Tips

Because grayscreen prototyping is the foundation of our development process, we have built many automated tools for creating and modifying prototypes efficiently. The principles of prototyping, however, are technology independent. Our first prototypes were built with straightforward HTML, some were even built using DreamWeaver. The following descriptions do not necessarily reflect how we implement grayscreen prototyping today, but rather how anyone can implement the process using whatever technology they are most comfortable with. The following tips will provide some practical help for building prototypes that will effectively communicate the structure, content, and functionality of a website.

Step one: establishing structure

The first round of prototyping usually focuses on the overall structure of a site, namely, the main site categories and subcategories. Separating main categories from site utilities (home, contact, search, etc.) is another aspect of defining structure. Identifying site features, functions, and sections is similar to developing a well worked-out site map. An HTML prototype is better than a site map, however, because it presents these elements in a more tangible and relevant way. For example, a prototype might represent the main sections of a site as "about us," "our products," "news," "partnerships," and "investors." These sections would be grouped together at the top of the page as its main navigation bar. In the upper right hand section of the page links for site utilities such as "home," "contact," "search," and "site map" can be added. These might be grouped separately from the main navigation elements to show that they would be displayed differently from the main navigation bar. Thus a prototype can describe the relationship between two different types of navigation without getting into specific issues of visual design.

Step two: representing content

Actual content for a site must be constructed once the overall structure has been worked out. Rather than using final copy for this step "dummy" copy can be used to mock up the content. The length of the "dummy" copy gives the client an idea of how long the final copy will

need to be. It can also be helpful to add a rough first sentence to dummy copy that describes its final purpose. For example, the copy on the main section page for a product area might read, "This is an initial paragraph to give the user a general overview of your product offerings. A very simple mention of primary benefits would be appropriate here, but keep it short. Lorem ipsum dolar yadda yadda yadda..." The thoroughness of this rough "place holder" copy helps the client create appropriate content for the site.

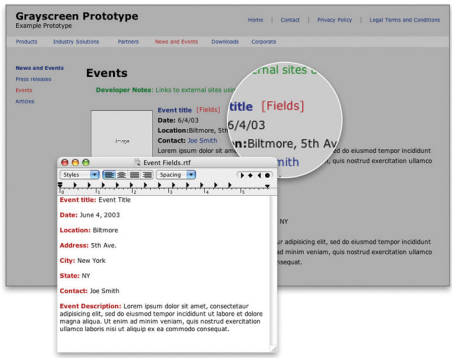
Step three: defining functionality

Sites using database driven content or other technical functions present serious communication problems due to the gap between the technical nature of these functions and a client's lack of technical background. Differences between dynamically generated and static content are often lost on clients without technical experience. In order to communicate clearly about technical functions, prototypes must translate these aspects into a form that anyone can understand.

This can be accomplished through the use of programmer notes and mock-ups. Mocking-up site features can simulate intended functionality. For example a page might show a series of form fields, pull down menus, or checkboxes that would simulate a site feature. Of course these elements would not be functional. Adding programmer notes to these screens takes the place of the actual functionality and informs the client and development team as to how a site function should work.

Practical tip: database field definition

When a website is database driven (as most sites are these days), it is critical to define database fields. Content that is generated from a database must be broken down into various fields. The client needs to understand what information will be drawn out of a database and what the related database fields will be. Adding "from database" links next to all database-generated content will help identify these instances. These "from database" links can then point to text files that list the actual database fields for a specific piece of content. This allows the client to confirm that the proposed database fields are appropriate to the content.



Programmer notes and links that lead to descriptions of database fields help the client and team to communicate clearly about how the site will work, especially when developing a database driven site.

For example, having reviewed a list of fields and having read the programmer notes, it might occur to someone that a particular piece of content needs a date field that is not listed. A date field can then be added to the prototype before the databases are built, thus avoiding a potential miscommunication. Miscommunication of database structure is the most common area where costly mistakes are made.

Practical tip: guessing

Sometimes the initial representation of site content and features requires a bit of "guess work" on the part of the developer. Ideally, sample content or data from the client forms the basis of the initial prototype. But sometimes we don't have much information to go on and have to guess.

For example, a contact form may have many fields or just basic contact information fields. The data might need to be stored, or emailed to several people. By taking a first guess at the form fields on a contact form and indicating what will happen to the information in a programmer note, the client will be able to respond to what they see and evaluate whether it meets their needs. When a client has something tangible to respond to they can more effectively refine the details of content and functionality. When details are addressed early, adjustments can be made, budgets can be altered if necessary, and conflict and wasted time can be avoided.

Practical tip: using templates

Using templates can help keep prototypes simple and malleable. Because

the layout of a prototype is not as important as the *structure, content,* and *functionality* it represents, you can save time by reusing previous prototypes or working from generic prototype templates. Revising existing prototypes or building a library of prototype templates will make the prototyping process faster.

Practical tip: using "includes"

There are some technological ways of increasing the efficiency of the prototyping process, for example the use of "includes." There are different technical means for implementing includes. In general, an include is a special piece of code on a web page that refers to another source document, and then includes the contents of that document in the web page at the point where the include code is placed. This allows for elements of a site that are consistently displayed from page to page, such as a navigation bar, to be maintained in a single location. In this way, changes to the navigation structure, which are to be expected and encouraged, can be made quickly and easily. For example, at some point in the prototyping process, a new section might be added or renamed in the navigation bar. Instead of having to adjust the navigation code on every prototype page, the navigation bar's include file is changed once, and all of the pages in the prototype then reflect the change.

Practical tip: content collection and organization

As was mentioned earlier, one of the more challenging areas for web development projects remains the creation and delivery of content by the client. Since grayscreen prototypes represent content, they help avoid the problem of receiving content that does not fit the site. A prototype can help the content collection process as well.

Once a thorough prototype exists, it can become a great content collection mechanism. One way to accomplish this is to assign a color to any text in a prototype that needs to be provided by a client. All of the instances where such content is specified in the prototype can then be listed on a content overview page. Each item can be linked to the specific prototype page where the content is needed, allowing the client to refer to its context. This list page can also contain due dates and assignments

associated with each item. This will help the client manage the content creation, editing, and delivery process.

Another way that a prototype can facilitate content collection is by placing a "mail to" link next to each area on the site where a client must provide content. These links can trigger an email window from which the client can copy and paste, or attach the specific piece of content for that page and email it to the developer. Coordination of content is facilitated by embedding the subject line of the email with the name and placement of the content (as it is labeled in a prototype). This helps identify where each piece of content belongs on the site.

As the client provides content, the color of the links on the content list page can be changed to distinguish content that has been provided from content that is still needed. This mechanism is extremely helpful to clients as they create and deliver content. The positive feelings such help produces are another example of how grayscreen prototyping not only plays an important role in getting the job done, but also creates positive relational dynamics.

Practical tip: approval process and policy

Ultimately the client needs to "sign off" on a prototype so the designers and programmers can build the working site. Since the developer is making a strong commitment to communicating about the project in detail, the client needs to be committed to understanding what is being described and giving approval to the prototype.

A prototype, once it is completed, is the blueprint from which the final site is built. At some point, the prototyping phase needs to come to a clear end, at which time the client will sign-off on the prototype. It can be helpful to include a reminder in the sign-off document that makes it clear that the approved prototype is the final specification and definition of scope for the project.

Because many things will change during grayscreening the client needs to be aware that what they are approving is the final definition of the

site. Any other documents, emails or even conversations that may have occurred are superseded by the prototype.

Practical tip: the role of the project manager

Grayscreen prototypes help improve the process of communicating the *structure, content,* and *functionality* of a proposed website. While the process itself is valuable, the role of the project manager is critical to the process running smoothly. While I am using the title project manager here, a developer may sometimes refer to this role as account manager, producer, or project leader.

The project manager is responsible for translating, communicating, and educating the client about the project using a grayscreen prototype as a communication tool. This is an important and challenging role. We have found that it is important for a project manager to have a good grasp of the underlying technology (enough to translate and educate), but not so much that they assume knowledge that the client may not have. Such a project manager will be able to explain the technical issues in a way that is highly relevant and understandable to the client.

The project manager should always make sure that the language in the programmer notes (or anything in a prototype) is simple and understandable to the whole team. Sometimes a project manager will need to reword a programmer note (with the programmer's knowledge) to clarify it for the client.

Pulling it all together

It may seem like prototyping is a lot of work - and it is. But it is far less work that has to be done when critical details are overlooked or misunderstood. The work is well worth it when as a result of solid thoughtful prototyping is a happy client and an effective website.

Part 2 - Information Design

Introduction to Information Design

The web development process is the chief subject of this book. Its corollary subject is the nature and role of information design in web development. The two subjects are inseparable. The web development process is designed to facilitate the skill of information design. The relationship between the web development process and the skill of information design is akin to the relationship between an operating room and the surgeon. The operating room (process) is an environment that contains all sorts of precise equipment. Within the room certain rules and procedures must be followed. Without the surgeon (skill), the environment and equipment of the operating room would not be very useful. Likewise the surgeon without the operating room might be able to perform some emergency procedures but would not be able to exercise the full extent of his skill.

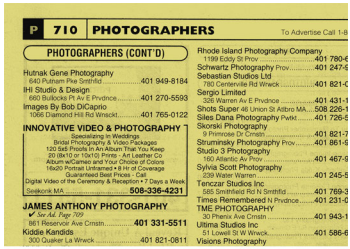
One of the primary benefits of grayscreen prototyping is that it minimizes the need for an information design specialist. While the role of the information designer is important, the grayscreen prototype process actually reduces the overall dependence a project has on the unique skills of the information designer. It does not eliminate the dependence altogether, but reduces it by extracting the information design insights of the client and the development team during the process. The more people that review and contribute to the overall information design of a site, the more likely the structure will reflect the needs of site visitors. The grayscreen process brings together a broad range of perspectives, which enrich the site's design.

A team approach to information design works because, in reality, everyone has a bit of information design talent. Some have a technical bent and others have an intuitive sense of usability. Some are more logical and can review the site prototype for proper and consistent categorization. Because prototypes are developed by groups of people, many individuals

can bear influence and contribute in unique ways that will perfect the final product.

Because of the close relationship between the skill of *information design* and the *web development process*, this section focuses on defining and fine-tuning the information design skills inherent in everyone. We are not writing to information designers, but rather to anyone who will have a role in the development of a website. This section will introduce you to the discipline of information design, touching on a few basic principles and presenting some practical tips for implementing information design. There are other helpful books on information design listed in the bibliography.

What is Information Design?


P 710 PHOTOGRAPHERS

To Advertise Call 1-8

PHOTOGRAPHERS (CONT'D)

Hutnak Gene Photography
640 Putnam Pke Smithfld 401 949-8184

IHI Studio & Design
660 Bullocks Pt Av E Prvdnce 401 270-5593

Images By Bob DiCaprio
1066 Diamond Hill Rd Wrwckt 401 765-0122

INNOVATIVE VIDEO & PHOTOGRAPHY
Specializing In Weddings
Bridal Photography & Video Packages
120 S&S Proofs In An Album That You Keep
20 (8x10 or 10x10) Prints - An Leather Cvr
Album w/Cameo and Your Choice of Colors
16x20 Portrait Unframed • 8 Hr of Coverage
Guaranteed Best Prices - Call
Digital Video of the Ceremony & Reception • 7 Days a Week
Seekonk MA 508-336-4231

JAMES ANTHONY PHOTOGRAPHY
✔ See Ad, Page 709
861 Reservoir Ave Cmstn 401 331-5511

Kiddie Kandids
300 Quaker La Wrwck 401 821-0611

Rhode Island Photography Company
1199 Eddy St Prov 401 780-6

Schwartz Photography Prov 401 247-6

Sebastian Studios Ltd
780 Centerville Rd Wrwck 401 821-0

Sergio Limited
326 Warren Av E Prvdnce 401 431-1

Shots Super 46 Union St Attbro MA 508 226-1

Siles Dana Photography Pwtkt 401 726-5

Skorski Photography
9 Primrose Dr Cmstn 401 821-7

Struminsky Photography Prov 401 861-6

Studio 3 Photography
160 Atlantic Av Prov 401 467-9

Sylvia Scott Photography
239 Water Warren 401 245-5

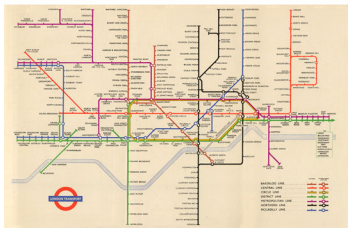
Tenczar Studios Inc
685 Smithfield Rd N Smithfld 401 769-3

Times Remembered N Prvdnce 401 231-0

TME PHOTOGRAPHY
30 Phenix Ave Cmstn 401 943-1

Ultima Studios Inc
51 Lowell St W Wrwck 401 586-6

Visions Photography



The yellow pages contain many print-based information design conventions that w e take for granted. The London underground shows information design conventions used to communicate transportation information. Simplified structures, colored routes, and circles to indicate transfer stations help to describe this complex system.

A DEFINITION OF INFORMATION DESIGN

What it is not

To define information design, it is helpful to first take a step back and determine what it is not. Information design is not simply visual design. Visual design (layouts, colors, typography and imagery) is ultimately applied to information design, but information design is an important prerequisite to good visual design. Good visual (graphic) design fits hand in glove with information design. Information design provides the framework for graphic design, and graphic design helps to further clarify the structures established by information design. However, the two are as distinct as blueprints are from the interior and exterior design elements that form a building.

A definition of website information design

Information design (as it relates to websites) is the ordering, structuring, naming (labeling), categorizing, condensing, prioritizing, and presenting of information. It begins with the details of a site's content and carefully relates them to the whole of the site. Information design encompasses information architecture (categorizing content), navigation systems (how we move from one piece of content to another) and interface design (how content is presented). Information design borrows skills from

graphic design, publishing, and interface design, combining them with those of library science. The role of the information designer is to be an advocate for the people who will ultimately use the site. The information designer must champion the cause of user experience and work to overcome the realities of individual bias, existing organizational structures, and politics that often interfere with good information design.


The New York Times

Some in G.O.P. Say Iran Threat Is Played Down

IN NEW METHOD FOR STEM CELLS, VIABLE EMBRYOS

Gaza Captors Of 2 Newsmen Pressure U.S.

Africa Adds to Miserable Ranks of Child Workers

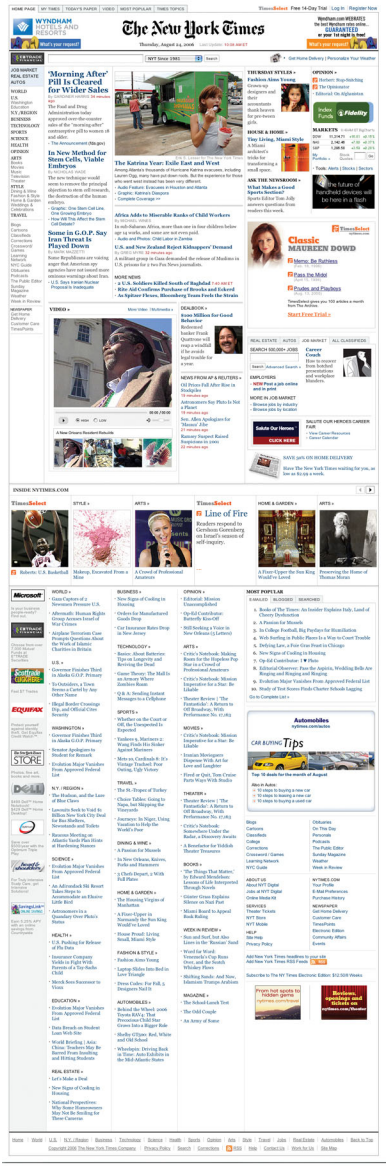
The New York Times, like all newspapers, uses information design conventions that we are all familiar with. Headlines, sub heads, sections referenced by letters and page numbers.

Comparing the print version with the online version, we can see the non-linear organization of Hyper Text systems requires a new way of structuring data and organizing information. The overall length of the online version (next page) highlights these structural concerns.

Other forms of information design

Information design, in its other forms, has been around for a long time. The practice of information design has played an important role in print design as it applies to books, magazines, newspapers, and other types of written communication. It is utilized in charts and graphs (see Edward Tufte's classic books). Information design can be seen in the signage and maps used in transportation systems.

Consider our modern yellow pages. It contains many design conventions that have been established to help us find and use vast amounts of information easily. Alphabetical sections, product and service categories, top of page topical references, topical sub-heads, indented addresses, and the alphabetical listing of names are all familiar information design conventions. Of course, we take these things for granted because they have developed slowly over hundreds of years.



Hypertext vs. linear structure

The information designer uses visual and structural design principles to simplify, organize, and display complex information. These principles are especially important with the advent of hypertext documents (i.e. web pages). Although hypertext has proven to be a wonderful invention and has contributed greatly to the rapid growth of the Internet, it has also introduced new ways in which we interact with information. The organization of printed information has evolved over hundreds of years. We are used to interpreting printed structures such as tables of contents, chapters, indexes, sections, and sidebars. In only one decade this new hypertext paradigm has arisen. The new system has very different structures, interfaces, and organizational principles than print. Every other form of publishing is linear in structure. Books and printed materials present information in a static linear way. Television and radio also present information in a linear fashion. With hypertext, people review information in non-linear ways. A single page can link to many others opening the possibility for a reader to experience a body of information in many unpredictable ways.

Dynamically generated content

The web utilizes technologies that disrupt the predictable linear presentation of information. Database driven websites, for example, allow content to be generated dynamically. Dynamically generated content means that pages can change based on an individual's preferences, the time-of-day, or many other factors. This further changes the nature of how information is experienced. Not only can a single page lead to many other pages, but the information on the page can be different from one person's experience to another's.

IMPORTANCE OF INFORMATION DESIGN

Proportional to the breadth of content

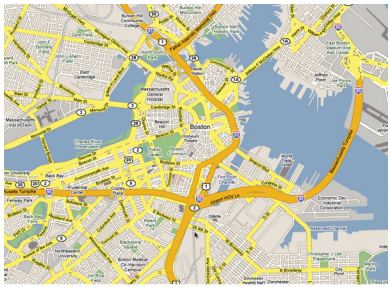
The importance of information design grows proportionately to the depth and breadth of information on a website. A small amount of information, dealing with a limited subject matter for a select audience (brochure-ware websites), needs less information design than a large site filled with hundreds or thousands of pages and covering many subjects for a diverse audience. The problems inherent in such vast sites multiply over time. The investment made in organization, navigational systems, and user interface become increasingly valuable as the complexity of a site grows.

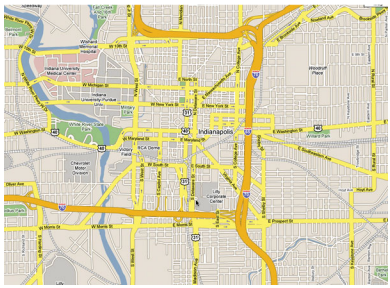
The Big Dig

Information design is most valuable at the outset of a significant web-based project. It is much harder to fix information design problems after a site has grown to include hundreds or thousands of pages than it is to structure the site properly when it is first built. A non-Internet example of how information design problems can grow exponentially is the "Big Dig." In New England, anyone who has traveled through Boston is aware of the massive engineering project known as "The Big Dig." A new and complex system of tunnels was constructed to run directly beneath the city. It marks the greatest civil engineering project ever undertaken. At every stage of the project, an incredible amount of work needed to be done re-routing traffic and utility systems, building temporary

structures, destroying old structures, and finally constructing sections of the new tunnel.

The complexity of the project results from the complexity of existing systems that have been built up slowly over time. Imagine if someone with a prophetic gift was able to foresee this project hundreds of years ago. The city could have kept a clear route free from other development so that the tunnel could be easily constructed. Of course it is impossible for anyone to have guessed such a project would be necessary such a long time ago. However, when it comes to websites we should be able to plan for the future, knowing that a site will grow in depth and breadth of content. Information can be structured in ways that assume growth. The more flexible and well thought out the initial design, the less work (destroying and rebuilding existing pages and structures) will need to be done in the future.





Information design needs to establish a clear structure that allows users to "get around" intuitively over time as more and more "streets and buildings" are added to the site.

A good example of this is the comparison of the metropolitan layouts of Boston and Indianapolis. The grid-like structure of Indianapolis' streets provides more intuitive navigation.

Boston vs. Indianapolis

Consider the map of Boston and think about how hard it is to get from one end of the city to the other. Imagine having to give driving directions to someone who is not already familiar with the city. Now think about giving directions in a more modern city such as Indianapolis. Proactive city planning has based the city's layout on a grid structure. Terms like "West North Street" and "East North Street" actually have some relationship to where you are in the city. It's fairly easy to figure your

way around the city, even if you've never been there before. Similarly, information designers need to establish a clear structure that allows users to "get around" a website intuitively over time even as more and more "streets and buildings" are added to it.

Internet example: the university website

Information design becomes critical as the depth of a site's content grows. Online publications and university sites are examples of websites with vast content. Failure to establish a solid, intuitive information design can cause an entire site to fail.

University websites are examples of particularly information-rich websites. They often contain hundreds or thousands of pages, all developed and maintained by multiple parties and departments. Maintaining consistent categorization, navigational structure, and sub-page layouts is a huge challenge. Often content is shared or duplicated from one section to another. It is not uncommon for a user to follow a set of sub-links within one area and arrive at a completely different site. Many universities have given up on trying to organize their sites effectively and instead have opted for a glorified site map as the main home page.

The complexities of such large sites are not easily overcome (politics and turf wars usually add to the struggle). The more content a site contains, the more critical it is to carefully work through the information design process using an inclusive, team-centered prototyping methodology.

Five Principles of Information Design

Gaining an understanding and appreciation for information design sensitizes us to the need for clear structure and organization. These five information design principles will be helpful to keep in mind as you work through the process of defining the structure, content, and functionality of a website.

Principle One: Don't Construct Decoration

Architects Robert Venturi, Denise Scott Brown, and Steven Izenour coined the principle, "It is all right to decorate construction, but never construct decoration."

Visual design vs. information design

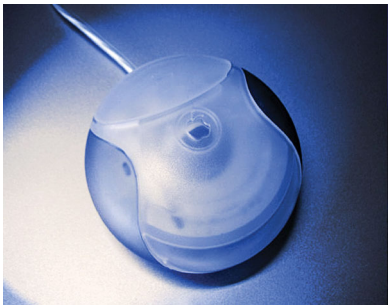
It is important to separate in our minds the distinction between information design and visual design. These two disciplines certainly meet in the final product of a website, but they are clearly distinct in the design process. Before visual design can be addressed, the right questions have to be asked, problems must be defined, and overall planning must be completed. Through these processes the information designer builds a deeper level of understanding about how a site and its various functions fit together. Finally, once the construction of a site is complete, the visual designer can add decoration and clarity to the structure. Good visual design will further refine the clarity of the information. There will be problems however, if the visual design process preempts the planning process.

Many web projects end up derailing when they move too quickly into visual design. If you find yourself choosing a color palette for a site before you have fully planned out its structure, content, and functionality, you are failing to properly complete the information design process. As a result you will usually end up constructing decoration rather than decorating construction. Visual design will never do the

job of clarifying information if the overall information design has not preceded it.

Apple's ill-conceived "hockey-puck" mouse

Consider Apple's ill conceived "hockey puck" mouse that came standard with their iMac computers. The mouse was redesigned as a circle rather than an oblong shape. This failed to consider the interaction of the user who cannot tell how the mouse is oriented in their hand without feeling around for the button. A person using the mouse would move the mouse left and the cursor would go up because they could not tell that the mouse was facing the wrong way. Here Apple constructed a decoration rather than decorating an existing (and superior) form. Visual design can add clarity to a designed structure, but it should never dictate what that structure should be.

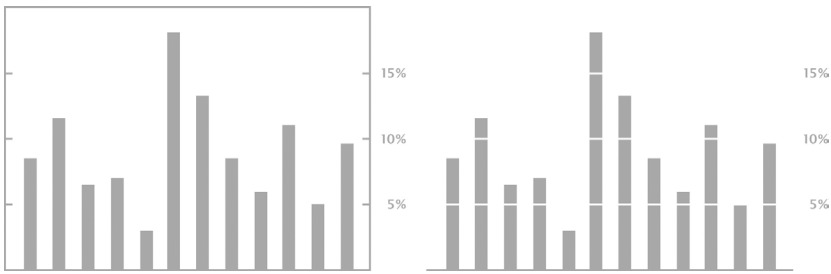


Apple's round mouse was a neat idea conceptually, but an extremely bad idea in terms of functionality.

Principle Two: Erase

Erase non-data pixels

Edward Tufte in his classic book The Visual Display of Quantitative Information, introduces his principle of erasing "non-data ink." To visually dis-



Having erased the bounding box, the hash marks, and the bars where the hash marks are indicated, Tufte shows how less ink clarifies the relationship of each bar to the others.

play data (in his case a graph) Tufte shows how, by erasing all lines, points, etc. which do not directly serve to communicate the data, the resulting graph will communicate the information more directly. Erasing non-descriptive ink also draws out relationships within data that would otherwise be missed. This simple principle applied to web-based information will help to cut through the clutter and confusion that typifies many websites. As is most often the case, more can usually be communicated with less.

Simplify vs. clarify

Richard Saul Wurman qualifies the principle of simplification (erasing) as just one aspect of the true goal of the information designer, that is, clarification. He writes, "The fundamental task of the information architect is to make information understandable. My passion has always been making the complex clear, clarifying, rather than simplifying it. The simplification movement is just another minimalist fashion. The goal is to clarify – make it easy to understand."

Clarification is the ultimate goal for the information designer. Simplification is a good place to start. Stripping away clutter helps to reveal the essence of information. It may be necessary to refine, and add back some of the information that initially gets stripped out, but everything should ultimately add clarity. Anything that does not add clarity should most often be removed.

Principle Three: GOOD QUESTIONS

Planning essential

Planning is the most critical part of the design process, and the most important part of the planning process is asking the right questions. Grayscreen is designed to facilitate the planning process. It enables the client and development team to ask the right questions and design the site appropriately in the beginning.

Finding the right questions

One of the primary skill sets of the information designer is the ability to

ask the right questions. The right questions are rarely the first questions. The first questions are like darts thrown in the dark, they don't often hit the target, but they can tell you where the target isn't. Through asking general questions and listening carefully to the answers, an information designer will help find the target. As the target comes into focus, new and better questions can be asked. These refined questions clarify the overall priorities for a project.

Taking the necessary time

Thoroughly defining a project can be difficult and time consuming. It can be tempting to skip over or superficially work through this process. The information designer needs to make sure that all problems are fully defined so that a design solution can be developed that will provide the right answers to the right questions. To find the right answers, you have to ask the right questions.

Principle Four: The Fallacy of "Professional" Testing

At some point in the development process it is helpful to do usability testing. Doing good usability testing can be quite tricky because of the real way that people perceive and experience things.

Beckwith's examples

Harry Beckwith in his insightful books, Selling the Invisible and The Invisible Touch, relates a few stories of how people really behave as opposed to how we think they behave.

He reminds us of the "Folger's Crystals" commercials, where they secretly replace the house coffee at some ritzy restaurant with Folger's. Of course, the unsuspecting customer raves about the coffee only to find out that it's Folger's. Did Folger's really taste as good, or will almost any coffee taste good when consumed in the context of such an overwhelmingly posh environment?

Another of his examples is the "McLowCal" burger. McDonald's spent a lot of time surveying consumers to ask if they would purchase a low cal alternative to the Big Mac. Of course everyone would like to think that

they would make the healthy choice, and implied this in the results. However, once on the market, the product failed miserably. Although we would all like to think that we would choose a healthy burger when provided with the option, in reality we choose the tasty burger. The truth is, we think we behave differently than we actually do.

These two examples demonstrate how the responses we get from focus groups, sample surveys, and user testing can be misleading. More often than not, these methods return the results we expect to get.

Utilize the experience and insight of many

We have found that the best results for usability testing (as well as the fastest and least expensive) come from applying the 80% - 20% rule*. That is, 80% of good design will come from an inclusive process that enables a range of people familiar with the needs of a site to work through the organization of its structure, content, and functionality. The remaining 20% of "testing" can be accomplished through having an outside group provide a basic site review, checking for typos, broken links, etc.

It's important that these "testers" not know what they are really testing. As soon as you tell someone to "test the user interface," they will immediately enter a critical frame of mind where they must find problems, or they will be thought to lack insight. By having someone check for typos or broken links, they will not only provide a useful service, but they will also experience any real interface problems as they work through the site. By asking general questions about their experience, you will get much better feedback. If such a person reports interface or organizational problems they are likely to be real issues.

*The 80% - 20% rule is based on the principle that 37.45% of all statistics are made up on the spot.

Principle Five: DESIGN

"To design is to plan, to order, to relate, and to control. In short, it opposes all means of disorder and accident" – Emil Roder, Typography

The opposite of accident

Design is the opposite of accident. Design is often thought of as the result of an artist's choices of color, layout, and type. While this is certainly a part of the graphic design process, the information design process is different. Design can also be thought of as a deliberate process of ordering. This is the kind of design an information designer practices. The tools of the information designer are not necessarily fonts and colors, but thought and analysis. In this respect, information design is akin to architecture or engineering more so than graphic design.

Pressures that work against good design

The pace of technology and the demands of the marketplace often fight against the design process. Sometimes there is no way to slow things down. Design opposes accident, but when things move too fast, accidents are bound to happen. The design process requires time to work through. Fortunately, the grayscreen process facilitates good planning and definition, while moving the project forward rapidly.

Practical Information Design Tips

Working through the prototyping process requires us to implement our information design skills. Keeping information design principals in mind will help us to sharpen our skills and evaluate a prototype. The following practical tips provide specific help for working through the common information design aspects of websites.

Establishing site categories

The fist step in evaluating a prototype is to review the overall organization and categorization of its contents. There is a tendency for people working within a company to categorize their content according to internal company structures. These internal conventions for organizing information are not always intuitive and can confuse people from outside the company. Therefore it is helpful to have a fresh, neutral frame of mind when organizing a prototype. We can protect ourselves from making organizational assumptions by stepping back and considering the different ways content can be grouped. Nathan Shedroff points out that there are basically six ways that information can be organized: alphabetically, numerically, by time, by location, by category, or randomly. Reviewing how we have organized and grouped the content in a prototype can help us be more objective, thus ensuring that a site's structure will be clear to others.

Labeling

Another aspect of organization and categorization is labeling. Once the overall grouping and categorization of the prototype has been settled, the names given to each section and page must be examined. When naming sections or pages, brevity is important. Clarity is also important. Generally speaking, the best labels are those with the shortest names that still clearly identify the content.

Navigation systems

Maintaining a consistent navigational structure is one of the keys to making a website easy to use. A navigation system should be established right from the start of a project. In some cases, a home page will use a

slightly different layout from the rest of a site and the navigation system might be displayed differently on it. If this is the case, the rest of the site should maintain a consistent navigation system. The main navigation should always appear in the same place on all pages.

The main navigation of a typical site is made up of the main sections and what are sometimes referred to as "site utilities." Main sections are the primary categories for a site. "About us," "Products," "What's New," and "Services" are examples of main categories. These are usually specific to each company and will include any number of sub pages. Site utilities are generally pages that are common to almost every website such as "contact us," "search," "site map," and "home." Site utilities can often be displayed as icons rather than being listed among the main sections.

In addition to maintaining a consistent main navigation bar, we recommend creating a consistently located sub navigation system. Sub navigation makes it easy for a user to toggle between pages within a main section. It also helps a visitor to know where they are in a site at all times. A navigation system should clearly identify and distinguish main sections (first tier), sub sections (second tier), and third tier pages. Indenting, changing color, or changing size can help distinguish between these navigation levels.

Drop down menus make for efficient site navigation, but they do create some interface problems that need to be carefully considered.

DHTML drop down menus

DHTML drop down menus are useful because they dynamically display the contents of main and sub sections as a mouse rolls over the navigation bar. A visitor can click on any of the sub links and go straight to a specific page. Drop down menus simplify site navigation. They also remove the need for site maps, since the navigation system itself acts as a dynamic site map.

"Drop down menus" are a useful devise for displaying and linking sub sections (more on this later). For download and speed considerations, graphics can be used for the main section navigation bar, but HTML text should be used for the sub page navigation.

A caution when using drop down menus

When using drop down menus, visitors tend to "skip over" main section pages, also referred to as a "landing pages." For example, when a mouse moves over an "About Us" section, several sub pages will be dynamically presented. A visitor will usually click directly to one of these sub pages, bypassing the content on the "About Us" page. If there is important content on theses pages, it will typically not be seen. There are a couple of techniques to address this drop down menu dynamic. One is to add a duplicate of the main section pages in its sub page list. For example, one of the sub choices under the "About Us" main section could also be called "About Us" and link to the same page. While this technique does not require a visitor to hit the main section page, it does increase the likelihood that they will see its contents.

The other technique is to make sure that no critical information is placed on main landing pages. Instead they should contain simply short descriptions of the content in the main section with links to each sub page. These kinds of main section pages act as a glorified table of contents pages. In the event that the visitor skips over these landing pages, they will not have missed any critical information.

Searching a site

Searching offers some powerful tools that can make sites easier to use. Search capabilities never replace the need for consistent site organization. In fact, using site search tools opens up other information design considerations.

Determining the usefulness of a search

Search engines can sift through vast amounts of information almost instantaneously. When a site contains a significant amount of content, a search is often the best way to locate specific content. The depth of information to be searched is a factor in determining if a search function

is appropriate and how it should be implemented. For example, it would not be very helpful to search a nationwide retail store finder that contains 200 stores by street address or zip code. More likely than not, the search would not find any records. Whereas searching by address or zip code could be very useful if a store locator has thousands of records.

Searching with too many criteria

Having too many criteria to search, compared to the number of database records will also result in failed searches. For example, a car database could be searched by the model, year, color, transmission, options, and mileage. The more search criteria presented, the more those criteria can be combined, and the fewer results will be found. If the range of search criteria is too broad for the number of vehicles in a database, the user will be frustrated by "not found" results. Reducing the criteria, or restricting the combination of criteria, would make the search more useful.

Boolean operators

Another consideration when using a search is whether to allow Boolean (and, or) operators. "And/or" options allow a user to enter multiple keywords and choose whether the results should contain any of the words in the query, or records with all the terms in the query. Again, one should take into account the depth of content so that an "and" operator will narrow results rather than eliminate them.

Ranking search engine results:

When a search engine finds results from a search query, it needs to know how to present the results to the user. They should be ranked in some order. Search engine ranking is often based on the number of occurrences of the search term. However, other factors can be added to this ranking system: how recent is the information, how popular is the content, what is the title of the document, etc. Such factors need to be considered in how a search engine determines relevance when listing results.

Conclusion

The process and skill of information design are essential to successful web development projects. The web is still a relatively young form of communication. It's to be expected, that mistakes will be made as clients and developers work through the process of communicating this new medium. Learning how to effectively communicate about the structural and technical complexities of websites is central to improving our ability to utilize the web to its fullest potential.

Calling a truce in the client, developer war

The web is truly a revolution in how we communicate information. As difficult as the web development process can be, companies continue to build sites, attempting to realize the rich potential the web. Imagine if the time commonly wasted in miscommunication during web projects could be poured into actually improving sites. Instead of bickering over problems, clients and developers could focus their attention on making their sites better. Exaggerated expectations could be uncovered and addressed if clients and developers worked through an effective hands-on process of defining their sites.

Web technology will continue to change and evolve. Every day new site features and capabilities become available. But no matter how much promise technology holds, good communication will always be essential to successful web development. Sooner or later, grayscreen prototyping might be outdated, but communicating technical information non-technically will always provide a solid foundation for effective web development. Only a deep commitment to discovering new and innovative ways of communicating technical information non-technically will allow us to keep the peace and end this war between clients and developers.

Recommended Reading

Visual Explanations: Images and Quantities, Evidence and Narrative
Edward R. Tufte. April 1997, Graphics Press.

The Visual Display of Quantitative Information
Edward R. Tufte. November 1987, Graphics Press.

Envisioning Information
Edward R. Tufte. October 1990, Graphics Press.

Information Architecture
Louis Rosenfeld & Peter Morville. October 1998, O'Reilly and Associates.

Don't Make me Think
Steve Krugg. October 2000, Que,

Information Design
Robert Jacobson, Ed.. 1999, MIT Press.

Information Architects
Richard Saul Wurman. 1996, Graphics Press.

Designing Visual Interfaces
Kevin Mullet and Darrell Sano. 1995, SunSoft Press.

Web Navigation: Designing the User Experience
Jennifer Fleming. O'Reilly & Associates, 1998.

Web Style Guide: Basic Design Principles for Creating Websites
Patrick J. Lynch and Sarah Horton. Yale University Press, 1999.

Designing the User Interface: Strategies for Effective Human-Computer Interaction
Ben Shneiderman. Addison Wesley Longman, 1998.

The Elements of User Interface Design
Theo Mandel. Wiley, John & Sons, 1997.

Data Smog
David Shenk. Harper San Franscisco, 1998.

Selling the Invisible
Harry Beckwith. Warner Books, 1997

The Invisible Touch
Harry Beckwith. Warner Books, 2000