Why is good UI design so hard for some Developers? [closed]

Asked 15 years, 10 months ago Modified 6 years, 1 month ago

Viewed 55k times

206

votes

Some of us just have a hard time with the softer aspects of UI design (*myself especially*). Are "back-end coders" doomed to only design business logic and data layers? Is there something we can do to **retrain our brain** to be more effective at designing pleasing and useful presentation layers?

Colleagues have recommended a few books me including *The Design of Sites*, *Don't make me think* and *Why Software sucks* , but I am wondering what others have done to remove their deficiencies in this area?

user-interface

Share

edited Nov 7, 2018 at 1:05

community wiki
13 revs, 3 users 100%
Chris Ballance

1 For myself (I suffer the same problem) I know a good UI when I use one. And I definitely know a UI that annoys me. But I have a very difficult time trying to design one myself. It's like the critical eye I have when I'm using someone else's UI doesn't work on my own designs. – JMD Feb 5, 2009 at 1:17

1 I hate the term "back-end coder" and was trying to keep it out of the title – Chris Ballance Feb 5, 2009 at 1:44

20 Graphic design != UI design. The former is about making things pretty. The latter is about making things useful and usable. – Esko Luontola Feb 10, 2009 at 22:24

2 +1 @Esko. Often 'pretty' means totally UN-usable. But the two CAN co-exist if handled with care and thought. – Dhaust Feb 10, 2009 at 23:25

12 Why is developing software so hard for UI designers? – Greg Hurlman Feb 13, 2009 at 20:33

Comments disabled on deleted / locked posts / reviews |

65 Answers

Sorted by: Highest score (default)

1 2 3 Next

359

votes

+75

Let me say it directly:

Improving on this does not begin with guidelines. It begins with reframing how you think about software.

Most hardcore developers have practically **zero** empathy with users of their software. They have **no clue** how users think, how users build models of software they use and how they use a computer in general.

It is a typical problem when an expert collides with a laymen: How on earth could a normal person be so **dumb** not to understand what the expert understood 10 years ago?

One of the first facts to acknowledge that is unbelievably difficult to grasp for almost all experienced developers is this:

Normal people have a vastly different concept of software than you have. They have no clue whatsoever of programming. None. Zero. And they don't even care. They don't even think they have to care. If you force them to, they will delete your program.

Now that's unbelievably harsh for a developer. He is proud of the software he produces. He loves every single feature. He can tell you exactly how the code behind it works. Maybe he even invented an unbelievable clever algorithm that made it work 50% faster than before.

And the user doesn't care.

What an idiot.

Many developers can't stand working with normal users. They get depressed by their non-existing knowledge of technology. And that's why most developers shy away and think users must be idiots.

They are not.

If a software developer buys a car, he expects it to run smoothly. He usually does not care about tire pressures, the mechanical fine-tuning that was important to make it run that way. Here he is **not** the expert. And if he buys a car that does not have the fine-tuning, he gives it back and buys one that does what he wants.

Many software developers like movies. Well-done movies that spark their imagination. But they are not experts in producing movies, in producing visual effects or in writing good movie scripts. Most nerds are very, very, very bad at acting because it is all about displaying complex emotions and little about analytics. If a developer watches a bad film, he just notices that it is bad as a whole. Nerds have even built up IMDB to collect information about good and bad movies so they know which ones to watch and which to avoid. But they are not experts in creating movies. If a movie is bad, they'll not go to the movies (or not download it from BitTorrent ;)

So it boils down to: Shunning normal users as an expert is **ignorance.** Because in those areas (and there are so many) where they are not experts, they expect the experts of other areas to have already thought about normal people who use their products or services.

What can you do to remedy it? The more hardcore you are as a programmer, the less open you will be to normal user thinking. It will be alien and clueless to you. You will think: I can't imagine how people could **ever** use a computer with this lack of knowledge. But they can. For

every UI element, think about: Is it necessary? Does it fit to the concept a user has of my tool? How can I make him understand? Please read up on usability for this, there are many good books. It's a whole area of science, too.

Ah and before you say it, yes, I'm an Apple fan ;)

Share

edited Apr 15, 2009 at 15:23

Chris Ballance

34.3k ● 26 ● 105 ● 152

answered Feb 5, 2009 at 14:42

Thorsten79

10.1k ● 6 ● 40 ● 54

8 Excellent comment! You've nailed one of the most fundamental hurdles in software design. A hard fact to swallow for hardened developers (like me), but the truth often is. – user34411 Feb 5, 2009 at 22:03

12 +1. I recommend reading "The Inmates are Running the Asylum", it goes into good detail about the differences in user/dev mindsets, as well as some remedies. – Richard Levasseur Feb 7, 2009 at 18:59

9 +1 To be honest, any developer that doesn't care about the user is a poor developer! – Gary Willoughby Feb 7, 2009 at 22:10

7 Very valid points, and I think that this mentality is also part of the reason that a number of developer-run projects (e.g. open source or what-have-you) have come across as difficult to use -- by and large, most developers write for themselves as the user, not for the "real" end user. – Cloudy Feb 10, 2009 at 22:19

25 +1 This is THE reason Linux still isn't ready for the average user's desktop. – Bryan Anderson Feb 11, 2009 at 15:16

215 votes

UI design *is* hard

To the question:

> **why is UI design so hard for most developers?**

Try asking the inverse question:

> **why is *programming* so hard for most *UI designers*?**

Coding a UI and designing a UI require different skills and a different mindset. UI design is hard for *most* developers, not *some* developers, just as writing code is hard for *most* designers, not *some* designers.

Coding is hard. Design is hard too. Few people do both well. Good UI designers rarely write code. They may not even know how, yet they are still good designers. So why do good developers feel responsible for UI design?

Knowing more about UI design will make you a better developer, but that doesn't mean you should be responsible for UI design. The reverse is true for designers: knowing how to write code will make them

better designers, but that doesn't mean they should be responsible for coding the UI.

How to get better at UI design

For developers wanting to get better at UI design I have 3 basic pieces of advice:

1. **Recognize design as a separate skill**. Coding and design are separate but related. UI design is not a subset of coding. It requires a different mindset, knowledge base, and skill group. There are people out there who focus on UI design.
2. **Learn about design**. At least a little bit. Try to learn a few of the design concepts and techniques from the long list below. If you are more ambitious, read some books, attend a conference, take a class, get a degree. There are lot of ways to learn about design. Joel Spolky's book on UI design is a good primer for developers, but there's a lot more to it and that's where designers come into the picture.
3. **Work with designers**. Good designers, if you can. People who do this work go by various titles. Today, the most common titles are User Experience Designer (UXD), Information Architect (IA), Interaction Designer(ID), and Usability Engineer. They think about design as much as you think about code. You can learn a lot from them, and they from you. Work with them however you can. Find people with these

skills in your company. Maybe you need to hire someone. Or go to some conferences, attend webinars, and spend time in the UXD/IA/ID world.

Here are some specific things you can learn. **Don't try to learn everything**. If you knew everything below you could call yourself an interaction designer or an information architect. **Start with things near the top of the list**. Focus on specific concepts and skills. Then move down and branch out. If you really like this stuff, consider it as a career path. Many developers move into managements, but UX design is another option.

- **Learn fundamental design concepts**. You should know about affordances, visibility, feedback, mappings, Fitt's law, poka-yokes, and more. I recommend reading *The Design of Everyday Things* (Don Norman) and *Universal Principles of Design* (Lidwell, Holden, & Butler)
- **Learn about user experience**. This is becoming the umbrella term for the human-centered design of web sites, applications, and any other digital artifact. The classic primer here is *The elements of User Experience* (Jesse James Garrett). You can get an overview and the first few chapters from the author's site.
- **Learn to sketch designs**. Sketching is fast way to *explore* design options and find the right design, whereas usability testing is about getting the design right. Paper prototyping is fast, cheap, and effective

during the early design stages. Much faster than coding a digital prototype. The key text here is *Sketching User Experience: Getting the design right and the right design* (Bill Buxton). Sketching is a particularly useful skill when working with IA/ID/UX designers. Your collaboration will be more effective. For a good primer on how and why designers sketch, watch the presentation How to be a UX team of one by Leah Buley from the 2008 IA Summit.

- **Learn paper prototyping**. The fastest way to iteratively *test* an interface before you write code. Different from sketching and usability testing. The definitive book here is *Paper Prototyping* (Carolyn Snyder). You can get a good DVD on this from the Nielsen Norman Group.
- **Learn usability testing**. Discount testing is easy and effective. But for many UIs, usability is hard to do well. You can learn the basics quickly, but good usability people are invaluable. If you want a book, the classic is *The Handbook of Usability Testing* (Jeffrey Rubin). It's older but offers thorough coverage of lab-based testing. The famous starter book is *Don't Make Me Think (2nd Ed)* (Steve Krug). I caution people about this one: Krug makes it sound easier than it is. But it is a good starting point. The user research books listed in the next point also cover this topic. And you can find piles about it online.
- **Learn about information architecture**. The main book here is *Information Architecture for the World Wide Web (3rd)* (Louis Rosenfeld & Peter Morville). A

good starter book is *Information Architecture: Blueprints for the Web* (Christina Wodtke). For more, visit the Information Architecture Institute or attend the annual Information Architecture Summit.

- **Learn about interaction design**. The main book here is *The Essentials of Interaction Design (3rd)* (Alan Cooper, *et al*). A good starter book is *Designing for interaction* (Dan Saffer). For more, visit the Interaction Design Association (IxDA) or attend the annual Interaction Design conference.
- **Learn fundamentals of graphic design**. Graphic design is not UI design, but concepts from graphic design can improve an interface. Graphic design introduces design principles for the visual presentation of information, such as proximity, alignment, and small multiples. I recommend reading *The non-designer's design book* (Robin Williams) and *Envisioning Information* (Edward Tufte)
- **Learn to do user research**. Where usability tests an interface, user research tries to model users and their tasks through personas, scenarios, user journeys, and other documents. It's about understanding users and what they do, then using that to inform the design instead of guessing. Some techniques are interviews, surveys, diary studies, and cart sorting. Good books on this are *Observing the User Experience* (Mike Kuniavsky) and *Understanding Your Users* (Courage & Baxter)

- **Learn to do field research**. Watching people in the lab under artificial conditions helps (ie: usability), but there is nothing like watching people use your code in context: their home, their office, or wherever they use it. Goes by various names, including ethnography, field studies, and contextual inquiry. Here is a good primer on field research. Two of the better known books here are *Rapid Contextual Design* (Karen Holtzblatt *et al*) and *User and task analysis for interface design* (Hackos & Redish).
- **Read UX design web sites**. Some of the big ones are Boxes & Arrows, UX Mag, UX Matters, and Digital Web magazine.
- **Use UI pattern libraries**. There are patterns for interfaces. For web sites, I recommend *The Design of Sites, 2nd ed* (Van Duyne, *et al*) and *Homepage usability: 50 websites deconstructed* (Jakob Nielsen & Marie Tahir). For desktop applications I recommend *Designing interfaces* (Jennifer Tidwell), and for web applications I recommend *Designing Web Interfaces: Principles and Patterns for Rich Interactions* (Bill Scott & Theresa Neil). Online you should check Welie pattern library, UI patterns, and Web UI patterns.
- **Attend UX design conferences**. Some good annual conferences are: Information Architecture Summit, Interaction '09 (IxDA), User Interface, and UX week.
- **Attend a workshop or webinar**. You can take workshops, webinars, and online courses. This is far from a comprehensive list, but you might try the UIE

virtual seminars, Adaptive Path virtual seminars, and UX webinars from Rosenfeld Media.

- **Get a degree**. A graduate degree in HCI is one approach, but these programs are mostly about writing coding. If you want to learn about the design of digital artifacts and devices, then you want a graduate program that's not in CS. Some options include Interaction Design at Carnegie Mellon, the d-School at Stanford, the ITP program at NYU, and Information Architecture & Knowledge Management at Kent State (disclosure: I'm on faculty at Kent; we are seeing more and more people with CS degrees moving into UX design instead of management, which is interesting, because management is the traditional path for developers who want to move away from writing code while staying in their field). There are many more programs. Each has their own perspective, areas of emphasis, and technical expectations. Some come out of the arts and visual design, others out of library and information science, and some from CS. Most are hybrids, but every hybrid has deeper roots in one or more fields. If this interests you, look around and try to understand the differences between these programs. Some offer online courses and certificate programs in addition to full-fledged degrees.

Why UI design is hard

Good UI design is hard because it involves 2 vastly different skills:

- **A deep understanding of the machine**. People in this group worry about code first, people second. They have deep technological knowledge and skill. We call them developers, programmers, engineers, and so forth.
- **A deep understanding of people and design**: People in this group worry about people first, code second. They have deep knowledge of how people interact with information, computers, and the world around them. We call them user experience designers, information architects, interaction designers, usability engineers, and so forth.

This is the essential difference between these 2 groups—between developers and designers:

- **Developers make it work**. They implement the functionality on your TiVo, your iPhone, your favorite website, etc. They make sure it actually does what it is supposed to do. Their highest priority is making it work.
- **Designers make people *love* it**. They figure out how to interact with it, how it should look, and how it should feel. They design the experience of using the application, the web site, the device. Their highest priority is making you fall in love with what developers make. This is what is meant by user experience, and it's not the same as brand experience.

Moreover, programming and design require *different mindsets*, not just different knowledge and different skills. Good UI design requires both mindsets, both knowledge bases, both skill groups. And it takes years to master either one.

Developers should expect to find UI design hard, just as UI designers should expect to find writing code hard.

Share

edited Aug 17, 2009 at 14:17

community wiki
23 revs, 2 users 99%
Karl Fast

8 This is the best answer here. Great links BTW! – Bernard Igiri Feb 11, 2009 at 20:32

Excellent overview of UI design! I've also observed your notion of different mindsets. I do both UI design and programming, and it's best that I concentrate on only one of them at a time. [...] – Esko Luontola Feb 19, 2009 at 23:28

If you make UI design decisions while programming, you think that what is the simplest way to implement, which often leads to poor UI design. And if you think about implementation while doing UI design, you might choose UI design patterns which are easier to implement, but not as good for the user. – Esko Luontola Feb 19, 2009 at 23:29

2 Why isn't this marked as the best answer? It seems far better than Thorsten79's answer. – AbdullahC Sep 5, 2010 at 9:19

1 Wish I could favorite this answer directly. :) – Dan J Apr 14, 2011 at 18:30

70 votes

What really helps me improve my design is to grab a fellow developer, one the QA guys, the PM, or anyone who happens to walk by and have them try out a particular widget or screen.

Its amazing what you will realize when you watch someone else use your software for the first time

Share

answered Feb 5, 2009 at 0:59

Kevin

30.4k ● 9 ● 78 ● 83

I've tried this approach a lot and found it to be very effective. Occasionally I have a non-technical friend use it to see what causes them pain when trying to use it. – Chris Ballance Feb 5, 2009 at 1:03

This is the same approach that I take. – Ed Swangren Feb 5, 2009 at 1:15

Could this approach be called "usability testing" perchance? ;) Yes, it's the approach you should take prior, during and after. – Ates Goral Feb 6, 2009 at 4:07

This would be almost my exact answer. QA and tech support are awesome. Developers suck at UI Design, iterate and have others test it out often. – Bill K Feb 7, 2009 at 2:02

7 I believe its called the "hallway usability test" – Kevin Feb 7, 2009 at 12:48

32

votes

Ultimately, it's really about empathy -- can you put yourself in the shoes of your user?

One thing that helps, of course, is "eating your own dogfood" -- using your applications as a real user yourself, and seeing what's annoying.

Another good idea is to find a way to watch a real user using your application, which may be as complicated as a usability lab with one-way mirrors, screen video capture, video cameras on the users, etc., or can be as simple as paper prototyping using the next person who happens to walk down the hall.

If all else fails, remember that it's almost always better for the UI to be too simple than too complicated. It's very very easy to say "oh, I know how to solve that, I'll just add a checkbox so the user can decide which mode they prefer". Soon your UI is too complicated. Pick a default mode and make the preference setting an advanced configuration option. Or just leave it out.

If you read a lot about design you can easily get hung up on dropped shadows and rounded corners and so forth. That's not the important stuff. Simplicity and discoverability are the important stuff.

Share

answered Feb 5, 2009 at 1:05

Jacob Mattison

51k ● 11 ● 108 ● 128

well put. Most often, attempts to add flexibility just result in clutter. Simpler == better. – SquareCog Feb 5, 2009 at 1:39

Can you put yourself in the shoes of your user if they are on the other side of the planet, in a culture you've never experienced before? Internationalization is a major consideration in solid UI design. Lets not assume that everyone lives in the USA.
– user34411 Feb 5, 2009 at 4:15

Absolutely; my current project is in fact targeted at users all over the planet (basically in every country other than the US, where I am). It makes empathy harder, no question -- and all the more important to attempt it seriously and vigorously.
– Jacob Mattison Feb 5, 2009 at 4:39

Also, I'd add that the vast majority of usability problems are shockingly obvious once you get a real user to look at it. Yes, there will be subtle ones that are culture-specific, but you can make tremendous improvements using the next guy walking down the hall. – Jacob Mattison Feb 5, 2009 at 4:42

+1, only because I can't +100 myself! I'll add that gaining experience using vastly different applications as they were intended helps you grow as a UI developer. Only using tools like Visual Studio or other dev tools will hamper this ability...
– BQ. Feb 5, 2009 at 15:34

26

votes

Contrary to popular myth there are literally no soft aspects in UI design, at least no more than needed to design a good back end.

Consider the following; good back end design is based upon fairly solid principles and elements any good developer is familiar with:

- low coupling
- high cohesion
- architectural patterns
- industry best practices
- etc

Good back end design is usually born through a number of interactions, where based on the measurable feedback obtained during tests or actual use the initial blueprint is gradually improved. Sometimes you need to prototype smaller aspects of back end and trial them in isolation etc.

Good UI design is based on the sound principles of:

- visibility
- affordance
- feedback
- tolerance
- simplicity
- consistency
- structure

UI is also born through test and trial, through iterations but not with compiler + automated test suit, but people. Similarly to back end there are industry best practises, measurement and evaluation techniques, ways to think of UI and set goals in terms of user model, system image, designer model, structural model, functional model etc.

The skill set needed for designing UI is quite different from designing back-end and hence don't expect to be able to do good UI without doing some learning first. However that both these activities have in common is the process of design. I believe that anyone who can design good software is capable of designing good UI as long as they spend some time learning how.

I recommend taking a course in Human Computer Interaction, check MIT and Yale site for example for online materials:

- MIT User Interface Design and Implementation Course

Structural vs Functional Model in Understanding and Usage

The excellent earlier post by Thorsten79 brings up the topic of software development experts vs users and how their understanding of software differ. Human learning experts distinguish between functional and structural mental models. Finding way to your friend's house can be an excellent example of the difference between the two:

- First approach includes a set of detailed instructions: take the first exit of the motorway, then after 100 yards turn left etc. This is an example of functional model: list of concrete steps necessary to achieve a certain goal. Functional models are easy to use, they do not require much thinking just a straight forward execution. Obviously there is a penalty for the simplicity: it might not be the most efficient route and any any exceptional

situation (i.e. a traffic diversion) can easilly lead to a complete failure.

- A different way to cope with the task is to build a structural mental model. In our example that would be a map that conveyes a lot of information about the internal structure of the "task object". From understanding the map and relative locations of our and friend's house we can deduct the functional model (the route). Obviously it's requires more effort, but much more reliable way of completing the task in spite of the possible deviations.

The choice between conveying functional or structural model through UI (for example, wizard vs advanced mode) is not that straight forward as it might seem from Thorsten79's post. Advanced and frequent users might well prefer the structural model, whereas occassional or less expirienced users — functional.

Google maps is a great example: they include both functional and structural model, so do many sat navs.

Another dimension of the problem is that the structural model presented through UI must not map to the structure of software, but rather naturally map to structure of the user task at hand or task object involved.

The difficulty here is that many developers will have a good structural model of their software internals, but only functional model of the user task the software aims to

assist at. To build good UI one needs to understand the task/task object structure and map UI to that structure.

Anyway, I still can't recommend taking a formal HCI course strongly enough. There's a lot of stuff involved such as heuristics, principles derived from Gestalt phychology, ways humans learn etc.

Share

edited May 23, 2017 at 11:46

Community Bot

1 ● 1

answered Feb 5, 2009 at 14:49

Vlad Gudim

23.5k ● 16 ● 71 ● 92

+1 for the MIT OCW link :) Those lectures notes are invaluable – Debajit Feb 10, 2009 at 22:17

You're right, users can become experts of a software themselves, and they like it when somebody carved a way for them through the "nice" UI to be faster. That's where all the key shortcuts are for, for example. usability is a very interesting subject! – Thorsten79 Feb 12, 2009 at 21:41

25

votes

I suggest you start by doing all your UI in the same way as you are doing now, with no focus on usability and stuff.

alt text

http://www.stricken.org/uploaded_images/WordToolbars-718376.jpg

Now think of this:

A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away. — Saint-Exupéry

And apply this in your design.

Share

edited Feb 7, 2009 at 21:00

community wiki
2 revs
Hoffmann

This is why I far, far prefer Office 2007! – Richard Ev Feb 7, 2009 at 18:55

If someone is wondering from which grave I dug that quote, I heard it from civilization 4. Great game. I have no idea who the author is, but he sure as hell could write a better UI than the designers from word. – Hoffmann Feb 8, 2009 at 2:27

There's another similar good quote from A. Einstein: "Make everything as simple as possible, but not simpler." I've found these ideas also applicable to UI design: www.presentationzen.com. ...and +1 – Pyry Jahkola Feb 9, 2009 at 0:58

Saint-Exupery was much more than a pilot and aircraft designer; +1 for quoting him. His children's book ("The Little Prince") is well worth reading, but I'm getting off-topic here. – David Thornley Feb 11, 2009 at 14:57

5 You should've selected the print view...can't see the rulers now. – Mussnoon Feb 13, 2009 at 14:55

16

votes

A lot of developers think that because they can write code, they can do it all. Designing an interface is a completely different skill, and it was not taught at all when I attended college. It's not just something that just comes naturally.

Another good book is *The Design of Everyday Things* by Donald Norman.

Share

edited Mar 6, 2010 at 10:50

F'x

12.3k ● 8 ● 73 ● 125

answered Feb 5, 2009 at 1:05

thursdaysgeek

7,926 ● 21 ● 81 ● 118

Thanks, "The Design of Everyday Thinks" is on my Amazon WishList – Chris Ballance Feb 5, 2009 at 1:11

+1. "The Design of everyday things" is also on the Coding Horror list of recommended reading (good list of books for developers). Recently finished reading it, and can also highly recommend it. – Mun Feb 7, 2009 at 19:39

I'm reading "The Design of Everyday things" at the moment - it's certainly very good, but as mentioned in comments on another answer, Alan Cooper's "The Inmates are running the Asylum" addresses exactly this topic. As a developer, the book was "life changing"! – Stuart Helwig Feb 10, 2009 at 23:47

My peeve: a lot of developers (and designers, and others) think that because I can code, I can't design the user interface. I don't know Photoshop very well, but I think I have a good eye for what works and what doesn't. (The Design of Everyday

things was my first UI bible.) – Barry Brown Apr 15, 2009 at 15:38

14 votes

There's a huge difference between design and aesthetics, and they are often confused.

A beautiful UI requires artistic or at least aesthetic skills that many, including myself, are incapable of producing. Unfortunately, it is not enough and does not make the UI usable, as we can see in many heavyweight flash-based APIs.

Producing usable UIs requires an understanding of how humans interact with computers, some issues in psychology (e.g., Fitt's law, Hick's law), and other topics. Very few CS programs train for this. Very few developers that I know will pick a user-testing book over a JUnit book, etc.

Many of us are also "core programmers", tending to think of UIs as the facade rather than as a factor that could make or break the success of our project.

In addition, most UI development experience is extremely frustrating. We can either use toy GUI builders like old VB and have to deal with ugly glue code, or we use APIs that frustrate us to no end, like trying to sort out layouts in Swing.

Share

answered Feb 5, 2009 at 2:23

Uri
89.6k • 51 • 226 • 322

"We can either use toy GUI builders like old VB and have to deal with ugly glue code, or we use APIs that frustrate us to no end, like trying to sort out layouts in Swing." You really boiled down my experiences with GUI building. If I may add: "or crazy DOM and CSS combination" – Hoffmann Feb 13, 2009 at 2:34

I'm sure everyone is familiar with Totally Gridbag? "madbean.com/anim/totallygridbag" – Uri Feb 13, 2009 at 2:42

12

votes

Go over to Slashdot, and read the comments on any article dealing with Apple. You'll find a large number of people talking about how Apple products are nothing special, and ascribing the success of the iPod and iPhone to people trying to be trendy or hip. They will typically go through feature lists, and point out that they do nothing earlier MP3 players or smart phones didn't do.

Then there are people who like the iPod and iPhone because they do what the users want simply and easily, without reference to manuals. The interfaces are about as intuitive as interfaces get, memorable, and discoverable. I'm not as fond of the UI on MacOSX as I was on earlier versions, I think they've given up some usefulness in favor of glitz, but the iPod and iPhone are examples of superb design.

If you are in the first camp, you don't think the way the average person does, and therefore you are likely to make bad user interfaces because you can't tell them from good

ones. This doesn't mean you're hopeless, but rather that you have to explicitly learn good interface design principles, and how to recognize a good UI (much as somebody with Asperger's might need to learn social skills explicitly). Obviously, just having a sense of a good UI doesn't mean you can make one; my appreciation for literature, for example, doesn't seem to extend to the ability (currently) to write publishable stories.

So, try to develop a sense for good UI design. This extends to more than just software. Don Norman's "The Design of Everyday Things" is a classic, and there's other books out there. Get examples of successful UI designs, and play with them enough to get a feel for the difference. Recognize that you may be having to learn a new way of thinking abou things, and enjoy it.

Share

answered Feb 5, 2009 at 15:22

David Thornley

57k ● 9 ● 95 ● 158

+1 for suggestion The Design of Everyday Things – Oskar Duveborn Feb 5, 2009 at 15:37

+1 for a very good answer. I think it's telling that you are still hovering at 1 up-vote (now 2); those that fit your description will probably not think they do, or simply dismiss the very idea :-(– mghie Feb 7, 2009 at 22:06

This is a blanket statement. Not every anti-Apple comment on Slashdot is a criticism of its UI, or verification of a lack of UI understanding. Apple has good UI and so do many of their competitors. – Bernard Igiri Feb 11, 2009 at 20:21

Another thought: an Apple isn't for everyone; for example, it doesn't provide the phenomenal awesome power that a CLI does. – J. Polfer Feb 11, 2009 at 21:22

2 I personally find Apple products to be extremely unintuitive to use. I imagine I am not the only person who was frustrated when their first iPod would start shuffling songs every time I started jogging, or why the iPhone screen is continually inverting itself. I think things should only occur from direct, unambiguous user actions. – Lotus Notes May 13, 2010 at 23:00

10 votes

The main rule of thumb I hold to, is never try to do both at once. If I'm working on back-end code, I'll finish up doing that, take a break, and return with my UI hat on. If you try to work it in whilst you're doing code, you'll approach it with the wrong mindset, and end up with some horrible interfaces as a result.

I think it's definitely possible to be both a good back-end developer and a good UI designer, you just have to work at it, do some reading and research on the topic (everything from Miller's #7, to Nielsen's archives), and make sure you understand *why* UI design is of the utmost importance.

I don't think it's a case of needing to be creative but rather, like back-end development, it is a very methodical, very structured thing that needs to be learned. It's people getting 'creative' with UIs that creates some of the biggest usability monstrosities... I mean, take a look at 100% Flash websites, for a start...

Edit: Krug's book is really good... do take a read of it, especially if you're going to be designing for the Web.

Share

answered Feb 5, 2009 at 0:57

James B

8,343 ● 4 ● 34 ● 40

and what is the title of this krug's book – defau1t Nov 9, 2011 at 14:44

@refhat It was prominently linked in the top post, so I didn't link it at the time, but it's this one: librarything.com/work/12322 – James B Nov 14, 2011 at 12:47

8

votes

There are many reasons for this.

(1) Developer fails to see things from the point of view of the user. This is the usual suspect: lack of empathy. But it is not usually true since developers are not as alien as people make them out to be.

(2) Another, more common reason is that the developer being so close to his own stuff, having stayed with his stuff for so long, fails to realize that his stuff may not be so familiar(a term better than intuitive) to other people.

(3) Still another reason is the developer lacks techniques.

MY BIG CLAIM: read any UI, human interection design, prototyping book. e.g. Designing the Obvious: A Common Sense Approach to Web Application Design, Don't Make

Me Think: A Common Sense Approach to Web Usability, Designing the moment, whatever.

How do they discuss task flows? How do they describe decision points? That is, in any use case, there are at least 3 paths: success, failure/exception, alternative.

Thus, from point A, you can go to A.1, A.2, A.3. From point A.1, you can get to A.1.1, A.1.2, A.1.3, and so on.

How do they show such drill-down task flow? They don't. They just gloss over it.

Since even UI experst don't have a technique, developers have no chance. He thinks it is clear in his head. But it is not even clear on paper, let alone clear in software implementation.

I have to use my own hand-made techniques for this.

Share

edited Apr 14, 2011 at 18:21

community wiki
2 revs, 2 users 93%
george kyaw naing

7

votes

I try to keep in touch with design-specific websites and texts. I found also the excellent Robin Williams book **The Non-Designer's Design Book** to be very interesting in these studies.

I believe that design and usability is a very important part of software engineering and we should learn it more and stop giving excuses that we are not supposed to do design.

Everyone can be a designer once in a while, as also everyone can be a programmer.

Share

answered Feb 5, 2009 at 0:58

Edwin Jarvis

6,120 ● 7 ● 38 ● 41

I don't agree that everyone can be a programmer. Some aspects of the trade cannot be taught, you can either hack it or you can't (no pun intended.) – Chris Ballance Feb 5, 2009 at 1:01

2 I mean, once in a while, that doesn't mean that the guy will be a *good* programmer or will be it forever. – Edwin Jarvis Feb 5, 2009 at 1:08

7

votes

When approaching UI design, here are a few of the things I keep in mind throughout (by far not a complete list):

- **Communicating a model**. The UI is a narrative that explains a mental model to the user. This model may be a business object, a set of relationships, what have you. The visual prominence, spatial placement, and workflow ordering all play a part in communicating this model to the user. For example, a certain kind of list vs another implies different things, as well as the relationship of what's in the list to the rest of the model. In general I find it best to make sure only one model is

communicated at a time. Programmers frequently try to communicate more than one model, or parts of several, in the same UI space.

- **Consistency**. Re-using popular UI metaphors helps a lot. Internal consistency is also very important.
- **Grouping of tasks**. Users should not have to move the mouse all the way across the screen to verify or complete a related sequence of commands. Modal dialogs and flyout-menus can be especially bad in this area.
- **Knowing your audience**. If your users will be doing the same activities over and over, they will quickly become power users at those tasks and be frustrated by attempts to lower the initial entry barrier. If your users do many different kinds of activities infrequently, it's best to ensure the UI holds their hand the whole time.

Share

answered Feb 5, 2009 at 1:28

Rex M

144k ● 34 ● 291 ● 315

You bring out a good point with **Knowing your audience** ... I also like "You are not your user" (Platt) and "Know thy user." (Platt) This cannot be emphasized enough – Chris Ballance Feb 5, 2009 at 1:39

7

votes

Read Apple Human Interface Guidelines.

Share

answered Feb 7, 2009 at 18:11

community wiki
Marco Luglio

You mean the guidelines that Apple themselves no longer follow? – Gregor Brandt Jan 7, 2011 at 19:41

"They're more like guidelines anyway" ^_^ – Marco Luglio May 13, 2011 at 1:23

5
votes

I find the best tool in UI design is to watch a first-time User attempt to use the software. Take loads of notes and ask them some questions. Never direct them or attempt to explain how the software works. This is the job of the UI (and well written documentation).

We consistently adopt this approach in all projects. It is always fascinating to watch a User deal with software in a manner that you never considered before.

Why is UI design so hard? Well generally because the Developer and User never meet.

Share

answered Feb 5, 2009 at 2:24

user34411

Absolutely agreed, but it's quite possible for a developer to apply solid basic UI principles. – nailitdown Feb 5, 2009 at 2:35

Solid Basic UI principles do not work in every country. We deploy software in many regions and there is a vast difference in UI design for different parts of the globe. Try sending your US-centric app to China sometime... – user34411 Feb 5, 2009 at 4:11

5

votes

duffymo just reminded me why: Many Programmers think "*Design" == "Art".

Good UI design is absolutely not artistic. It follows solid principles, that can be backed up with data if you've got the time to do the research.

I think all programmers need to do is take the time to learn the principles. I think it's in our nature to apply best practice whenever we can, be it in code or in layout. All we need to do is make ourselves aware of what the best practices are for this aspect of our job.

Share

answered Feb 5, 2009 at 3:02

nailitdown

7,978 ● 11 ● 38 ● 38

I think that you can't discount aesthetics completely. Perhaps an ugly UI can be just as useable as a pretty one, but that isn't all that a UI is for. After all, the UI is the face of your program, much like a book's cover: people are much more likely to pick up a program with a pretty UI. – Tikhon Jelvis Aug 16, 2010 at 3:59

Absolutely, aesthetics are important, but that's Graphic Design, not UI design. – nailitdown Aug 16, 2010 at 9:46

5

votes

What have I done to become better at UI design?
Pay attention to it!

It's like how ever time you see a chart on the news or an electronic bus sign and you wonder 'How did they get that

data? Did they do that with raw sql or are they using LINQ?' (or insert your own common geek curiosity here).

You need to start doing that but with visual elements of all kinds.

But just like learning a new language, if you don't *really* throw yourself into it, you won't ever learn it.

Taken from another answer I wrote:

> Learn to look, really look, at the world around you. Why do I like that UI but hate this one? Why is it so hard to find the noodle dishes in this restaurant menu? Wow, I knew what that sign meant before I even read the words. Why was that? How come that book cover looks so wrong? Learn to take the time to think about why you react the way you do to visual elements of all kinds, and then apply this to your work.

Share

edited May 23, 2017 at 12:34

Community Bot

1 ● 1

answered Feb 6, 2009 at 4:12

Dhaust

5,550 ● 9 ● 55 ● 81

+1 for the zen advice on learning to look, and think. Very few people question what meets the eye. – Debajit Feb 10, 2009 at 22:23

5

votes

However you do it (and there are some great points above), it really helped me once I accepted that there is NO SUCH THING AS INTUITIVE....

I can hear the arguments rumbling on the horizon... so let me explain a little.

Intuitive: using what one feels to be right or true based on an unconscious method or feeling.

If (as Carl Sagan postulated) you accept that you cannot comprehend things that are absolutely unlike anything you have ever encountered then how could you possibly "know" how to use something if you have never used anything remotely like it?

Think about it: kids try to open doors not because they "know" how a doorknob works, but because they have seen someone else do it... often they turn the knob in the wrong direction, or pull too soon. They have to LEARN how a doorknob works. This knowledge then gets applied in different but similar instances: opening a window, opening a drawer, opening almost anything big with a big, knob-looking handle.

Even simple things that seem intuitive to us will not be intuitive at all to people from other cultures. If someone held their arm out in front of them and waived their hand up-and-down at the wrist while keeping the arm still.... are they waiving you away? Probably, unless you are in Japan. There, this hand signal can mean "come here". So who is right? Both, of course, in their own context. But if you travel to both, you need to know both... UI design.

I try to find the things that are already "familiar" to the potential users of my project and then build the UI around them: user-centric design.

Take a look at Apple's iPhone. Even if you hate it, you have to respect the amount of thought that went into it. Is it perfect? Of course not. Over time an object's perceived "intuitiveness" can grow or even fade away completely.

For example. Most everyone knows that a strip of black with two rows of holes along the top and bottom looks like a film strip... or do they?

Ask your average 9 or 10 year old what they think it is. You may be surprised how many kids right now will have a hard time identifying it as a film strip, even though it is something that is still used to represent Hollywood, or anything film (movie) related. Most movies for the past 20 years have been digitally shot. And when was the last time any of us held a piece of film of ANY kind, photos or film?

So, what it all boils down to for me is: Know your audience and constantly research to keep up with trends and

changes in things that are "intuitive", target your main users and try not to do things that punish the inexperienced in favor of the advanced users or slow down the advanced users in order to hand-hold the novices.

Ultimately, every program will require a certain amount of training on the user's part to use it. How much training and for which level of user is part of the decisions that need to be made.

Some things are more or less familiar based on your target user's past experience level as a human being, or computer user, or student, or whatever.

I just shoot for the fattest part of the bell curve and try to get as many people as I can but realizing that I will never please everyone....

Share

edited Jul 16, 2010 at 6:11

community wiki
3 revs
exoboy

4

votes

I know that Microsoft is rather inconsistent with their own guidelines, but I have found that reading their Windows design guidelines have really helped me. I have a copy on my website here, just scroll down a little the the Vista UX Guide. It has helped me with things such as colors, spacing, layouts, and more.

Share

answered Feb 5, 2009 at 2:54

David Anderson

13.7k ● 5 ● 52 ● 80

+1 reading the platform's design guidelines is definitely something one should do (and discover how often Microsoft breaks their own rules ;) – Oskar Duveborn Feb 5, 2009 at 15:38

And in addition read the Apple ones (Human Interface Guidelines). They have more basics and principles in them. That makes you understand what should be universal and what is platform specific – Stephan Eggermont Feb 7, 2009 at 19:11

And while you are at it, read the GNOME HIG as well. Between those three it starts to become visible what is just a specific thing of a single platform, and what seems to be a good idea in general. – mghie Feb 7, 2009 at 22:00

Most of these are based on IBMs, but that is pretty out of date now. For example, the standard cut and paste commands were Shift+Del and Shift+Ins (they still work, btw). – Simon Buchan Feb 20, 2009 at 4:20

4

votes

I believe the main problem has nothing to do with different talents or skillsets. The main problem is that as a developer, you know too much about what the application does and how it does it, and you automatically design your UI from the point of view of someone who has that knowledge.

Whereas a user typically starts out knowing absolutely nothing about the application and should never need to

learn anything about its inner workings.

It is very hard, almost impossible, to not use knowledge that you have - and that's why an UI should not be designed by someone who's developing the app behind it.

Share

answered Feb 5, 2009 at 14:34

Michael Borgwardt

346k ● 80 ● 486 ● 723

4

votes

"Designing from both sides of the screen" presents a very simple but profound reason as to why programmers find UI design hard: programmers are trained to think in terms of edge cases while UI designers are trained to think in terms of common cases or usage.

So going from one world to the other is certainly difficult if the default traning in either is the exact opposite of the other.

Share

answered Feb 5, 2009 at 20:14

MSN

54.5k ● 7 ● 78 ● 107

3

votes

To say that programms suck at UI design is to miss the point. The point of the problem is that the formal training that most developers get go indepth with the technology. Human - Computer Interaction is not a simple topic. It is not something that I can "mind-meld" to you by providing a simple one line statement that makes you realize "oh the

users will use this application more effectively if I do x instead of y."

This is because there is one part of UI design that you are missing. The human brain. In order to understand how to design a UI, you have to understand how the human mind interacts with machinery. There is an excellent course I took at the University of Minnesota on this topic taught by a professor of Psychology. It is named "Human - Machine Interaction". This describes many of the reasons of why UI design is so complicated.

Since Psychology is based on Correlations and not Causality you can never prove that a method of UI design will always work in any given situation. You can correlate that many users will find a particular UI design appealing or efficient, but you cannot prove that it will always generalize.

Additionally, there are two parts to UI design that many people seem to miss. There is the aesthetical appeal, and the functional workflow. If you go for a 100% aesthetical appeal, sure people will but your product. I highly doubt that aesthetics will ever reduce user frustration though.

There are several good books on this topic and course to take (like Bill Buxton's *Sketching User Experiences*, and *Cognition in the Wild* by Edwin Hutchins). There are graduate programs on Human - Computer Interaction at many universities.

The overall answer to this question though lies in how individuals are taught computer science. It is all math

based, logic based and not based on the user experience. To get that, you need more than a generic 4 year computer science degree (unless your 4 year computer science degree had a minor in psychology and was emphasized in Human - Computer Interaction).

Share

edited Mar 6, 2010 at 10:52

community wiki
2 revs, 2 users 80%
jwendl

I disagree: computers are math and logic based, by their nature. Any teaching of computer programming must be math and logic based. – Paul Nathan Feb 11, 2009 at 18:16

1 Yes, you are right Paul. Computers are math and logic. UI design involves a human element though, which has proven in history to be far less math and logic based. – jwendl Feb 12, 2009 at 3:10

It's not so much that they are less math and logic based, it's that the math and logic behind them is much more complex, mostly beyond anyone today. That said, I bet good UI designs are all, to some extent, based on geometry. – Tikhon Jelvis Aug 16, 2010 at 4:03

2 votes

Let's turn your question around -

Are "ui designers" doomed to only design information architecture and presentation layers? Is there something

they can do to retrain their brains to be more effective at designing pleasing and efficient system layers?

Seems like them "ui designers" would have to take a completely different perspective - they'd have to look from the inside of the box outwards; instead of looking in from outside the box.

Alan Cooper's "The Inmates are Running the Asylum" opinion is that we can't successfully take both perspectives - we can learn to wear one hat well but we can't just switch hats.

Share

answered Feb 7, 2009 at 19:04

community wiki
igouy

---

2 votes

I think its because a good UI is not logical. A good UI is intuitive.

Software developers typically do bad on 'intuitive'

Share

answered Feb 10, 2009 at 23:30

community wiki
Gregor Brandt

Excellent point, requires the **other** side of the brain
– Chris Ballance Feb 11, 2009 at 15:20

There's no such thing as intuitive for UI. There is similar to what I already know, but that is somewhat different.
– Stephan Eggermont Feb 11, 2009 at 16:25

Nothing is "intuitive" everything is learned. Intuitive is a word thrown around by business people to make programmers sweat. – Bernard Igiri Feb 11, 2009 at 20:15

spoken like a true programmer....and that then is the root of the issue. – Gregor Brandt Feb 12, 2009 at 0:52

@gbrandt Intuitive is some combination "familiar" and "simple". It's still a goal to strive for though. The hard thing about usability is that what is intuitive to some is not necessarily intuitive for others. @Stephan is right that "intuitive" is not some hard "fact", it's relative to the user. – TM. Feb 13, 2009 at 15:28

2
votes

A useful framing is to actively consider what you're doing as designing a process of communication. In a very real sense, your interface is a language that the user must use to tell the computer what to do. This leads to considering a number of points:

1. Does the user already speak this language? Using a highly idiosyncratic interface is like communicating in a language you've never spoken before. So if your interface must be idiosyncratic at all, it had best introduce itself with the simplest of terms and few distractions. On the other hand, if your interface uses

idioms that the user is accustomed to, they'll gain confidence from the start.

2. The enemy of communication is noise. Auditory noise interferes with spoken communication; visual noise interferes with visual communication. The more noise you can cut out of your interface, the easier communicating with it will be.

3. As in human conversation, it's often not what you say, it's how you say it. The way most software communicates is rude to a degree that would get it punched in the face if it were a person. How would you feel if you asked someone a question and they sat there and stared at you for several minutes, refusing to respond in any other way, before answering? Many interface elements, like progress bars and automatic focus selection, have the fundamental function of politeness. Ask yourself how you can make the user's day a little more pleasant.

Really, it's somewhat hard to determine what programmers think of interface interaction as being, *other* than a process of communication, but maybe the problem is that it doesn't get thought of as being anything at all.

answered Feb 10, 2009 at 23:54

community wiki
chaos

2

votes

There are a lot o good comments already, so I am not sure there is much I can add. But still...

- Why would a developer expect to be able to design good UI?
- How much training did he had in that field?
- How many books did he read?
- How many things did he designed over how many years?
- Did he had the opportunity to see the reaction of it's users?

We don't expect that a random "Joe the plumber" to be able to write good code. So why would we expect the random "Joe the programmer" to design good UI?

Empathy helps. Separating the UI design and the programming helps. Usability testing helps.

But UI design is a craft that has to be learned, and practiced, like any other.

Share

answered Feb 11, 2009 at 10:42

community wiki
Mihai Nita

Exactly what I was thinking, How does "developer" relate to "User Interface Designer"?? – Jorge Córdoba Feb 11, 2009 at

18:23

@Jorge here's now it relates: a lot of developers have to design UIs for their employers, even if they aren't designers. At my company, we don't have ANY artists. Our apps don't look too bad. In this case, you've got to have group reviews and stuff to make sure the design is nice. – TM. Feb 13, 2009 at 15:24

@TM: Design is not necessarily UI/UX Design. Sure it might *look good* but that doesn't mean it **works good**. – Kevin Peno Nov 18, 2009 at 22:18

2

votes

Developers are not (necessarily) good at UI design for the same reason they aren't (necessarily) good at knitting; it's hard, it takes practice, and it doesn't hurt to have someone show you how in the first place.

Most developers (me included) started "designing" UIs because it was a necessary part of writing software. Until a developer puts in the effort to get good at it, s/he won't be.

Share

answered Feb 11, 2009 at 20:00

community wiki
MattBelanger

2

votes

To improve just look around at existing sites. In addition to the books already suggested, you might like to have a look at Robin Williams's excellent book "The Non-designers Design Book" (sanitised Amazon link)

Have a look at what's possible in visual design by taking a look at the various submissions over at The Zen Garden as well.

UI design is definitely an art though, like pointers in C, some people get it and some people don't.

But at least we can have a chuckle at their attempts. BTW Thanks OK/Cancel for a funny comic and thanks Joel for putting it in your book "The Best Software Writing I" (sanitised Amazon link).

Share

answered Feb 11, 2009 at 20:04

community wiki
Rob Wells

---

2 votes

User interface isn't something that can be applied after the fact, like a thin coat of paint. It is something that needs to be there at the start, and based on real research. There's tons of Usability research available of course. It needs to not just be there at the start, it needs to form the core of the very reason you're making the software in the first place: There's some gap in the world out there, some problem, and it needs to be made more usable and more efficient.

Software is not there for its own sake. The reason for a peice of software to exist is FOR PEOPLE. It's absolutely ludicrous to even try to come up with an idea for a new

peice of software, without understanding why anyone would need it. Yet this happens all the time.

Before a single line of code is written, you should go through paper versions of the interface, and test it on real people. This is kind of weird and silly, it works best with kids, and someone entertaining acting as "the computer".

The interface needs to take advantage of our natural cognitive facilities. How would a caveman use your program? For instance, we've evolved to be really good at tracking moving objects. That's why interfaces that use physics simulations, like the iphone, work better than interfaces where changes occur instantaneously.

We are good at certain kinds of abstraction, but not others. As programmers, we're trained to do mental gymnastics and backflips to understand some of the weirdest abstractions. For instance, we understand that a sequence of arcane text can represent and be translated into a pattern of electromagnetic state on a metal platter, which when encountered by a carefully designed device, leads to a sequence of invisible events that occur at lightspeed on an electronic circuit, and these events can be directed to produce a useful outcome. This is an incredibly unnatural thing to have to understand. Understand that while it's got a perfectly rational explanation to us, to the outside world, it looks like we're writing incomprehensible incantations to summon invisible sentient spirits to do our bidding.

The sorts of abstractions that normal humans understand are things like maps, diagrams, and symbols. Beware of

symbols, because symbols are a very fragile human concept that take conscious mental effort to decode, until the symbol is learned.

The trick with symbols is that there has to be a clear relationship between the symbol, and the thing it represents. The thing it represents either has to be a noun, in which case the symbol should look VERY MUCH like the thing it represents. If a symbol is representing a more abstract concept, that has to be explained IN ADVANCE. See the inscrutable unlabled icons in msword's, or photoshop's toolbar, and the abstract concepts they represent. It has to be LEARNED that the crop tool icon in photoshop means CROP TOOL. it has to be understood what CROP even means. These are prerequisites to correctly using that software. Which brings up an important point, beware of ASSUMED knowledge.

We only gain the ability to understand maps around the age of 4. I think I read somewhere once that chimpanzees gain the ability to understand maps around the age of 6 or 7.

The reason that guis have been so successful to begin with, is that they changed a landscape of mostly textual interfaces to computers, to something that mapped the computer concepts to something that resembled a physical place. Where guis fail in terms of usability, is where they stop resembling something you'd see in real life. There are invisible, unpredictable, incomprehensible things that happen in a computer that bare no resemblance to anything you'd ever see in the physical world. Some of this

is necessary, since there'd be no point in just making a reality simulator- The idea is to save work, so there has to be a bit of magic. But that magic has to make sense, and be grounded in an abstraction that human beings are well adapted to understanding. It's when our abstractions start getting deep, and layered, and mismatched with the task at hand that things break down. In other words, the interface doesn't function as a good map for the underlying software.

There are lots of books. The two I've read, and can therefore reccomend, are "The Design of Everyday Things" by donald norman, and "The Human Interface" by Jef Raskin.

I also reccomend a course in psychology. "The Design of Every day Things" talks about this a bit. A lot of interfaces break down because of a developer's "folk understanding" of psychology. This is similar to "folk physics". An object in motion stays in motion doesn't make any sense to most people. "You have to keep pushing it to keep it in motion!" thinks the physics novice. User testing doesn't make sense to most developers. "You can just ask the users what they want, and that should be good enough!" thinks the psychology novice.

I reccomend Discovering Psychology, a PBS documentary series, hosted by Philip Zimbardo. Failing that, try and find a good physics textbook. The expensive kind. Not the pulp fiction self help crap that you find in Borders, but the thick hardbound stuff you can only find in a university library. This is a necesesary foundation. You can do good design

without it, but you'll only have an intuitive understanding of what's going on. Reading some good books will give you a good perspective.

Share

answered Feb 12, 2009 at 3:29

community wiki
Breton

2

votes

If you read the book "Why software sucks" you would have seen Platt's answer, which is a simple one:

1. Developers prefere control over user-friendliness
2. Average people prefere user-friendliness over control

But another another answer to your question would be "why is dentistry so hard for some developers?" - UI design is best done by a UI designer.

http://dotmad.net/blog/2007/11/david-platt-on-why-software-sucks/

Share

answered Feb 13, 2009 at 17:05

community wiki
dotmad

---

1 2 3 Next