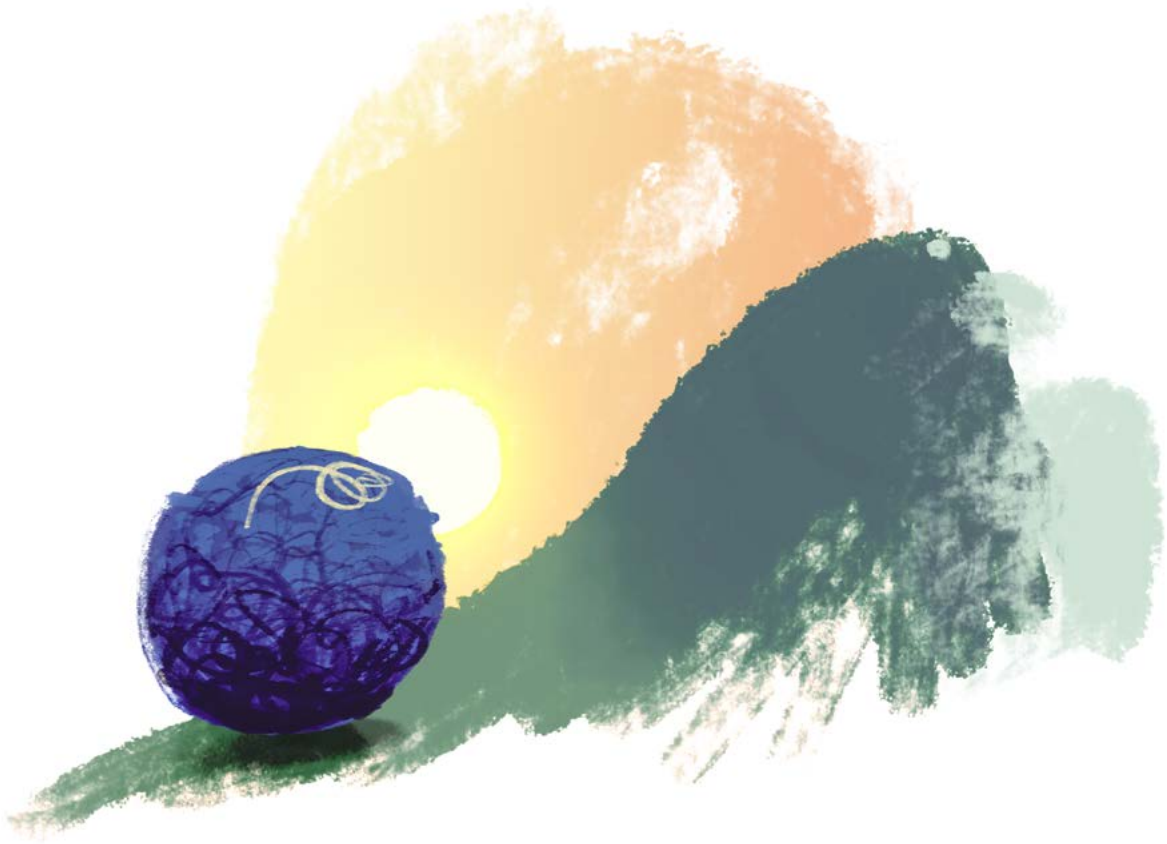
The Modern Digital Tragedy

Why Development Teams Fail to Make First-Rate User Interfaces (And What They Can Do About It)

By Dean Schuster
Partner user experience strategy, truematter

Creating exceptional digital products is not easy. Most development-centered firms focus on technical concerns to the exclusion of usability. As a result, their sites, apps, and software are confusing and difficult to use. They have sacrificed form on the altar of function.

As the competitive landscape increasingly requires more user-friendly products, development teams employ a variety of tactics to improve user interfaces. Unfortunately, these approaches often fail because they don't address core user experience problems.



Section 1.

The Impossible Standard of Digital Product Perfection

We want to make software that is more useful, usable, and wildly accepted. Development teams have worked toward this ideal for decades. Noteworthy digital products are rare because they are so hard to create. Success requires managing endless details, expectations, and circumstances. It feels like a problem that can't be solved.

The Age of Digital Products

We live in a world dependent on digital products. Technology utterly dominates our lives. You can hardly throw a rock without hitting someone using a connected device.

Because we interact with screens all day long, our attention on user interface has moved front and center. This presents daunting challenges for those of us who craft interactive products or services. Everyone prefers that their apps, websites, or software solve critical problems while being amazingly easy to use. Deep down, we also want our products to be head-over-heels loved. Digital products with these traits have a far greater chance to achieve profitability, overall success, and even market dominance.

Development teams have pursued this particular elusive goal since the first lines of code were written. Given the volume of industry chatter dedicated to making better apps, the world should be awash in stellar, intuitive software. But we all know this isn't the case. Despite the existence of the software self-help industry, most web apps, mobile apps, interactive software, and intranets are mediocre at best, and barely usable at worst.

You know you can do better. But there are things in your way.

Where You're Starting

First, let's imagine your digital product fulfills basic requirements for success:

- **It's a strong idea.** Your product has legs. It represents a feasible, workable concept that offers value.

- **The market is ready.** Your digital product fulfills an unmet market need or boasts functionality that serves your market better than others.

- **You're implementing a reasonable strategy.** Your firm has a credible, not necessarily perfect, go-to-market plan.

- **Your product actually works.** It is functional. This should go without saying. But we have to say it.

But assuming your product meets these criteria, the question still remains, "How do you make it remarkable, easy-to-use, and loved?

It can be done.

The marketplace is full of high-profile, well-planned products (physical and virtual). Several come immediately to mind, including Nest thermostat, Lego bricks, the iPhone, Google Maps, Basecamp, Aeron chairs, Netflix, the Oculus Rift, Slack, Airbnb, and MailChimp. Their fame is not merely the result of clever marketing or storytelling. Not all began with fountains of cash. These products succeed because they serve people beautifully. In both the real and virtual worlds, they deftly merge form and function. They are easy to use. The market has embraced them.

Unfortunately, most development-centered organizations do not understand why their software fails to live up to this promise fully.

We're going to dig into the reasons.

The Core Conundrum

Development teams feel burdened with high expectations. Despite efforts to create excellent work on the level of admired products, you probably don't often reach your goals. Instead, you spend endless time fighting the same battles for quality, buy-in, and process over and over again. Fate seems to conspire against you.

This has all the earmarks of an ancient Greek tragedy.

The Story of Sisyphus

Sisyphus, famed character of Greek myth, was a notoriously clever adversary of the gods. His repeated efforts to cheat death ran him afoul of Hades. Finally brought to heel by Zeus, he was condemned to spend an eternity rolling a massive boulder up a hill, only to see it endlessly roll back down. He was forced to endure an eternity of fruitless, pointless toil.

Are we modern boulder pushers?

Creating digital products can feel very much like pushing a massive chunk of rock uphill that never quite gets to the

mountaintop. What developer, program manager, or interactive team cannot sympathize with this everlasting frustration and torment? We are the present-day Sisyphus.

Despite lofty goals, we find ourselves mired in self-defeating patterns. We repeat the same process, take the same actions, and stubbornly adhere to the same standard approaches. Naturally then, software looks and behaves much as it ever has. People respond much as they ever will. Worse, like Sisyphus, we are conscious of our fate. We know it's happening. And we feel like the cycle is forever. But it doesn't have to be.

Your Sisyphean Burden

So you're not dealing with gigantic rocks. You're making digital products. But like Sisyphus, your job isn't easy. Development teams are responsible for software that solves highly complex, intricate problems. This may involve traditional software, mobile apps, or web apps, and a multitude of factors are involved, all of which must succeed.

Development leaders must thrive in each of these (often conflicting) areas:

- **User Need** - Your digital product serves a wide variety of people who have high and ever-increasing demands. They have real-world needs that must be addressed.

- **Market Knowledge** - You are expected to have a profound understanding of the market.

- **Thorough Planning** - Effective software requires extensive planning, from corporate strategy through finished release (and beyond).

- **Stakeholder Expectation** - You have a diverse group of stakeholders to keep happy. Their needs often conflict. Worse, organizational leadership often focuses on short-term utility over long-term quality.

- **Time Pressure** - Good work takes time, yet schedule pressure is a primary driving force for most digital products.

- **Complexity** - Intricate functionality can be exceedingly difficult to build.

- **Requirements** - Requirements must be well-defined, clearly communicated, and agreed upon by everyone involved. This is always tougher in practice than in theory.

- **Usability** - You must balance depth of features and functions with the usability of the final product.

- **Quality** - Everything must work perfectly.

- **Budget** - Creating solid software requires a high investment in a world where margins demand greater and greater efficiency and cost reduction.

- **Competitiveness** - At the end of the day, your product must be competitive in an ever-changing marketplace.

- **Team Building and Dynamics** - It can be exceedingly difficult to find and keep the right people to make great software happen.

An Overwhelming Ideal

Balancing these imperatives is difficult to say the least. It's such a problem, an entire industry is devoted to digital product improvement. Scores of books have been written on product strategy alone. But what about the people who are responsible for actually building the end products? Development teams are not always well connected with corporate strategy, market strategy, or even users themselves. Yet these teams are expected to make consistent and constant progress toward better and better interfaces. They are expected to make interactive products that stand above the rest, products that are as useful as they are usable. Everyone *wants* better usability.

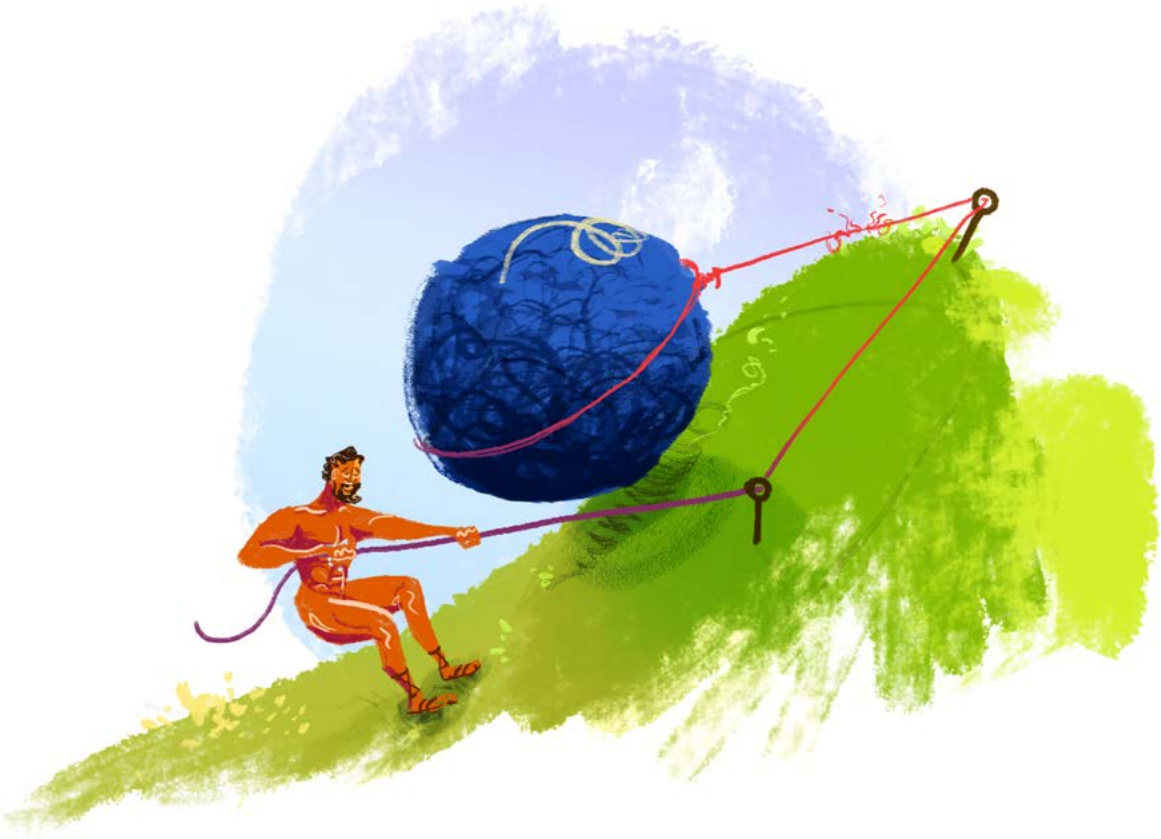
It can feel like too much to bear. It can feel impossible. Perhaps the myth of Sisyphus teaches us to accept our place in the world and embrace the inevitability of failure.

But what if you didn't have to resign yourself to this fate? What if you could get that boulder once and for all to the top of the mountain?

Attacking the Problem

At this very hour, developers are employing a wide variety of tactics to revolutionize what they make and how they make it. They are trying to create better user experiences. To do so, they naturally lean on their unique points of view. Development teams are filled with brilliant, analytical, process-driven engineers. And they play to their strengths.

We're going to examine the things they try, and why they often fail.



Section 2.

Stop Relying on Process Change to Fix Poor UX

Development teams, unsure how to make their digital products more user-friendly and competitive (beyond adding or refining functionality), shake up their processes or internal organization, hoping this will be a catalyst for improving interfaces and usability.

Doing Things Differently

Sisyphus was doomed to endlessly roll his boulder up a hill. At some point, he surely wondered if he was going about it the right way. Maybe he needed to change his approach. Maybe if he pushed the boulder differently, he'd achieve his goal.

Similarly, many organizations feel if they become more efficient or communicate better, digital products will improve. They look inward for solutions and view interface quality as something to be squeezed from better methods. With this in mind, they may try:

1. Refining Processes

Teams expend great effort to do things more efficiently, to achieve superior results. Development teams have been seeking the perfect process since the beginning of the software age. Over time, the names have changed (Waterfall, Agile, Lean, etc.) but the quest remains the same, a more perfect software methodology for more perfect software. Although improved user experience measures are sometimes folded into process tinkering, its primary goal is not to create better user interfaces.

And true, changing a development approach may better incorporate user feedback, yield greater efficiency, and produce fewer software defects. Certainly, development teams flock to this approach.

WHY IT FAILS

Process alone is insufficient.

Refining product lifecycles can be a good thing. But transitioning to a new process is never as easy as advertised, especially for larger, more established organizations. Often the move is partial.

How many teams employ a dysfunctional hybrid of waterfall and agile? Few new, iterative development processes include UX/UI best practices. Most projects continue on without focus on users or usability. A development-centric approach will always yield development-centric solutions.

Let's be honest for a moment. Will any different process inherently improve usability? If you didn't engage users in your old process, you will likely not engage them in your new process.

2. Merging Silos

Most large organizations are hierarchically stratified. Worse, these strata exist in separate business units or divisions, not all of which share information or goals effectively. Marketing usually exists wholly apart from software development, which exists wholly apart from executive strategy.

Compartmentalized organization, a still-prevalent relic of industrial manufacturing, is not the ideal. Flatter structures, at least outside of the public sector, are more in vogue. A company may seek to more fully integrate departments so the organization moves forward with a more unified voice and purpose. Better internal communication and cooperation almost always yields a more coherent approach. Software firms show significant innovation on this front, breaking down barriers.

WHY IT FAILS

Silos endure.

Unfortunately, silos are notoriously difficult to eradicate. Turf wars and politics remain. And even if a development organization becomes flatter and communication flows more effortlessly, does this change the way teams understand or approach user interfaces?

3. Reorganizing

Often, firms shake up internal roles, hierarchy, or reporting structure. This can tangibly disrupt old patterns. It's something of a reset. It can simplify development and may provide a fresh start. This approach is common to growing development organizations.

WHY IT FAILS

Reorganization rarely changes culture.

Culture is not something easily defined. It simply exists. It is built on mission, fashioned by leaders, and carried forward by the larger team. No amount of org chart calculus will transform it overnight.

Great products, digital or otherwise, have their genesis in correspondingly admirable culture. If corporate culture doesn't value the user or their perspectives and needs, interfaces will skew to the mystifying, confusing, maddening end of the spectrum, every single time. It matters little who leads a specific team if the organization doesn't value users at its core. Changing an org chart cannot make apps, sites, or software better. It certainly cannot make them usable. If a company's culture didn't focus on users before reorganization, it won't focus on them after. It doesn't matter what tactic Sisyphus uses to get that boulder to the top of the hill, it's still coming back down.

Change is a means, not an end.

For some reason, organizations think shaking things up leads to better, more beautiful, more usable products. If a team changes itself structurally or begins to work differently, it may become more efficient. It may become a well-oiled development machine. But this doesn't mean digital products will necessarily get better. A team may simply have discovered a more

competent, efficient way to create the same old products. That boulder gets up the hill faster, but it still rolls back down.

To move beyond stagnant interfaces, organizations must do more than change inner workings. Those who understand this, move on to different tactics. They turn their gaze outward.

Next, we'll look at who they get to help push the boulder up the hillside.

Section 3.

Feedback From Customers, Users, and Competitors Can Lead Your Dev Team Astray

In an effort to improve adoption and usability of digital products, development organizations engage customers or users for guidance. They rely on this feedback to plan and change their software, apps, or sites. Though the instinct for advice is sound and input may help, this approach often backfires.

Seeking External Feedback

Let's imagine Sisyphus, rolling that boulder up the hill for the what, zillionth? time. He thinks, "I must be doing something wrong. I can never get this dang rock to stay at the top." Being a worldly, clever man, he decides to seek help. He looks for opinions and advice from people who use or purchase boulders – engineers, landscapers, or perhaps stonemasons. Surely, they can give him some insight about this boulder and this hill.

Development-centered organizations gravitate toward this approach. Customers buy and use the products they make, so they must have opinions about them. Those opinions could be the catalyst to making better, more user-friendly digital products. They hope independent, objective input will help them identify and fix highly specific problems with their apps, sites, or software.

Some tactics include:

1. Establishing Customer Advisory Boards (CABs)

Organizations sometimes gather a group of current (and sometimes prospective) customers together to offer periodic feedback to development teams. Often, a CAB includes participants with a strong (even symbiotic) connection to the development organization. They offer high-level strategic advice and may review products in progress.

WHY THEY FAIL

CABs are inherently flawed.

Customers have vested interests in themselves, not your product. Everyone on a CAB has an agenda, particularly the dominant, important clients. If they can, they will steer a product wholly toward their preferences. What begins with innocuous prodding or desire for specific features devolves into direct visual prescriptions for the interface. This feedback may

be useful, but it's more likely driving digital products toward a laundry list of features that serve few. First-rate interfaces cannot be created this way.

CABs rarely include actual users.
CABs are often populated by account representatives and project managers who specialize in squeaky-wheel diplomacy. Even the most well-meaning clients are focused on their own unique problems. Helping create the best possible interface for the most users is hardly their primary motivation.

Account or project leads typically lack visual design skills, or for that matter, software design skills or experience. When amateurs drive functionality and interaction, feature glut and interface confusion rule the day.

CABs cede control to clients.
Some development teams are forced into this arrangement by management that believes customer direction is the wellspring of product success. Perhaps this is carefully considered opinion. Perhaps it reflects a management fad or has been gleaned from the latest software development book. The answer could be simpler. Managers may be keen to produce better digital products, but they are likely even more desperate to keep a key client happy.

It doesn't matter how well a customer is served (or placated). They will not like the ineffective, confusing digital product they have helped create. After all the work devoted to giving them what they want, they may still leave for a competitor.

2. Organizing Focus Groups

Focus groups are collections of people (often representative users) who gather together to offer direct feedback about software products. These moderated sessions are usually held prior to development (to obtain directional ideas) or after development (to obtain immediate feedback). Interface screens may be shown on screen or printed. Focus groups typically involve group discussion. These groups can be referred to as "user tests," even though they don't involve any interface testing. Feedback gained from these sessions is quite different from that discovered during formal user tests.

WHY THEY FAIL

Focus groups rarely work.

Focus groups are inherently problematic. Some people share opinions more forcefully than others. Groups can be dominated by single, powerful personalities, causing a "group-think" effect. Focus groups work well for reactions to advertising campaigns or films in development, but are unreliable for software usability. People are notorious for their inability to accurately predict their behavior, particularly with interactive products. When focus group participants comment on a completed product, they do so apart from actually using it. Their opinions are speculative at best. Changing an interface based on such feedback could actually harm usability.

3. Querying the Source

When confronted with user discontent, developers attack the problem logically. They point-blank ask users what they want. These are not theoretical interviews. Specific end-users are directly contacted (informally or formally) and asked to provide detailed feedback about what an application should do and what changes should be made to the interface. Prescriptions are plugged-in to the project plan as schedule allows. The development team believes they are now user-centered.

WHY IT FAILS

Users can be inadvertently misleading.

Development-driven interactive products are often difficult to use. Users recognize glaring flaws and suggest changes. Fixes are made. Everyone wins, right?

Talking to users is not wrong. When done properly, it's absolutely right. Unfortunately, it is rarely done properly. No user input should be accepted at face value. Opinions and preferences should always be treated with skepticism. Not all user comment should be afforded the same emphasis or assigned the same importance. Regrettably, direct end-user suggestions are often implemented without deep investigation, provided the development team agrees with the feedback.

While people are generally clear on what they want to accomplish and can identify what annoys them, they almost always lack the perspective and skill to prescribe solutions that truly fix their problems. They communicate as best they can, but don't know when their suggestions make an interface worse rather than better.

User to-do lists are not a solution. Intended to make a product more usable, they can deepen the original problem.

4. Imitating Competitors

When pressured to perform in a competitive market, the most talented teams can be tempted to solve problems precisely as their competitors do. This is particularly true for smaller firms competing with larger, more established organizations. Lacking confidence and courage, they perceive imitation as a shortcut to success.

WHY IT FAILS

Imitation attacks the wrong problem.

Direct reproduction of another product may help a team leap forward, but that leap will have solved another organization's challenges (and for different users). Most small, online stores seek to imitate firms like Amazon, a multinational, enormous company dealing with entirely different economies of scale, markets, and strategies. Their problems and context couldn't be more different.

Further, the grass is not necessarily greener on the other side of the browser. The organization being copied may itself feel rudderless, lacking strong direction. It may have made a mistake. It may already be copying someone else.

Looking outside is not enough.

Copying others represents the least viable option for development teams. Yet, ironically, it is the most often trod path.

When internal teams lack interface know-how they are more apt to rely on external input or sources. Seeking inspiration from customers, focus groups, end users, or competitors makes sense. External perspectives can spark product improvements. Organizations realize they often operate in an internal echo chamber. What better way to counteract this than seeking connections beyond corporate walls?

This input instinct is admirable, but can be counterproductive. When we solicit opinion from outside groups, we assume they'll show us a direct path to more user-friendly interfaces. Unfortunately, input is more likely to be unintentionally or intentionally biased (customer advisory boards), misleading (focus groups), or myopic (end-users). We cannot take this

feedback at face value, though we often do.

We can't pat ourselves on the back because we've pursued external direction for our digital products. We haven't done anything if we are unsure of the usefulness or effectiveness of the feedback we get, and our products are in trouble if we can't interpret feedback into practical interface improvements.

What do we do then?

Well, development teams may think, if we cannot properly learn from customers or users how to fix our interfaces, perhaps we can simply fix the users themselves.

Section 4.

Training and Online Help Can't Fix Poor Digital Product Design

When users complain that digital products are confusing, development organizations often bolster training programs and help systems. In doing so, they mistakenly seek to fix users rather than fixing interfaces. This approach avoids the real problem of poor usability.

Fixing the User

One imagines Sisyphus monumentally frustrated with that infernal boulder. Every time he pushes it to the top of the hill, it defiantly rolls back down. The cycle repeats forever.

In extreme exasperation, he breaks down and angrily yells, "What the %&$@# is wrong with you? Why won't you stay at the top of the hill? Mercury's wings! This is maddening! You are flawed!"

Poor Sisyphus is losing it. He's displaced his aggression onto an innocent object. He is livid with the boulder when he really should be angry with Zeus, who consigned him to his eternal fate. Better, he should be upset with himself. His antics and trickery got him in trouble with the gods in the first place.

We often react this way.
When users find our digital products confusing, we decide the problem lies with the users themselves. They must be somehow flawed. Why else do they struggle with software, apps, or sites that are so obviously easy to use? Essentially, we blame them for interface problems. But because we are enlightened, merciful, and eminently understanding, we don't yell at them. We train them. We offer them help.

Instead of a greater focus on interface usability, development organizations often rely heavily on instruction and explanation. They teach people the proper way to use a system. On the surface, this approach feels obvious. Explaining software, managers claim, is cheaper than fixing it. But over time, underlying issues go unsolved. Interfaces remain confusing and infuriating, even to the people who have been taught to use them.

Neither the boulder nor the users have done anything wrong. Yet, here's what development teams and firms do:

1. Leaning excessively on training.

Walk the halls of development and you may hear statements like this:

- "This interaction is not ideal, but we'll deal with it in training."
- "That's a training issue."
- "We'll teach users how to do that right."

Organizations that rely on software products tend to rely heavily on training. They boast an array of classes, programs, videos, and "onboarding" options. The more mission-critical the software, the more elaborate the instructions. Even simple apps are created with the assumption that users will be taught proper behavior. This is a standard arrangement. Training will occur. It's how things are done: death, taxes, training.

WHY IT FAILS

Training is not a cure-all.

Organizations act as if training is an expansive answer to all software ills. The trainer is the doctor, and the user is the patient who must be fixed. Beyond being flagrantly insulting to users, this deflects culpability for poor interfaces away from the development team. This approach blames the user. It's the user who can't figure things out. They are not as intelligent as those who built the product.

This perspective also leads to complacency. If the user just can't get it, why should you focus on making your products better? Worse, cutting corners is far easier when trainers make it right later. If training solves all problems, why bother making excellent, usable software in the first place? Why try harder? Welcome back to our vicious Sisyphean circle.

Training is limited by subject matter.
The best trainers may not train textbook use of the system. More likely, they identify hyper-efficient "work-arounds," tricks and short-cuts that circumvent idealized use. Work-arounds are a tacit admission that an application is flatly difficult to use. Trainers know this. They spend significant time with users and hear the litany of complaints. They openly criticize the user-friendliness of interfaces, perhaps to avoid the embarrassment of association.

Let's remember, users feel helpless in the face of confusing software. Patient, clear training rescues them. Trainers are loved. It's not a trainer's fault if most interactive systems are terribly difficult to use. No amount of stellar instruction can make a bad digital product into a good one. Training is decidedly not the problem. Reliance on training is. When we presume teaching makes up for poor usability, we offer resigned acceptance of poor, underperforming interfaces.

2. Creating robust help.

Everyone makes mistakes. We fail to understand the simplest software. Even power users encounter problems when using interactive systems. This is normal. When we get stuck, we need targeted, concise guidance, and we need it fast. For this reason, nearly all apps, web sites, and software provide some form of "help." It is a non-controversial best practice. But where to start? What kind of help? A dizzying array of options and techniques exist, including:

- Help Sections, Menu Options, or Help Desk (online and offline)
- Contexual Help, Inline Help, or Tool Tips
- In-Person, Real-Time Assistance (really)
- User Manuals, Style Guides, or Knowledge Bases (online and offline)
- Wiki, Chat, or Bots (oh my!)

From this list, a future information archeologist might deduce the information age was either chock-full of mystifying products or inordinately dense people. Based on the sheer volume of choices, they'd at least infer help must have been extremely important.

WHY IT FAILS

Help is usually not helpful.

Help, conventional wisdom tells us, exists to explain interfaces. Organizations embrace this virtue, creating voluminous help content. Users routinely encounter copious explanation: paragraphs of tool-tip text, endless F.A.Q. pages, blisteringly expansive help sections, and biblically long introductory copy. Development teams act as if content alone equals comprehension. They assume people devour user manuals in full, and everyone loves reading online. If some is good, more is better.

But when is the last time you found help, well, helpful?

People seek help as a last resort, when nothing else works. But what if the glorious help content (if or when found) is just as confusing as the interface? Rules for appropriate content and baseline usability don't stop at the entrance to the help section. People react poorly to jargon-heavy copy, assumed knowledge, poor typography, and annoying interactions wherever they are found.

Sadly, most help features would not pass a standard usability audit. Just because help exists doesn't make it inherently useful or remotely effective. When we add help for its own sake, we are merely painting by the numbers. Ostensibly understanding and courteous, we are merely making another lazy assumption.

Maybe it doesn't matter. Maybe people just aren't very smart. They'd be lost even if our help were perfect. Perhaps so. But this again deflects attention from the real problem. What is more likely: all people are dullards, or most software is bewilderingly complicated and most help content just as needlessly confounding?

Training and Help Mask the Real Problem

As end-user frustration increases, organizations must do something. Increased attention on training and help are logical, positive responses. Training and helping users isn't wrong, but relying on either instead of fixing interfaces can be disastrous.

Misplaced Trust

How can self-evident best practices go astray? How can teaching enable poor interfaces? The danger comes from confirmation bias. When we want something to be true, we believe it to be true.[1] Our pre-conceived notions affect our opinions. We believe training and help must be beneficial. Why? Because we know help is helpful. We know training works.

What if this isn't true? If help and training made interactive experiences intrinsically wonderful, most organizations would have happier, more satisfied users, adoption rates would be off the charts, and customers would never flock to the competition. Something else must be going on.

That something is poor user experience. When we assume the interface cannot be the problem, or is only a minor issue, we

1 Heshmat, Shahram Ph.D. "What is Confirmation Bias?" *Psychology Today*. Sussex Publishers, LLC, 23 Apr. 2015. Web. 10 May. 2017.

blame users instead of ourselves for our products' shortcomings. When we assume training or help will compensate for bad design, when we refuse to acknowledge and fix deep rooted usability problems, we are in denial.

Misplaced Blame

Sisyphus is beside himself, absolutely certain the boulder is at fault for his predicament. But just as he cannot blame a boulder for being a boulder, we cannot blame people for being people. No matter how they are helped or trained, people will bristle against confusing, difficult interfaces. Digital products cannot be fixed with expensive help systems or training programs. They must be fixed at their core.

Training isn't the only afterthought technique development firms use to try to make their products better. We'll explore another tactic that often comes too little, too late next.

Section 5.

Pretty Colors and Pictures Won't Make Your Digital Product's Experience Any Better

Sometimes organizations think their interface flaws are visual. The problem is the product is ugly, and someone just needs to make it pretty. Development teams try, but because they often have a limited understanding of the relationship of visuals to user experience, their efforts fail.

Making It Pretty

Exhausted and perhaps delirious from his strenuous efforts, Sisyphus is making odd choices and now thing are really going downhill. He has resorted to art.

For some reason, he believes that changing the look and feel of the boulder will solve his eternal problem. If he makes it attractive enough, that massive hunk of granite will leap to the summit and stay there. He will finally be able to rest in peace.

It seems everlasting punishment makes one do strange things.

Are we doing something eerily similar?
Much like Sisyphus the artist, development teams routinely believe they can improve interfaces by making them look better. They consider this view completely rational. In this model, visuals fuel and determine user experience. Visual fixes can thus work miracles. Everything development-centered organizations do to improve visual interfaces stem from this belief. Here are the tactics they pursue:

1. Enlisting developers as visual designers.

Few development-centered shops can boast in-house creative talent. When no visual professionals are available to work on user interfaces, someone else must step up. Usually, the most visually attuned developer is tasked to create an app's graphical user interface (GUI). Often, a group of developers create the interface in ad hoc fashion.

Based solely on technical requirements, the developer builds the GUI based on their power user point of view. The interface is entirely subordinate to functionality, and is created either on-the-fly or after core development is fully complete.

WHY IT FAILS

Most developers are not visualists.

Someone must craft user interfaces, but expecting left-brain dominant, analytical engineers to be visual designers is a recipe for disaster. It is doomed to fail. Developers don't make things pretty. It's not their job. It's not what they think about. They have not been trained for it. They focus on the functional requirements and quantifiable tasks.

Real disparities in talent and inclination exist between developers and visual designers. Though developers may sense an interface isn't ideal, most can't see precisely why something isn't right visually. They realize this, so they don't like to wade into the right-brain pool. Instead they prefer to solve problems as any respectable left-brainer would.

The trouble occurs when developers are forced into the roles of visual design, information architecture, or content strategy, creating the classic wrong tool for wrong job scenario. This happens frequently in development teams. It's one of the chief reasons most software coming from development-centered organizations is so frustrating and confusing.

Granted, some developers are more able to create "aesthetically pleasing" software than others. However, this elusive developer/designer hybrid is remarkably rare and, when you find it, usually more focused on front-end development.

Power users create dreadful interfaces.

When back-end developers create user interfaces, bad interfaces result. Unfortunately, poor visuals are the smallest worry. Usability will be based on a developer's power-user mentality. The system will work, but mere mortals will be flummoxed and helpless before the clunky interaction. Everyone involved knows this, but feels trapped by budget, leadership, clients, feature-set,

use cases, requirements, product path, competitive pressure, or other stress du jour.

Imagine a performance automobile created only by engineers, without constant collaboration with industrial designers and automotive artists. Can you make a Porsche this way? Maybe, but the odds are not in your favor.

2. Hiring a freelance graphic designer.

When development teams feel visual pizzazz is needed, the team may look outside the organization for help with the graphics. This approach will get the organization limited project-based help, but the results will be directed by developers. All work will continue to proceed from a development perspective.

WHY IT FAILS

Few graphic designers have real interface experience. Most graphic designers are print designers by training. They create identity systems, brochures, signage, posters, or annual reports. They may possess years and years of experience and win industry awards for their work. This does not mean they have the skills to design software or app interfaces. Interactive systems are qualitatively, wildly, monstrously different from print or advertising. Traditional designers do not readily grasp the nuances of non-linear, online systems. Further, their old-school perspective will prevent them from relating to and understanding developers.

Graphic designers are caught in the beautification trap. Lacking interactive experience, print-focused designers do the best they can. Often, they beautify the existing user interface after it has been developed, reproducing a classic development error. Their embellishments may be better than a developer's, but they're still just embellishments, lipstick on a pig. Core usability and the nature of the user's experience remain

fundamentally unchanged. Beautification cannot solve software problems.

3. Asking marketing for help.

Some development shops have active, even large marketing departments that offer graphic design expertise. Collaboration here feels logical. Sometimes marketing assistance is forced upon development by desperate executives.

WHY IT FAILS

Marketing and interface design are radically different.
User experience and usability flaws are often seen as visual problems. Marketing departments are regarded by product developers as specialists in visual problem-solving. Certainly, in-house marketing teams usually include people with graphic design skills. Why not give marketing a crack at the user interface?

Ostensibly a logical step, a marketing-focused approach rarely works for software or app interface design. The skills needed to market the company strategically and visually don't translate well to digital product design. The rules for interactive usability have very little to do with traditional marketing. Best practices for digital content and interaction are often diametrically opposed to best practices for traditional product marketing (and even digital marketing).

Marketers seldom bridge the technical gap.
Like traditional designers, most marketers tend to exhibit an imperfect grasp of the technical issues driving UI design. The best marketers know this about themselves. Conversely, well-meaning, naive marketers feel their understanding of graphic design will allow them to create usable, elegant interfaces. Caught in the beautification trap, they can end up making an underperforming UI worse.

These problems reinforce the general (and unfortunate) divide between marketing and development. These two groups, thrust into a common arena by the advent of the commercial web, are like parties communicating in different languages without the benefit of an interpreter.

4. Calling in an agency[1].

When a project feels too large for a freelancer or the marketing department, or if working with either has failed in the past, organizations may bring in outside firms to complete interface work. Consultants come in all shapes and sizes, from graphic design firms and advertising agencies, to web design shops and UI specialists. These firms fill the developers' needs as best as possible based on their proclaimed UI skillset. Finding firms requires referral, searching, interviews, and heaven help us, RFPs. Cost drives choice.

This approach can offer limited or even solid success, but can be fraught with pitfalls.

WHY IT FAILS

Consultants are not magicians.

Despite what you may hear in elaborate RFP presentations, consultants cannot perform the impossible. They are limited by their expertise and scope of work. They cannot be anything other than what they are, and they can only do what you hire them to do. The very act of bringing on a consultant does not guarantee success.

The process can backfire.

Finding a solid creative firm with UI experience can be difficult,

---

2 Truematter is a UX/UI consulting firm. We have a vested interest promoting user experience strategy and design services. In articles like this, we offer our expertise freely and openly.

particularly in smaller markets. Traditional or digital agencies may claim user experience expertise, but nearly always lack it. Their expansive portfolios typically include websites, web marketing, SEO, social media, branding, print, PR, and advertising services (to name a few). They try to be everything to everyone. They bring an agency knife to a UX/UI gun fight.

Short-term thinking rules the day.
Consultants are often brought in for one-off projects. Assuming the right skillset and focus, tangible value can be gained from limited engagements, and they can yield significant improvements. However, the project-based approach is superficial. It can only solve limited interface problems, and by definition cannot address underlying systemic issues (the source of most UI problems).

No consultant can, by themselves, transform a company into a user experience powerhouse. This is especially true if the organization's leaders fail to recognize the strategic nature of investing in usability, or are unable or unwilling to move culture in a user-centered direction.

Consultants move on.
Even if hired to catalyze change, consultants are temporary. They come and they go. When a talented services firm has left the scene, are internal teams better off, or do they utterly falter? Does an organization return to previously unsuccessful tactics? Success depends less on the consultant and more on the organization's motivations, commitment, and leadership.

5. Hiring a dedicated visual designer.

Teams burned by one or more of these tactics may attempt to build design muscle internally. They may hire an interface design resource permanently dedicated to interface work. This move is more typical for larger development shops or teams focused on websites or mobile apps.

Hiring a professional interface designer (rather than a graphic designer) is a solid step in the right direction. It shows long-term commitment to making better digital products. However, even this step may not lead to the changes the development firm desires.

WHY IT FAILS

Finding the right designer is not easy.
Most software creators are development-centered firms with development-centered instincts, biases, and processes. Development firms often cannot differentiate solid interface designers from mediocre interface designers. Since the first UI hire will become the de facto interface guru, this initial professional is pivotal. Hiring an excellent designer can require luck. Hiring a so-so designer is easier, but can lead to a false sense of progress.

A dedicated designer may have to swim upstream.
Even if the hire is excellent, development firms find assimilating interactive designers problematic. They don't know where UI specialists fit, so they often tack their contributions on to the end of the process (after planning, definition, and development are largely complete). Even a highly talented professional can only do so much to improve a complete or nearly complete product. They can change the veneer, but the substance below the surface, where user experience is determined, will remain unchanged.

Too many shops can't overcome this institutional inertia. Lost in the paradigm that high-quality visuals (created after the fact) lead to better usability, development firms relegate UI professionals to second-class status. Interface designers become order-takers serving an established process. Great effort and expense has yielded the equivalent of an exceedingly fancy bandage on a recurring injury.

When a lone visual designer is stuck navigating development culture, they fight two major battles. They fight to create excellent interfaces, and they fight to be allowed to create excellent interfaces. Either fight could be all-consuming. In such an environment, the more talented the interface specialist, the more likely their stay at the firm will be short.

Visual fixes don't address systemic problems.

Some of the words and phrases in this article intentionally reflect real terms development teams use when talking about user interfaces.

- Look and Feel
- Graphical User Interface
- Beautification
- Pretty
- Aesthetically Pleasing
- Visual Pizazz
- Lipstick on a Pig

These are flighty terms describing a veneer or patina. And looking at design this way is a serious problem for usability.

Misunderstanding Visual Design
Most engineers and left-brainers view visual design as an entirely subjective façade. When a thing is subjective, it is art. It is merely beauty. Because engineers tend to think this way, they do not esteem visual design. Unsurprisingly, they fail to understand its proper role, often with devastating effects on user interfaces.

If visuals are only embellishment, why do most development

teams insist on fixing flawed interfaces through “beautification?” Can visual design simultaneously be window dressing and the key to improving interfaces? No. It cannot be both.

In fact, it is neither.

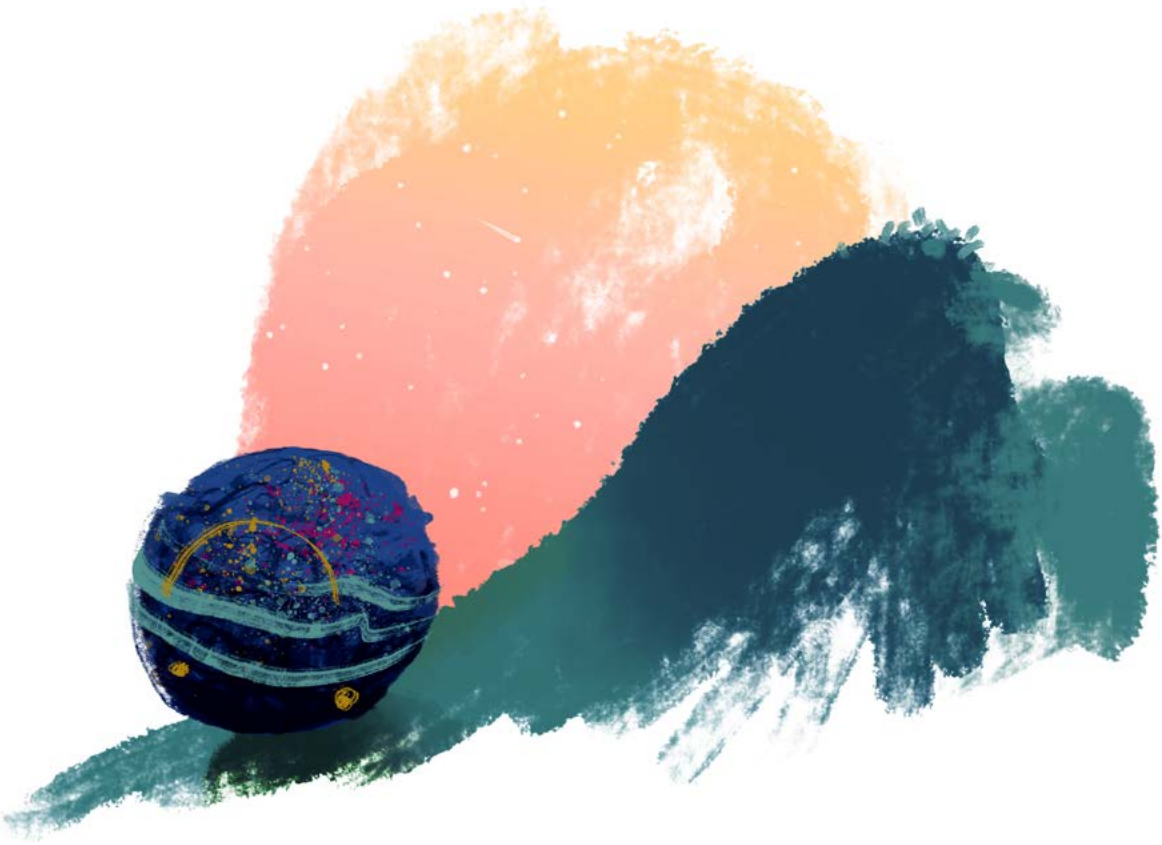
Why Visual Fixes Fail

Visual design is far more valuable to interfaces than most realize, yet it is no miracle elixir. User interfaces are only the tip of the iceberg. If you want to make a digital product better and easier to use, the entire user experience must be addressed. This includes function, form, content, context, and interaction, all of which are linked inextricably. Focusing on visuals fails because development teams misunderstand what it takes to make high performing sites, apps, or software. It takes a holistic approach.

This is why employing freelancers, working with the marketing team, bringing in an agency, or even hiring an interface designer can so often flounder. If an organization devalues and misunderstands the role of design and nature of interfaces, nothing any of these folks do will succeed in the long term.

A Common Thread

We’ve looked at a number of ways development teams attempt make their digital products more user-friendly. Changing internal processes, seeking external advice, hyper-investing in training, or upgrading visuals are frequently employed. Unfortunately, they will not solve the problem of poor user experience. Without a complete change in perspective, they cannot.



Section 6.

Why Development-Centered UI Approaches Fail

As development-centered firms pour their energy into functionality at the expense of usability, their digital products grow more complicated and confusing. Their efforts to fix user suffering rarely succeed because they don't address core problems.

Getting the Boulder Up the Hill

Sisyphus is right back where he started. He has tried everything. Nothing works. Every day that massive rock sits at the bottom of the hill. Every day, he rolls it to the summit. Every day, it rolls back down. The next morning, he begins again. Sisyphus is playing out an ancient tragedy. He will never succeed.

It's different for us. Unlike Sisyphus, we can reach our goals. Our predicament is not eternal. The boulder represents our digital products. Arduously rolling it up the hill represents our efforts to make that product truly exceptional, loved by users, and spectacularly accepted in the market. When we get the boulder to the top of the hill and keep it there, we've achieved something remarkable.

It's not easy.
We want to make more friendly and usable products, but face unyielding demand for advanced, complex functionality. Competitors constantly push the envelope with better interfaces, simpler interactions, and stronger usability. We know our products must get better. They must evolve. That's why we work so hard at it.

But that's the thing. Our efforts often fail.

We pull out all the stops to improve interfaces. We refine internal organization or processes. We seek help from external sources. We teach people to use our products correctly, and we try our hand at visual design fixes. Unfortunately, each of these approaches have one thing in common. They don't work.

To be sure, some tactics may help slightly, producing uneven results. Some initiatives may yield short-term benefits. Most fail outright. The methods we choose simply don't create fundamental change.

And fundamental change is needed if we are going to emerge from the interface dark ages.

The question remains. Why do we fail?

Development teams are locked in a faulty, incomplete perspective.
Developers, program managers, and software architects do what they have been trained to do. They know how to program, prioritize functionality, plan systems, and manage development projects. This focus on function, while completely understandable, is ultimately myopic. Development teams experience precious little training on usability, user-centered design, or visual interfaces. Can we blame them for lack of understanding?

Development leaders know how to run software firms. They don't know how to incorporate effective UI, what they call "art," into their processes, estimates, or deliverables. They don't know how to identify and hire the right UX or UI people. Nothing in their business experience has prepared them for the new competitive interface landscape. Their perspective prevents them from seeing beyond standard approaches.

Some firms refuse to change.
With the advent and universality of the commercial internet, the world has transitioned from the era when software can be made solely with development resources. Competitive digital products require broader skill sets. Even if they agree in theory, in practice too many firms simply do what other development firms do. Talented developers can be trapped in organizations that blithely accept poor usability. It is par for the course. These firms think they're doing just fine, thanks. They don't want to embrace a broader point-of-view and so are vulnerable to savvy competitors. Rarely diverging or evolving, they repeat the mistakes of old methods. Their products ossify. Our Sisyphean boulder rolls back down the hill.

Changing Your Perspective

No one sets out to purposefully create unintuitive, barely tolerated digital products. If development teams can embrace a broader point-of-view on what makes interfaces truly great, they can thrive. They can radically change their organizations for the better.

What must development teams embrace to do this? What must they unlearn? What is the fundamental change that must take place in leadership and corporate culture that will break the cycle of unusable digital products?

We tackle that next time.

About truematter

Frustrating screen experiences are everywhere. You deal with them, we deal with them, our older relatives deal with them, and they make us all want to take a hammer to whatever device we're using.

Truematter exists to make all of our lives easier any time we have to deal with a website, app, or piece of software. Our team is always thinking about how to improve user experience to help create digital products that are usable, useful, and loved. You can read more of our thoughts at blog.truematter.com.

Credits

Author
Dean Schuster

Editor
Bailey Lewis

Illustrator
Daniel Machado

Whitepaper Designer
Rannah Derrick