The Grand Design in Improving Agile Success

Larry Marine
Success PragmatiQ
LMarine@SuccessPragmatiQ.com

Theo Mandel, Ph.D.
Success PragmatiQ
TMandel@SuccessPragmatiQ.com

You don't need to be told that user experience expertise can vastly improve your application's design. Then why, asks two UI experts, isn't design front-and-center in the Agile development process?

While many companies are embracing Agile-style development processes, few seem to be achieving the envisioned successes that initially drove them towards this new development process. Some might too easily wag a finger at developers, when in fact the crux of the problem lies with those who request products from the developers. You know the old saying: garbage in, garbage out.

What is Design?

One reason many Agile teams don't achieve any greater product success than they did with previous development efforts is that many companies continue to rely solely on their developers to design their products.

The problem stems from the misunderstanding of what developers do and what needs to be done. Developers are developers, not designers. And Agile is a development process, not a design process.

Moreover, the term design is often misused in the context of product development. There are three common forms of product design: Visual, technical, and solution design.

Visual Design is the artistic aspect of a product design. It is also referred to as graphic design, which focuses on establishing a visual language that appeals to the users. Visual design may also include the industrial design aspects of the product, as well.

Technical Design is the engineering form of design. To engineers, this is the guts of the product that users typically don't see, such as the code or how a function is engineered to perform behind the scenes.

Solution Design… ah, that's something else again. Einstein is quoted as saying that given one hour to save the world, he would spend 55 minutes defining the problem and the last 5 minutes solving the problem. In most cases, developers solve the wrong problems. They solve the wrong problems very well, but it's not the developer's fault. Given inaccurate or vague problem definitions, how can a developer be expected to create an accurate and precise solution?

Solution design depends on an accurate problem definition and an accurate solution related directly back to the problem. Solution design also addresses all user interaction and navigation when using the product.

This is the most underrepresented definition of the term “design.”

When product managers ask developers to design something, they are thinking in terms of requirements and features, as well as visual design, while developers are thinking in terms of code and functions. Product managers are considering how the software will look while developers are concerned with how it will act. They are both using the same word, but in different contexts.

How many times have you been on your way out of an office building or hotel and noticed the nicely designed glass door with a chrome handle? You reach out to the handle and grasp it, barely noticing how nicely the handle fits your hand, and, as you would expect to do, you pull on the handle – but the door doesn’t open. Then you see it: the little sign that says PUSH. Then you twinge at the notion that you can't open something as simple as a door.

But it’s really not your fault. If something as simple as a door requires instructions, there is something inherently wrong with that design. While the door is visually appealing, the solution is not aligned with the problem.

Solution design not only includes, but depends on an accurate problem definition. The problem definition is based not only on the technical problem, but considers the human factor, or cognitive problem, as well.

The door-handle problem might have been stated as “Provide a mechanism that only allows the user to open the door outwards, to comply with federal fire safety regulations. Successful operation of the device must not depend on any instructions and the system must avoid any user errors.” You can guess that door handle would not have been the first solution that the developer would have considered. A developer would more likely consider a flat panel, which invites the only correct user action to push the door open.

But you can’t rely on a more accurate problem definition. You must also design a solution that focuses on the problem. Leaving developers to “design” a solution for the stated problem allows them the flexibility to design a solution based on their understanding of the technology, not necessarily (or typically) the users and their needs.

Of all the different groups in an organization – business, marketing, training, customer support, sales, and development – which group has the least amount of knowledge of or insight into business objectives, marketing objectives, and user objectives? You guessed it: developers. Companies hire developers not for their “people” skills and business acumen, but for their technical prowess. So why do so many companies entrust their product designs with the very same people that they would not trust to run their business or marketing efforts?

This isn't simply a rant. It's a suggestion on how to improve the Agile development process by improving the lines of communication between those who need the product and those who build it.

Define the Problem Well, First

This actually is much easier than you might guess, but is quite poorly done in most organizations. Most companies don't put enough effort in defining the problem and typically base their problem definition on unsubstantiated assumptions. And those companies that do conduct some degree of user research tend to perform the wrong kind of research. Any type of self-reporting method, such as focus groups, interviews, surveys, and journals rely on the users' ability to accurately describe their problems, which they are actually not very good at.

There's a well-worn quote by Henry Ford that goes "If I had asked my customers what they wanted, they would have said 'faster horses.'" While trite, the premise is still true. Users are more apt to describe incremental improvements to existing solutions than they are to accurately define the problem.

The most successful method of user research is ethnographic observation. This is not a time and motion study, but a cognitive analysis to gain insight into the users' cognitive perspective through observable behavior.

The power of observation versus interview is demonstrated in a redesign effort we performed on the e-commerce website ProFlowers.com. We observed men buying flowers in several brick and mortar stores and realized that men know very little about flowers (duh…). They would stare at the display case until someone would ask what they needed. The men would typically respond with something like "I need to buy flowers because I forgot my wife's birthday."

And there it was. The ah-ha insight: men buy flowers for a reason. We designed the ProFlowers website around occasions and bouquets, not flower types. That simple solution design helped them enjoy some of the highest e-commerce conversion rates for the past 10 years.

At the time we designed the ProFlowers site, there were about 1,200 on-line florists, all selling flowers by flower type, not bouquets organized by occasion. ProFlowers is the most successful e-florist and one of the most successful e-commerce sites in the world.

Interestingly, the developers did not follow our initial redesign because "nobody else was doing it that way" and their site was a miserable failure. The ProFlowers executive staff then made the developers rebuild the site in 1998 following our design and it has succeeded ever since.

Design a Solution Concept

The most common point of failure with Agile implementations is born from the misperception that you *can* design while you build. In theory, that *is* true, but the reality is that designs resulting from such efforts are no better than any other engineering-

driven design efforts and are more often than not relative failures from a user experience perspective. Such poorly designed products rarely achieve any sustainable success and are prime targets for competitors.

Designing while you build is basically implementing tactics without a strategy. Many aspects of a product require a certain amount of research, prioritization, and strategic planning. As Lewis Carroll said, "If you don't know where you are going, any road will take you there." Without a plan, many products end up including features and functions that tend to justify the product more than serving the users' tasks or needs.

You would not expect bricklayers to start laying bricks before the architect provided the blueprints, so why expect developers to start writing code without knowing what the product should do? Too many strategic issues typically cannot be addressed in the short amount of time available in a typical Agile sprint.

The pre-development work should provide a general conceptual plan in the form of design wireframes, not narrative requirements descriptions. A more visual set of design blueprints, with brief explanations of how a feature should behave, is a much more successful design communication device than are the typical requirements given to developers. Those guidelines are open to interpretation and usually results in designs that only other programmers can understand and use.

This conceptual solution design must be based on the problem definition, not on individual whims, if the product is to succeed. This high-level design need only capture the general flow and organization of the product, which typically accounts for 70-80% of the design. The remaining 20-30% of the design is completed in the design sprints, which should precede the development sprints. Very little design should actually occur in the development sprints. Development should occur in the development sprints.

Add User Experience Design to the Process

Incorporating Agile processes beyond just the development efforts includes creating an Agile-oriented solution design process, as well. Given that the high-level conceptual design provides 50-80% of the design up front, the interaction design team can complete the last design efforts in sprints, much like developers complete code in sprints. The difference is that the interaction design must occur one sprint cycle ahead of the development sprint. So while the developers are coding out in their sprint 3, the designers are designing in their sprint 4.

The design team may also need to help the developers with a specific design issue during a development sprint, but that usually is only a minor issue and usually doesn't impact either sprint too much. The development team may also need to assist the design team to help determine the technical feasibility of a design approach, but again, this only slightly impacts either team.

This is quite different from the typical Agile implementation in that design and development are performed in the same sprint cycle. It is imperative to separate the two efforts since they each rely on different skill sets and priorities.

Allow for Correction of Poor Results

What's the point of testing the UI or the code if you aren't going to make time to fix any problems? Each sprint is supposed to have a certain amount of room built into the schedule to redesign or recode something that tested poorly in a previous sprint. Unfortunately, this is a common corner that most teams cut. One means of addressing this is to create a priority list of sections of the product that must be built to serve as a roadmap that is continually reviewed with respect any new concepts or items that need additional effort (rework). Rework items could be reinserted into the priority list to ensure that they are addressed in a future sprint. The items that will be addressed in each sprint are derived from the highest priority items.

This is another advantage of having quite a bit of the conceptual design done prior to any coding since it provides an informed picture of what needs to be done and identifies any dependencies.

Our experience suggests that the most successful approach to implementing Agile processes is to recognize that design and development are NOT the same thing. Somewhere between old-school Big-Design-Up-Front (BDUF) and pure Agile Design-as-You-Build methods is an integrated process where 80% of the design is done up front and the rest is done in design sprints that lead development sprints. Designers and developers also need to interact with each other prior to each sprint to ensure the feasibility of a design before submitting it for development. Separating the workload between design and development leads to even more efficient coding since the developers have less to do in a development sprint and can focus on the technology without being burdened to design the interface *and* the technology. And isn't that what Agile is intended to achieve – more efficient development?

About the Authors

Theo Mandel and Larry Marine have formed a new company, Success PragmatiQ, combining their decades of user experience consulting expertise to guide clients integrating successful UX practices with evolving development environments, such as stage-gate and Agile.

Theo Mandel, Ph.D. leveraged his 11 years at IBM, conducting user research and establishing user interface guidelines and standards for the PC platform, to launch a 20 year successful consulting career as a well-respected expert in user experience design and usability. Theo is the author of two well-known books. **The Elements of User Interface Design** was one of the first interface design textbooks translated into Russian.

Contact Theo at TMandel@SuccessPragmatiQ.com or 480-664-1202

Having learned from the master of usability (Dr. Don Norman), Larry has set himself apart as a thought leader and successful practitioner in product design. His mix of enterprise software, website, and medical device clients includes such luminaries as Proflowers, Novartis, American Airlines, Cardinal Health, and FedEx, to name a few. Such a diverse mix provides him with unparalleled design perspectives that result in truly innovative and profitable solutions.

Contact Larry at LMarine@SuccessPragmatiQ.com or 719-488-4566

SUCCESS
PRAGMATIQ