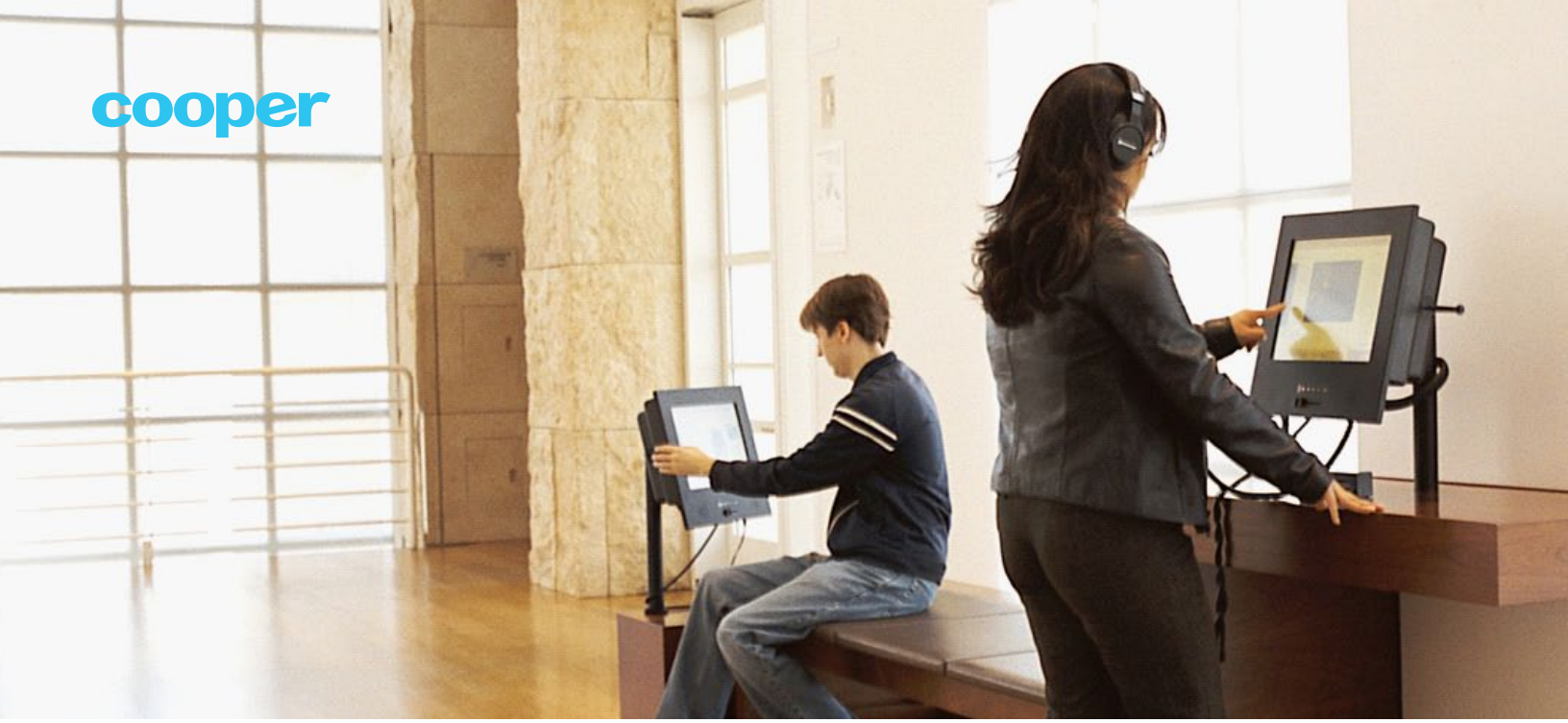
cooper



An Insurgency of Quality

Alan Cooper

Code Freeze 2009
Minneapolis, Minnesota

© 2009 Alan Cooper

Introduction

My name is Alan Cooper. As a pioneering independent software developer since 1975, I've written lots of code and created several successful applications. Many of you know me as the "Father of Visual Basic". I invented much of the underlying technology of Microsoft's software development tool and sold it to Bill Gates in the late 1980s.

In 1992, I founded what was arguably the first consulting interaction design company. Over the last 17 years, we've worked with literally hundreds of companies, large and small, on products of every conceivable variety. Located in San Francisco, Cooper continues every day to tell the user's story to practitioners and business people in a useful and actionable way.

I've made several significant contributions to the field of design. Many of you have used my invention, design personas. My books, including The Inmates are Running the Asylum, have helped to define a rigorous discipline of software design.

For the last two years I've focused my attention on the growth and success of agile development methods. There is nothing in the history of software quite as significant as the agile revolution. While I'm thrilled by the awesome potential of this new way of thinking, I remain aware that most revolutions in history have been co-opted and have failed to live up to their potential.

Alan Cooper
Chairman
Chief Strategy Officer
Cooper

We are here to celebrate our success

We are here today to celebrate the success of agile development.

The craft of agile has clearly demonstrated its immense power to enhance software development. It has proven its ability to deliver better quality software, in far less time, and with happier teams.

Similarly, the craft of interaction design has demonstrated great power to enhance the quality of the software user's experience. Skillfully applied, interaction design can also speed the delivery of better quality software, in less time, with happier teams.

"Skillfully applied, interaction design can also speed the delivery of better quality software, in less time, with happier teams."

Courtesy flyingwithoutfear.com

We are here to make mid-course corrections

We are also here to consider what mid-course corrections would improve the results of our work. Today's search for betterment is in no way a criticism of our existing practices, and any apparent disapproval you may hear is simply me grappling with the big problem of more effectively integrating two groups of equally intelligent, headstrong practitioners.

Courtesy http://rcrawford79.wordpress.com/

Terminology wrestling

An unfortunate footnote to all of this success is the constant wrestling with terminology, titles, and practices in both the design and agile worlds. Agilists argue interminably; about the differences between agile coaching and Scrum Mastering; which is better, Scrum or XP? Or whether you are really agile if you don't pair program.

Designers argue endlessly about the differences between User Experience Designers and Interaction Designers; which is better, contextual enquiry or goal-directed design? Or whether personas are real or just made up.

Like all of you, I have a position in these battles. I'm in support of what works best.

In the design world, I have seen a clear difference between those design practices that are craft-based and those that are art-based. The latter is based on someone's opinion, while the former is based on the demonstrable improvement in the actual end user's experience.

"In the design world, I have seen a clear difference between those design practices that are craft-based and those that are art-based."

Terminology wrestling

As in all crafts, there exists a broad collection of skills and techniques, and the craftsman uses the appropriate subset to solve the problem at hand. It isn't about style or authority and it isn't about ego; it's about synthesizing the correct solution. For over 15 years I have used the term "interaction design" to describe authentic craft-based design, and that is the term I will use in this talk to differentiate it from other, less effective forms of design.

This is not to say that what someone calls "experience design" isn't an effective craft-based method. You say potayto and I say potahto. It's just that there are many practitioners who, although they have the right words, they lack the fundamental imperative of all agile, craft-based methods: the success of the end-user.

Not process for process' sake

Agile's success comes from its ideals more so than from its practices. Our focus cannot be about process for process's sake.

Over the last decade, several groups of developers independently derived the basic practices of what we know today as agile. These methods showed promise, but weren't yet a coherent approach.

The explosive success of agile only came when a cadre of responsible craftsmen came together and stated that their goal was to build products that made the end user more successful.

Responsible craftsmanship

The Agile Manifesto states that “Our highest priority is to satisfy the customer”. The document clearly communicates that while agilists care about individual success, they care more about product success. Interaction designers agree with these principles completely. I call this attitude “responsible craftsmanship”.

But software design and development don’t live in the world of potters and painters, cabinet makers and blacksmiths. We live in the world of fast-paced, big money, high technology business. And business hasn’t had to deal with craft on this scale since the dawn of the industrial revolution.

Courtesy of Plusmo.com

A struggle for power

Over the years, when interaction designers ask me which design technique works best, I have assured them that this is not so much a battle of technique as it is a struggle for power. The same holds true today as we, the responsible craftsmen, wrestle power away from those people who insist on living in the past.

". . . we, the responsible craftsmen, wrestle power away from those people who insist on living in the past."

There is a similar power struggle in the agile world today. Many of the recent converts to agile are merely going through the motions, but are not truly committed to the ideals of agile. Lots of developers paper their walls with post-its, iterate rapidly, and have stand-up meetings, but like skateboarders who wear Van's and read Thrasher but never actually get on a skateboard, these people are poseurs.

The recent "Agile Roots" conference in SLC was an attempt to highlight this difference and return to the quality ideal. Similarly, the emphasis of today's conference is a reassessment of agile to determine what's real and what is mere posing.

Courtesy ThoughtWorks

Agile comes from programmers

Agile evolved in its own universe of programmers, invented by programmers, and centered on code. Better, more responsive code is a good thing, but it isn't the only thing. Arguably, the greatest strength of agile stems from its emphasis on collaboration with other disciplines.

While this emphasis on collaboration is new to programming, its value is immediately apparent to agilists.

Collaboration insures that the critical initial steps in software development are gently guided down more appropriate paths, resulting in multiplicative benefits as the project proceeds. Even rudimentary collaboration between relatively inexperienced practitioners yields significant benefits. But collaboration between disciplines is a tougher problem than collaborating within a discipline.

"Even rudimentary collaboration between relatively inexperienced practitioners yields significant benefits."

Courtesy of thephotoholic

Collaboration

Ironically, the very success of agile collaboration can blind its adherents to the scale of the problem: there is collaboration where there was none before, and then there is skilled, trained, effective collaboration between mutually experienced craftsmen.

I am not criticizing collaboration when I say that most agile practitioners (and most interaction designers, for that matter) have yet to engage in high-quality collaboration.

Probably the weakest link in the agile chain is the part where developers work with customers, users, marketers, analysts, and subject-matter-experts to determine what the software should do and how it should behave.

Often, when agilists give these stakeholders just what they ask for, they still don't create a successful product.

"The weakest link in the agile chain is the part where developers work with customers, users, marketers, analysts, and subject-matter-experts to determine what the software should do and how it should behave."

Collaboration

Part of the problem lies in the sad truth that these other disciplines are similarly isolated and internally focused on their own particular specialty which keeps them from understanding the software side of the problem.

My personal 35-year long software odyssey has made abundantly clear to me that the task of satisfying the user can only be achieved by fully integrating both the development and the interaction design practitioners into a single, self-organizing, self-directed team.

This means that those agile enthusiasts who believe that all other disciplines, including interaction design, need to march to the agile drummer are thinking too narrowly. The two disciplines won't integrate on one discipline's turf. They will unite on some common ground, slightly new to both of them.

Smaller difference

But don't despair; the difference is smaller than you might think. Interaction designers have been using agile methods (under other names) for years. Interaction designers have always designed in pairs, have always rapidly iterated in a discardable medium, have constantly refactored their work, always begin with a fuzzy, out of focus, yet complete image of the final product, always are open to outside advice, suggestion, and direction, and have always judged that advice on its merit, not on the authority of its source.

IxD is the developer's best friend

I believe that interaction design should be the developer's best friend, and an integral part of the agile team. Their common goals and values make this inevitable.

Most of the other disciplines that developers need to collaborate with don't share the responsible craftsman's concern for product success and quality.

Typically people in roles such as "business analyst", "database administrator", "documentation writer", "chief executive officer", "graphic designer", "project manager", or "code librarian" don't have the big view in mind.

cooper

Problematic collaborationists

This is not a condemnation of these people—frequently they care deeply and feel hamstrung by their inability to do the right thing. Most people are merely stymied by their job limitations—they are way too busy doing the rest of their job, or their discipline hasn't sufficiently evolved to address the broader goal of customer satisfaction. Often the sheer size of their other responsibilities forces them to assess all software design and programming activities as "merely technical" and this blinds them to the vital strategic nature of how their software behaves.

While pursuing their own careers, they can inadvertently obstruct the development process. Even if they are willing to collaborate, they might do so only on their own terms.

Interaction designers can be the developer's most effective collaborator. This is not true of all those who call themselves "designer". In fact, their willingness to collaborate effectively is an excellent litmus test of their authenticity.

A true interaction designer places the user's satisfaction at the forefront; a true interaction designer isn't too busy with "their" job to work hand-in-hand with developers; a true interaction designer is willing to collaborate on the developer's terms; and a true interaction designer discusses the trade-offs and participates in the difficult decision making instead of issuing directives to the programmer.

"A true interaction designer places the user's satisfaction at the forefront."

White collar

Probably the most troublesome group of necessary collaborationists is business management. Very few managers have any skill or interest in software development. All of them are too busy and far too preoccupied with things that they deem more important than software development.

Courtesy of Library of Congress, US Dept of War information, Howard R. Hollem,photographer

Blue collar

Most of them are stuck in the industrial age paradigm, where managers are white-collar, smart, well-educated, and responsible for deciding on the proper course of action, while practitioners are blue-collar, poorly-educated, and incapable of making informed decisions.

http://www.flickr.com/photos/maxbraun/149270753/sizes/o/

No collar

Today, in the post-industrial age, “no-collar” knowledge workers are typically better educated, equally intelligent, and better able to make informed decisions than any manager is. Worst of all, managers insist on managing; they restrict resources unnecessarily, issue ill-considered directives, and let their guesses preempt the work of the expert practitioners.

After years of trying to reason with managers, I finally came to realize that the true power lay in the hands of practitioners; of responsible craftsmen; of us.

Responsible craftsmen

I am now convinced that we are engaged in an insurgent war, and that responsible craftsmen are the only constituent that can honestly see the problem and the solution, and that we will fight for the greater good, rather than merely for our own, short-term career goals.

At the turn of the millennium, software developers in this country were in a bad way. Their basic methodology and approach to development was rooted in academia or old-skool companies like Microsoft. Their methods worked, but they were also guaranteed to take a long time and produce unlovable, unusable products.

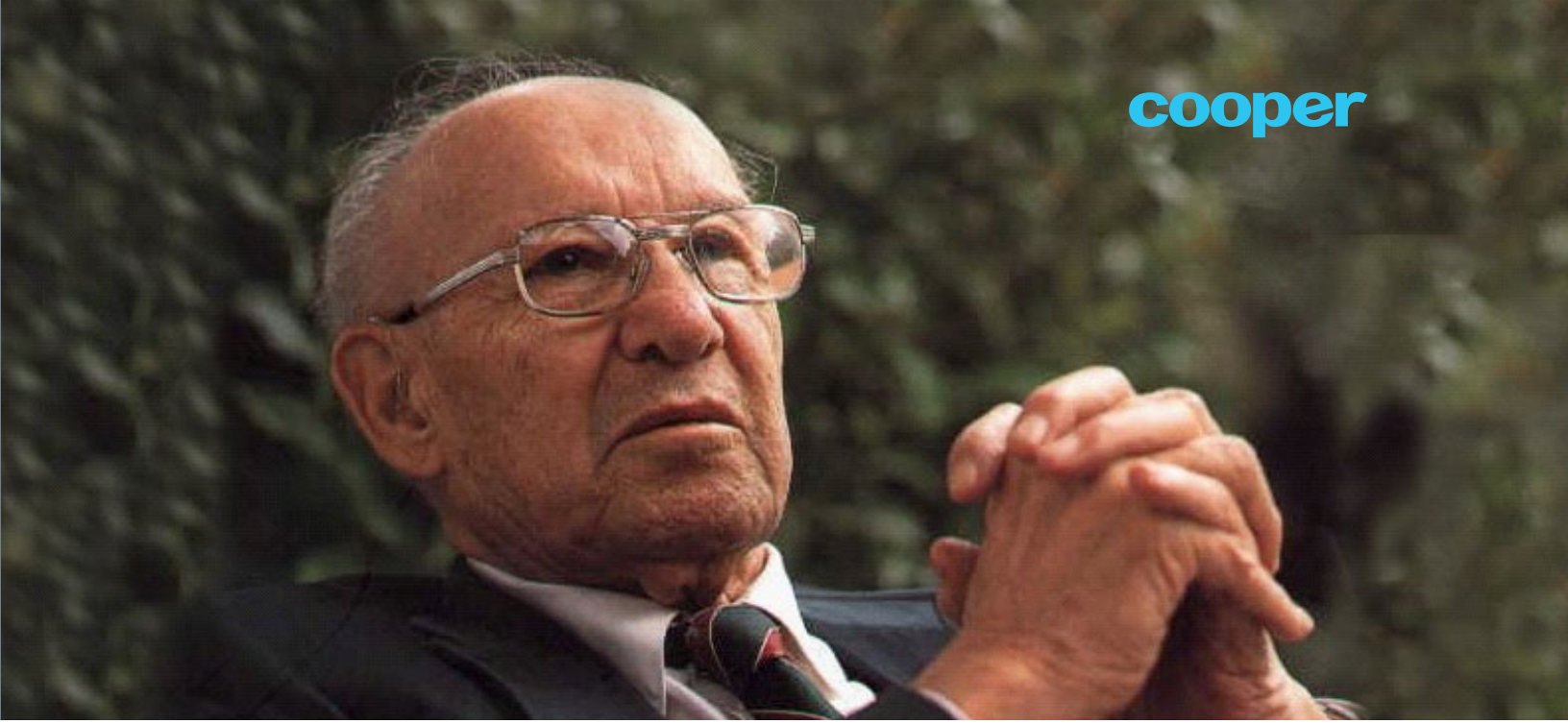
The booming open source movement was clear evidence that developers were unhappy at their day jobs. And those jobs were drying up, being outsourced to Bangalore, Talinn, and Chengdu. Communications within the development community were good, but dialog with the business and design communities was weak.

"The booming open source movement was clear evidence that developers were unhappy at their day jobs. And those jobs were drying up. . . "

Responsible craftsmen

While I have no proof, I simply can't credit the rapid ascent of agile to anything other than a collective sense shared by the software intelligentsia that, being at a professional nadir, there was nothing to lose by gambling everything on agile's radically new vision.

Coincidentally and concurrently, I was arriving at a similar state of frustration in my attempts to convince business executives that they needed to pay more and better attention to their software development efforts. I struggled to explain that software wasn't some technical backwater of the corporation, the way it might have been when punched cards walked the earth.



Peter Drucker: management guru

Ironically, reading the words of one of the greatest business management thinkers, Peter Drucker, gave me the courage to finally turn my attention from business executives to focus instead on practitioners as the true source of power in the modern company. Drucker presciently foresaw the rise of the knowledge worker and the growing ineffectiveness of conventional management.

Author's collection

IxDA conference in Savannah

In February 2008 I was honored to give the very first keynote speech on the first day of the first annual conference of the IxDA, the Interaction Design Association, in Savannah Georgia. That talk was brand new, different from this one, but I also called it "An Insurgency of Quality". In it, I exhorted the audience of 450 practicing interaction designers to cease trying to convince managers to respect, support, and adequately fund software design and development. Instead I encouraged them to directly seek out developers for some indigenous, unsanctioned, under-the-radar, yet highly-effective collaboration. As yet ignorant of the term, I was encouraging them to be agile.

The essence of insurgency

While there are many enlightened executives who clearly see that agile, collaborative teams are the blueprint of the future, there remain many more executives who don't. The war is still in progress, and it is fought by teams of agile practitioners who believe that by caring exclusively about the quality of the product and the success and satisfaction of the user, their career standing will take care of itself. To me, that selfless commitment is the purest essence of insurgency.

Shortly after that talk in Savannah, I began to investigate the burgeoning agile movement. To my immense pleasure, I saw many parallels between the disciplines of interaction design and agile development. I saw a hunger in agilist's eyes for collaborative assistance on the part of designers (and I also saw frustration in the eyes of developers who had been handed pretty pictures by old-skool graphic designers and told to implement them without argument).

I'm a big fan of cross-skilling, but the practical limits are real. Any programming of release quality is beyond the ken of most non-professional programmers. Likewise, any form and behavior synthesis of release quality is beyond the interest level of most non-interaction designers.

"Despite the overlap between their practices, they each have different ways of getting things done. Each practice will need to accommodate the other to some extent. Effective integration is a symbiotic partnership."

A symbiotic partnership

Some but not all of the interaction design process fits directly into the agile framework. In particular, the activity frequently referred to as “requirements gathering” doesn’t fit well into the agile framework.

The foundation of agile is the rhythm and tempo of the iterative loop. Coincidentally, the same is true of much of interaction design, however, not all of interaction design is generative; much of it is empirical and analytical.

Empirical and analytical

This empirical and analytical work, while also iterative, rarely fits into the rhythm and tempo of software development.

In particular, the universe doesn't begin at scrum zero. The work necessary to understand and identify the user, to understand and aim the business objectives, and the work necessary to imagine the product's end-state doesn't fit into regular time boxes and therefore should best be performed before programming begins.

The empirical part consists mostly of what is called ethnographic research. This typically involves lots of travel and coordination with users and stakeholders scattered around the world. The analytical part consists mostly of skull-work, organizing and reviewing the research results, and developing personas from the material. Once this real-world insight is in-hand, it needs to be written-up for further reference, and the team can do some basic sanity checking on their initial assumptions.

Most agile developers would agree that this work has to get done, but many mistakenly think that this "requirements" stuff is handed to the designer by some outsider, like marketers or business analysts. On the contrary, synthesizing the "requirements" is the interaction designer's most important job.

"The analytical part consists mostly of skull-work, organizing and reviewing the research results, and developing personas from the material."

“Requirements” aren’t wine

Outsiders will certainly present the team with long lists of demands, but satisfying “requirements” isn’t the same as satisfying users. What’s more, not all “requirements” are actually required.

“Requirements” handed in from various stakeholders have to be regarded as raw data to be factored into the larger vision. Converting “requirements” into design is akin to converting grapes into wine. In order to do it well, interaction designers need to wrap their heads around the big vision early in the process, and this critical work is rarely seen by developers.

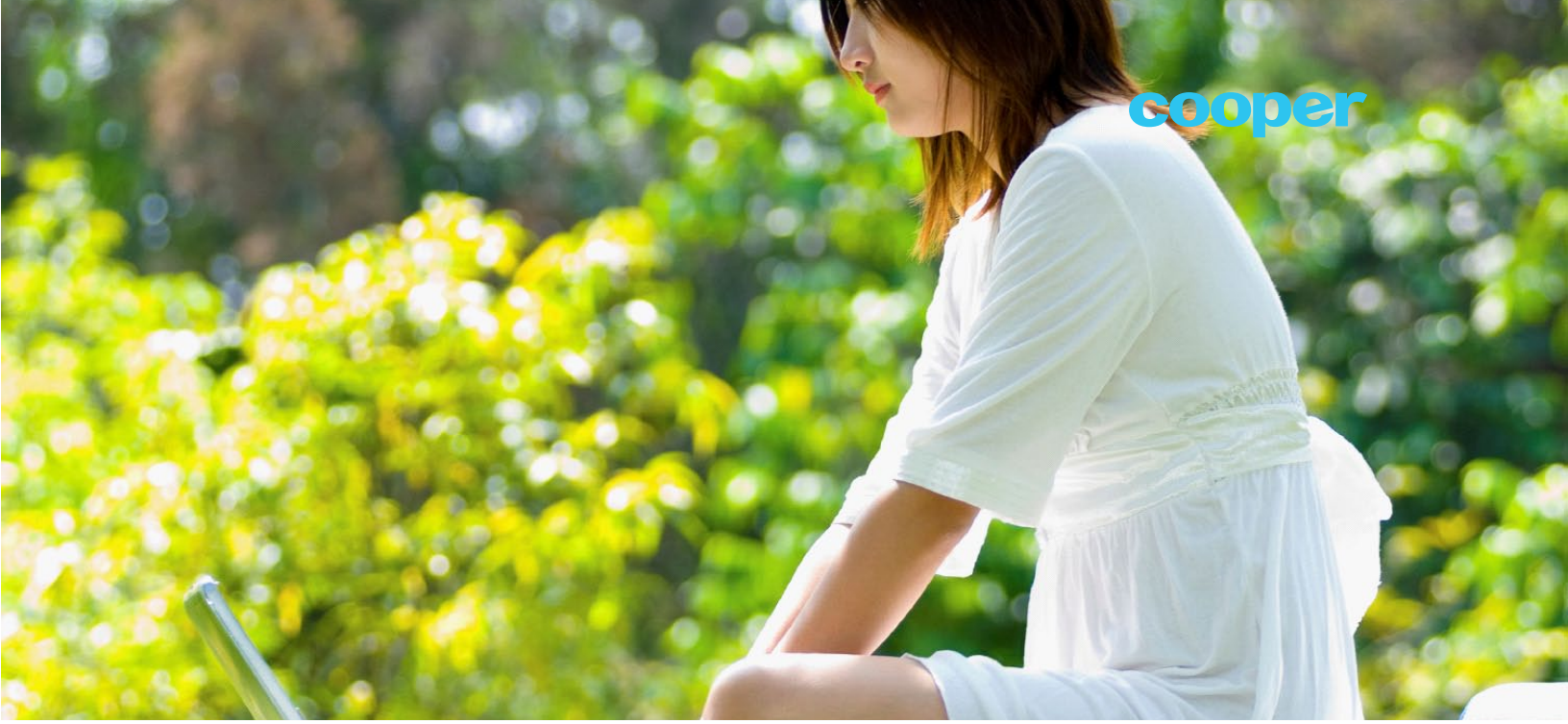
Agilists have found that trying to imagine a complete, finished program is not just a waste of time and effort for them, but can often lead them into project-killing featuritis, second-system-effect, and terminal bloat. They know that “you aren’t gonna need it” and worrying about it or coding for it now is a dangerous trap.

That’s certainly true for developers, but an integral part of designing the form and behavior of a software solution for end users is imagining the end-state of the product. One of the core competencies of interaction designers is this ability to visualize software behavior without having to code it.

“While well-disciplined developers stay away from end-state thinking, it is wrong for them to restrict others from doing so.”

Agilists fear BDUF

Agilists fear that interaction designers imagine the end-state so they can freeze it into massive Big Design Up Front documents that will forever after force the project down some obsolete blind alley to eventual failure (and this fear is quite justified, as many of our business manager clients do just that). But that is just an echo of the bad-old-days. Enlightened collaborating peers simply don't do that crap.



Who is the end user?

Interaction designers imagine the end-state so that they can know with certainty who the end user will actually be. The end user imagined by marketers or analysts is often incorrect or inadequately representative. And if you don't know who your user is, your software is virtually guaranteed to fail to satisfy them.

What is the actual business case?

Interaction designers imagine the end-state so they can determine what actual business case the product will address. Just because business executives and marketing professionals identify an unfulfilled market need, doesn't mean that their vision of the product to fill it is effective, efficient, or even correct. That's what interaction designers do better than any other.

How can we judge functionality?

Interaction designers imagine the end-state so that they can establish a performance metric of the necessary tradeoffs that development demands. They construct a functional vocabulary of the product's capabilities and behaviors and the benefits that the user can gain from them. This is the cauldron where "requirements" are rendered into something useful. It is also the raw material for the product roadmap, showing what should bubble to the top of the backlog.

Some things don't fit

Of course, not all of the empirical work can or should be done in advance; to keep pace with business fluctuations and to test our hypotheses against the real world must be an ongoing part of the design process, and it is best done in concert with the programming.

While interaction designers are willing, nay eager, to fit their work into the agile rhythm, it would be inappropriate to try to squeeze every aspect of interaction design into the procrustean bed of agile time boxes.

Just as interaction designers must respect the iterative tempo of the agile programmers, those programmers need to respect the portion of the interaction designers work that falls outside of that cadence.

Interface design fits well

As you can see, much of what interaction designers concern themselves with is strategic vision and product conceptualization. But at least half of the interaction designer' job is tactical, focused on screen design and behavioral problem solving. This work dovetails well with the agilists time boxed iterations.

This time-boxable, tactical part of interaction design goes by many names, such as "user centered design" or "user experience design". I have always called it "interface design". Dave Hussman calls it "visual design". Regardless of what it's called, it gives him tremendous value every single day.

Some developers think that this tactical design is the only thing that interaction designers do, and many junior practitioners are satisfied with this role, and the resulting collaboration can be very smooth. The only downside is that it doesn't—by itself—result in satisfied users or successful products.

What makes such tactical design so valuable to developers is that it is effectively supported by the strategic empirical and analytical work done previously and that is usually invisible to developers.

"Some developers think that this tactical design is the only thing that interaction designers do. . . "

Bricklaying

When a mason, following a line on an architect's blueprint, builds a brick wall, he lays one brick after another, repeating the same process a thousand times. Whenever a programmer finds him or herself writing the same line of code more than once or twice, he or she will abstract it into a subroutine or method. In this way, programming differs from all other crafts: repetition is rare; each line of code is unique, bespoke, custom-made, and each one brings to the surface a new set of questions, problems, opportunities, and trade-offs between effort, commitment, and investment on the developer's part and power, usability, and appropriateness on the user's part.

Fractals

Like fractals, this pattern of trade-off-decisions is identical to those made in the board room, just made more frequently and on a somewhat smaller scale. Sometimes, these trade-offs can be just as important in the long term as those made in mahogany-paneled offices. Because these questions arise endlessly on a day-to-day basis at the lowest level of implementation, it's really easy for business managers to imagine that they are insignificant and only of interest on a technical level. This is not true.

Courtesy IMDB.com

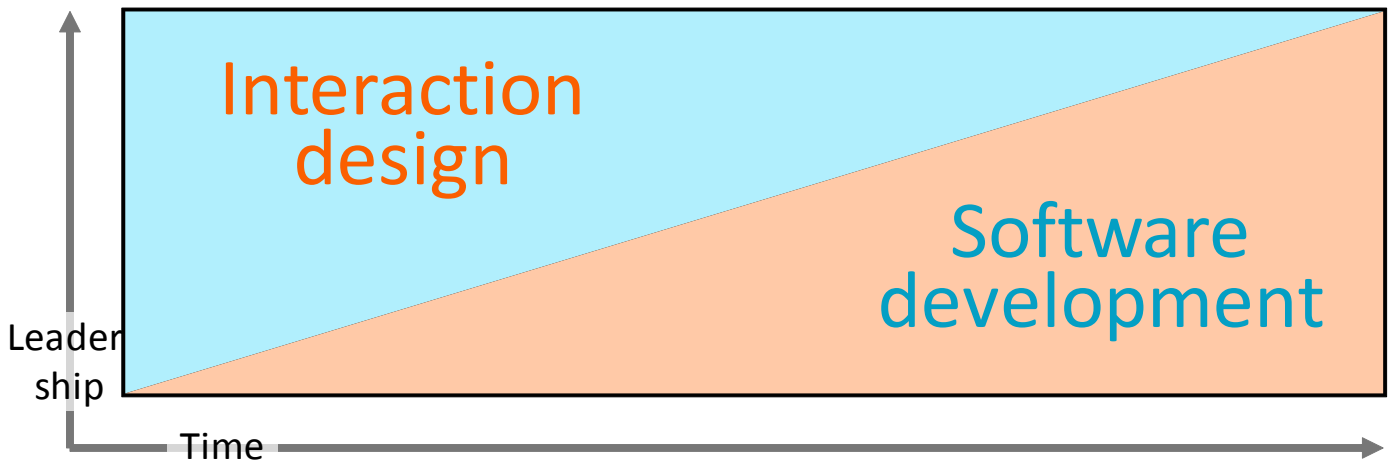
Battlefield choices

These myriad, tiny, yet often vital trade-off-decisions are unique to software, and I've come to think of them as "battlefield choices". Much of the impetus for the development of agile methods comes from a desire to better address such battlefield choices. And this is exactly where the interaction design discipline excels. Interaction designers can work side-by-side as an integral part of the agile team, providing informed guidance for each battlefield choice, helping to keep the implementation details true to the bigger product vision.

In particular, each one of these battlefield choices usually has a technical component and a user-facing component. The technical issues typically can be vanquished with logic and deduction; however, these most effective technical tools are frequently ineffective at solving the human-interaction side of the problem.

Cognitive illusions

There is a large and rapidly-growing body of scientific evidence proving that all humans are subject to a plethora of cognitive illusions and perceptual distortions. Applying logic or deduction may seem a logical approach, but the illusions that bedevil Homo sapiens doom such logical design to a tragically bad user experience. Interaction designers are trained in the craft of unraveling and extracting the underlying goals that drive a user's sense of satisfaction. They can then design software behavior to satisfy these goals rather than the stakeholder's firmly-stated but equally firmly distorted "requirements."



Collaboration leadership over time

I visualize the leadership role in a project changing over time. This graphic shows what I mean.

On the left side of the chart, at the very beginning, the interaction designer drives the project by determining the contextual issues of who the user is and what the business purpose will be. While stakeholders may desire, or even "require" many features and behaviors, the team relies on the interaction designer to assess the tradeoffs involved. He must seek out and assess the potential market advantage and weigh it against the cost of development. The interaction designer leads the team to decide with confidence on the strategic issues facing the product. This keeps the big, conceptual iterations front-loaded in time, where they are far cheaper to make. This means that the developers must be available to consult with the designer to assess the relative cost of each potential feature.

"I visualize the leadership role in a project changing over time."

Collaboration leadership over time

As the project proceeds, the agile developer begins to assume more of the leadership role simply because the project's focus shifts from the user and market to the growing code base. The tempo of time-boxing now drives the content and pacing of battlefield choices. The developers and the interaction designers are working shoulder-to-shoulder in full and constant communication solving the complex problems that emerge. All team members work to assure that there are no unforeseen effects.

Soon, the agile developers have taken over the leadership role. Every day the team's efforts reveal dozens of new, unforeseeable battlefield choices. If the choice regards user-facing behavior, then the interaction designer must be instantly available to consult with the developer to assess the relative benefit of each potential solution.

The small company Product Owner

One of the reasons that agile is so successful in small projects is that the person who actually owns the product, the entrepreneur, plays the role of "product owner".

One of the reasons why agile is often less effective on large projects is because the person who plays the role of "product owner" doesn't know how to own the product.

The Scrum Alliance defines "product owner" as the "single person with the final authority representing the customer's interest in backlog prioritization and requirements questions. This person must be available to the team at any time."

There are two inherent weaknesses in this approach. First, the "final authority" concept is much weaker than the self-organizing team concept. Wouldn't it be better to have the team itself have the information and perspective it needs to make such prioritization decisions within its own group?

"One of the reasons why agile is often less effective on large projects is because the person who plays the role of 'product owner' doesn't know how to own the product."

The small company Product Owner

The second weakness of this approach is that, in the real-world, the nominal product owner is simply not going to be "available to the team at any time". The product owner is undoubtedly very busy with other product and project management duties that keep him or her sufficiently out of the loop so that their attention is never fully focused on the particular battlefield choices that bedevil the team.

In small start-ups, the single-minded focus of the successful entrepreneur is always on customer satisfaction. What's more, he or she will be, of necessity, hands-on and participatory. All of this is obviously compatible with the beliefs, values, and practices of agile.

cooper

The big company Product Owner

In larger companies, however, the product owner is likely just a salaried employee who has other demands on his or her time, is pulled by outside commitments, and typically lacks the skills to make critical decisions regarding how a product should behave. What's more, any middle manager on a large, established product line is constrained by convention, by the sheer magnitude of the existing code base, by the inertia of the existing team, and by his own lack of self-confidence in the face of a proven product.

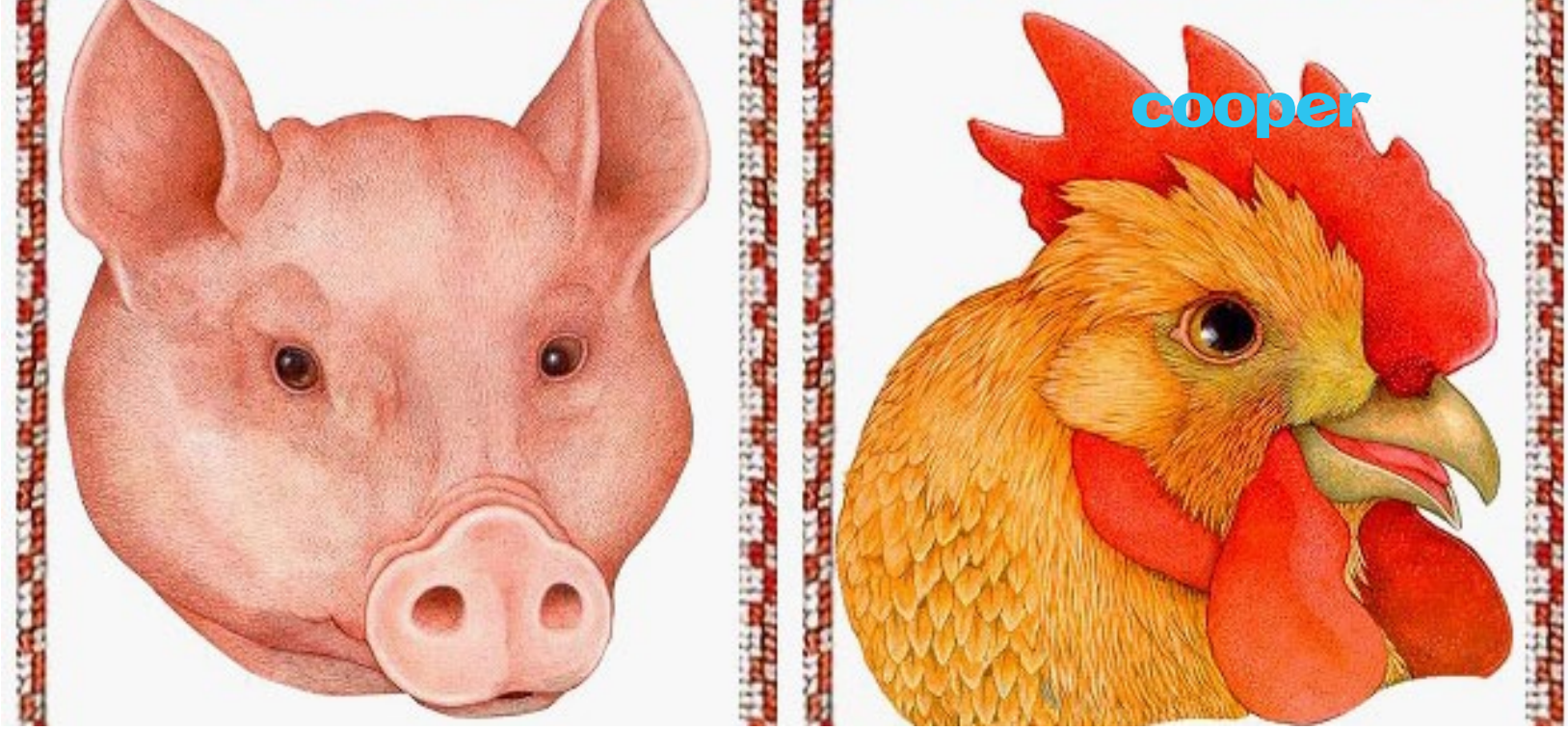
When the agile development team demands tough, decisive action from the "owner", they are most likely to get prevarication, procrastination, trepidation, and tradition.

Being a skilled and effective product owner is a much more difficult job than most people think.

Interaction designers play product owner

Interaction designers, at their strategic best, are superbly equipped to play the role of product owner effectively, efficiently, and collaboratively. They speak tech to the developers, speak business case to the managers, speak user personas to the marketers, and—being responsible craftsmen—are beholden only to the success of the project.

Please note that I am not saying that the interaction designer should "own the product." I simply believe that the demands of the product owner role are too difficult and too important for an amateur or part-timer. What's more, a weak stick in the product owner role can have a devastating effect on the entire project. It is very much in the development team's best interest to assure that a skilled and experienced, responsible craftsman play the part. The role is a perfect fit for a senior interaction designer.



The pig and chicken

Most of you have heard the allegory of the pig and chicken. At their restaurant they serve ham and eggs, which means that the pig is committed but the chicken is merely involved.

Developers see themselves as fully committed pigs: their professional standing is linked to the project's success or failure. Most developers also view designers as chickens: as mere advisors to the developers, and their professional standing is independent of the project's fate.

But a true interaction designer desires and demands to be a pig; to be committed. They don't want to merely contribute advice; they want to buy in to the project and be a full, oinking member of the team.

I realize that for this to happen, developers must put their trust in interaction designers, and in turn the designers must prove their commitment by making absolutely sure that their decisions are correct.

I want each developer here to demand pig-quality commitment from their interaction designer teammates, and I want each interaction designer here to take pig-level responsibility to do the necessary homework to make well-researched, defensible, and correct form and behavior design so that they have the courage to stand by their work.

". . . for this to happen, developers must put their trust in interaction designers, and in turn the designers must prove their commitment by making absolutely sure that their decisions are correct."

Courtesy http://swervechurch.wordpress.com/2008/12/21/fox-holes/Photo

There are no chickens in foxholes

There are no chickens in foxholes.

I believe that many committed teams of equals, integrating agile development with interaction design, practicing responsible craftsmanship, will be an immense force. The power of our combined efforts will be unstoppable in the war for satisfied users and professionally fulfilled practitioners. This insurgency of quality will show the world how software was meant to be created.

Thank you!