Appears in: *Proceedings of the Second International Conference on Usage-Centered Design*. Portsmouth, NH, 26-29 October 2003, pp. 1-15.
http://www.foruse.com/2003/

Contact Info:
bill@billbuxton.com
http://www.billbuxton.com

[M10]

Performance by Design: The Role of Design in Software Product Development

William Buxton

Abstract

This article could just as well be titled "What I have learned about software product design in 8 1/2 years of working with some of the best industrial designers and film makers in the world." The underlying premise is that filmmakers and industrial designers approach the design of new products in a fundamentally different way than the software industry. More often than not, software products are green-lighted, and then work begins. With films and product design, green-lighting comes at the end of a front-end process, not the beginning. Stated another way, software projects tend to go directly to development/engineering, leapfrogging over anything that an industrial designer, for example, would recognize as a design process. Our argument is that our industry's bypassing such an explicit and formal front-end design (or in film terms, pre-production) process lies at the root of many of our problems of quality, cost over-runs, and late delivery. Furthermore, I would argue that the absence of this front-end process lies at the root of the software industry's abysmal track record in bringing out successful new (as opposed to n+1) products. To put my argument into perspective, I will briefly summarize the process followed in film and product design, and discuss how it can apply to software product design[1]

Introduction

For a number of years I have struggled with a frustration with my own industry, software product development. The roots of this frustration lay in two basic questions:

1. Why were we so bad at bringing out new products—and by this I mean new as opposed to "n+1" products, or upgrades?
2. Why were all of our development efforts (both new and n+1) always late, over budget, and so far in quality from what we wanted them to be, and knew they should be?

At first, I took this frustration personally, both as an individual, and as an executive of a software company. But gradually, I looked around and realized that it wasn't just us. The whole industry seemed to suffer from the same malaise. In fact, despite our failings, we were better than many. However, this was small comfort. After all, how good can you feel about your own performance if the best that you can say is that you are not as bad as some of your competitors?

The reality was that, as a highly competitive individual and company, I felt that standards should be set according to what you knew you could and should be doing, not by industry norms that fall well below that mark.

Despite new approaches to code reviews, agile programming, iterative design, object oriented design, and the like, there had to be a better way.[1] The good news is that it became evident as soon as I lifted my gaze beyond my own industry. In fact, it was staring me right in the face in the guise of my primary customers in the film and industrial design industries.

Why Film and Industrial Design

Why might film and industrial design have some relevance to software product design, and why have I chosen them rather than some other industries, by way of example?

[1] At this point I need to make a clear statement, or confession. Although trained as a computer scientist, I am not, never will be, and do not pretend to be a software engineer or an expert in that area. I have too much respect for the discipline and its complexities—and too intimate a knowledge of my own limitations—to go down that path. So, it is critical to take what I am saying in that context. However, this is not an apology, any more than what I have to say is a critique of that profession. I strongly believe that the process that I outline here is complimentary to, as opposed to being in opposition to, sound software engineering. My position is this: software engineering is a critical component to successful products and quality. It is simply not sufficient. My objective is to shed some light on that part of the overall process that I feel is missing.

The second question is the easiest, so I will deal with it first. The fact is that I was in the business of selling software tools to the creative talent in some of the world's leading design studios, as well as some of the best people in computer animation and visual effects. These were people that I had access to, that I knew, respected, and from whom I could learn.

This brings us to the first question concerning their relevance. While perhaps seeming banal, the reason that their process interested me was that they generally succeeded in exactly the areas where we were weak, at best, and failing, at worst: namely, getting new products out on time, on budget, and with a high standard of quality and innovation.

Implicit in the above is the assumption that the process of both filmmaking and industrial design had something in common. I saw this as a well-defined process that preceded the project being green-lighted.

In film, this phase of the process is called "preproduction." Most film buffs will know this term and may even be familiar with some of its components, such as story development and assembling the key players. But it actually goes much further than that. Typically, you only consider a film for green-lighting after it has assembled a complete "package." This package not only includes script, director, and key cast members but also budget, release date, marketing plan, and a detailed production plan. In other words, before committing the resources to actually shoot the film, there must first be an extremely detailed creative, production (technical), and business plan in place. While this represents a significant up-front investment before even knowing if the film will happen, the costs and competitive nature of the industry are such that it is unthinkable to commit to something where these things are not nailed down in advance. There is still room for creativity and innovation during production. However, it could be argued that it is precisely the up-front planning in the pre-production phase that provides this freedom, and lets the director and producer manage it, while staying within the framework of what was approved.[2]

[2] Of course there are noticeable exceptions where things have run way out of control. However, more often than not, these have occurred when the director or star have so much "clout" that they are able to work outside the normal checks and balances of the process. Sometimes one gets away with it, and that clout is proven justified at the box office. Think of Apocalypse Now, for example. Other times, it results in disaster, as in such films as Ishtar, or Heaven's Gate. While it is easy to find exceptions to what I am saying, nevertheless I argue that the general concepts remain valid.

Likewise, when a company like BMW or General Motors wants to bring a new car to market, there is a very similar process that precedes making the decision to put the car into production. This phase involves people from engineering, marketing, and design. By the time that the executive team meets to decide whether to green-light the project or not, they know what they are going to build, how and where they are going to build it, how much it will cost, who will want it, and how they are going to sell it. But what struck me most strongly about the process was this: despite my having a fair bit of experience, when confronted by a new model vehicle in the design studio, I still had to ask, "Is this real or a full-size clay model?" The design process was such that at the end, before making the decision to go into production, they output a model of the proposed vehicle that even experienced people could not tell if it was the final product or not. And yet, despite this refinement, if the vehicle were approved for production, there would still be at least a year of engineering before it could be manufactured. And, as in film making, during the product engineering and manufacturing phases, there was still room for innovation and improvement.

(a) (b) (c)

Figure 1: Three of the many phases of the automotive design process. (a) Early "ideation", or concept sketching; (b) Design refinement using 3D clay scale models; (c) Design review using full size clay model. Credit: Frank Tyneski (a); General Motors (b & c).

It is this front-end process that I see as generally missing in the software industry, and it is my belief that its absence is largely responsible

for our industry's poor performance with respect to the quality of our products and bringing new products to market.[3]

Design and Engineering: A Social Contrast

The focus of our discussion thus far has been on various industries and their processes. However, the over-riding concern in all of this is human. Not only are people, in the form of users, the ones who appreciate and are affected by product quality, but it is also people who populate the processes that create it. Consequently, to discuss a process, or change in process without taking into account the people who carry it out would be to violate the basic tenants of a human-centric perspective.

In this regard, I have to walk the slippery slope of making some comments about the human side of the differences between the up-front design phase, versus the production, or engineering phase. Simply stated, design is a separate process from engineering. More to the current point, design takes a very different mind-set and training than does engineering. In my experience, for example, employees who were a disaster in product engineering were outstanding in the design phase and vice versa.

In a product engineer I want someone who is extremely well organized, bordering on anal-retentive, and who is technically strong. While open to suggestions, I don't want the engineer innovating with the product concept. I want their creativity directed at improving the quality of the product and how its vision can be effectively brought to market. However, for this to happen, the engineer needs to know what that vision is. Therein lies the problem in most of our software products, since (more often than not) the engineer is making design decisions at the same time that the product is being built. And, to make matters worse, since there is no centralized responsibility for design (much less an explicit design process), various engineers are making design decisions that are consistent with their own model of the product, but inconsistent with those of other engineers. The best analogy that I can think of is with a film that has no director, much less a script, storyboards, or production design.

[3] It is important to remember that there are limitations with all analogies. Film making, for example, has little or no equivalent to after-shipping support, maintenance, or upgrades. Yet, these are clearly important factors that need to be taken into account in the design and engineering of software products. Nevertheless, I believe that film is still a valid and useful analogy. Furthermore, these consideration do exist in my other analogy, automotive design and engineering.

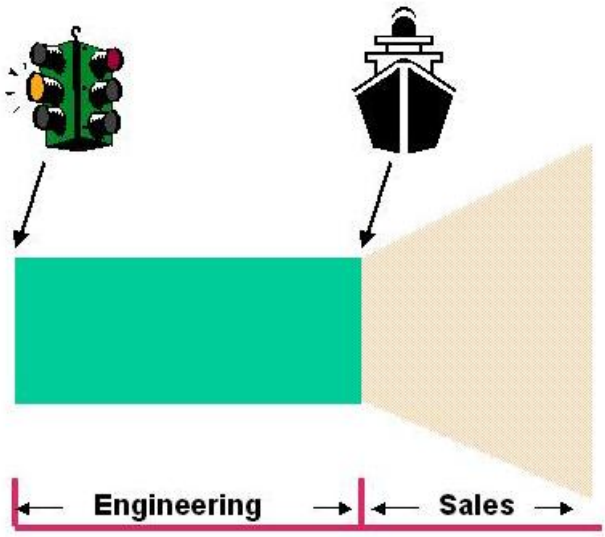

Figure 2: The status quo in software product development. Projects get a green light right at the start, and go directly to engineering. The next phase is when they ship—usually late, with bugs, and over budget and missing functionality.

Designers are a different breed altogether. They went to art college rather than engineering school. They tend to work different hours, dress differently, and use different references. Their primary skill is not making things that work, that can be maintained, or that can be built. Rather, it is sketching. That is, quickly working up product concepts into a concrete form sufficient to serve as the basis for discussion, evaluation and questioning. From the perspective of quality, the metric is in terms of concept and speed, not refinement of execution.

While I fully acknowledge that I risk falling into the trap of characterizing both sides by banal stereotypes, I nevertheless hold to three key observations:

1. Regardless of how you characterize the two, design and engineering are very different skill sets, and the two professions are made up of two very distinct populations.
2. Each skill set is essential for the production of quality products, but neither is sufficient on its own.
3. For the most part, software companies and their process have a huge deficit in the design arts and process, and much of their

failure is due to their attempting to integrate design into the engineering processes, and have it done by engineers.

Since I fully acknowledge the shaky ground on which I am treading, let me give a brief example to illustrate the differences in these two populations.

I suspect that if you asked both a designer and engineer involved in software design, you would find that they both advocated the notion of rapid prototyping and iterative design. “So!” you might say. “Here is a similarity, not a difference.” Well, here is what I have seen.

One company had an idea for a new product. After looking at the problem that the product was trying to address, and consulting with customers, the design group started doing some concept sketches. These evolved into working prototypes that captured the essence of the evolving design, in particular in terms of its feel and interaction involved in performing the desired function. At the end of the process, which included customer validation, the result was a prototype that was, to the intended software product, what a full-sized clay model would be in automotive design.

Based on this, the project was green-lighted and the product taken over by product engineering. Partially because of the fidelity of the prototype to the basic concept and intent, what emerged (in record time), was a product that not only was faithful to the prototype, but was better. Yet, it was on time, met the specifications, and was as usable as it was free of bugs.

You might conclude, therefore, that this was an example of a successful interaction between design and engineering. Well, not quite. In fact, there was a significant part of the engineering team that felt that the design group had failed, since they could not use any of the code from the design prototype in their product. The resulting reimplementation was seen as a waste, and just another example of the design team’s technical incompetence.

The heart of the problem was perhaps best articulated to me by a junior industrial designer who explained it thus, “Engineers view prototypes as part of the process of building things. Designers build prototypes to criticize and tear apart.”

In this case, the engineers in question viewed the prototype in engineering terms, as a piece of not-production-quality software. The designers, on the other hand, viewed it as an “interactive sketch.” It was to the final product what a rough sketch of a car is to the final vehicle. The

key to appreciating the underlying point here is that the notion of sketching was extended to include feel, and not just look; to serve its purpose, the sketch had to be interactive. However, since the medium that the sketch used to accomplish this included some of the same tools and technology that would also be used in the engineering phase, it was susceptible to a misinterpretation of its intent.

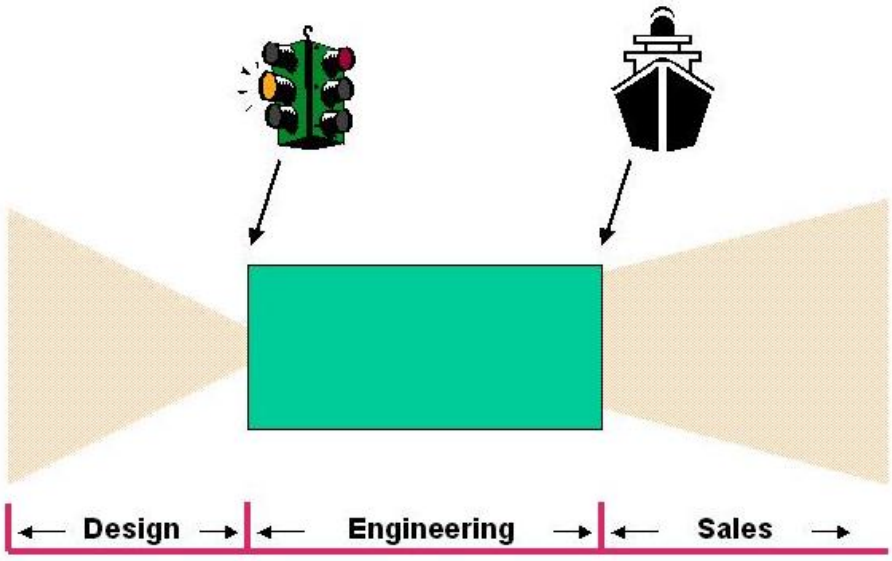

Figure 3: Inserting an explicit design process at the front end, prior to green lighting the project. The process is represented as a funnel, since the number of concepts to emerge is always anticipated to be fewer than enter.

Still, it is unfair to blame the engineers, as there is typically nothing in their training or experience that would lead them to think any differently. Design, its role, and its process are simply not part of the culture or educational process of software engineering, all of which leads us to the situation in which we find ourselves today.

A Sketch of the Process

At best, in the time available, we will only be able to gloss over the surface of the process. Hence, the best that we can do is consider this

explanation, and the resultant level of detail, as analogous to a sketch, rather than a final rendering.

In the simplest terms, the difference between Figure 2 and Figure 3 illustrates the basic argument that I am making: the insertion of a design process at the front end of product development. The primary assumption in advocating doing this is that the cost and time lost due to this additional stage will be significantly less than the cost and time lost due to the poor planning and over-runs that will inevitably result if it is not included.

My perspective is that the bulk of our industry is organized around the demonstrable myth that we know what we want at the start, and how to get it, and therefore build our process assuming that we will take an optimal, direct path to get there. Nonsense. The process must reflect that we don't know and acknowledge that the sooner we make errors and detect and fix them, the less (not more) the cost.

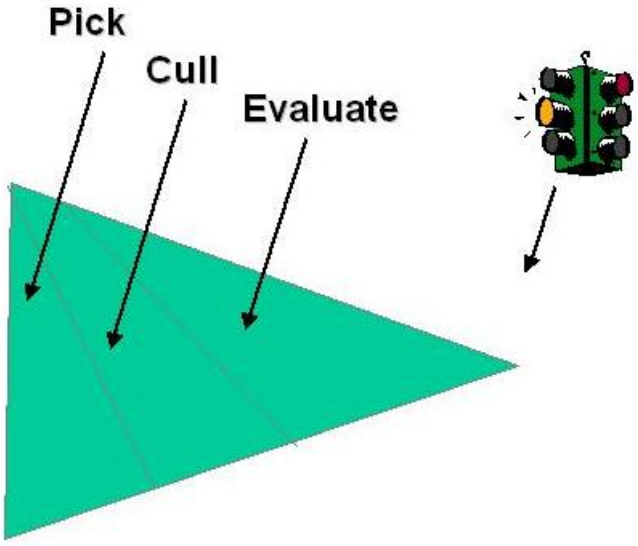

Figure 4: Changing criteria for evaluating ideas at one progresses through the funnel.

While the back-end engineering process is about getting things right, the front end is loaded towards making errors sooner rather than later, detecting them, and learning from them. One key to "sketching" in this context, is ensuring that the cost of errors, and the subsequent lessons, is contained within acceptable limits. The other key is recognizing that the

generation of ideas is targeted at converging on a design, rather than adding diversity for novelty's sake. Hence, the process is represented as a funnel, in which ideas are explored and refined. At the beginning, ideas are "cheap" and lightweight processes are all that are needed to evaluate them. As they become more and more refined, so does the investment that we have in them, so therefore the rigor with which they are examined increases. This continues up to the point where they are brought forward for formal go/no go evaluation as part of the green light process.

What is critical to note in all of this is that despite breaking the product lifecycle into three basic phases, as shown in Figure 3, this does not mean that the process should be viewed as three silos, roughly partitioned among design, engineering, and sales. One only has to go back to our précis of the process used in film to see this point. The "Design Phase" as I intend it to be understood, does not imply just the design of the actual product. Rather, as in film, it also includes the design of the engineering process (the technologies that will be used that is, the engineering/ manufacturing plan), the design of the marketing plan, and the refinement of the overall business model.

Likewise, green-lighting a project is not a signal that the designers are now off the project, or that they in any way "throw the project over the transom to engineering." I have included Figure 5 in order to try and capture the ongoing responsibilities and involvement of the key factions throughout the lifecycle of a product.

One can think of the ongoing relationship and distribution of responsibilities between design and engineering by analogy to the role of the architect and structural engineer in designing a building. The architect has prime responsibility at the front end but must clearly work with the engineer in order to ensure that the design can be built.[4] On the other hand, the architect has ongoing responsibilities, after green-lighting, for the evaluation and monitoring of the product as it moves through the construction phase, ensuring that the design intent is respected.

Returning to our comparison with film, if we consider the director as the head of the main creative department of a film, a useful exercise or test on any software product would be to ask: "Who is the equivalent of the director?" "Do they have comparable power and responsibility?" "If

[4] Of course, in the best of cases, the relationship goes well beyond that. The expertise of the structural engineer can often open up opportunities for the architect, and contribute to a much-improved design.

not, why not?" "If not, why do we believe that the integrity of the design will be maintained?"[5]

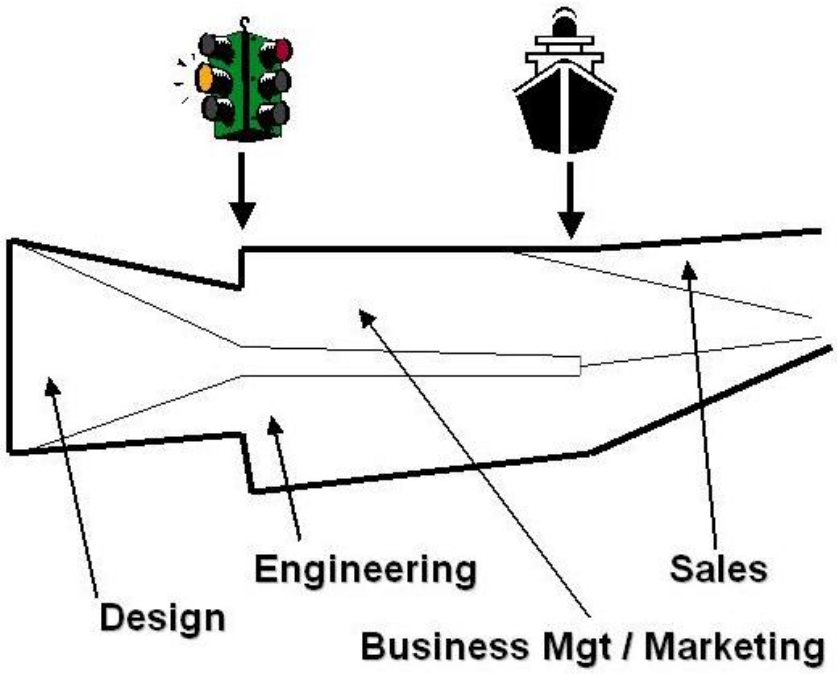

Figure 5: No Silos. Here we try to capture the nature of the ongoing responsibilities of the various teams throughout the overall process. Of central importance for our purposes is the notion that "design" includes the design of the business/marketing plans and engineering/ manufacturing plans, as well as the product itself. In this, it is no different than filmmaking.

Where are the Users in All of This?

At this point we risk being so bogged down in discussions about processes, roles, responsibilities, diagrams, and so forth, that we lose sight of why we are doing this in the first place: to design products that people want, need, like, and can use. Yet, so far I have not said anything about user-centered or usage-centered design, usability testing, or any of

[5] To be balanced, one has to recognize that there is an analogous set of questions that need to be asked in order to ensure that the integrity of engineering and business concerns are sustained through the design process.

the other associated buzz words. At least for anyone who knows me, this may seem especially curious. Therefore, let me inject a few comments.

First, the absence is explained partially because I simply take this part of the process for granted. While I may be deluding myself, I think that we should be approaching the point that we do not need to point out such basic things any more than a paper on mathematics needs to include a discussion of basic arithmetic.

My second comment is that it is precisely a concern for users that underlies the value of the proposed approach. Techniques fundamental to the design phase, such as sketching and prototyping, mean that iterative user testing and validation can occur much earlier and in a form that captures the interactive nature of the system. The consequence is that user input can begin early enough to influence the design of the product. User involvement then obviously continues through the engineering phase, in the form of usability testing.

The Anatomy of a green Light

Perhaps the most important step in any of this, in terms of it working, is the nature of the green-light process, that is, the criteria and process whereby the decision is made to go to market or not. We tend to get what we measure and reward, so it is critical to ensure that these criteria are in alignment with our objectives. The process needs to ensure that the proposed product has a good fit to the intended market. But it also must ensure that everything has been done to make sure that it meets well-defined measures of quality in the process.[6]

Furthermore, the engineering and manufacturing plans must be presented in order to determine not only what is to be built and that it can

[6] This is another case of "you get what you measure." In many companies, QA consists of finding and eliminating bugs. But what is a bug? In too many cases, bugs have more to do with programming errors, as opposed to design or more significantly, usability errors. If the specifications are in terms of functions in the product, then as long as the functions work, and can be demonstrated, then many would consider the product as meeting the specifications. But what if the specifications are along a completely different dimension? For example, in the product referred to earlier in this paper, one of the prime specifications was: "Any potential user of this product needs to be able to learn 80% of the functionality of the program in 3-5 minutes of first exposure, and have a 90% retention a week later." Such a specification not only has a wonderful effect on restraining "feature creep", it also means that a product with no programming bugs, and which has all of the specified functionality, may still fail QA as having a "stop ship" bug. It must meet the user-oriented specifications, or it does not ship.

be build, but also how it will be built. The higher the fidelity of the final design prototypes, the more we know on the what side of the equation. And, if you don't have a very clear idea of what you are building, how can anyone responsibly or reliably say if it can be built, how it can be built, when it can be built, or how much it will cost to do so?

Likewise, the green-light process needs to include a serious evaluation of the analysis of the product market, and the sales and marketing plan—and not just of the version one product, but of the product over its larger life-cycle (including how well it scales, from a business perspective, and how it fits into other products offered by the company).

From the holistic perspective, one also needs to go beyond this analysis and determine what else one might otherwise be doing with the resources thus consumed, in terms of maximizing ROI as well as service to the customer.

I see the green-light process as the establishment of a number of "contracts" among the key executives responsible.

1. The Executive Responsible for Design: agrees to deliver products to the green-light process according to criteria established by the executive as a whole, and is evaluated on doing so.
2. The Executive Responsible for Engineering: on green-lighting a project, agrees to deliver that product, meeting specification, at a specified time and within a specified budget, and is evaluated on doing so.
3. The Executive Responsible for Sales: on green-lighting a project, agrees to sign up for a specific revenue target, in a specific quarter, if the product is delivered meeting specifications (including quality), at the specified time, and is evaluated on doing so.

The executive team, as a whole, takes responsibility for establishing the strategic direction of the product agenda and roadmap, as well as the green-lighting of projects.

However, while the above seems "common sense" and "obvious," see how much it contrasts with common practice, where we green-light products before we know what they are, and forecast revenue before there is any design, much less meaningful validation. The consequence is that we "have" to ship whatever we have to meet our numbers, and the product that we start to engineer is seldom the same product that actually ships (if, in fact, it ships at all).

A Bit of History

In order to better guide future directions, as well as prevent us from lashing ourselves too much for our past behavior, it is worth briefly looking into what got us into this situation in the first place. Dan Olsen explains it in terms of three factors:[7]

1. Immaturity of the field.
2. Flexibility and low manufacturing costs of software.
3. Moore's Law.

He describes the software industry in terms of the old American frontier, where the winners got there first without much concern with how they got there. Behavior was largely driven by the large rewards to be won, coupled with the fact that they could be (and were) won without such niceties as design or usability. Remember, safety standards and the rules of the road (much less comfort and styling) came after, not before, the introduction of the automobile.

Furthermore, unlike film or automobiles, where there is a significant cost of goods, including materials and manufacturing costs, software (while hard to write) was relatively fast to create, modify, and bring to market. With less upfront expenses, the rush to market was coupled with relatively less risk, thereby further fueling the rush to the frontier.

Finally, the reality of Moore's Law meant that there was a "perpetual frontier," and therefore little or no time for things to settle down to the point where such "niceties" as design, quality, and usability standards might become established. Again, during a gold rush, nobody involved thinks about the environment. Just think of this as a sustained gold rush.

From this characterization by Olsen, I draw two main observations. First, as long as the gold rush exists, things are more likely to change by introducing a new process. Second, the economic downturn in the technology sector over the past few years can be largely interpreted as suggesting that the value of the gold is rapidly diminishing precisely due to the absence of associated quality of design, usability, and manufacturing. All of this argues, I would suggest, all the more for the need for the changes that I have advocated in this essay.

[7] Personal communication.

Is the Glass half Full or Empty?

In many ways, the picture of our industry (at least in terms of how I see it and paint it) is pretty depressing. There are very few exemplary role models to follow that might lead us to see it in a more positive light. Perhaps I am perverse, but I actually take heart from that. I think that this is rooted in my inherently competitive nature. What better situation it there to be in than one where you not only see the general problem, but also see a path around it?

I do not think that the problems facing our industry are insurmountable. Hard, yes, but not hard to see, just to implement. I have a theory that says that the world generally prefers hard solutions that are easy to implement, whereas I think that the answers we need are in simple (verging on common sense) solutions that are hard to implement; that is, solutions that require vision, leadership, and perseverance to make happen.

The good news is that the approaches that I am suggesting here have already proven themselves in other industries. They also "make sense" and can be articulated in a clear way. And, more to the point, I believe that I can demonstrate that they can work, through the case studies that will form part of the live presentation accompanying this paper.

There is another way to do things. My hope is that this effort provides some help in moving along this path.

Acknowledgments

The ideas expressed in this paper evolved over the 8 ½ years that I spent as Chief Scientist of Alias| Wavefront (now Alias Systems) of Toronto. I owe a great deal to both my Research Group and Design Group at the company for originating, refining, critiquing, and most significantly, deploying these ideas. I have also benefited from comments on drafts of the manuscript from my friends and colleagues Dan Olsen, Andy van Dam, and especially Dave Kasik.

While what I have expressed here did not emerge from a vacuum, but rather was the result of an evolving collective exercise, I nevertheless take responsibility for any distortions or weaknesses that may be reflected in my telling, while fully acknowledging my indebtedness and respect for my colleagues, who helped my thinking evolve along this path.