



creating the intelligent interface why wouldn't you do this?

By Sue Hardman

(ten steps to engineering high-impact software products)

Imagine a concept where you could create a seductive user interface not only at the front of your technology products, but through all aspects of your technology -delivery across your company or organization.

Good service doesn't have to be face-to-face or product-to-user. It can be over the counter or over the internet or even across a simple screen display. It matters and it is providing a competitive edge for some of the world's largest consumer companies who openly admit to wanting to provide leading-edge technologies and a better 'customer experience'.

As a Development Manager, if there was a better way... would you use it? If there was a way you could reduce the customer-reception risk to virtually zero.. would you use it? If you could find a way to reverse engineer your customers exact expectations into the interface of your products and the interaction layers of your company.. would this not make sense? Today's interface layers present an opportunity to layer conversation between you and your user.

There is a better way, but it involves letting people with alternate skills into our development labs. Human Factors and the science of 'usability' is evolving out of academia and into the labs of the worlds largest software companies, buying them time, allowing them to fully engineer the product experience in order to produce attractive, transparent and human-centric intelligent interfaces as well as enlightened, progressive user/customer interaction layers.

The customer experience can be captured, orchestrated and replicated even from a distance!

This is the new way of thinking in terms of creating an intelligent user interface (intelligent UI) and an intelligent interaction stream (remotely) between your customer

and you. Once studied and harnessed, this experience can be engineered in and can conduct your consumers or other stakeholders on a journey through your products, processes and organization as if they walked through the front door.

Hear me out...

Paper Abstract and Introductory Notes:

Why would you buy a Jaguar car? Is it the look? It can't possibly be the price; one doesn't purchase a Jag because one is price-conscious... Maybe it's the look and the fact that the marketing assures you that the world around you knows you're a Jaguar buyer – you must therefore by association, be successful. The look of the car and the marketing positioning work around the associated prestige serves to broadcast a certain image emanating to one's peers to say you've 'made it'... of course!

But here's the thing, once you've loved the look which got you into the showroom in the first place, and you've got past the price tag; Jaguar, and most other prestige car manufacturers, are superb at constantly reinforcing your wise purchasing decision by applying engineering work to the feel of the car. It's in the layout of the dashboard, the instrumentation and the feel of the seats, the change of the gears, the feel of the engine feeding back to you, even in the feel of the buttons, the sound of the engine and how the door opens and closes.

This is the level of engineering that will stop you from looking at a different car and will constantly work to make you feel really good about your superb choice of car into perpetuity, into potentially buying a second one and in continuing to feel particularly good about paying those on-going maintenance fees. Many things around us are engineered for 'feel'. The 'feel' of the product is what makes or breaks your product cycle – life through death of the product series.

It is the 'feel' of the car and of the dealership that allows you to enjoy the entire experience. You're not actually buying a car... you're buying a way of life.

Why then do we as software engineers, business application builders, web developers and systems engineers **not** pay attention to, or even respect the 'feel' of our solutions? If we did the same amount of 'feel-engineering' would it pay off in spades like it does for "consumer products"? A number of world-class companies believe so...

Background:

Technology engineering, and more so, software engineering remains something of a cottage industry. It's built in back rooms from the ground up, pieced together, assembled and tested, reassembled and retested continuously until a single result is achieved – the ultimate in feature presentation. The product is 'put on the market' and constantly upgraded as we begin to fix bugs, add features and so on. Isn't it time we realized that a lot of our 'product' stays out there for a long time and that the maintenance cycle is going to ultimately cost us margin? It is this feature bundle that has become the focal point of our development and our marketing – and it is to our detriment.

We haven't really come a long way in the 30 odd years I've been involved in technology-based product development. In fact, I suspect we used to be better at testing, witness the steady stream of 'hiccups' experienced even in this latest calendar year¹. We may have got better and faster at putting code together, and we may be more open to using tools, but we really haven't emerged very far when it comes to producing and manufacturing world-class people-centric software products.

This is less true for our colleagues in other disciplines. The hardware guys and the telecom guys have, by comparison, come a long way. Their managers appreciate the small things that make a product more saleable, more installable and more serviceable and even more importantly, more appealing in the eyes of the buying user. In fact, many telecom and the odd technology product companies actively work to engineer the entire customer experience and translate it inside and outside the product functionality and into the company operations. In extremes, company operations are reworked to better fit the operating model that will be more palatable or more appealing to the end user. It's worked spectacularly well for companies like mmO2 (UK), Cognos and RIM who actively strive to better the entire product and customer experience.

It's about time we in the software industry begin to look at doing the same. We need to realize that there is more than just hard work needed to produce winning product, we can work smarter. One of the ways to do that is to proactively engineer the product and the customer experience – add standard technology ergonomics into the interface layers. You effectively reverse engineer the buying users' expectations into your product or service in order to create an interesting and responsive experience in buying, finding, using and depending on your product.

This paper presents 10 ways you can use to begin to approach understanding what is needed in assessing the value of the human experience and human behaviour side of your product, capture it, orchestrate it for effect... and engineer it in.

¹ RBC banking, American and United Airlines outage, Hydro Outage, CIBC banking...

Number 1 – Know your customers’ users – all the users.

Without doubt, every systems and software engineer on the planet absolutely knows their end-users – right? How could we possibly build anything without this knowledge? Before we start to code, we meticulously research the user and we meticulously assemble the code so that our users can access every possible feature and combination of features, every bit of on-line help and every little cob-webbed and richly featured alleyway in the inner workings of the product. On completion, all the product features have reveal-paths, they are (hopefully) clearly laid out and, the only management challenge we might face is in putting a set of standards under our growing UI team so that we code the correctly sized and annotated buttons, icons and gizmo’s. That’s the way we’ve always worked and we’ve produced masses and masses of really successful product. Well... have we?

In reality, what we’ve done during the product-build is create the “centre of the universe user” or a mental picture of a single user who has the same intimate familiarity with every feature just like we (the Dev managers) do. This is effectively a single hypothetical feature-user who is intimately familiar with every single aspect of our products, every function, every soft-switch and every repercussion in every field, attribute or database table. By the time we’ve finished, this user who knows absolutely everything can’t possibly exist – if they did, they would know more about our product than most of the individual members of the development team.

But that is not reality. This feature-user doesn’t actually exist!

Once on the open market, our product will face a series of the customers’ staff that haven’t the first idea of what our aims were in building the product, couldn’t care less how elegant it looks or how technologically -advanced it is, they only care how it works and they just want to get their jobs done. They are effectively being forced to use our products in order to do their job.

I want to suggest, there are actually three types of ‘user’ we need to be aware of:-

- the *feature-user* conjured out of the worst-case scenario developed in and by R&D,
- the *buying-user* that Product Management and the marketing guys are so concerned about, in reality a buyer-user or at least a buyer-champion, and
- the *using-user*, the real human being at the end of the chain who actually has to use your tool/product in order to do their job or at least reap the benefits the marketing guys promised.

To get a first hand view of this gap between users, perform usability testing per the example above, look at the real end users and watch closely as the point-of-astonishment dawns, how fast it takes to reach there and look what you could do to push that point

further away or work to make it disappear completely. This is the very heart of human factors work.

Example:

I did a similar thing and had a rude awakening in producing a leading-edge ASP for JetForm Corporation, a web-based product called Formsplanet. Formsplanet was a collection of interesting business forms out on the web for small businesses to use.

We didn't provide only one of anything. We had up to 10 invoice layouts, about 15 types of timesheets, order forms, plain stationary, statement of accounts, that kind of thing. We'd coded the front end to be supremely simple, after all, we couldn't actually reach our users, they entered the product via the internet and had to be self-serving and self-sustaining as far as possible. The concept was; our user would go into a library of forms, tag those that they wanted, and build a small but specific collection. When they had decided which they wanted, they could go into a very simple graphics package and put in a logo, size it, lay it out and put (generally) a header and footer on the form. Simple!

The forms were electronically fillable, you could protect the face so that no one could tamper with it and you could pass it through a series of "states" which in turn, the business owner could define. The whole thing was a very simple 4 or 5 step process and so we had the opening page set up that way.

Prior to launching, we were smart enough to get some target small-business customers into the office and have them, with no hints, go through the interface. Without exception... not a single customer could navigate this 5 step process. Somehow we hadn't framed the product correctly and we certainly hadn't positioned it correctly in the users' eyes. Coupled with the functional mishaps, all of our test subjects thought it should interface directly with QuickBooks (or other) accounting package. Not only that.... all our users got completely stuck in how to go from having empty, nicely logo'd forms, to that of having semi-filled and fully-filled forms in the "state-machine" for basic processing. It was excruciating to watch our users try and navigate through. We were incredulous! What wasn't obvious about typing into the open fields? They hadn't even realized that the Step numbers were "clickable".. What should have been a very simple interface turned out to be a very frustrating exercise for both my target users (who were nice enough to come in and help us) and my development team who couldn't grasp why the users couldn't just click on things. What is it that makes users "click"?

Bank on every using-user knowing only (approx.) 7 features... Don't think the 'user' is going to read the books and use the help-information you provided, they don't. If you don't believe me, take a serious look at how many features you know in products you use everyday – Microsoft Word for instance!

Either because your customer pays you, or because these users are so frustrated looking for what they consider fundamental things, confounded and confused users will continue to phone you... guaranteed! If you change the way a field works, it doesn't matter how many times you document that change, the phone will continue to ring with users telling your support guys that it works differently. The more usable your product, the less the customer will call...

In summary, we need to realize that there is clearly three kinds of 'users' – the mythical ultimately -literate *feature -user* that we build the product for and is the focus of the development group; the *buyer-user* that the Marketing guys think is out there (and is providing competitive pressure because they pose a constant threat by looking at a competing product rather than yours – and – they use showcase features of your competitors interface), and our real *using-user* who has had your product foisted on them to make them 'more effective'.

It is the using-user whose emotions and imagination can be triggered either negatively or positively by your product. Their behaviour can be observed, captured, re-orchestrated and engineered into the product by Human Factors and human interaction specialists. Strangely enough, this is the **ONLY** user that gets ignored in most R&D product initiatives – but is the focus of a new-age product supplier that will present you with the largest threat. Get to know all of these users, use them to your advantage but always test for the latter using-user, study their behaviour and learn to build to capture their imagination moving forward. This is the key to longevity and it is the key to producing an intelligent interface.

Number 2 – Thinking outside the Window box..

Stop! Put your hands in the air... and walk away from your computer.

Play Team Fortress or Myst – do you see a single Windows box? When you pick up a gun or you walk down a path, did you have to do it with a drop-down menu? Why do we continue to think in the Windows metaphor? Worse still, in the Windows/VB metaphor! How many web sites do you see with a Submit button... and a form to input and an OK button? The internet has a one-click interface and no limit on what can be rendered from your imagination. Do you know how totally inappropriate these Windows buttons might appear as an interface in a medium outside the Microsoft environment?

In my opinion, in the modern software age, only the video gamers get the interface right. You enter the game, you get the game landscape from the look, but the 'look' interface becomes irrelevant once you are oriented. The only thing the gaming user cares about is not getting (figuratively) killed or at least getting to the end of the game or to the next level successfully. Play Solitaire on a standard Windows PC. Are you manipulating

Windows – or are you playing cards? This level of interface transparency adds not only a level of functional elegance, but also captures and holds the imagination of your using-user.

Why then do we continue to create these horrendous interfaces that require interminable interruptions by our end users? Cascading menus, drop down selection lists, radio buttons.... Is there not a better way to do it? Think completely outside the box; fire the imagination of your using-users!

Example:

Over the last few years I've been heavily involved in delivering accounting-based systems. Green screen isn't very modern but it has a few attributes that are helpful to accounting people, one is the complete lack of 'drag and drop'.

There still exists a function in Finance where accounting clerks enter vast arrays of input data based on a controlling hash-total. The only way to do this fast and for your user to get through the work is to strip off the interface features, let them enter numbers followed by a de-limiting character (preferably their choice) in a columnar fashion. Decrement the hash total and let them have at it. No dragging and dropping, no multiple tabbing through irrelevant or repeating fields, no mouse-work whatsoever.

Green screen really worked in this kind of application... don't put Windows specific or "cool" things in the middle; after the first 'wow', it's just a feature that needs to be got rid of.

Rethink the interface... totally. Disassociate it from your feature and function array, look at the domain expertise of your real using-user and find another metaphor, potentially a better one and one that fits the mental model of what the using-user expects. Find a better way!

Number 3 – Model before you build.. go on, it's easy..

Are you still writing reams and reams of functional specifications? Does it contain 'screen layouts'? Is this document key to the materials you pass around to get approval to move forward? While this kind of spec is functionally necessary and useful or even key to the Development team, it really has little relevance to how a real user will perceive your product. Seriously, how you interface your product and how you display and use technologies can be disconnected. More to the point, it can be re-engineered to model an 'interaction' with your using-user and your company/product group.

This is evident if we look at some of the more-established technologies, those that have been around in automated forms for decades. Accounting systems are a good case in point. Other than the interface and the machine horsepower and the outright scalability (which is worth millions and shouldn't be totally discounted), there is very little

difference between Oracle Financials on a mainframe and QuickBooks on a Windows PC. However, when you look at the interface, they are a million miles apart. Another good example of a tailored interface is Airport Arrivals and Departures – one glance gives you all the information you need, this is a green-screen interface with the addition of colours denoting on-schedule status. Neither of those applications bares any relationship to the type or size of data that lies beneath it.

Before you build... construct a wire-frame or model and reveal it to real customers and real human beings. Use PowerPoint or Paint or even folded cardboard or modeling clay if you have to. Get their feedback; study what they do and how they go about it. If possible, leave your model with them so that they can cascade through the process formality and ponder on their aspects. Learn to connect with real users and what integral part of their lives and tasks your product will be to them – study and learn from their behaviour. Point-n-click and drag and drop is of no use to someone who is literally keying in numbers in columns (order entry) or receiving EDI data for validation and audit.

Wire framing or modeling has long been used on projects that routinely cost mere thousands of dollars. If you purchase a new booth for the marketing guys, if you architect a building or even if you build a floating platform and tow it out into the ocean (the multi-billion dollar Hibernia project) you would expect that some kind of model would be constructed and used to communicate shape, size, fit, form and function. Forget the detailed way it looks – it's irrelevant. An artist's rendition would do it. If there is a storyboard.. cartoon it out, put it into frames or any other means as long as you can communicate how it is going to come together at the end. Give it to someone and let them try it – you'd be surprised at the feedback you get. Oh, and in the web, there's no better way than to do this 'cartooning' or 'storyboarding' to demonstrate page navigation (product flow) and click through (feature selection).

The same is true if you are purchasing a new house – who in their right mind would buy sight-unseen from a written spec?

Why then is this **NOT** the case with massive software projects? We routinely spend hundreds of thousands of dollars, if not millions, with no model, no wire-frame, no straw-man, no diagrams.... and no pre-use study! There is no technical limitation to this, it is done at the conceptual design layer (the layer we've typically dispensed with in order to design at the logical layer - the layer of our own understanding as it is constrained technically), find a way and, more to the point, find a way to present it to real people that are going to be using your product as a tool.

For example:

Look at it this way; your code base is easily as large as the screenplay and script for a full-featured movie.

In order to communicate to the “team” (makeup, costume, prop-guys, camera people, stunt team, actors, continuity people... and so on) the management group pulls together teams of visual artists to render a full storyboard. This is used to communicate the flow, the shooting sequence, costume requirements and the overall plot development and story layouts – it even drives the final editing team. Before the shooting starts, there’s not a single person on the entire team that doesn’t have a mental picture and visual model of what the ultimate outcome is going to be.

The cost of these movies is routinely dwarfed in comparison to some of the software teams I’ve actually worked on and managed – and hey, we just ‘winged it’ – how irresponsible is that? You have to wonder how we ever got to the end of the project... come to that; where is the end of the project?

Do it the safe way – work at a conceptual level, create the conceptual design before moving into the logical design, build a visual model of what is proposed and pre-test it before Development begins.

Number 4 – Your competition is probably doing this...

Microsoft, Cognos, SAP, IBM all have sizable human interaction groups and human factors engineers, they test, they study and are responsible for the receptivity, assembly and usability of the using-users technical interface and its design. More to the point, they are responsible for the customer or users experience inside the product or web site... or even Support Group.

These companies care about how using-users perceive the product, how they behave inside the process, how they use it and, more to the point, how they can’t use it and where the point-of-astonishment is. It is those ‘points-of-astonishment’ that make the phone ring with support calls and put their product in a negative light against the competition. All these companies want to be perceived by their market as agile, progressive and responsive.

Human interaction people and human factors engineers are the ones that can tell you if your product is going to fit the using-users mental model of how they would expect things to work. They are behavioural specialists and they work within the intuitive model of the user so that you can reproduce it in your product to create comfort and convey confidence – this is a winning advantage and it absolutely creates loyalty and integrity.. take their advice! When geared towards the buying-user, you can create a competitive edge. They can tell you what will appeal, what will work better, what is more likeable, what is more installable and what is going to be sustainable over time.

Consider this;

Do a quick check on the people inside your own Development group. Chances are; you have 'Microsoft' people or 'Oracle' people or even 'ColdFusion' people... ask yourself how the owning organizations have managed to get and to keep this level of loyalty. Oracle, Microsoft and Macromedia all have sizable human interaction engineers working inside their products architecting the interaction sequences between your people and the products. Check out MSDN, it is a lifeline to developers and a conduit back to Microsoft – a hugely powerful conduit.

Secondly, on informational and transactional web sites readability and usability have to be very high. If not, then what's the point in producing them in the first place? If you seriously want to digitally service your buying-user or your using-user remotely and you need them to actually complete the process then this has to be heavily studied and orchestrated accordingly. How can you do it without in-depth study and rework? Make sure you measure the using-users exit-points (past the point-of-astonishment and into the realm of abandoning all interaction with you – hitting the close-button for instance), work to adjust these points and find a way to get more and more of your using-users (effectively the consumer or citizen) to complete the process.

A human interaction engineer will study all these points, tell you the exhibited (versus desired) behaviour and make recommendations of how best to play into the users mental model. This can be usability tested and captured and no, your developers are **NOT** qualified to do this. It needs real behavioural specialists!

Number 5 – Deadlines vs. dead-aim...

Pressure to meet deadlines is a factor that R&D managers work with everyday. It is the reason we compromise (functionality), put out upgrades (with features that were missed out of release 1.0) and it is the source of nightmares and the bulk of our professional stress. Code freeze date is a published day when the entire company is going to come together in a single (and usually massive) release. No wonder we have 'high-fives' parties!

Will doing usability work and human interaction work compromise the deadline?

Answer; only if you didn't factor it in!

Real world-class companies factor it in. And factor in that this is potentially the only way they will get it right or at least get-it-close the first time around. If not, and your beta using-users hate the product, then that is the **ONLY** valid cause for slipping the deadline.. no? (Besides, if your product goes to beta and your using-users really do hate it or can't use it; you have other, more pressing, problems possibly involving a new, start-from-scratch interface strategy – and who ever does that? It's more likely you'll launch anyway with something piloted that you already know 'misses the mark'.)

Example:

Twice since I've joined Maskery, a period of only three months, I have had cause to talk to software product organizations who publish Knowledge Bases out to their premier supported customers in the installed base. Both companies really care about the customer -quality feedback they get and really care that their customers can get to the right information at the right time, preferably without having to resort to making a phone call. In reality, this rarely happens for them, customers report dissatisfaction and potential solutions have been under review for quite some time. One company is a global software publisher where customer support quality is highly valued and the other is a sizeable Government of Canada department who wishes to better service Canadian citizens remotely.

Both organizations conclude that customers are having difficulty finding information – both organizations have had their best technical people on the problem and both companies report they are now seeking a better search engine.

How can this be? There are only three real search engines on the market and both companies use one of them. Clearly the customer is asking for information based on criteria that doesn't match their view of how it should be structured – if they did, they would be able to articulate the search string to get the 100% answer 100% of the time. Both companies are actively budgeting for a "rewrite" - one has a budget of \$250k to solve the problem.

The real question is; if you, as an R&D manager or webmaster, blow \$250k on a new solution – what assurance do you have that it will be better the second time around?

The real problem therefore is; your customer isn't asking for things inside the model you choose to publish, they are not using the 'correct' search criteria. You need to get someone studying what the customer (or citizen) thinks they want and then re-engineer the information model to create the right solution for the using-user. If you could do this; then any search engine (even old ones) would do the job. Get a human interaction person involved, they can tell you for way less than \$250k how to solve the real problems.

If you don't look at this problem as a human issue, then what you spend to solve what you perceive as a technology problem might be irrelevant – you will remain a million miles away from your customer regardless of the size of your budget and regardless of the state of your technology solution.

Deadlines become irrelevant ... if you are on the wrong track to begin with. Do you want to get it right and solve the problem, or do you want to meet the deadline with a potentially different set of problems?

Doing real using-user studies and getting a real behavioural specialist on the problem not only simplifies the development sequence but also goes a significant way towards 'de-risking' the outcome. Getting it close on the first pass... is way better than putting out

'surprise' products to your installed base or even new customers. Find the customer's mental model, work within it and engineer around it.

Take advantage - why wouldn't you do this?

Number 6 – The phenomenon of repeating errors...

For decades, in a very concerted, heads-down mode, we have continued to develop more and more code. The amount of code we've produced is massive and much of it is still deployed... however; much of it is also out of date, has worked in the same manner for thirty odd years, is continuously upgraded and retested only for functional validity.

When a plane falls out of the air, we meticulously piece it back together and look at the mechanics, the hydraulics and all the systems (including human systems) that kept it in the air – only on total post-crash reconstruction do we make recommendations for change which are universally implemented. How come this doesn't happen with software and with business applications? Don't tell me we continue regardless? Well, it seems we do....

Consider this:

A recent publication² actually won the National Book Prize 2004 in Canada. On thoroughly enjoying this book, I went off looking for others and I was incredulous to learn that there are many, many books dedicated to the subject entitled 'technology disasters' (many of them software based). Some of the instances that Kim Vicente outlines are nothing short of horrific. I was stunned! I had no idea...

My surprise is not that some of the software we develop causes sometimes-fatal consequences, but the fact that, over time, we continue to repeat those errors – time and time again. On reviewing the available materials, I was shocked to learn that although many of our failed-systems are studied and reviewed, very little is done to ensure retro or reverse implementation. We never seem to learn and I, for one, wasn't aware this material was available. The sad fact is; all the errors we coded into the mainframe environment and subsequently redeveloped into the mid-range environment and then the database environment are alive and well inside the Windows environment. Chances are; it will get ported into the IP environment. Worse still; we don't seem to be passing on the 'hard-knocks' that we learnt the first time around to the next generation of software producers.

When you look at large, highly organized bodies like the military and such like, they do their project work and then they go into a 'debriefing'. No rank, no prestige, nothing hidden, all is revealed and it is studied extensively. They look at what worked, what

² Kim Vicente – The Human Factor published by Alfred A Knopf, Canada 2003

didn't work, what could have worked better and where the obvious errors were – and what could be learnt for the future. How come we don't (as a body) do that? There are masses and masses of materials on failed projects and technology disasters. People perform in-depth study of the weak points, the cascading series of events, the failure points and the humans in the systems that made it happen. As an industry, we are guilty of either not caring or not looking or not responding or not having budget. For whatever reasons, the lesson from previous mistakes or misunderstandings (user confusion) is rarely taken into account when designing new systems. The consequences of this... repeated mistakes... over and over. People died because a using-user made what they thought was a logical decision!

We work hard, long hours on projects with very tight budgets and crushing deadlines – but clearly, inside our development work we could at least look to get a better and different perspective. Study our failures and get serious about continuous improvement³.

The one thing that is abundantly clear; using-users reaching the point-of-astonishment inside your code base will **NOT** make the decision you assumed they would make – they will make the decision based on their own 'educated' guess usually applied under stress – and you, as a developer of these tools, have no idea what that will be. No idea, at least, unless you actually study it, allow for it, build for it, capture it – and redirect things.

There is no such thing as a dumb-user, just one you haven't studied.

Number 7 – If you're serious about your product being usable, put a number on it...

How usable is usable? Well, if you're serious and you really want to get better, work to put a value on it.

For instance;

In the Knowledge Base example above, if we could work towards making things more 'findable' or 'remotely solvable' what realistically should we expect to happen? If customers successfully find their information or solve their problem (and they don't make a phone call) would be considered a successful resolution?

Logically then, our usability expectation should be that a number of customers (say, 20%) manage to complete the process and that telephone volume for first level support (say, 10%) should drop. If we were to absolutely get this right – then our usability metric would be that 20% more customers complete the process (which you can now measure through standard web management tools) and that our phone volume (for customers seeking

³ Maskery, twice per calendar year, put on a seminar called 'The anatomy of IT disasters', its worth attending and you'd be surprised at how many catastrophic elements come together to which we, as an industry, contribute to. Register via the web site at www.maskery.ca

information that we already know exists) would drop by 10%. Not unreasonable you might think.

Usability is not something you can just expect to ask for; you have to work at it. Maskery does work for Government and you'd be surprised at how many RFP's openly state "must be easy to use" or "simplified for the user" without putting a clear and measurable metric expectation on it.

If you really do put effort into re-doing the on-line Help system for instance, put a performance expectation on it that would demand a measurable increase in its use. It is imperative that you work to measure increase in use and increase in success – how else would you know you got it right? These things are workable, measurable and much more realistic than the intangible 'have it work better' or 'must be easy to use' approach we tend to take.

If the marketing guys tell you not to worry about usability because you plan on selling training, then put a usability value on training. Report the number of trained customers that phone in with first-level questions – how effective then would the training be? The world's most successful technology companies constantly work at bettering the customer experience manipulating it to their own end and engineering it into the product. By that, I mean the entire product cycle including the support and upgrade sequences.

In Government, for instance;

Over the last few years, I've worked extensively with Government departments (both in Canada and the US) who are putting a concerted effort into reaching and remotely servicing their citizens. Oftentimes, the RFP states, "must be easy to use" or "the citizen must be able to remotely process". These are unmeasurable, untestable and unreachable goals. The outcome is going to be someone's opinion of a binary negative or positive result.

If you think that (for instance) internet -savvy citizens should be able to remotely process their tax return or get their fishing licenses, or renew their drivers license (no one, to my knowledge has achieved that one yet) – then state a number. What's reasonable? If you think 20% of internet -savvy tourists visiting your web site should be able to get their fishing and hunting licenses on-line – then ask for that and test for that. It can be modeled and tested prior to investing the development dollars needed to bring up a 'cool' (read: expensive) new web site.

If you think that students should be able to register for classes on-line, again, stick a number on it that is reasonable. What would be a good number? You would expect students to be internet and technology savvy these days – 60%... 70%? Model it that way, build according to the wireframe and test for it as a part of the completion and deliverables. On 'go-live' day, this has to be an achievable goal.

Measuring how usable or functional a digital service or a new technology product is... should be an integral part of the delivery cycle.

Number 8 – It's going to cost you less to get it right the first time...

Usability is a science, it involves study and empirical testing that will get you hard numbers and a behavioural model that will dictate the essence on the 'feel' side of your design to create an intelligent interface. That design may not be the final, ultimate and only design, but it's a much better way than the 'fair guess, make it look cool' approach that I used to take.

Usability engineering is expensive especially if you didn't budget for it in on the first pass. It costs a similar amount to get a Human Interaction Designer as it does a Senior Engineer doing the same job for the same amount of time; these are highly qualified and highly educated people. The cost however, is offset by the receptivity and usability of the product when you launch. Users will actually like your product, complaints will be fewer and a little more realistic and you might be able to dispense with all those support calls that get logged as 'UBF's'⁴. And you will get a blueprint design that will drive the development team in a cohesive fashion.

Human interaction engineers can gain you as much traction out-of-the-box with software products as Industrial Designers do with attractive-looking appliance hardware and office chairs. Your buyer will like it, it will ergonomically fit their business objectives and processes, it will ergonomically fit their staff and your customers will get on board with fewer phone calls to you because they are stuck. Your support costs will be lower, your users will be on-line and active a lot faster – and they will actually like what the product does for them. All of this to say... this is the clear way to de-risk the development cycle. The shape, size, fit, form and function will drive the product further into the account than the cool-look you were working on – it can be tested for, it can be engineered for and it can be delivered in the same amount of time with significantly less worry of failure. Marketing costs are reduced if the product is received positively through the product evaluation cycles.

Not only is this a new approach, its one that guarantees an emotive response down your buying-user and using-user chain once the product is launched and it gives you an edge in the marketplace. This means less support nightmares, more targeted on-line help, more targeted documentation and training – and less on-going costs. You can also begin to

⁴ User Brain Failure – a term commonly used to close calls in telephone support centres. Usually indicative of the user not understanding, not comprehending, forgetting commands or failing to respond in quite the right manner.

predict feature need and support and sales models... the human factors people will tell you the 20% new features that will solve 80% of the support phone calls.

One final word here; I continue to get asked by R&D VP's if you can express human factors or human interaction engineering in terms of ROI. The short answer is... yes, of course! It is always financially provable that better product and better product experience management manifests in significant cost savings (and revenue-ramp) both to the buying-user (customer) and to the software product producer. This is a subject I don't want to get into here, however, one needs to look at the ROI issue in a more simple fashion.

If your Development team have ever re-tooled to learn another language (Java or OO for instance), or you've written and rewritten your process functionality a few times across a number of databases and a number of machine platforms – I'd love to have a serious ROI discussion with you. Let's compare that payback cycle!

Similarly, no one in their right mind would ask a bunch of Quality Assurance or Quality Control (QA/QC) people to justify their ROI from testing. You just wouldn't do it! Testing, designing, building and deploying highly usable, tailored-to-fit products has always worked and will continue to do so.

Food-for-thought:

R&D departments are currently spending more money on ergonomic designs of their developer's office furniture than they are on ergonomics designs inside the products they produce.

Good customer service and first-rate product invariably wins the ROI battle... getting it right the first time wins costs vs profit battles as well. If you can build-in attraction, simplicity, comfort, elegance, installability and supportability and that 'feel-good' feeling by doing this.... why wouldn't you do this?

Number 9 – Engineering the customers' entire experience...

Usability is defined as “*the science of interfacing your customer with your [organization] company*”. The science of usability isn't limited to the product internals but in fact, spans the entire delivery cycle and the company.

How the customer finds the product, how they receive it, how they acquire it, how they purchase it and how they use it right through to how they find the after-sale service. This is far more than the normal product cycle – perception and particularly potential-customer perception, is everything!

Working with Maskery, I've been surprised at how many industries use usability to gain a measurable competitive advantage. I've been particularly surprised at how other parts of

the technology industry outside software development actively use the science of usability across the entire spectrum – the entire product and customer experience engineered in order to win mind-share and additional business. More to the point, how these industries study and reverse-engineer a positive buying and using experience back into their products lifecycle in order to invoke and manage the customer's and user's response.

If you know who your customers are... you can engineer the entire product and customer experience. Companies like Nokia, O2, Bell Sympatico, Amazon, Telus, Disney and the like have done this and continue to win mindshare. Disney actually calls it “imagineering”, a wonderful phrase that typifies what these processes are engineered to do. The entire product life cycle, the support cycle, the upgrade cycle and the delivery mechanisms are engineered for impact, for penetration and for seamless adoption.

This is proactively being done today by cell phone manufacturers, telecom industries, consumer products and consumer electronics – electronic service delivery organizations, direct marketing companies, the retail sector, even the coffee machine producers... everyone but seemingly the software and soft technology industry. Have you ever wondered how it is we can feel really good about spending upwards of \$300 on a coffee machine? Everything you aspire to purchase may have been engineered in this fashion.

Just imagine; if you could engineer from your customer back... through the front doors of your entire company... capture the entire ‘customer experience’ inside your product and reflect it back from using your product interface.

Why wouldn't you do this?

Number 10 – The road to commercialization...

Finally, I live and work in Ottawa, Canada. We are an R&D hotspot for leading-edge telecom and software technology boasting a population of almost 100,000 software engineers in a valley that stretches almost 100 miles. There are almost as many software engineers in our population as there are civil servants. We are highly creative and technology savvy. Most of the major software providers have R&D facilities in the area.

At the strategic advisory and board level, I work closely with many small technology companies whose innovative product value can take your breath away but who continue to struggle on how best to take those products to market. Taking advanced technology products to market, finding a way to optimize development of them, doing competitive analysis and market research, putting them through the manufacturing process, selling them, installing them and servicing them is done largely on a ‘best guess’ hit-and-miss basis. It is our one weakness – innovation commercialization!

Strategically, using the science of usability with empirical testing techniques and real behavioural-based ‘feel’ engineering, we can birth innovation-based products faster, with

as little risk as possible and with the impact needed to create business value to the customer – optimized to the customer. These services are available in companies that boast human interaction people and human interaction engineering such as Maskery.

Human interaction people are skilled behavioural specialists that can perform user-centred quantitative and qualitative analysis and testing with the necessary objectivity and technology-savvy to allow you to look ahead, predict the response and make qualitative decisions **before** you formulate the business, marketing, support and sales plans. They work at the behavioural level of users and are psychology based. They look at how people respond to technologies and can help you get the right emotive response, evoke or at least predict the desired using-user reaction. Human interaction specialists live in the interaction layers of technology based products allowing you to “bend” your interface in order to take advantage. This is the key to creating intelligent interfaces and intelligent high-energy interaction paths into, and out of, your organization.

Commercialization is vastly simplified if you can objectively look in-depth at potential sales models (and research how people buy), potential marketing models (and how people search for and find your product under this model), potential delivery models (and how people receive and expect to pay for), competitive models and products (and how those potential buyers perceive them in light of your product), down to service models (and how people expect to talk to your service people).

This sequence is typified as; Reach, Touch, Innovate rather than Innovate, try to Reach, hope to Touch – completely the opposite of what we have historically done.

Why wouldn't you do this?

Nutshell:

We technology developers are characteristically the most hardworking people who will do anything we need to do to 'win'. We work horrendous hours, we're meticulous, we manage ourselves and we are 'guru's' in our industry. We will stop at nothing to get a release out... why then do we consider human interaction engineering and human interaction work “airy-fairy” stuff? It just doesn't make sense to me. Even given that we are skeptics at heart... why wouldn't you do this?

Human interaction engineering is relatively new but is proving to be the difference between costly, hard-to-support, difficult-to-manage and difficult-to-implement products and those that are much more readily received, fun to work with and simple to support. We have to accept that our products decay over time – but can be replenished by polish. The provision of a new, intelligent interface is the easiest and most cost effective way to go.

Look at your competition. Do they have products in the hands of customers who pay? Are those products well received? Is the company seen as 'up and coming'? If so, chances are they have human interaction engineering specialists working with them. Think about it...

Do you have a select collection of your favourite products? Microsoft... Oracle... Macromedia..? All of these companies actively use human interaction and human factors engineering to create and deliver product to you. You are a developer and you are their using-user. Your behavioural traits have been studied to get the highest possible impact inside those products and in order to make it an integral part of your day. If you spend hours connected to their web sites looking for information and relying on them to feed you and you're skeptical about new things that do the same but are from different vendors... then you are living proof that this works.

Some products just work better for you than others! Surely you've wondered why that is?

- get to know your buying user and your using-user, find someone who can help you study their behaviour and play to it inside your products,
- think outside the window interface box, no constraints, no buttons.. immerse your using-user inside your interface, orchestrate the product interface directly into their personal domain
- build wire-frames and storyboards to communicate how your product or web site will communicate with real people, rework it and reintroduce new things,
- look at your competition, they may be playing a smarter game with their using-users and potentially with their buyer-users,
- go for dead-aim rather than deadlines – target the right product at the right person, make it an integral part of the product design, reflect the company's values outwards and replicate this over every aspect of the product, the organization and the delivery cycle,
- look at the errors we've repeated over time.. find a way, a better way next time,
- usability, serviceability, upgradeability and simplicity can be engineered and measured. Put a number on it.. it can be measured
- reduce your development costs by getting it right the first time,
- replicate and orchestrate your using-user and your buying-users expectations back into your products, capture the expectations and engineer the experience-path into the product and reflect it back to the real people outside,
- commercialization means finding the right buying, using and supporting experience for your using-user and your buyer-user – excitement can be captured

and replicated back throughout the entire product and even the company. Sales models, support models, upgrade models and so on, really matter.

When it comes to technology interface and technology company interaction, the bottom line becomes harnessing both the product and customer experience inside your innovation in order to capture and hold mindshare.

It's all about shape, size, fit, form and function... its not about the 'look', it's about the 'feel' and our ability to create an intelligent interface with our products and intelligent interaction streams both in and out of our companies through the products we develop.

Why wouldn't you do this?

About the author:

Sue Hardman is a high technology industry and business applications R&D veteran. Her background is in Product Development and Product Management/Marketing and spans over 30 years. For the last 15 years, Sue has worked at a senior level in high technology companies, mostly software-focused, and often consults to both Canadian and US Government on strategic citizen-focused digital service delivery. She sits on the board of 8 companies in Canada, US and UK.

A long time proponent but new recruit to human factors and human interaction engineering, Sue is actively involved with Maskery in a business development capacity.

Sue can be reached via Maskery at hardman@maskery.ca

About the company:

Maskery is a consulting company staffed by human interaction engineers and human behavioural specialists with extreme domain expertise in the delivery and engineering of technology products and digitally delivered services.

Core expertise includes the human and engineering disciplines of psychology (cognitive and organizational), physiology, software engineering, anthropometry and ethnology as it pertains to visual interaction design, statistics and industrial design, user interface design, usability research, usability testing, accessibility design and rapid prototyping.

Maskery helps companies and government organizations “bend” technology and the delivery of digital services to fit the human behavioural model in order to orchestrate the whole product experience and ultimately the entire customer experience to your advantage. Maskery belongs wherever real people meet technology.