Measuring the Impact of User-Centered Design

by Paul Gokin

The paper was originally submitted as course paper for the Measuring User Experience course taught at the Bentley College's Master in Human Factors and Information Design program by Joseph Dumas, Ph.D. The paper received a grade of A, and was praised for being "thoughtful and well argued."

Introduction: Why measure UCD ROI.

The main reason for measuring the benefits of user-centered design (UCD) is to show its impact on important business metrics. Since companies are run by the numbers, it is wise to express this impact in terms that management can use to make decisions: numbers. These numbers can then be used to justify the addition of UCD methods to application and product development projects. For example, at eBay, all proposals, including user experience enhancement projects, must have an ROI analysis attached to them; otherwise they are not even considered for approval (Herman, 2004, p. 1415). This paper examines some of the most important measures for costs and benefits of user-centered design and looks at how they can be used to demonstrate the value of UCD.

Categories of Benefits of UCD.

From the standpoint of the income statement, the business benefits of user-centered design are similar to every other business activity aimed at improving profitability: (a) decrease costs and/or (b) increase revenue. In the case of UCD, it is useful to think about the cost saving measures at two different levels: product development and the company as a whole. This gives rise to three categories of benefits of user-centered design: (1) reducing development costs (and risks), (2) reducing operating/ownership costs (costs outside of development), and (3) increasing revenue.

Summary of UCD benefits.

The table on the following page summarizes the UCD-attributable measures of system improvements and their impact on business metrics for three different types of applications: internal, external, and web-based. Please see the list of assumptions for project environment in all three cases (page 4) before looking at the table.

Business Metrics	Corresponding usability improvement measures:		
	Internal Application	External Application	Web-based Application
Reduced risk of project failure: the product is successful.	More accurate user needs assessment, improved user-analyst communication, result in a product that people want to use. Competitive and internal product evaluations catch product flaws early enough to correct.		
Reduced development cost	Product flaws are identified earlier in the process and are less costly to correct. Only the required features are built. Less documentation is required.		N/A
Reduced deployment costs	Installation and configuration takes less employee time; fewer user errors save even more time.		N/A
Reduced maintenance costs	User requirements are met better so fewer changes are required.	N/A	User requirements are met better so fewer changes are required.
Reduced training costs and increased productivity	Product is easier to learn so less training is required. Employees spend more time doing their work rather than helping others learn the system.		N/A
Reduced support costs, higher data quality, and increased productivity	Design invites fewer user errors and provides better error recovery. Users waste less time troubleshooting problems.	N/A	Design invites fewer user errors and provides better error recovery.
Reduced operating costs: increased productivity	*Fewer* employees are required to complete the *same* amount of work.	N/A	N/A
Increased revenue: increased productivity	Employees can do *more* revenue-generating work in the *same* amount of time.	N/A	N/A
Increased revenue: higher conversion rate	N/A	More customers buy the product based on better organizational fit, lower cost of ownership and improved ease of use.	More customers are able to get comfortable using the web site quickly and transact successfully.
Increased revenue: more potential customers and happier current customers	N/A	More customers are attracted/referred to the product based on its reported ease of use.	More users return to the site. Good customer experience helps “word of mouth” marketing.

Application environment assumptions: Internal application is developed/deployed/maintained/supported by the company. External application is maintained/supported by the customer, but the vendor deploys it and provides free initial training. Training is a part of the product's purchase price. Web-based application is purchased from a vendor who deploys it (rather than developed in-house), but is maintained/supported by the company. Only the customer-facing portion of the web-based application is considered.

UCD-attributable improvements in business metrics in more detail.

Reduced risk of project or product failure.

Janice Rohn cites some disturbing numbers when talking about the costs of poorly run IT projects: 30% are cancelled before completion, 46% of development costs are spent on failed projects, and the total cost of failed projects is an estimated $80 billion (Rohn, 2005, p. 206). Aaron Marcus cites a concrete example where an entire application was scrapped because a user acceptance test found a fatal flaw in the assumption about how data would be entered (Marcus, 2005, p. 24). IT project failures are often attributed to poor user needs analysis and inadequate user-analyst communication (Rohn, 2005, p. 206)—exactly the areas where user-centered design can make a positive difference.

Unfortunately, the risk-reducing impact of UCD is very difficult to measure and can only be estimated after the fact and only on a failed project. For example, we may determine that the project failed because an important user requirement was not addressed. If we determine that the requirement would have been addressed by adding the appropriate UCD activity, we could count the entire cost of the project as a sadly unrealized UCD-attributable cost savings. On the other hand, retrospective analyses such as this help build a strong case for UCD's social internal ROI—the *belief* that UCD reduces development time, risk, and cost.

Reduced application development time and cost.

While it seems counterintuitive at first, adding user-centered design activities throughout the project lifecycle can actually reduce the time it takes to develop an application. The main source of savings here is the reduction in the number of changes made to the product late in development. Better definition of user requirements and validation of workflow, organization, and labeling through usability testing and inspections *early in development* are just a few examples of UCD-attributable system improvements that apply here.

Just like risk management, measuring the impact of UCD on development time and cost is difficult, because of the difficulty associated with linking UCD activities to development time savings. Having said that, some rough estimates can be made. For example, let's assume that in early testing with a paper prototype we discovered that the proposed workflow omits an important step. If the problem is significant enough that the application would not pass user acceptance testing at the end of the development cycle, we could estimate how long it would take to fix this issue at that point and compare it with the cost of fixing it now. The difference is the benefit of UCD involvement.[1]

Reuse: reduced development time/cost on other similar pojects.

Just like programming code that is re-used on other projects within an organization, UCD products such as user profiles and style guides can also be reused. Wilson & Rosenbaum also suggest a database of solutions to common UI problems so that the designers don't have to re-invent the proverbial wheel (Wilson & Rosenbaum, 2005. p. 231).

[1] There are of course caveats with hypothetical estimates such as this one. We've already seen this when trying to estimate the cost of project risk reduction. Here we assume that someone outside the UCD group would not have discovered the flaw earlier. In addition, lack of UCD would probably leave more than one issue unaddressed, in which case it would be nearly impossible to estimate the cost of fixing just that one issue in isolation.

Measuring the impact of reuse is relatively easy. For example, in the case of an existing solution to a design problem, we could simply subtract the cost of adapting the existing solution to the new problem from the documented cost of developing the original solution.

Reducing deployment and maintenance costs.

An application that meets user's requirements well does not need to be updated and patched as often. One way to measure this is by looking at the historical frequency of software deployments, analyzing the trends, and correlating the trends to UCD efforts. Unfortunately, both internal and external confounds may reduce the effectiveness of this approach. For example, changing user requirements may necessitate more frequent deployments. Increased software "bugs" also may make deployment more frequent (internal confound); and fewer "bugs" may mask the effect of UCD, making it seem that the reduction in updates is due to fewer bugs rather than more usable software. However, if we are aware of the reasons for deployments, it will be easier to adjust for non-UCD factors to isolate the effect of UCD on deployment frequency.

Reduced training and support costs.

A product that is easier to learn and is less error-prone will require less user training and support. In cases where training and support is provided by the company using the product, the cost savings are obvious. In fact, easier to learn applications will also have a "hidden" benefit—enhanced productivity—because employees will spend less time helping each other learn the application and more time using the application to do productive work (Rohn, 2005, p. 206). While it may be difficult to estimate the productivity benefit, reduced training and support costs are relatively easy to measure by tracking the changes in training expenses and customer support calls and normalizing those to the number of employees using the system (we've found another confound!).

The benefits of reduced training and support are not quite so obvious in vendor companies that provide support and/or training to their customers. Here, things will depend on how the vendor charges for these things. If support and training are included in the product price or charged at a fixed rate, then the savings are obvious. If support and training are charged by the hour, then improvements here may actually hurt the bottom line! So the lesson here is to be mindful of your company's business model and any short-term losses that usability improvements may produce.

But what about the long term? Can usability have a strategic impact and provide sustainable competitive advantage? Of course it can! Let's continue with the training example to illustrate this. Let's assume that the learning time has been dramatically reduced and, therefore, so was the amount of training required. A vendor company that previously charged for training separately can now include it in the price of the product without having to raise that price as much as other vendors would. This lowers the product's Total Cost of Ownership (to the customer), making it more competitive in the marketplace. This example illustrates that the effect of UCD is not limited to quick, tactical savings, but can have a long term strategic impact.

Increased productivity.

Productivity gains are about doing more work in the same amount of time and/or taking less time to do the same work. Productivity benefits include improved task efficiency (i.e. starting out on the correct path, less time spent completing unnecessary steps), as well as improved task effectiveness (i.e. fewer errors and less time spent recovering from errors). A reduction of errors has an additional "hidden" benefit beyond productivity: better quality of data (Rohn, 2005, p. 205).

Increased productivity is among the most often cited benefits of user-centered design. A possible reason for his may be the fact that they can be easily estimated with relatively high precision. To estimate an increase in productivity all we really need to do is (a) measure the improvement in the time it took to accomplish a task and (b) multiply the time difference by the fully loaded employee cost to get the dollar value of the improvement. Karat suggests also multiplying the value by productivity ratio—"the proportion of time that people are working productively while on the job"—to come up with a more conservative, but realistic estimate (Karat, 2005, p. 122).

Unfortunately, there is a problem. And it doesn't have anything to do with the calculation itself, but with *realizing* the savings. A couple of years ago I was on a client assignment where I used this simple calculation to identify a potential for about $37,600 worth of yearly productivity gains to be achieved by re-desiging the client's internal application. I estimated the cost of the redesign to be around $3,000. "$34,700 net. Nice," I thought. But as I was getting ready to present the figures to the company president, I showed the calculations to a colleague of mine. His remark was sobering: "That's great, Paul. After we implement the changes, whose salary will be cut first?"

The lesson here is that for productivity, cost savings will not make for a persuasive argument unless the management is prepared to lay off employees or cut their salaries. There is an alternative, however. Let's look at it next.

Increased revenue with the same staff.

This is another way of looking at productivity. Basically, instead of calculating the cost savings we calculate the time savings and present these savings in terms of employee hours saved—the hours that those employees can now devote to working on additional revenue-generating projects.

To continue my consulting example, I could have expressed the $37,600 in employee cost savings in terms of the hours the employees could now devote to other projects. It is important to break up the time savings in a meaningful way; here, it is useful to separate the total savings by the type of employee (programmer, project manager, support specialist, etc.) and express the savings in terms of hours per week to keep things manageable. Let's consider salespeople. If you look at the calculation grid (see Appendix A), you will see that each salesperson would be able to do 1 (one) additional hour of work per week. If it takes a salesperson on average 16 hours of prospecting and negotiating to close a deal, and the average deal adds $2,000/year in revenue, then we could estimate that the proposed changes will yield approximately $2,000 x (50/16 = 3 more deals per year) = $6,000/year per salesperson. Therefore, the client's entire 5-person sales force could grow annual revenue by $30,000 *every year*! Even factoring in the 25% commissions of $7,500, the 3-year simple return for the project is $22,500 + ($30,000 + $22,500) + ($60,000 + 22,500) = $157,500! This is a figure a manager will find hard to ignore.

If I knew the time required to complete one unit of work, I could have also expressed productivity gains in terms of the additional units of work. For support personnel, for example, I could have expressed the time savings in terms of additional calls the person could handle. In our case, each support employee would be able to devote 3 more hours to taking calls. If one call took 15 minutes to answer and log, each support employee could handle 12 more calls per week, or 60 calls/week for the company's entire support staff. But we don't have to stop there! Let's assume that each customer makes 2 support calls per week on average. This means that the company can now sign up 30 additional customers and still be able to handle their needs with the current support staff!

The overall idea here is that benefit estimates must be tailored to the needs of the decision-makers.

Think of it as a user-centered approach to reporting benefits: put things in terms that management will find easy to understand and act upon. Sometimes it is dollars and sometimes it is not.

Increased revenue: higher conversion rate.

We've just seen how increased productivity can bring in additional revenue for internal applications by letting employees devote more time to revenue-generating activities. This is perhaps the extent of revenue-generating benefits for internal applications. For external and web-based applications, however, UCD-attributable revenue benefits can be much more dramatic.

A survey Nielsen/Norman Group conducted a few years ago revealed that, on average, a "usability redesign" doubles web site conversion rates (Nielsen, 2003)—basically the number of unique visitors who transact with the site divided by the number of those who don't. The general argument is that increased ease of finding the right product or service and an easier purchasing process result in higher conversion rates. Naturally, confounds—concurrent events like the company's marketing efforts and competitive environment, product/service pricing and availability, technical "bug fixes," new technical "bugs," added features, removed features and so on—make it difficult to make historical comparisons to determine how much of the gain is attributable to usability improvements alone. However, the impact of some confounding factors is reduced due to the real-time nature of web site updates: it may take as little as a few hours to roll out a new design and measure the impact of the changes. In fact, making small changes, rolling them out, and measuring the real-time effect on conversion rates is one of the best ways to show the real-world contribution of UCD.

Vendor companies can also measure conversion rate changes and correlate them to UCD-attributable improvements. For example, those companies that offer downloadable trial versions of

their software can track changes in the ratio of trial downloads to purchases. If the try-to-buy ratio improves after a UCD-driven update, then the impact of the update on the revenue can be measured. Of course, the issue of confounds remains. However, methods like trend line analysis or industry benchmarking can be used to isolate and measure the magnitude and direction of the UCD-attributable change. Customer surveys can also be used to determine whether improved usability does, in fact, affect purchase decisions. So the point is that with a little effort the effect of UCD efforts can be separated from that of the confounds.

Increased revenue: more customers.

Wilson & Rosenbaum suggest that comparisons don't have to be across time: we can compare our product against a competing product/site (Wilson & Rosenbaum, 2005, p. 241). This could be an important strategy for vendors offering products that have competing analogues in the marketplace. In this case, vendors can usability test their product against the competition (or the previous version of their own product) to demonstrate their product's superiority. They can then advertise this fact to attract more customers.

Web-based applications can also demonstrate the effect of better usability on attracting customers. For example, they can use cookie-based tracking to easily measure how many customers return to buy more and correlate any changes with usability improvements over time.

UCD Costs.

"[U]sability costs can range from a small expense of a cubicle or an office and the cost of the employee's time, to well over $1 million for multiple high-quality labs, equipment and employees" (Rohn, 2005, p. 193). In most cases, the most significant cost of adding user-centered design to a

development process is the cost of the people who perform the UCD activities. The cost of their time is usually measured by adding their salary, benefits, work related expenses (i.e. travel), and per person operating overhead to come up with fully loaded employee cost.

Unlike the benefits of UCD that sometimes can seem as elusive as a four-leaf clover, the costs of user-centered design investment are easy to calculate and basically consist of any combination of the following: a one-time expense of setting up a lab, UCD in-house staff's fully loaded labor cost, UCD contractor staff's hourly rate, the cost of recruiting and compensating participants, and ongoing expenses associated with running the lab (videotapes/DVDs, software updates, etc.).

Confounds.

Confounding factors, or confounds, are things that make it difficult to establish (prove the existence of), describe (prove correlation vs. causality) and measure the relationship between the costs of UCD activities (the independent variable) and the corresponding change in business metrics (the dependent variable).

In particular, the redesigned product usually incorporates changes contributed by groups other than the UCD group including additional features (ideas for which came from outside of the UCD group), technical "bugs," and so on which makes it difficult for the UCD group to claim full responsibility for the successes or failures of the final product. Even the impact of UCD activities on the development process itself can be difficult to estimate if, for example, major changes in management or programming staff occur between projects (i.e. budget/staffing cuts or offshoring).

In addition to the *internal confounds* discussed above, changes in the product's environment may impact its success or failure. These factors include things like geo-political events (i.e. wars or elections), overall economic climate (i.e. boom or recession), trends (i.e. industry or seasonal), changes in the company's legal or social environment (i.e. litigation, un/favorable legislation, or publicity), changes in the company's competitive environment (i.e. entering or exiting competitors), changes in the company's marketing/sales efforts (new marketing campaign or hire of new salespeople), business infrastructure problems (i.e. supply chain breakdowns, poor order fulfillment or unhelpful customer service), and IT infrastructure (i.e. viruses or denial of service attacks). The problem is that these factors may impact the project metrics as much or more than any changes in user experience, therefore, potentially negating the impact of the entire UCD effort.

Not accounting/correcting for the confounds makes statements about the impact of UCD activities on business metrics internally invalid. Fortunately, multivariate statistical methods such as factor analysis or principal component analysis can be effective in removing the effects of the various confounding factors. Another approach is the use a metric that is less affected by confounds, but lets us demonstrate the effect of UCD-attributable improvements. For example, we would use conversion rates that than sales to measure the impact of a redesigned checkout process on an eCommerce site. Sale can be impacted by a number of confounds (i.e. changing number of customers visiting the site), whereas conversion rates measure the look-to-buy *ratio* rather than the sales *volume*. The point is that we should be aware of the confounds and use the appropriate techniques to adjust for their effect, or select measures that are less affected by them. The challenge is identifying what the confounds are and gauging the magnitude and direction of their effect.

Putting it all together: calculating the return on investment in user-centered design.

Before looking at the case study I prepared for this paper, let's review what a UCD-specific cost-to-benefit ratio is and what it is not. The UCD cost-to-benefit ratio is a relatively crude measure that compares the cost of UCD activities on a project to the expected or realized financial benefits of these activities. While this sounds pretty good in theory, the examples in the book (Bias & Mayhew, 2005) illustrate the rampant misuse of the method as well as the method's managerial uselessness. Here are some of the issues:

a. ***All* of the project's benefits are attributed to UCD activities.** Clare-Marie Karat cites a study where $450,000 in UCD expenses "yielded" $60 million in revenue giving a head-spinning cost-benefit ratio of 1:133 (Karat, 2005, p. 125). Nigel Bevan attributes Israel Aircraft Industries MPC's entire $400,000 increase in sales (and the additional development and support cost savings of $380,000) to the $27,000 investment in UCD for a cost-benefit ratio of 1:29 (Bevan, 2005, p. 592). Even in the absence of all confounds, only those projects where *all* of the new or changed features championed by the user-centered design effort would qualify for this generalization. This is never the case.

b. **The additional costs associated with the changes in the benefit measures are ignored.** Bevan's calculation completely ignores the costs associated with additional sales that must be incurred. In this case these expenses may include additional deployment, training, support, and materials costs. In the case of an eCommerce site selling physical goods, the insult is worse still where most of the sales revenue is "eaten up" by the cost of goods sold.

c. Claims of future benefits **don't take into account capacity, time, and other "bottlenecks"** outside the areas touched by the project **that may prevent the benefits from materializing** to the estimated extent**.** To use the physical goods eCommerce example, the merchant may

not have the warehouse space or staff to fulfill the additional orders. Construction of additional facilities and hiring/training additional staff take time and money.

d. **The time value of money is ignored.** Any calculation of return on investment that claims monetary benefits to be realized at some point in the future must calculate the present value of those benefits by discounting them at a predetermined rate. This is basic accounting.
e. **Useless in decision making.** In order for management to evaluate the attractiveness of an IT project as an investment, all relevant costs and benefits should be included. Stating that for every dollar of UCD investment $29 in savings and sales will result (a 1:29 cost-benefit ratio that Bevan cites in his study) is meaningless to management trying to decide whether or not to invest in the project, because none of the additional expenses for realizing the benefits are included. It also helps to know the total dollar value of the project rather than ratios or percentages.

The point is that while cost-benefit ratios may help usability professionals feel good about themselves, they don't help management make the decision to fund the project.

The case for Net Present Value.

Calculating a project's Net Present Value (NPV) is a much more precise and authoritative way to express the impact of a usability redesign than a simple cost:benefit ratio. Let me illustrate this by performing the necessary calculations for my client case.

The business, product, and roles. The client's business is to sell templated web sites and lead/sale management systems to their business customers. The salespeople sell the product, the programmers customize it for the customer, and managers oversee each project until it is handed off to the support specialists who train the customers, provide technical support, and respond to customer requests.

The application and the project. All of the client's employees use an internal application to track their hours, log their activities, interact with clients, manage projects, and manage potential and current customers. The project I undertook for the client was to evaluate how people use this internal application and determine how it can be improved.

Project costs. As a part of the evaluation, I conducted observations and contextual interviews with all stakeholder groups excluding the salespeople who received a survey instead. I also performed a heuristic evaluation and a task-based walkthrough. Here are the costs for this evaluation:

Loaded employee hourly costs:[2]

Programmer: ($40,000 + $40,000 * .4 + $40,000 * .4 * .15) / 240 / 8 = $33/hour

Manager: $40; Support: $26; Sales: $46 (extrapolated from typical commissions)

3 hours of contextual interviews (1 hour with each stakeholder):

$33 + $40 + $26 + 3 * $35 (my rate) = $204

3 hours of observations (participants were about ½ as productive during observations):

$16.5 + $20 + $13 + 3 * $35 = $155

3 30-minute questionnaires created by me and completed by the salespeople:[3]

$35 * 2 + $46 * .5 * 3 = $139

Heuristic evaluation + task-based walkthrough:

$35 * 10 hours + $35 * 10 hours = $700

Deliberation + documentation + creating/presenting the proposal:

$35 * 10 hours + $40 * .5 * 2 = $390

TOTAL cost to analyze the application: $1588

Proposed work to implement the suggested changes:

Design and prototyping (me): $35 * 8 = $280

Feedback (me + users): $35 * 4 + .25 * $145 (about ¼ hours from each) = $176

[2] Since this is a completely internal project, everyone except the salespeople is on overhead. So the .35 overhead factor is not used to arrive at the fully loaded cost.

[3] I could have calculated the salespeople's time in terms of lost revenue, but used their commissions-derived loaded cost instead for convenience.

Programming, debugging, rollout: $33 * 40 = $1320

Learning curve (30 minutes with the system for everyone): $573

TOTAL cost of the project: $3,937 ≈ $4,000, including $1300 in UCD costs.

Project Benefits. We've already established that the biggest gains from improved usability will come from additional customers whom the salespeople will be able to sign on to use the software (see page 9). We've already estimated the after-commissions 3-year revenue benefit of $157,500 (cash flows of $22,500, $52,500, $82,500 in years 1, 2, and 3, respectively) which represents a growth rate of 15 new customers per year. We can now calculate a preliminary 3-year NPV for the project (using a 5% discount rate):

$NPV = 22{,}500 * (1/(1.05)) + 52{,}500 * (1/(1.05))^2 + 82{,}500 * (1/(1.05))^3 - 4{,}000 =$

21,429 + 47,619 + 71,267 – 4000 = **$142,267**

The $142,267 looks great, doesn't it! However, there is a catch. Two, actually. In order to realize this NPV, we need to be able to handle 15 new customers per year. Unfortunately, given our current staffing situation, we are at capacity when it comes to programmers and the updated system will not produce enough appreciable time savings to enable the programming staff to handle the addition of even one additional customer. So the catch #1 is that we don't have enough staff to handle the additional customers. Catch #2 is that we need to factor the cost of handling those additional customers into our NPV formula.

Let's assume that the president agreed to hire an additional programmer in year 1 and an additional support person at the end of year 2 to help grow the customer base. We now should have enough staff. Now lets calculate the cost associated with adding 15 customers per year.

Let's assume it takes 20 programmer-hours to customize and deploy a customer web site, 2 support hours to train each new customer, and ½ hour/week to support them. To reduce complexity, let's assume that there are no additional costs associated with adding and supporting a customer.

Annual programming costs here are: $33 * 20 * 15 = $9,900 / year. Now for the support costs. The redesigned application freed up 720 support hours per year. Support requirements are as follows:

Year 1: 2 * 15 + (15 * 48 * .5) – 720 = 0

Year 2: 2 * 15 + (30 * 48 * .5) – 720 = 30

Year 3: 2 * 15 + (45 * 48 * .5) – 720 = 380

The cost of extra support in years 2 and 3 is: $26 * 30 and $26 * 380, or $780 and $9,880, respectively. We can now subtract these outflows from our NPV formula cash flows to include the additional costs:

Final NPV = (22,500 – 9,900) * (1/(1.05)) + (52,500 – 9,900 – 780) * $(1/(1.05))^2$ + (82,500 – 9,900 – 9,880) * $(1/(1.05))^3$ – 4,000 = $12,000 + $37,932 + $54,180 – $4,000 = **$100,112**

Compare the above NPV with the relatively meaningless non-discounted UCD cost-benefit ratio of 1,300 : 157,500 or 1:121!

So if I were going to pitch this project to the company president, I would say:
"The project has an estimated 3-year NPV of $100,000 contingent on the provision of 350 additional programmer-hours per year and 30 and 380 hours of support hours in years 2 and 3, respectively to handle the additional customers. The fully-loaded cost of additional programmer and support hours has already been included in the calculation."

Conclusion.

The main theme of this paper has been that the benefits of UCD should be expressed in terms managers will find useful. I believe that it is only in the larger context of the project as a whole, with all of its relevant costs accounted for, that this is possible. This sentiment is echoed in the way UCD project proposals are put together at eBay: the proposal justifies the expected results *rather*

than the process in terms of how they affect the "financial levels that drive the business" (Herman, 2004, pp. 1414-1415). Simple cost:benefit ratios are meaningless and should be avoided.

References.

Bevan, N. (2005). Cost-Benefit Framework and Case Studies. In R. G. Bias & D. J. Mayhew (Eds.), *Cost Justifying Usability: An Update for the Information Age* (pp. 575-600). San Francisco, CA: Morgan Kaufmann.

Karat, C-M. (2005). A Business Case Approach for Usability Cost Justification for the Web. In R. G. Bias & D. J. Mayhew (Eds.), *Cost Justifying Usability: An Update for the Information Age* (pp. 103-141). San Francisco, CA: Morgan Kaufmann.

Marcus, A. (2005). User Interface Design's Return on Investment: Examples and Statistics. In R. G. Bias & D. J. Mayhew (Eds.), *Cost Justifying Usability: An Update for the Information Age* (pp. 17-39). San Francisco, CA: Morgan Kaufmann.

Rohn, J. (2005). Cost-Justifying Usability in Vendor Companies. In R. G. Bias & D. J. Mayhew (Eds.), *Cost Justifying Usability: An Update for the Information Age* (pp. 185-213). San Francisco, CA: Morgan Kaufmann.

Wilson, C., & Rosenbaum, S. (2005). Categories of Return on Investment and Their Practical Implications. In R. G. Bias & D. J. Mayhew (Eds.), *Cost Justifying Usability: An Update for the Information Age* (pp. 214-263). San Francisco, CA: Morgan Kaufmann.

Herman, J. (2004). A Process for Creating the Business Case for User Experience Projects. *CHI '04*, 1413-1416.

Appendix A. Case study calculations.

Available upon request.