

Good design is effective and profitable

Design Objectives

By Daniel Mordecki

When design is assumed as Interaction Design, the objectives are automatically established and became a powerful tool to align sites and systems with business objectives

10/14/2001

5 objectives of design

Setting that we talk about Interaction Design each time we mention the word 'design' we can determine at least 5 objectives for design:

- ✓ To define the finished product
- ✓ To limit and reduce costs
- ✓ To focus at the user
- ✓ To remove the pressure that design implies for the programming team
- ✓ To make schedules credible and achievable

The 'a priori' knowledge of generic objectives for Web site and application design, beyond the specific objective of designing this application or site, allows us to know if we are taking the good or wrong way. Asking for each objective we can determine easily if we are or we are not designing properly

It is worth the trouble to stand out once again that the design **precedes** to the programming. First we design, then we program and at the end we decorate the Interface. It's usual to confuse decoration with design, it is a common and expensive confusion. The objectives raised here must be obtained therefore before the first line of code is written.

It is worth the trouble to stand out once again that the design **precedes** to the programming. First we design, then we program and at the end we decorate the Interface

To define the finished product

To define the end target before beginning to code the program is something so obvious in the theory as difficult to find in the real world practice.

Other sciences more mature than computer science, like for example architecture and civil engineering, work the time necessary to fulfill this objective before beginning with field tasks. All tools are valid: scale models, blueprints and diagrams, descriptive memories, animations and an enormous baggage of techniques that allow customers, constructors, bricklayers, subcontractors and all the people involved in the project to have a complete visions of the finished product, with no need to put the first brick. It's valid to transfer the example to the IT industry. At the IT industry, as soon as the first outlines of the system are known, the programming begins, with the illusion that programming and designing at the same time saves both time and money. Unfortunately to avoid the previous work of design and documentation it's often considered a hacker trick inside the programmers community.

Nevertheless, to define precisely the final product has two highly positive consequences: it allows to draw up the way and it allows to handle customer expectations.

To draw up the way

Only when you already defined the final product, it's possible to draw up the way, to develop a plan for the project. Without a destiny, without knowing to where we are going, a way cannot be marked. As Lewis Carroll says in Alice's Adventures in Wonderland: 'If you don't know where you are going, any road will take you there'. It seems to be written specially for this occasion.

Code development, necessary based on the previous definition of data structures and models, freezes the project. The assumption that on the way it is possible to modify system characteristics is an optical illusion that doesn't take into account all the decisions and balances that are needed for software building, and that when you write the first line of code, you have taken each one of those decisions. This can be done in explicit form, separating the design of the programming or in implicit form, assuming that 'we program a little and we decide later'.

Design before the programming is the only way that offers project management the ability to define the way to be followed by the project.

It's not possible to have satisfied customers, beyond some luck you can sometimes have, if we are not able to handle the expectations of these customers

Unfortunately to avoid the previous work of design and documentation it's often considered a hacker trick inside the programmers community

To handle customer expectations

All of us want that our customers can feel happy, satisfied with the product that we gave to them, are these internal customers, external customers or both.

It's not possible to have satisfied customers, beyond some luck you can sometimes have, if we are not able to handle the expectations of these customers. The vulgar idea that a customer is satisfied when the product is 'good' is

essentially false. A customer is satisfied when its perception on the product that is given to him is equal or better than the expectations that he had.

Feature accumulation in a list does not allow us to handle any customer expectations. Following Architecture example, feature definition of a house says something, but not enough to generate expectations: '3 bedrooms floor, 2 bathrooms, 110 m2, high quality ceramics, central heating' includes an enormous amount of completely different apartments. For that reason, among other things, architecture offers additional tools so that their customers can compose a finished idea of the product that they will receive at the end. At sites and applications design it happens exactly the same. Whereas an 'excellent search' for one can mean a generous dowry of logic operators, proximity, synonymous, etc. for another one can mean speed. For a third one it can imply ability to distinguish between languages of different documents and for a fourth one not only searching texts, but also images, sounds, videos and other file formats. Without design, that is to say, without a correct handling of customer expectations, being lucky is the only way to be saved of our customers being defrauded when they see the finished product.

This lack of design prevents us to find the way, a reliable schedule, and without this schedule there is no reliable budget

Whereas an 'excellent search' for one can mean a generous dowry of logic operators, proximity, synonymous, etc. for another one can mean speed. For a third one it can imply ability to distinguish between languages of different documents and for a fourth one not only searching texts, but also images, sounds, videos and other file formats. Without design, that is to say, without a correct handling of customer expectations, being lucky is the only way to be saved of our customers being defrauded when they see the finished product.

It's not possible to have satisfied customers, beyond some luck and guesswork, if we are not able to handle their expectations.

To limit and to diminish the costs

Murphy's Law say that a computer project consumes 90% of the resources in first 90% of the work and another 90% of resources in the reminder 10%. A corollary says that it is not worth to budget 180% to avoid the problem. It's tragical, but this approaches dangerously the reality.

The list of deviations in budgets and schedules of IT projects is wide and includes from small projects to very big ones, like for example the delay of 2 years in the fourth version of Windows, that at the end was brought to the market under the 'Windows 95' name.

Difficulties in determining real costs for system development derive mainly from the lack of design. This lack of design prevents us to find the way, a reliable schedule, and without this schedule there is no reliable budget. This worsens much when first system advances begin to appear and (internal or external) customers get disappointed. This imposes changes, that must be added to the delayed system definitions, both against the implicitly taken definitions that were taken when programers begun the coding. The altercates and friction that this kind of processing and successive approaches generates, in some cases became frontal battles inside company management. When costs begin to go out of control, the IT department begins to blame the rest

of the company by the lack of definitions, and the rest of the company requests the head of systems by the delays. The damage is already done.

This process has a complement in the fact that the description of the systems as a features list is the key tool to evaluate programmer hours by the IT department to make the budget, assigning the programming hours that will take each feature to be finished and the price of these hours. This way, IT determines unilaterally which features are more important and which aren't. When costs go out of control, time of cuts begins and the more expensive features are discarded without mercy. This perverse process, far from helping, only adds firewood to the fire, harming the quality of the end outcome and intensifying the discussion of the different areas that promoted at their moment the inclusion of features which now are being eliminated.

The medicine for this disease is preventive, not corrective. Interaction Design allows to properly define each one of the system interaction areas, both with customers and end users. This gives them a comprehensive vision of the program they will obtain at the end. Only on this bases they will be able to prioritize the functional areas of the system. Also Interaction Design allows to develop the way from the present situation towards the known destiny, to determine the work that must be requested to each area involved with the project, not only IT. On this basis it is possible to develop an optimal budget, and in addition realistic.

To focus at the user

Excepting the speech, the user is not represented in the site and system development. It's a truth

Grandiloquent expressions as 'human Interface' or 'natural interaction' can be added to descriptions and feature lists of different products. Nevertheless, the common users frequently hate the computers and feel that the computers hate them

The medicine for this disease, that one that make costs grow without control, is preventive, not corrective

that we must recognize before being able to correct it. Pamphlets can shout to declare that the user Interface is intuitive and the system easy to use. Grandiloquent expressions as 'human Interface' or 'natural interaction' can be added to descriptions and feature lists of different products. Nevertheless, the common users frequently hate the computers and feel that the computers hate them.

Dissociation between intentions and reality also has one of its deep causes in the lack of design at the site and applications creation process. The intention to focus at the user is sincere, but when terms pressure, costs pressure and competition pressure begins, developments end up following paths that move them away of good intentions and approach them to reality: they lack definitions, the time is finished and development begin to behave as if everything were allowed. A forceful evidence of this reality is documentation: when it is available, most of the times it's terrible. Hardly it overwhelms the contract line that said: 'End user

documentation'. You can argue that users hate the computer, but you must agree that they hate documentation.

Building with the greatest precision the finished product definition, and with this definition determining the path to follow, comes up with deviations that pressures impose and approaches the systems to the user. It is always possible to twist the way towards an erroneous map course, taking one or two mistaken decisions during the development process is enough. But whereas the design does not guarantee that the user is going to feel satisfied with the site or system, that he is going to reach his objectives, the inverse one is guaranteed: the absence of Interaction Design generates sites and systems built in function of programmers requirements and preferences, time and cost pressures and disputes between different departments.

To remove the pressure that design implies for the programming team

Who worked as programmer knows how despairing is to develop a poorly specified system. Whereas when you are a rookie you may think that that it is an opportunity to shine, life teaches us that poor specification gives others unlimited power to criticize our work with impunity, to include functionality that was not specified at the beginning, and to force us make enormous changes that otherwise had been unnecessary.

Emails, meeting notes and all the stuff that we have won't be enough: such-and-such function is *obvious* that had to be included and although at the right time nobody heard our questions, nobody got interested in meetings, nobody filled out the forms, we must program. Programmers assume thus, by default of the rest of team, the heavy task of designing: a task that they are not skilled to do and that is committed to them in the worse conditions.

Design has different theory, objectives, foundations, techniques and methodology than programming. More than that: in many cases ways are opposed. For example, whereas the programming implies the permanent and exhaustive analysis of edge situations, Interaction Design practically ignores them.

Forcing programmers to design is in the first place unfair and secondly extremely counter-productive: you get a very poor design and programmers are taken away of the work they really know and must do.

To make schedules credible and achievable

Although this item would have been included within the costs item by its kinship, deserves a separate section by its consequences.

Causes are practically the same ones, but consequences are different: whereas the problems of costs hit in projects profitability, credibility is an intangible that pierces much more deep and

credibility problems have deeper consequences for the companies. Cost deviations, at the end, can be fixed with more money, but there is no money that can fix lack of credibility.

Process that we already described, where programmers begin to code applications without knowing clearly the finished product, implicitly making design decisions that freeze the capacity to incorporate certain functionality, generating a process of permanent revision, a test and error approach, generates uncertainty margins much more important than those that would be due to assume. Changes and delayed definitions aggravate this process, that often becomes literally infinite. Thus it is frequent to find decisions that in an arbitrary date, the product must be brought to the air 'as it is', giving an eloquent sample that long ago the full project went out of control.

Although you can judge it is an IT department problem, making the assumption that only IT has a credibility problem is being myopic. Nowadays, without systems there is no business. Without new systems there are no new businesses. The lack of credibility in system development and deployment becomes lack of credibility in businesses development, lack of credibility in the company as a whole.

Interaction Design and System Analysis are both essential tasks but disjoined. The design precedes to the analysis.

Design and System Analysis

It is worth to finalize commenting out about the relation between design and system analysis.

We as programmers tend to confuse both, since a good analysis as a good design leads to a good system. Nevertheless they are not equivalent. System analysis, the techniques and methodologies of computer science start exactly at the point where design ended its work. The system analysis takes a problem specification, defines for it a development strategy, an associate data mode and allows the accomplishment of a suitable codification. The Interaction Design starts from scratch to specify the problem.

Interaction Design and System Analysis are both essential tasks but disjoined. The design precedes to the analysis.

As a conclusion

Changes and delayed definitions aggravate time deviation, that often becomes literally infinite

The uncertainty and lack of credibility that site and system development imposes to companies, the impotence when seeing projects flooding, that form change tool become to ballast that maintains the status quo, that from competitive weapon become disadvantage, deserve a deep change in the conception and facing of these processes.

Correcting Interaction Design deficiencies appear to be one of the ways to revert this tendency. Maybe it's not the only one. Perhaps

not even the best one. But it is within reach, with clear objectives, technically defined and attainable implementations. It does not suppose adding expenses but all the opposite: we have no doubt that Interaction Design is for the company an efficient and profitable option: a great business.