LOOKS DO MATTER

By:
Yogita Sahoo
Mindfire Solutions, India
www.mindfiresolutions.com

My paper on Beauty Testing, "Testing the Bold and the Beautiful", originally published at Stickyminds fetched me some interesting comments and intelligent questions about the importance of aesthetics in software. Most readers believed strongly in the concept and termed it as a revolutionary thought in testing. Few confused it with usability testing. At the same time all of them agreed upon its implementation difficulties.

More than the testing aspect, my primary goal here is to create awareness about Beautiful Software among test professionals. This paper also intends to answer the thoughtful questions proposed by the readers as well as help those starting out on a road to test Beauty.

Table of Contents

Introduction

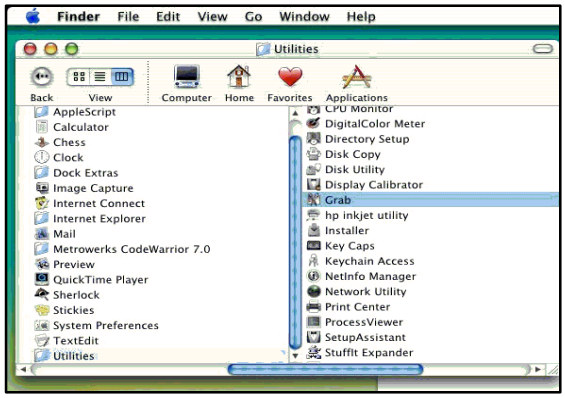
Software organizations in today's world face increasing challenge in order to adequately address the varying need of their demanding customers. The fact that there are enough market players already existing with the same product or service that you offer, and the number of competitors increasing day by day, one can sustain only if your offerings are outstanding in all aspects. Increasingly sophisticated users are very blunt in making the perfect choice when it comes to buying any software product or services and quite intolerant of accepting even a minor glitch. To put it more upfront, customers in today's world want to be pampered with the best of features and qualities in any software. A user today is very much aware and choosy about what exactly he wants to see in the software and not ready to compromise on any inadequacies. And hence the sense lies in delivering software with ever-higher levels of functionality, reliability, responsiveness, usability, performance and of course Beauty.

To weather such a competitive environment, it is critical that every user-software interaction must generate positive customer experience. User experience is of paramount importance; to be successful, software need to eliminate user discomfort. It's unfortunate that User Interface design is sill considered a secondary activity in the development life cycle of many software applications and beautifying the same is believed to be unnecessary till date. When software professionals are waking up to the ideas of Human Factors and Human Computer Interaction, beauty is deeply neglected while addressing issues related to positive user experience. With the inception of attractive web sites, operating systems and skinned custom applications, indications are clear that users are becoming conscious about aesthetics in software. Customers are expectant of beautiful stuff along with fine-tuned functionality. I guess, that is reason enough for us to think about software beauty now.

Beauty – The growing trend in software

Beauty is becoming an essential part in our daily lives. It's obvious from number of beauty pageants held around the world, offices and homes designed by interior designer, designer apparels and accessories in fashion houses, beautifully garnished food presentations in restaurants, trendy looks of electronic gadgets etc. etc. You will find the same trend in software business too. Let's look at some recent products from software giants and their trend-setting efforts in software beauty.

Mac OSX: Macintosh has always surprised its users with utterly charming graphical user interface. The new theme for OS X is "Aqua", which combines the pleasure of compelling user interface with powerful core foundation. With brilliant new features and an aesthetically refined use of color, transparency, and animation, Aqua delivers a well-organized and cohesive user



experience available to all applications developed for Mac OS X.

Windows XP: Much attention has been devoted to the new user interface in Windows XP. A dramatic update of the classic Windows looks, with a sleek and colorful new visual design makes user interaction a pleasure. The desktop is presented to the user in a clean state (unlike the default My Document, My Computer, Recycle bin icons in prior Windows) for the user to design his desktop the way he wants. Windows XP's new visual styles and themes use sharp 24-bit color icons and unique colors which creates an appealing visual effect

Skins: Skin applications have the flexibility to adopt different appearance as desired by the user. User may use the same application, but with a different look every time. A good example of such software would be the Audio Player by Win Amp. Leaving the commands in user hands helps to satisfy the aesthetics preferences of a broader user base which is not possible with applications with a stagnant UI.

Websites: With the emergence of web as a strong marketing media, companies are quite experimental at making their websites attractive, flashy and more life-like. Usage of flash and audio/video components on the net makes them more attractive, interactive and user friendly. Websites have come a long way from the initial static HTML pages to dynamic, robust and beautiful web pages.

Beauty Testing – A domain often misunderstood

Software beauty is perceived wrongly among s/w professionals. The fact that the general definition of beauty differs from person to person increases the number of definitions of beautiful software. To some, software is beautiful when it's bug free, to others beautiful software is the one rich with features. Few may define it as painlessly usable. The rest may say a neat and colorful UI makes software beautiful. How do you define beauty? May be a mixture of the above or an entirely different statement. Hence there arises the need of a common agreement on the definition of beautiful software.

What is Beauty Testing?

To test the attractiveness, pleasantness, splendor, elegance, brilliance and gracefulness of every single user interface element in the software; so that they collectively create a magnificent look.

Beauty vs. Usability:
Beauty testing is entirely UI oriented. Oftentimes, people mistake it for usability testing. Hence it's all the more difficult to convince them to execute an independent testing endeavor to check aesthetics. Usability is all about the ease, speed, and pleasantness with which the user can use the application. And during usability testing, the ease and speed factor pull so much of importance that pleasantness often takes a back seat. If the "pleasantness factor" is well taken care of, we will certainly succeed in creating beautiful software with high-end functionality. Beauty actually serves towards acquiring better usability.

Aesthetic are often considered as ornamental additions to the software without having much contribution to the application serviceability. But in practice, an application with beautiful UI is not only an immediate crowd-puller, but also retains its users to be explorative on them. This approach is top-down, enumerating and glamorizing the look of an application in order to create a magnificent impression on the user at the very first time. Those impressions would invariably turn into prospective business, when the user further experiments the overall functionalities of the application.

Beauty Testing – A domain often ignored

Poor practice: Beauty testing is often ignored because of poor testing practices. Software people believe in a pre-convinced notion that any appearance defects will be caught while performing functional testing on the application. The simple reality of it is this – No tester really bothers so much about UI bugs when she commences with functional testing only to discover that there is a long list of more important bugs to be fixed. Hence UI bugs move to the bottom of the list to be attacked later. They actually move so down in the list that when the team allocates time to attack them, the delivery deadline is round the corner. Software projects working on such scenarios, finally decide to improve UI as a part of the software maturation process, which would mean a wait till next version release.

For most developers, a UI defect is never a worthy bug. Appearance defects are mostly regarded as enhancements or suggestions instead of bugs due to their technical simplicity. Hence their severity and priority lessens in the bug list. As long as a defect doesn't induce a serious crash in the application or affect the general functioning, it can be ignored. Developers suggest that testers keep these types of issues in the bug list as lower priority ones. "If time permits," they say, "we may think of looking into those. After all, the application runs fine without them." Such developers often discourage testers to report UI bugs.

Beauty is ignored: Beauty testing is ignored because Beauty itself is ignored in the software lifecycle. First of all, user interface is thought of very late in the process, thereby eliminating the scope of beautifying the same. Secondly, developers mostly design and implement UI, which is biased towards an easier coding process. They also follow the UI norms and industry standards very strictly, which restricts creativity. If you consider examining development lifecycle of any serious desktop software, UI consideration probably falls in the later stages of development. Developers first do core programming

and get something visible on the screen. After a successful attempt they write more codes to implement functionalities and create screen elements to test the same. For example, if code were written for 'save' action, the developer would create the conventional gray, rectangular 'save' button to test that on clicking it, the common 'save' dialog box opens. Likewise when all functionalities are acquired, the developer would then arrange and alter the size and layout of the visual controls in order to induce a proportionate look. There is no involvement of a creative person anywhere in the scene. Which is most likely not the case in a website development, where a graphic designer, who probably has no knowledge about programming, designs the display and layouts of all screens and other visual elements. Programming then follows the design laid by the designer. This probably answers why websites are much more attractive than their desktop counterparts.

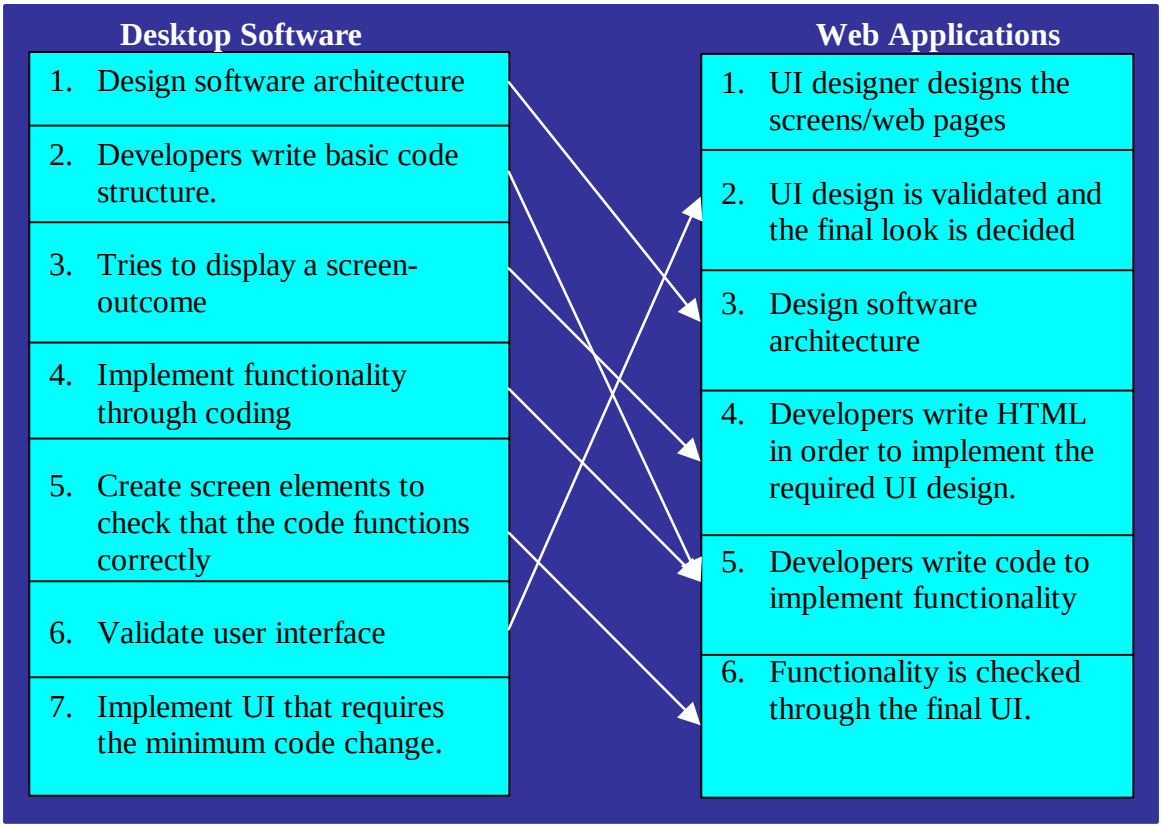


Fig 1.1 UI implementation in SDLC of desktop software and web application.

Importance of Aesthetics in Software

GUI is necessary and beautiful UI is important for effective user interaction. Most testers who have ventured into UI testing are well aware that an extremely adequate UI can have negative impacts on the users' experience because of poor display styles.

Applications that have a GUI, can display a much greater range of output styles than a character-based application. Most of these screen-based outputs have become norms for the majority of users as applications have standardized on the use of items such as

windows, icons, menus, dialog boxes and other UI controls. Hence usage of non-standard visual elements may restrict the user from capturing the actual information at a logical level. This leaves us with little opportunity to experiment on the medium that represents information exchange between the user and the software. But beautifying the way these mediums appear to the user can be a successful approach to create non-conventional display attributes, while keeping the UI standards intact.

Imply Professionalism: Beautiful applications are assumed to be into serious business. If we compare two web shopping malls with offerings at par with each other, the one that's beautiful would attract more users doing transactions over the site, because it creates an impression worthy to be reliable - An application that puts so much of effort into beauty issues couldn't have neglected important business features and functionality. An ugly application, in the other hand, creates an air of unprofessional attitude.

Positive business impact: Software not only must push its contents and pull in visitors – it must successfully convert them into revenues and retain its clients. The below figure represents how an application with beautiful looks has an edge over its competitors in retaining the users since their first attempt, also capturing dissatisfied users from equivalent products. User 1, after experiencing a product with poor UI, moves on to explore similar application with a better interface. An acceptable UI also doesn't satisfy him till he discovers a similar application with beautiful UI and he sticks on to it. User 3 comes across the application with excellent UI at the beginning, barring him the need to look out for more products in the market.

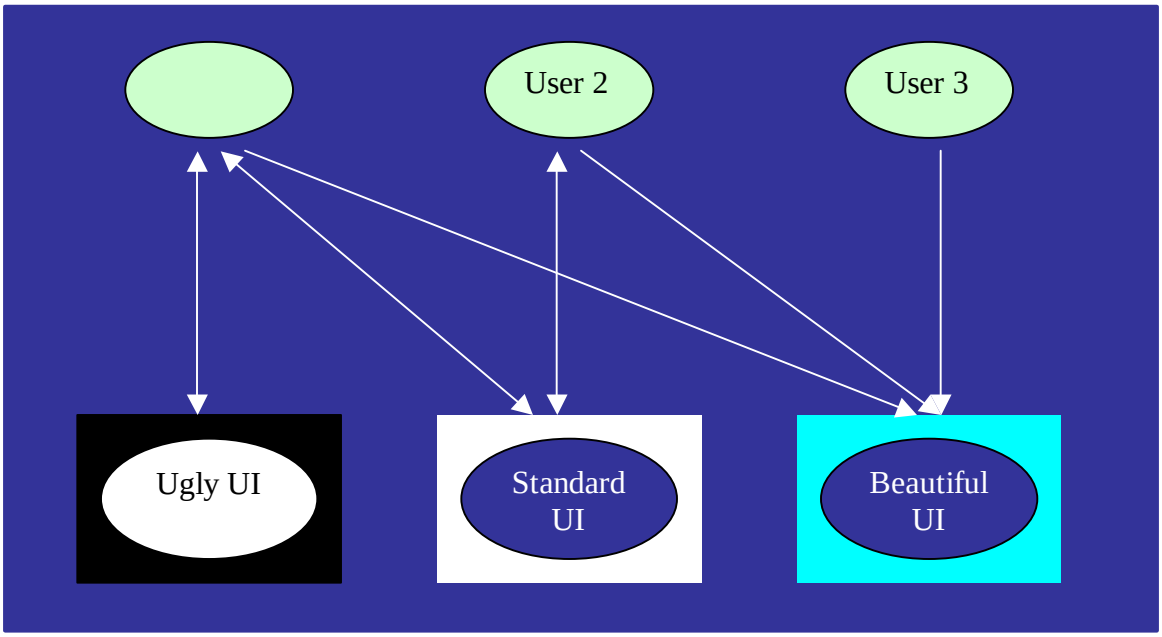


Fig 1.2 Beautiful UI winning over ugly and standard UI

Can real beauty be tested and How?

Beauty is a vague term. One man's beauty is another's ugliness. Moreover beauty can't be measured in quantitative terms. So how do we test the correctness and implement something that has no specific requirements or guidelines to be checked against?

In my experience, attempts to test beauty will not only be impossible, but also incorrect. What alternatively could be done is to test and decide upon a look that appeals to the average user mass and be the best option available in it's category.

Who should test?
Software testers should facilitate the beauty testing process, but they should not be the actual ones to test it. The simple reason being, technical professionals are mostly more bothered about functionality issues than beauty and creativity. This makes them non-eligible to carry out such tasks. For similar reasons, I find Usability Engineers to be equally inefficient to check beauty. Users in the other hand may suggest significant improvements. But you can't anticipate every user's expectations, and even if you could, you probably can't satisfy everyone's need without losing the program's integrity. I suggest that aesthetic issues should be dealt with by a representative group (mostly testers, developers and users), formulated by knowledgeable experts in consultation with graphic/UI designers, experimenting on a separate copy of the application, and then finalized. Such representative groups are more productive because the group forms a consensus about aesthetic issues, which is better than a one-on-one debate between a tester and a developer. The representative user group should have a mixture of ignorant users as well as knowledgeable users having prior experience in similar domains.

When should Beauty Testing begin?
It is recommended that aesthetic importance should be assessed and realized early in the software development life cycle. Start thinking about beauty on the first day you start designing the User Interface. It is not advisable to keep this process ongoing till the final stages of development. Get the representative group experiment and finalize on the final looks. Implement the same before the UI freeze happens and then there should be no mentions of altering it further.

Many organizations believe in freezing UI prototypes early. In such cases, either the beautification process has to be finalized and implemented early or only such features which are unlikely to have any effect on the application's working can be induced in the later stages.

UI prototype survey
Though the representative group masters the whole beautification process, it is always beneficial to include a feedback session on their findings, before the final look is decided. After the visual effects (decided by the group) are implemented to the UI prototype, conduct a survey in order to find its effectiveness. Surveys or feedback cycles not only help to gather valuable inputs, they also initiate positive criticism. The survey results should be analyzed and valuable suggestions implemented.

UI Prototype Survey for Beauty Testing

1. Would you assess the software as Beautiful? Where do you rate the aesthetic appeal of the application? (**>8:** Can do with the present look, **>5:** Need to improve. Suggestions from the survey have to be implemented, **<5:** Not acceptable. Have to rethink the UI.)
   ☐ >8 ☐ >5 ☐<5

2. Do you think at any point the UI standards are drastically ignored in order to implement beauty? Does that hamper usability of the application?

3. How do you rate the following attributes of different UI elements?
   Colors: ☐Overused ☐Properly used ☐Underused ☐Others _____
   Layout: ☐Uncluttered ☐Systematic ☐Proper Space Utilization ☐Others _____
   Fonts: ☐Significant size ☐Appropriate font face/styles ☐Highlights important info ☐Used differently for different types of info ☐Others _____
   Shapes: ☐Symmetric look of similar elements ☐ Sized for effective space utilization ☐Look changes after action on objects ☐Smooth look ☐Adequate usage of graphics/3D objects ☐ Others _____

4. Do you think the applied/suggested beauty enhancements go well with the product line, business objective; and end-users' technical sophistication and requirements?

5. Do you think by giving importance to beauty in the software, the product's sale, serviceability and popularity will be positively affected?

6. List the Visual elements that you think can look more beautiful.

Element	Area	What could improve	Impact

7. List Visual elements that are ugly and need to change at the earliest.

Element	Area	Alternative	Impact

**** *The supplied UI is a prototype only.*
**** *This survey intends to assess only the aesthetic drawbacks of the UI. UI testing is however conducted independent of this survey.*

Fig 1.3: UI prototype survey for Beauty Testing

Institutionalization of Beauty Testing

Beauty Testing, to be sincerely followed, should be introduced the QA process of the organization. Institutionalization of the concept is where its success begins. Implementing beauty testing into your current process will not be difficult. Since it only involves UI elements, it can be conducted parallel to the UI design activities and end with the UI freeze. Let's see how the beauty testing process blends into your existing process.

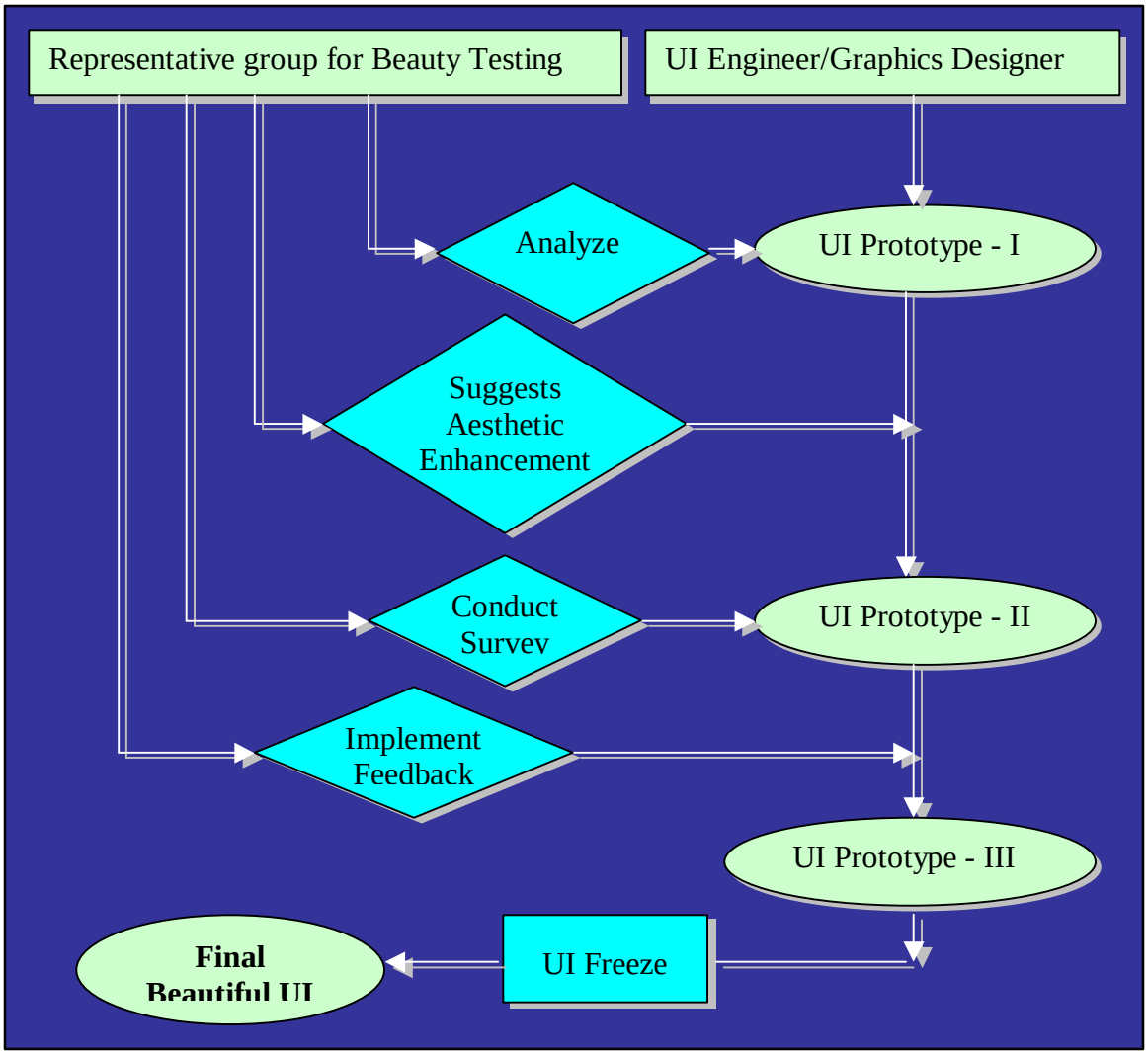


Fig 1.4: Introducing Beauty Testing to an existing process

The representative group will work together with the UI engineering team in the UI design phase. The UI engineering team designs the very first UI prototype. Then the representative group analyzes the prototype and suggests beneficial aesthetic enhancements. The UI engineering group now induces those suggestions and produces the next prototype version. The representative group conducts a survey on this version of UI, analyzes survey feedback and identifies potential suggestions. Finally these features are implemented and the next prototype version is designed. If the UI design phase also includes some other engineering activities, they all are executed as they were supposed to

in the old process. After all the process activities for the UI design phase are completed, UI is frozen.

Elements of Software Beauty and how to test them

In order to successfully execute beauty testing, one must first understand what elements does the testware consist of and what attributes of such elements should be tested. Testware in simple terms is the application/feature under test. And depending upon the application and type of testing effort, elements of testware change. But the attributes of these elements remain more or less the same.

Let's examine what could possibly fall into testware category during beauty testing. As expressed earlier, beauty testing is completely UI oriented. Depending upon the application's interface, the UI elements would also change. Here you will find the basic attributes of these elements that can be found in any software application, which could be considered for testing.

Elements could include frames, windows, screen controls (command buttons, list box, radio buttons, etc.), interactive dialogs (error message, alerts) and all other visible elements on the screen. Attributes to be tested will define the way these elements are laid out on the screen, their physical appearance, color schemes and text representation. Let's look into each one of them closely.

Layout:
Layout talks about the arrangement of different elements on the screen. A good layout of screen not only creates an aesthetic appeal but also optimizes efficiency of visual access. Here are some keys to create beautiful and efficient screen layouts.

- Do not put too many elements on one screen. Avoid congested layouts. E.g. a web form that has user registration page full of labels and text boxes should be considered to be broken into two less cluttered forms. Crowding up all information in one page may force the user to overlook some contents.
- Group elements according to their functionality. E.g. while collecting credit card information of user, group the related text boxes in one box. While displaying long list of information, group similar texts into one column and caption the same. Grouping makes the screen look cleaner and well perceived.
- Induce correct spacing. Uniform spacing among elements makes the page look neat and clean.
- Avoid too many empty spaces. Blank screens make the display look undisciplined and scattered.
- Align placements to create a uniform look. Too many elements to one side make a page look disproportionate.
- Follow eye movement. People generally move from top to bottom and left to right. Place important items at the top or left in order to have the user look at them early.

- Arrangements should follow a logical flow. When a website is loaded, user expects to see the home page first. You shouldn't straight away take him to the service page, though business is your main motive.
- While placing child windows, message boxes or dialog boxes, make sure that important parts of the parent windows are not obstructed.
- Follow standards and guidelines. You shouldn't put command buttons at the beginning of the form, and labels after the text boxes.

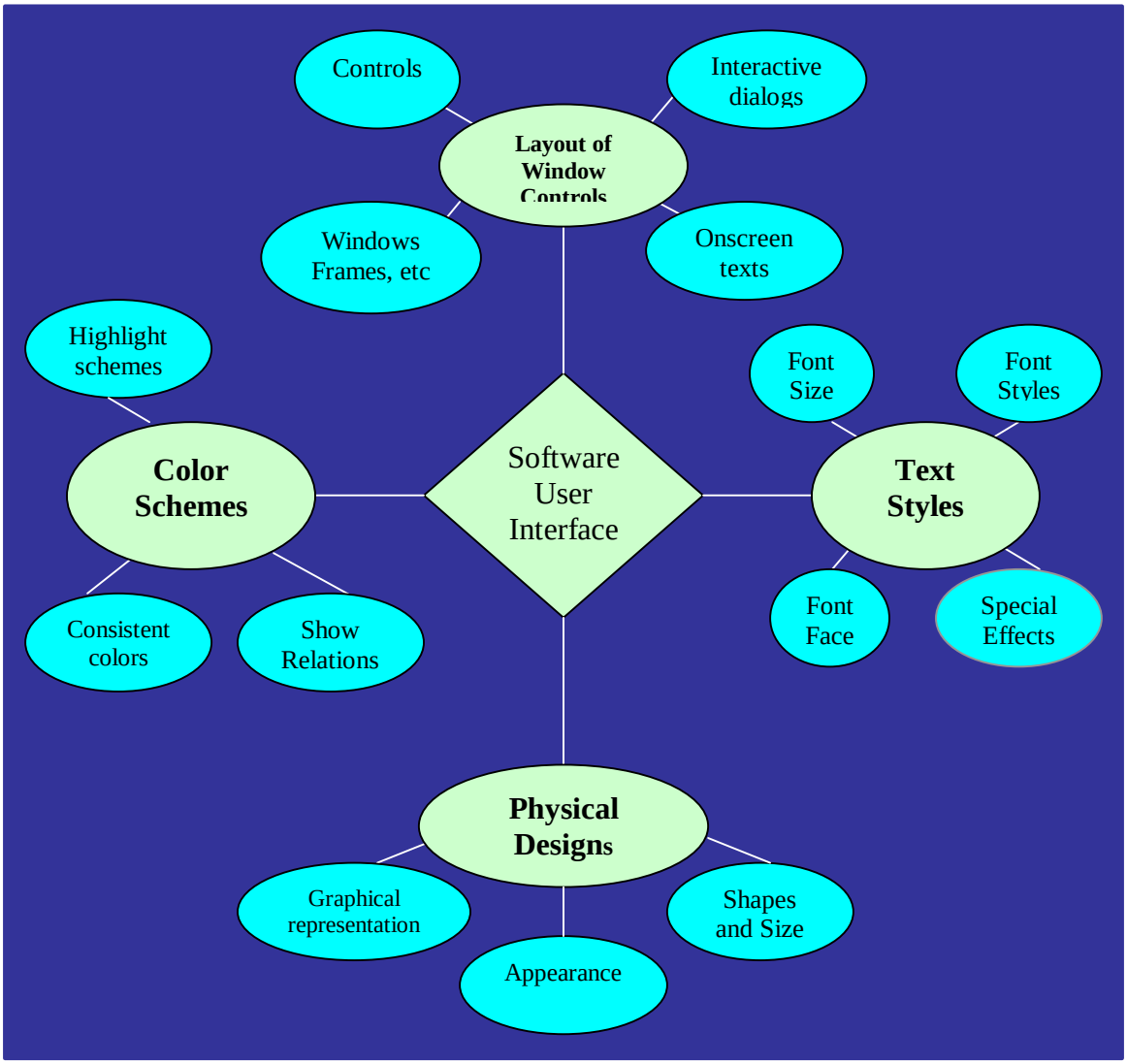


Fig1.4: Elements and attributes of software beauty

Text Styles:
Onscreen information should be beautifully presented. Size, styles and font faces can be liberally used for a marvelous look. But overdoing the approach may make the usability suffer. Here are some tips to make text usage beautiful and usable.

- Text size plays critical importance. Significant size of text makes information clear and saves space. Use different size for headings, sub-headings and captions.
- Use beautiful font faces for captions, highlighted information and short labels. But large piece of text should be presented in easy-to-read font size and styles.
- Color important information. Bolded and Italicized text stands out in the paragraph.
- Special effects like blinking text, moving texts, texts that change color create appealing looks.
- Do not use all capital letters at the same place. It makes the look cluttered and reading becomes difficult.

Physical Appearance:

Each UI element needs to be designed well in order to make the whole display beautiful and portray its importance to the user. Here are some keys to create beautiful display elements.

- Symmetric look is important. All parent windows should have similar looks and so do the child windows. Similar buttons should have same design patterns.
- Size elements to use space efficiently. At the same time, size should also indicate importance. A login button could be bigger than the other buttons on the website.
- 3D designs look trendy and more life-like.
- Make use of icons and graphics wherever needed.
- Change looks for controls that have undergone some action like a click or scroll-over. It induces variety in looks and creates dynamisms.
- Use rounded buttons instead of square ones. They create a smooth look.
- Usage of graphical buttons and tool bars make the screen look interesting.

Color Schemes:

Users like to see variety of color displays on the screen. Colors, if thoughtfully used, can not only beautify appearance but also increase usability. Here are some tips to usage of colors.

- Care should be taken to maintain color consistency throughout the application. At the same time, avoid monotony.
- Colors should be used only if they are needed. Lot of different colors may make the screen look so cramped that users face difficulty in finding information.
- Use colors to implicitly deliver a message. Some colors already have meaning you can use e.g. Red means error or danger. Incorrect usage of such colors may make the information perceived wrongly.
- Use bright colors to highlight important areas to which you want to draw immediate user attention.
- Same colors should be used for similar items. Users mostly tend to group items of same color. Use colors to create relationship.
- Create contrast efficiency while merging colors. A dark background with dark colored text makes the contrast poor and difficulties in reading. At the same time using high contrast colors like pure reds or blues creates discomfort in eyes.

Test everything that affects display:

We talked about few elements of beauty and their attributes that can be tested. But that's not all that defines beauty testing completely. You may not be aware of some strongest impact on beauty caused by small changes in the value of some parameters, which directly affects display on your computer. Make sure that your software looks beautiful even when the user alters these values. Here are some tips.

- *Resolution:* Resolution has the most critical impact on displays and can ruin the looks of software to a large extent. Software that looks clean with a1024x786 pixel resolution may look clustered, compact and ugly with 800x600 setting, as the screen area reduces.
- *Color:* The color settings of monitor can make the display components loose color and appear poor. If beauty is optimized at high color resolutions (32 bit True colors), then at lower color settings, screen looks dull.
- *Text Size:* While designing a web page, care has to be taken that if the user changes the text size (increase or decrease), the alignments and placement of text don't change drastically. Similar is the case for desktop software, in case user alters windows font size.
- *Window Size:* If window size is altered from the default size, some section may disappear until the default size is restored. Such windows should either adjust the contents so that they are visible even in a smaller area or provide scrolling as and when required.

Beauty Testing – Facts and Myths, Dos and Don'ts

Fact 1: UI testing won't improve beauty

Testers are often seen to believe that UI testing also covers testing for application's aesthetic drawbacks. Beauty testing is not about finding bugs but suggesting aesthetic improvements. Hence simply carrying out UI testing will not make your application. Beautiful. If you are serious about Beauty, consider Beauty Testing as a separate and independent activity.

Myth 1: Beauty won't improve program's serviceability

A misconception widely believed in our trade is: Aesthetic improvements will do no good to the application's serviceability towards its users. And this probably answers why we ignore beauty till date. The immediate effect of a good-looking application is to attract new users. Beautiful applications are a pleasure to use and can hold back users to use them repeatedly.

Myth 2: Beauty Testing can't be formalized

Beauty testing is rarely conducted because testing team lacks the resolve to conduct it effectively. Testers feel reluctant to carry out any such activity that's not a part of the process and lacks clear methods of execution. Beauty Testing could me methodized by introducing it as a part of the QA process and then clearly under-laying each activity that

should be performed as a part of it. The section “Institutionalization of Beauty Testing” focuses on implementation of beauty testing into an existing QA process.

Myth 3: Beauty implementation and testing can wait till the later stages of development

Beauty testing has to be carried out early in the project and possibly before starting with functionality testing. Do not wait till late when a lot of work has been sunk into the UI that the user won’t like. Then you are forced to alter features, which may introduced bugs into the system or induce inconsistencies at a later stage.

Dos 1: Beauty and Usability should go hand in hand

While implementing beauty, take care that the application usability is not hampered. Innovative User Interface artifacts will no doubt make the UI more appealing, but it could also affect the usability of the application. The testing team should stay conscious and repeatedly validate usability while beauty testing is conducted and enhancements are implemented.

Dos 2: Involve users

One excellent way to verify your guesses about beautiful UI, of course, is to test your design with potential users. Before the design phase begins, we must understand our users' concerns and learn to empathize with them; their feedback guides and inspires us while we explore different design possibilities; and late in a project, they help us refine and build the chosen design. Studies on Human Computer Interactions teach the value of an iterative design process that directly involves the end users. Involving users costs us far less, in terms of time and usability testing, to make good guesses and design choices right at the beginning. End users will not only tell you what’s beautiful and what’s not, they will successfully pinpoint potential usability blunders that you may have introduced while implementing beauty.

Dos 3: Be flexible while using UI standards/guidelines

UI guidelines, though provide a standardized usage of display elements, can restrict creativity to a large extend. Style guides not only ensure correctness, but also help maintain consistency across applications and inject inherent usability. But consistency in itself doesn’t ensure usability and beauty. It is a mistake to think that consistency in the surface properties of the interface will lead to good design and a beautiful application. You need to be able to design the user interface for your specific users, goals, and tasks. Guidelines may be a reasonable starting point, but they are only a starting point. The value in UI consistency lies in effective learning, by making it easy to transfer knowledge from another product. However, sometimes ease of learning can get in the way of ease of use. You must test your product with users to make sure that your initial design decisions are the best for your users and the specific work for which they are using your product. Blindly implementing UI standards will make the software look general and monotonous. Learn to strike a balance and make combined usage of standards as well as creativities.

Dos 4: Choose among alternatives
While deciding upon aesthetic issues, make alternative choices and then decide upon the best choice available. Like an interior designer presents several designs to her clients before deciding upon the one to be implemented, different designs of software appearance should be drawn before deciding upon the most eligible look. Availability of alternatives not only eliminates confusion, but also makes the decision process simple, easy and quick.

Dos 5: Look at competitors
Study existing software that offer similar services and could be treated as competitors. Examine their visuals and then decide what could make users prefer your application to the ones in market. The best advice is to avoid monotony and think of non-identical designs. Highlight areas in order to promptly indicate creativity and uniqueness.

Don'ts 1: Do not apply beauty without reasons
Apply beauty that makes sense. Beauty is not just crayon work but also a creation of mind. While implementing beauty to your application, see how compatible, necessary and helpful it is to the business focus and functionality proposed by the software. Implementing unnecessary beauty not only costs time, money and skills, but also indicates unclear service objectives.

Conclusion

Let's face it; building good UI is hard work and beautiful UI is harder. It's possibly tougher than writing the code itself. Creating beautiful UI is not logic-driven. There is no rules-on-stones to be followed. It's not even sheer magic or co-incidental that you end up creating beautiful looking software. But who can deny the importance of a good UI and afford to ignore the fact that looks can be a valid constraint in the success and failure of any application. The UI of your application is the actual representation of your code. Regardless of how well your application functions, a ugly looking UI will turn software reviewers rabid and send users running away screaming.

With all said and done about importance of aesthetics, beautiful software is ultimately worth only what its users can get out of it. Good looks, alone, cannot compensate for lack of functionality and usability in software. Neither good looks paired with fine tuned functionality can make the software click. Beauty has to be in co-ordination with the service objective, applied to suit the end user's sophistications and aesthetic preferences and nevertheless be thought-driven. If the success of beautiful artifacts in software engineering is any indication, both our industry and our customers will benefit greatly from this effort.

It's not necessary that each program be a visual work of art.
But it's important that it not be ugly.