GUILayout++: Supporting Prototype Creation and Quality Evaluation for Abstract User Interface Generation

Francisco Montero and Víctor Manuel López Jaquero

Laboratory of User Interaction & Software Engineering (LoUISE)
Instituto de Investigación en Informática (I3A)
University of Castilla-La Mancha, 02071 – Albacete (SPAIN)
{ fmontero | victor }@dsi.uclm.es

ABSTRACT

User interface development is not an easy task. One of the key elements is tackling user interface design by using user-centered development (UCD) techniques, more specifically user interface prototyping and iterative design based on evaluation. In this paper, some facilities are introduced to support the aforementioned activities. These facilities have been included in a tool called GUILayout. The new tool created has been called GUILayout++. By using the extra functionality it is possible creating high or low-fi user interface prototypes, analyze the structure and the look of the user interface according to the different areas that it comprises, and compute the Layout uniformity metric proposed by Constantine & Lockwood. Furthermore, starting with the prototypes created it is possible to automatically generate an abstract user interface compliant with UsiXML (http://www.usixml.org [23]) user interface description language.

Keywords

User interfaces, prototyping, UI evaluation, UI design.

INTRODUCTION

The user interface is a set of commands or menus the user uses to communicate with a program [15]. The main idea behind user interface concept is the mediation between the human and the machine. In a broad sense, the user interface is the link between the user and the software, that it, is the mean for the user to communicate with the machine.

This paper elaborate on the idea that in order to develop any software application with quality, the user must be considered from the very beginning of development, to avoid errors or flaws in the product created. This design philosophy is known as user centered design [18], and remarks the fact that interaction must be seen from the user's point of view [6].

A paramount ingredient to carry out this design philosophy has been using prototypes, that is, partial or total representations of the final product (ranging from simple mockups to more elaborated designs created by using software tools) that support, along the full process, the implication of the end-user in the design activities.

Using prototypes will support the designers in communicating with the end-users in a more effective way. Thus, from the very beginning prototypes will be created (more elaborated each time) that will be presented to the users. Users will interact with these prototypes as they would with a final product.

In this paper, the need to jointly consider prototyping and evaluation in the development of user interfaces is exposed. To do so, analysis and evaluation criteria for different kinds of prototypes are identified. To support considering jointly prototyping and evaluation activities in user interface design a tool has been created, namely GUILayout++.

This paper is organized as follows. First, different tools and methodologies supporting prototyping are discussed. Next, those techniques and evaluation and analysis metrics that have been proposed are gathered. GUILayout++ tool is described next. Lastly, some conclusions and future work are provided.

PROTOTYPING TOOLS AND METHODOLOGIES

There are several taxanomies to describe the different kinds of prototyping [19]. Regarding their fidelity degree prototypes can be either high-fidelity or low-fidelity. The look of low-fidelity prototypes (see Fig. 1) is far from being closed to the look the final application will exhibit. Nevertheless, high-fidelity prototypes (see Fig. 2) look more alike the final product.

There are some other terms related to prototyping methods. For instance, there is quick prototyping and modular prototyping methods. Quick prototyping consist on developing quickly new designs, evaluate them and discard the current prototype to create a new prototype for the next design iteration. Modular prototyping seeks reusing prototypes by modifying an existing prototype as the design cycle goes on. For this reason, this prototyping method is also known as incremental prototyping.

Figure 1. Example of low-fidelity prototype.

Figure 2. Example of high-fidelity prototype.

The importance of prototyping activities for user interface design is evidenced by its inclusion in many user interface development methodologies. These methodologies are supported by different tools and software products coming from both academia software companies. Next, an overview of some methodologies and tools where prototyping and evaluation are considered is provided.

Methodologies

Usability engineering methodologies are software development methodologies with a special emphasis on optimizing the usability of the system that is to be developed. User interface prototyping activity with different degrees of detail is often considered in many methodological proposal included in Usability Engineering. Examples include the Star Lifecycle [6] Usability Engineering [17], LUCID (Logical User-Centered Interface Design Method) [21], the Usability Engineering Lifecycle [13], Usage-Centered Design [4], MUSE (Method for Usability Engineering) [10] and ACUE approach [19].

Examples of usability engineering techniques include affinity diagramming and Canonical Abstract Prototypes [4], evaluation activities and usability inspections [17], heuristic evaluation [17], and task analysis and different kinds of prototypes [13]. In these methodologies, prototyping is a recurrent activity for user interface development.

Tools

Prototyping has been used by many developers [22], and it is supported by several tools. Some meaningful examples from academia are: GUILayout [2], DENIM [11], CanonSketch [3], SketchiXML [5]. There are also some other commercial tools such as Pencil (http://www.evolus.vn/pencil/) or SketchFlow (http://www.microsoft.com).

In some of the aforementioned tools, for instance in DENIM, experience has been integrated into prototyping, but no tool supports assessing or analyzing the prototypes created by using these tools.

In this paper the joint consideration of prototyping and analysis is proposed. Nevertheless, to support the evaluation of the prototypes the identification of some quality criteria is required. In the next section, some quality criteria for prototypes are discussed.

QUALITY METRICS IN PROTOYIPING

In user interface development methodologies considering usability there are plenty of quality principles and criteria than can be considered. Shneiderman [2], Nielsen [16, 17], Norman [18], Mayhew [12], Constantine and Lockwood[4], Tognazinni (www.asktog.com) and Bastien and Scapin [1], are probably the most recurrently cited in the literature.

The problem is that those criteria are hard to apply and evaluate automatically, and designer experience plays an important role when putting them in practice and assessing whether they are achieved or not.

We have identified some user interface analyzing activities that can be automated to contribute to the evaluation of user interfaces, and that can be used for prototyping assessment. These analyzing activities will be described in depth in the next section, and they were proposed in [16] and [4].

Zone Analysis by Nielsen and Tahir

In the book "Homepage Usability. 50 Websites Deconstructed" [16], the authors describe several heuristic usability evaluations of different websites. For this evaluation the authors consider some style guidelines (113) defined at the homepage level of websites. In the book, they analyzed also the use and availability of the different zones of the homepage of each website. Thus, the authors analyzed the purpose each zone was devoted to in the homepage.

Color	Purpose	Ideal value
	Content of interest	6%
	Advertising and sponsorship	25%
	Self promotional	25%
	Filler	6%
	Unused	2%
	Welcome and site identity	11%
	Navigation	25%

Table 2. Breakdown of screen real state.

The purpose for the different zones in a website for real state is shown in Table 1. The purpose of each zone can be specified when prototyping the user interface in GUILayout++.

An Structural metric by Constantine and Lockwood

Constantine and Lockwood [4] identified several flavors of design metrics for user interfaces: structural metrics (which are based on surface properties), semantic metrics (which are content sensitive) and procedural metrics (which are task sensitive).

The simplest to compute are those measures called structural metrics, which are based on those surface properties of the configuration and layout of user interface architectures that can simply be counted or toted. In this sense, not every designer who ends up responsible for user interface design necessarily has a graphic designer's eye for layout.

Layout uniformity is a structural metric that gives a quick handle on one important aspect of visual layout.

Layout Uniformity (LU) measures only selected aspects of the spatial arrangement of interface components without taking into account what those components are or how they are used. LU is based on the rationale that usability is hindered by highly disordered or visually chaotic arrangements.

$$LU = 100\left(1 - \frac{(N_h + N_w - N_t + N_l + N_b + N_r) - M}{6N_c - M}\right)$$

where N_c = total number of visual components on screen, dialog box or other interface composite and N_h, N_w, N_t, N_l, N_b and N_r are, respectively, the number of different heights, widths, top-edge alignments, left-edge alignments, bottom-edge alignments, and right-edge alignments of visual components. M is an adjustment of the minimum number of possible alignments and sizes needed to make the value of LU range from o to 100. $M = 2 + 2\left\lceil 2\sqrt{N_{components}} \right\rceil$.

As a structural metric concerned only with appearance, Layout Uniformity should not be given undue weight in evaluating designs. It can, however, be useful to the designer who lacks on aye for layout to know when a visual arrangement might be improved.

This metric, Layout Uniformity, was implemented and included in GuiLayout++.

GUILAYOUT++: PUTTING TOGETHER PROTOTYPING AND EVALUATION OF USER INTERFACES

GUILayout++ improves GUILayout tool. This tool was developed by Kai Blankenhorn [2] for his master thesis. This work identified the limitations that Unified Modeling Language (UML) provides for the specification of user interface. To overcome the identified limitations he created a UML profile for user interface development and for web site prototyping.

GUILayout is a profile that provides an easily comprehensible abstract representation of an actual screen based on designers' sketches. A GUI Layout Diagram consists of a Screen, which contains multiple *ScreenAreas*. Each may be decorated with one or more Stereotypes, representing performed functionalities like text, image or link. By nesting and arranging properly stereotyped ScreenAreas within each other, the developer is able to create an abstract version of a user interface (see Fig. 3). The Navigational Diagram provides UML-based support for common design artifacts like storyboards and sitemaps.

GUILayout represents an interesting contribution in the scope of user interface prototyping, and a complement for the different diagrams defined for UML. Nevertheless, this tool can be improved with evaluation and analysis facilities for software products that assist designers in the development of the user interface, which is always a hard task. Furthermore, as aforementioned, some analysis and metrics can be automatically computed to help in the different iterations of the design process.

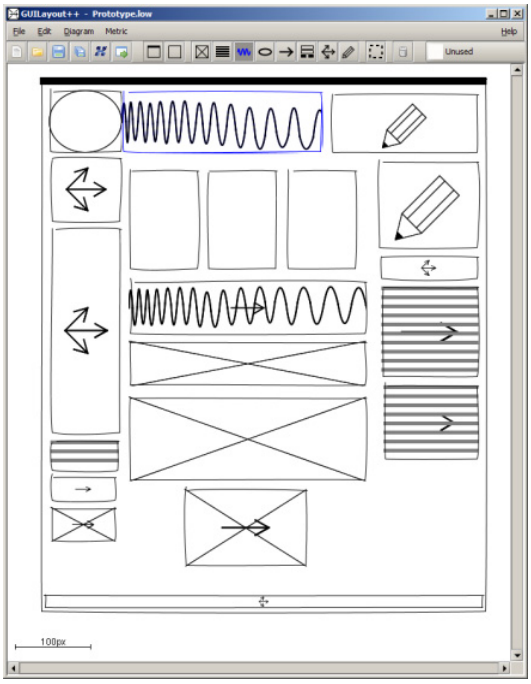

Figure 3. GUILayout tool and low-fidelity prototyping.

Using GUILayout++

One of the improvements GUILayout++ provides with respect to GUILayout is supporting low-fidelity prototyping while visually specifying, by using color codes, the purpose each area or zone is devoted to (see Fig. 4).

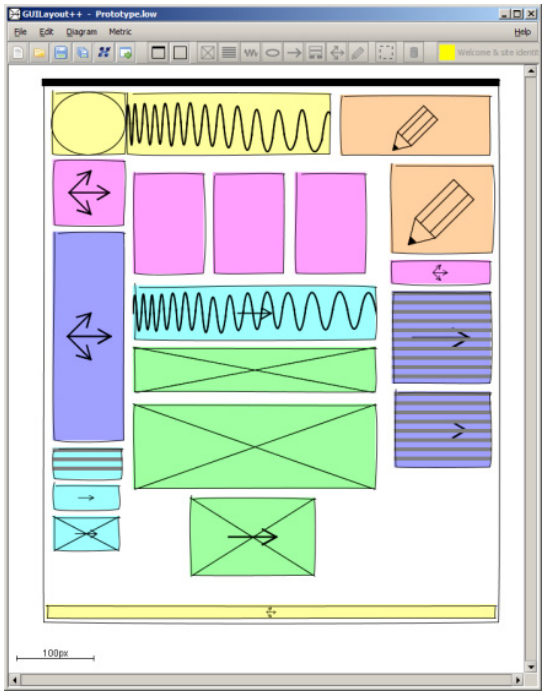

Figure 4. GuiLayout++ and low-fidelity prototyping.

By using the color codes, designers can identify visually and numerically the space devoted to the different concepts described by Nielsen and Tahir [16] (see Table 1), but working with a low-fidelity prototype. The resulting values can be compared with the ideal values for each application domain. Depending on how the resulting values match the ideal values they are tagged as: *ok*, *warning* o *error* (see Fig. 5). The ideal values used in the tool can be changed to match the application domain of the user interface designed.

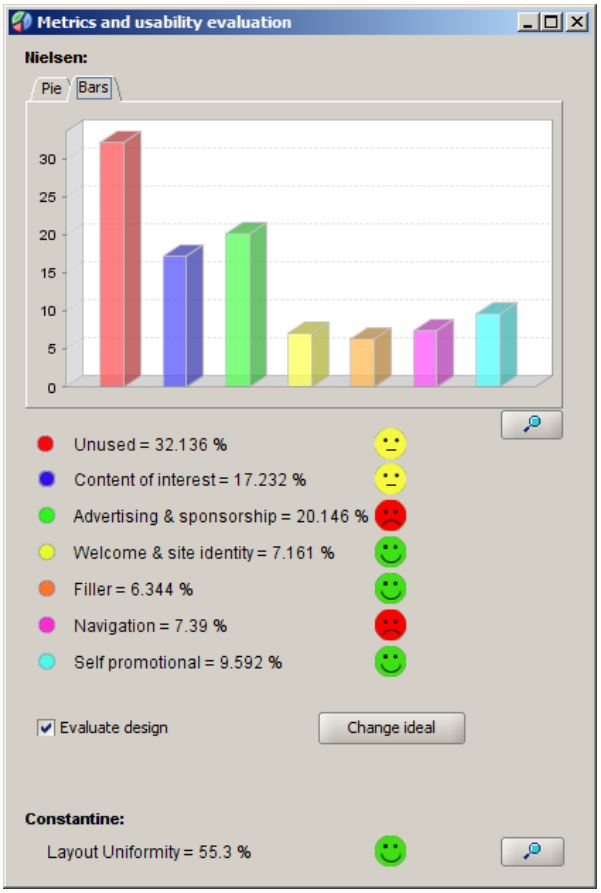

Figure 5. GUILayout++ and high-fidelity prototyping and metrics

Another facility provided by GUILayout++ is computing metrics related to structural design features found in the prototype. Currently GUILayout++ support computing *Layout Uniformity* metric, as proposed by Constatine and Lockwood [4] (see Fig. 6). Additional metrics could be considered in GUILayout++, for instance, [24]. Another UI metrics, for instace Ivory [7, 8], are not useful in this version of GuiLayout++. GUILayout++ cannot only work on low-fidelity prototypes, but also on high-fidelity prototypes. By using screenshots the designers can use high-fidelity prototypes to carry out the analysis previously described. This other way of working with GUILayout++ is illustrated in Fig. 7. In this figure, a screenshot of the web for tourism of Wallone region in Belgium (http://www.wallonie.be/fr/index. html) has been used. Then the designer has marked the different zones in the website, specifying the purpose of each zone.

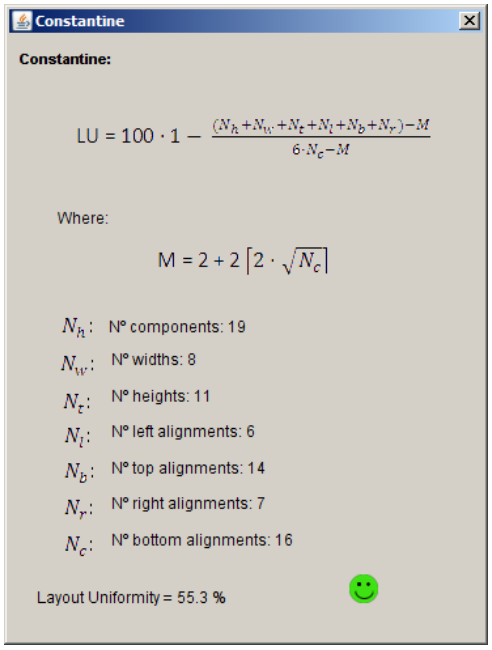

Figure 6. Layout Uniformity metric in GUILayout++.

Figure 7. GUILayout++ and high-fidelity prototyping.

Generating usiXML Abstract User Interface

Creating prototypes contributes to the subsequent user interface development, or to its refinement. In GUILayout++ the designer can use the prototype created in the user interface development process by storing the prototype using the abstract user interface components proposed for UsiXML abstract user interface model. These components are represented visually by using the graphical notation for the abstract user interface model proposed in IdealXML [14] (see Table 2). The abstract user interface for the high-fidelity prototype in Fig. 7 is shown in Fig. 8.

Image	Element
	Container
	Component
	Input
	Output
	Control
	Navigation

Table 3. Abstract User interface elements defined in UsiXML.

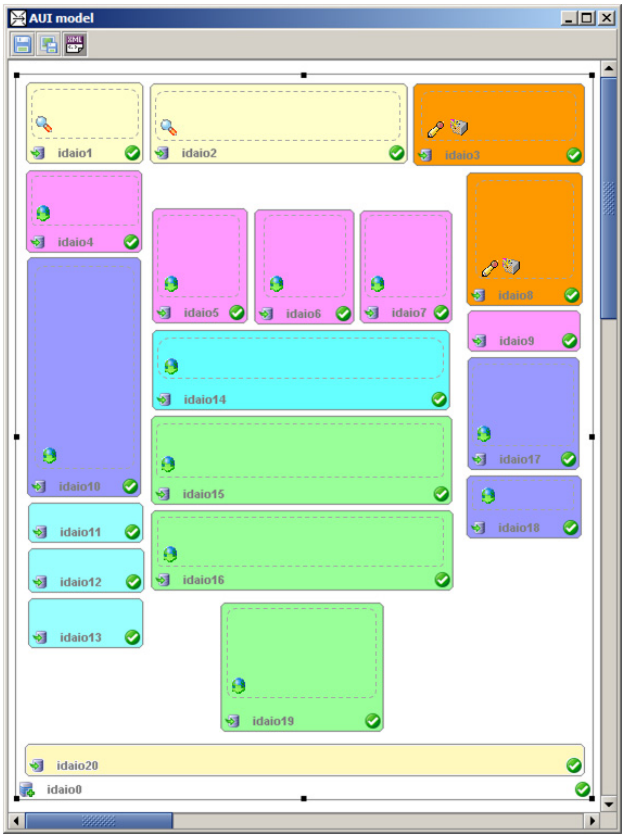

Figure 8. Abstract user interface specification by using UsiXML.

Starting from the abstract user interface generated by GUILayout++ the designer can follow different development paths for the user interface design, as described in [9]. For instance, the designer can derive the task model by using abstraction transformations, or he can go on refining the abstract user interface to generate a concrete user interface, that latter will be transformed into the final user interface the user will interact with.

CONCLUSION AND FUTURE WORK

In this paper the need to combine two activities, prototyping and evaluation, when tackling user interface design is exposed. In this sense, we have identified that those activities are usually included in many user centered or Usability Engineering methodologies available in the literature. In the scope of user interface evaluation several metrics haven identified that can be considered in multiple prototyping tools. In this sense, some of them have been included in the prototyping tool GUILayout, to create GUILayout++. Currently, we are including additional metrics in the prototyping tool so behavioral and navigational aspects are considered also in GUILayout++.

ACKNOWLEDGMENTS

This work is partly supported by the Spanish PEII09-0054-9581 and CICYT TIN2008-06596-C02-01 grants. .Also, we gratefully acknowledge the support of the ITEA 2 Call 3 UsiXML European project under reference #2008026.

REFERENCES

1. Bastien, J., Scapin, D. Preliminary findings on the effectiveness of ergonomic criteria for the evaluation of human-computer interfaces. In *Proc. of INTERCHI Adjunct Proceedings 1993*: 187-188
2. Blankenhorn, K.: A UML Profile for GUI Layout, Master's Thesis, University of Applied Sciences Furtwangen, Department of Digital Media (2004)
3. Campos, P. Nunes, N. J. CanonSketch: a User-Centered Tool for Canonical Abstract Prototyping, In *Proceedings of EHCI-DSV-IS'2004*, Hamburg, Germany, 2004.
4. Constantine, L., Lockwood, L. Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design, 1999.
5. Coyette, A., Vanderdonckt, J., and Limbourg, Q. SketchiXML: A Design Tool for Informal User Interface Rapid Prototyping. In *Proc. of Int. Workshop on Rapid Integration of Software Engineering techniques RISE'2006* (Geneva, 13-15 September 2006). Lecture Notes in Computer Science, Vol. 4401, Springer-Verlag, Berlin, 2007, pp. 160-176.
6. Hix, D., Hartson, H.R. Developing user interfaces: ensuring usability through product & process. Wiley, 1993.
7. Ivory, M., Sinha, R., Hearst, M.: Empirically validated web page design metrics. In *Proc. of CHI 2001*: 53-60.
8. Ivory, M., Sinha, R., Hearst, M.: Statistical profiles of highly-rated web sites. In *Proc. of CHI 2002*: 367-374.

9. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., and Lopez-Jaquero, V. UsiXML: a Language Supporting Multi-Path Development of User Interfaces. In *Proc. of 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCI-DSVIS'2004* (Hamburg, July 11-13, 2004). Lecture Notes in Computer Science, Vol. 3425, Springer-Verlag, Berlin, 2005, pp. 200-220.
10. Lim, K.Y, Long, J. The MUSE method for usability engineering. Cambridge ; New York : Cambridge University Press, 1994.
11. Lin, J., Newman, M. W., Hong, J. I., Landay, J. A. DENIM: finding a tighter fit between tools and practice for Web site design. In *Proc. of CHI 2000*: 510-517
12. Mayhew, D. J. (1999). The usability engineering lifecycle: A practitioner's handbook for user interface design. San Francisco, California
13. Mayhew, D. J., Principles and Guidelines in Software User Interface Design, Prentice-Hall, 1992
14. Montero, F., López-Jaquero, V. IdealXML: An Interaction Design Tool. In *Proc. of CADUI 2006*: 245-252
15. Myers, B. and Kosbie, D. Reusable Hierarchical Command Objects. In *Proceedings of CHI'96: Human Factors in Computing Systems*, Vancouver, BC, Canada, April 14-18, 1996.
16. Nielsen, J. and Tahir, M.: Homepage Usability: 50 Websites Deconstructed. New Riders Press. Nov. 15, 2001.
17. Nielsen, J. Usability engineering. Boston, MA: AP Professional, 1993.
18. Norman, D. A. (1988). The psychology of everyday things. New York: Basic Books.
19. Seffah A., and Metzker, E. Usability Engineering Methods Plethora. Adoption-centric Usability Engineering Systematic Deployment, Assessment and Improvement of Usability Methods in Software Engineering. Springer.
20. Shneiderman, B. Designing the User Interface: Strategies for Effective Human-Computer Interaction, 1st edition. Addison-Wesley. 1987
21. Smith, A., Dunckley, L. User Centred Design: The Application of the LUCID Interface Design Method. HCI (2) 1997: 563-568
22. Szekely, P. User Interface Prototyping: Tools and Techniques. Technical report, Intelligent Systems Division, University of Southern California. 1994
23. Usixml. http://www.usixml.org.
24. Vanderdonckt, J., Gillo, X., Visual Techniques for Traditional and Multimedia Layouts, in *Proc. of 2nd ACM Workshop on Advanced Visual Interfaces AVI'94* (Bari, 1-4 June 1994), T. Catarci, M.F. Costabile, S. Levialdi, G. Santucci (eds.), ACM Press, New York, 1994, pp. 95-104.