A Typology of Widgets for UI Design

Tanguy Wettengel
University of Limoges (France)
tanguy@teamtim.com

Dinh Quang Trung
International University (Vietnam)
dinhquangrim@gmail.com

Do Anh Vu
International University (Vietnam)
vuda.it@gmail.com

Abstract—Widget selection when designing user interfaces is a rather intuitive task. No clearly established methodology containing in-depth descriptions of widgets or explicit rules to choose them has been included in the major development processes. Widget catalogs are only partially specified and do not overtly consider the domain-field knowledge representation as bound to current tasks. Moreover, **these representations have not undergone any systematic approach from the point of view of their structure. The first aim of this paper is to formally describe the major types of content that widgets can host from this perspective. We call these types "Information Structures" and divide them into 5 categories ("atom", "collection", "hierarchy", "taxonomy" and "network"). We then specify a series of features allowing to classify widgets alongside with their informational content structure and set a framework for widget description.**

Index Terms—information structure, interface design, widget

I. Widget Choice in User Interface Design

User interface has proven to be a strategic component of most software packages and websites. The increasing awareness of this fact on a wide scale has been lately expressed by Wilbert [1] as follows: "User interface is one of the most critical contributors to the efficiency of any human operated computer system". Surprisingly though, most software development processes do not include an explicit UI specification method.

One example is the Rational Unified Process (RUP). While building up the interface, designers start by looking at the requirements to be considered in the current iteration (especially Use Cases and/or Storyboards), then identify the primary windows of the system's user interface [2], then select the appropriate widgets and organize them in space relying on their experience. Such a strategy seems bound to risk, mainly because the accuracy of the design choices can only be revealed when the product reaches its end-users, not less because the skill of designers can vary dramatically.

Extreme Programming (XP) seems to better tackle the problem, as all phases of an XP project require communication with the customer [3]. But, although customers are often experts in their own domain, they can seldom contribute to interface design in an efficient way (the problem at hand being not domain-knowledge itself but the rendering of it in a comprehensible task-oriented fashion).

Within interface design, the choice of widgets is a critical step, because it determines to a great extent the user-friendliness of an application. Besides general recommendations to be found in the major operating system development guidelines, the choice of widgets encounters the same pitfalls occurring in other phases of interface design, namely the fact of relying on implicit knowledge. The lack of sound methods to meet deductive decisions in this field becomes even more dramatic when considering that not merely usage conventions but domain knowledge structure within the current task, type of hosted data and value selection mechanism should be jointly considered to allow a systematic choice of widgets, as will be pointed out below.

This paper will promote a framework for classifying widgets as a tool to meet such decisions. It builds on a formal representation of major structures organizing domain-knowledge in a task-oriented perspective and a frame flavored analysis of functional features.

II. Practical Approaches

Outside process-based approaches (RUP, XP, etc.), widget selection has been considered as such, rather by practitioners than by scholarly research projects. Concerning widget typology, a practical guide by Jenifer Tidwell [4] grounds widget choice on a set of design patterns, tailored to the situation at hand. Patterns are defined following the "Gang of Four's" approach (what, when, why, how) and considered from the perspective of Human-computer interaction, software engineering and Web design. The aim is to allow designers to select the appropriate widget taking into account data types and usage situations.

In Web design, according to Jodi Bollaert [5], an IBM Usability specialist, the choice of widgets should be based on three main factors: user characteristics, content and technology. Content impacts the selection of Web widgets: if users can select from a set of known values, options should be expressed in the form of radio buttons, checkboxes, dropdowns, or list boxes, rather than text boxes, for example. Technology affects widget selection, as target platforms, browser types and versions, screen resolutions, and access speeds narrow the choices theoretically available.

Peter Picone [5], an independent Usability expert, suggests taking into account screen resolutions when defining the position of Web widgets (as list boxes and dropdowns placed too close to the fold may obscure the user's capacity to see the listed items). The impact of user characteristics on widget choice has equally been underlined by David Unsworth [5] (Saga Services Ltd.), who suggests that the age of intended users should be in focus when selecting Web widgets. He found out by experimental studies that mature adults (over 50) rarely use dropdowns and that their navigational behavior is dominated by the use of “visible links”.

As may be seen, the above case studies are closer to practical recommendations than to fine-grained analyses of widgets in terms of a comprehensive set of features. As for the distinguished relation between content types and the visual rendering of them (which plays a major role in the process of widget choice), inspirational background can be found in the work about analytical design by Edward Tufte (for a complete bibliography, see http://www.edwardtufte.com/tufte/index).

Less ambitious, a typology of cognitive information patterns and the best way of rendering them visually has been sketched by Bernie Dodge [6], who reminds that “Information can be portrayed visually in a number of ways. Some structures provide a better fit with the data than others”. Patterns (such as Cluster, Hierarchy, Venn Diagram, Timeline, Flowchart, Concept Map, Causal Loop Diagram, Comparison Matrix, Inductive Tower) are coupled with types of content and described by templates and examples.

III. Task-driven Domain-knowledge Structures

Choosing the right widget when setting up interfaces heavily depends on the domain knowledge that is being invoked and on the task being performed. Moreover, different user points of view can lead to different choices of widgets for the same data. For example, for a set of courses, a designer can simply choose a list box if he is setting up an interface for a non-student user (guest, parents...). This presentation would be suited to view the available courses and their description. If the designer is planning an interface for students that register online, he can take advantage of a tree view, because this presentation allows to show the pre-requisite relationship between courses (which course should have been already taken to register in some other one), an information that is crucial for the intended user . We here see that the same data can be expressed by different widgets depending on how it is thought of (and used) by the targeted audience:

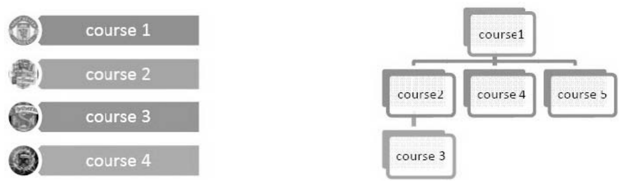


Fig. 1. A group of courses as represented by list and tree view

Our aim is to establish how the presentation specific to a widget (or widget cluster, as radio buttons) relates to what can be viewed as the target’s operative image[1] of information (how it is organized according to a task’s purpose - tree, list, etc. -, and how it can be used - single vs. multiple choice, for example -). As the main activity performed via widgets is the selection of some value, the information stored in a widget (or widget cluster) generally comprises many items. It is mainly the relation between these items that grounds the appropriate choice of the widget that will allow the user to select one or more of them.

For this reason, we primarily focus on this topic and propose a typology of structures, each one exhibiting a different relation between its elements. We call the four following basic types “Information Structures” and use set representation to describe them. An additional Information Structure consisting of a single item is termed “Atom” and not described in Table 1.

For Hierarchies[2], we assume that, even if more than one element of the domain set could (theoretically) map onto the same value of the co-domain set (in our above example, a car and a van having exactly the same loading capacity), a one-by-one ordered presentation of the domain set elements is usually required to populate a widget. This is why we describe hierarchies via functions (and not relations, admitting more than one image for an element of the domain set) and we map the domain set onto places in a sequence and not onto values of the ordering factor viewed as an attribute.

In networks, members of different collections are linked by some semantic relation with neither classifying nor ordering purposes. The semantic nature of these links is manifold, but can be described generally by ways of meronymic and pragmatic relations [7]. We here call them "association factor".

IV. A Typology of Widgets

Widgets are classified in Tables 2 and 3 according to their functional properties (a to c) and to their compatibility with the five Information Structures described in Section 3 (d). The use of this typology when designing interfaces can enable a systematic choice of widgets and thereby bridge the important gap Section 1 points at in software development processes and, of course, in Web design strategies.

a) Hosted data features

- *Number of levels* (Single/Multiple): Whether the data content comprises multiple levels or not.
- *Contrast* (Low/High): Whether the semantic contrast of the data elements is perceived as exclusive or not.
- *Type* (Text/Image): Whether the widget can handle text, numbers or other data types.
- *Variation* (Fixed/Dynamic): Whether the content of data the widget hosts changes over time or not.
- *Chunking* (Continuous/Discrete): Whether the data contrast is perceived by the user as being on a fuzzy border scale or not.

b) Control

- *Content Controller* (Data/User/System): Whether the included content can be changed or not by the user or by states of the system.
- *Selection controller* (User/System): Whether the user or the system select the current value.
- *Selection range* (Single/Multiple): Whether more than one value can be selected or not.

TABLE I
INFORMATION STRUCTURES

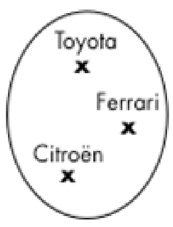
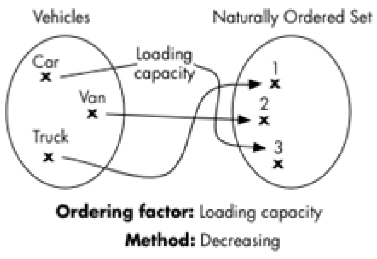
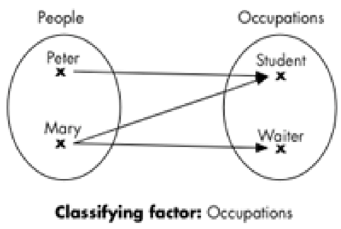
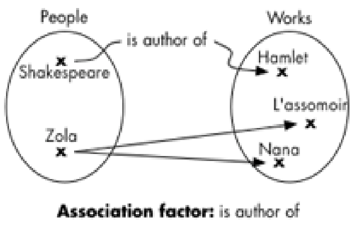
Collection	Hierarchy	Taxonomy	Network
A group of elements	An ordered group of elements	A classified group of elements	Elements semantically connected to other elements
A group of car brands	*A group of vehicles ordered by loading capacity*	*A group of people classified by occupation*	*A group of people tied to a group of works by an "authoring" link*
	 Vehicles Naturally Ordered Set Car Van Truck Loading capacity 1 2 3 Ordering factor: Loading capacity Method: Decreasing	 People Occupations Peter Mary Student Waiter Classifying factor: Occupations	 People Works is author of Shakespeare Zola Hamlet L'assomoir Nana Association factor: is author of
Mathematical Representation			
Set	Function	Relation	
	function label: **ordering** factor	co-domain: **classifying** factor	relation label: **association** factor

TABLE II
WIDGET FEATURES

	HOSTED DATA FEATURES					CONTROL			PRESENTATION			
	Number of Levels	Contrast	Type	Variation	Chunking	Selection Range	Content controller	Selection controller	Capacity	Content representation	Range comparability	Position variation
Slider	Single	Low	Img.	Fixed	Cont.	Mult.	System	User	Single	Anal.	Yes	Fixed
Scrollbar	Single	Low	Img.	Dynamic	Cont.	Mult.	Data	User	Single	Anal.	Yes	Fixed
Spin Button	Single	Low	Img.	Fixed	Discr.	Mult.	System	User	Single	Anal.	No	Fixed
Progress Bar	Single	Low	Img.	Fixed	Cont.	Mult.	System	System	Single	Anal.	Yes	Float
List View	Single	Low	Text	Dynamic	Discr.	Single	Data	User	Mult.	Symb.	Yes	Fixed
List Box	Single	Low	Text	Dynamic	Discr.	Single	Data	User	Both	Symb.	Yes	Fixed
Dropdown List	Single	High	Text	Dynamic	Discr.	Mult.	System	User	Single	Anal.	No	Fixed
Spin Dropdown	Single	High	Text	Dynamic	Discr.	Mult.	System	User	Single	Anal.	No	Fixed
Graph Presenter	Single	High	Img.	Dynamic	Cont.	Single	Data	System	Mult.	Anal.	Yes	Fixed
Menu	Mult.	High	Text	Fixed	Discr.	Single	Data	User	Single	Symb.	Yes	Fixed
Context Menu	Mult.	High	Text	Fixed	Discr.	Single	Data	User	Single	Symb.	Yes	Float
Tree View	Mult.	High	Text	Dynamic	Discr.	Single	User	User	Single	Symb.	Yes	Fixed
Multiple Choice Tree View	Mult.	Low	Text	Dynamic	Discr.	Single	System	User	Mult.	Symb.	Yes	Fixed
List Tree View	Mult.	Low	Text	Dynamic	Discr.	Single	System	User	Mult.	Symb.	Yes	Fixed
Check Boxes	Mult.	High	Text	Fixed	Discr.	Single	System	User	Mult.	Anal.	Yes	Fixed
Radio Buttons	Single	High	Text	Fixed	Discr.	Single	System	User	Single	Anal.	Yes	Fixed
Hyper Links	Single	Low	Text	Dynamic	Discr.	Single	Data	User	Single	Symb.	No	Fixed
Button	Single	Low	Text	Fixed	Discr.	Single	System	User	Single	Anal.	No	Fixed
Textbox	Single	Low	Text	Dynamic	Discr.	Single	User	User	Single	Symb.	No	Fixed
Mask Textbox	Single	Low	Text	Dynamic	Discr.	Single	User	User	Mult.	Symb.	No	Fixed
Label	Single	Low	Text	Fixed	Discr.	Single	System	System	Single	Anal.	No	Fixed
Tooltip	Single	Low	Text	Fixed	Discr.	Single	System	User	Single	Anal.	No	Float
Picture Box	Single	Low	Img.	Dynamic	Discr.	Single	User	User	Single	Anal.	No	Fixed
Group Box	Single	Low	Text	Fixed	Discr.	Single	System	User	Single	Symb.	No	Fixed
Toolbar	Single	High	Img.	Fixed	Discr.	Mult.	System	User	Single	Anal.	No	Fixed
Tab Control	Single	Low	Text	Fixed	Discr.	Mult.	System	User	Both	Symb.	Yes	Fixed
Palette	Single	High	Img.	Fixed	Cont.	Mult.	System	User	Single	Anal.	Yes	Fixed

c) Presentation

- *Position variation* (Fixed/Floating): Whether the position of the widget is fixed on the interface or not.
- *Range comparability* (Yes/No): Whether the user can compare values available to be selected or not, or whether he can see the whole range of selection during the widget usage or not.
- *Capacity* (Single/Multiple): Whether the widget can contain more than one data unit or not.
- *Content representation* (Analogical/Symbolic): Whether a widget codes values in analogical or symbolic format.

d) Information Structure compatibility

Possible values (Atom/Hierarchy/Sequence/Taxonomy/ Network)

TABLE III
WIDGET COMPATIBILITY WITH INFORMATION STRUCTURES

	Atom	Collection	Hierarchy	Taxonomy	Network
Slider			X		
Scrollbar			X		
Spin button			X		
List view		X	X		
List box		X	X		
Dropdown list		X	X		
Spin dropdown list		X	X		
Graph presenter		X	X		
Menu		X		X	
Context Menu		X		X	
Tree view				X	X
Multiple choice tree view				X	
List tree view				X	
Check box		X	X		
Radio button		X	X		
Hyperlink					X
Button	X				
Textbox	X				
Mask textbox	X				
Label	X				
Tooltip	X				
Picture box	X				
Group box		X			
Toolbar		X			
Tab control		X			
Palette		X	X		

V. LIMITATIONS AND FUTURE WORK

The number of widgets brought into consideration can be increased to cover a broader spectrum (we have only taken twenty seven fundamental widgets into account). Moreover, as widgets are classified by properties that could appear in future ones, this proposal can act as a framework for widget engineering. Even if the parameters considered in this framework are further questioned or modified, the interest of the framework itself is to put forward the idea that a structured and systematic picture of input facilities (widgets) is achievable.

An in-depth analysis of the relation between the task being performed and the way the information relevant to it is structured (allowing the same information to be structured in different ways for different tasks) can increase the knowledge of dynamic cognition processes and thereby inforce the thesis of task-sensitive knowledge organisation.

[1] Internal representations related to action. Ochanine, the propounder of this concept, underlines that the image of an object involved in action contains only the features relevant to the process [8].

[2] "Partial orders are the basic abstract structure describing hierarchical organisation. A partial order consists of a collection of elements and an order relation on these elements, describable as "domination." The order relation is required to satisfy three axioms: 1. Reflexivity: Each element dominates itself. 2. Transitivity: If a dominates b, and b dominates c, then a dominates c. 3. Antisymmetry: If a dominates b and b dominates a, then a and b coincide. A pair of elements is said to be comparable if one dominates the other. The partial order is said to be total or linearly ordered, or described as a chain, if each pair of elements is comparable. The opposite extreme is represented by antichains, which are partial orders in which no two distinct elements are comparable. In other words, in an antichain, the elements are all unrelated to each other." [9]

REFERENCES

[1] O. G. Wilbert, Essential guide to user interface design. IN: Widley Publishing, Inc., 2007.
[2] IBM. (2005). Rational Unified Process: Best practices for software development teams [Online]. Available: http://www.ibm.com/developerworks/rational/library/253.html
[3] J. D. Wells, (1999). The rules and practices of Extreme Programming. Available: http://www.extremeprogramming.org/rules.html
[4] J. Tidwell, Designing Interfaces : Patterns for Effective Interaction Design, Sebastopol. CA: O'Reilly Media, Inc, Nov. 2005.
[5] J. Bollaert (2002). Using web widgets wisely. Available: http://www.ibm.com/developerworks/web/library/uswidget/
[6] B. Dodge (1998). A Taxonomy of Information Patterns [Online]. Available: http://projects.edtech.sandi.net/staffdev/tpss98/patterns-taxonomy.html
[7] E. Winston, R. Chaffin and D. Herrmann, "A taxonomy of part-whole relations", Cognitive science, vol. 11, pp. 417-444, Dec. 1987.
[8] D. A. Ochanine, (1966). The operative image of controlled object. In "Man Automatic Machine Systems: 27th International Psychology Symposium. Moscow.
[9] J. D. H. Smith. Three Key Questions about Complex Systems [Online]. Available: http://orion.math.iastate.edu/jdhsmith/math/SEEtkqcs.htm

Tanguy Wettengel holds a PhD in Computational Linguistics and a Higher Doctorate in Language Sciences. He is a regular professor at the University of Limoges and an invited professor at the International University of Vietnam.

Dinh Quang Trung and **Do Anh Vu** graduated in Computer Science at the International University of Vietnam in 2008.