Journal of Visual Languages and Computing 38 (2017) 1–8

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

Iconic languages: Towards end-user programming of mobile applications

Rita Francese*, Michele Risi, Genoveffa Tortora

Department of Computer Science, University of Salerno, Italy

ARTICLE INFO

Keywords:
Visual languages
Mobile development
Iconic languages
Visual programming

ABSTRACT

After tracing the steps that led to the current generation of iconic languages starting from the original idea of S.K. Chang, we describe an iconic language, named MicroApp, for modeling pervasive mobile applications directly on the mobile device. MicroApp exploits generalized icons for composing mobile applications: services are represented by icons and are composed of adopting colors for representing data-flow. We also qualitatively evaluate the visual environment that implements this iconic language.

1. Introduction

Pictorial representations may be an effective communication means because they exploit the powerful and highly parallel human visual system [1], which is able to get and efficiently process images. Indeed, around a quarter of the human brain is devoted to vision, more than all the combination of the other senses [2]. Visually represented information may also easier to remember, due to the picture superiority effect [3]. Visual communication may be more intuitive for humans than textual one, but not all the visual representations are useful for human beings [4]. In addition, visual communication may be more ambiguous than textual one [5]. Thus, Computer Science researchers started to investigate whether humans can be facilitated in communicating with a computer through images rather than by using text.

In 1973, Xerox proposed the Alto personal computer PARC,[1] the first computer offering a bitmapped screen, the desktop metaphor and a graphical user interface.

S.K. Chang was one of the leading proponents of the research area of Visual Languages, whose history started in 1983. Thanks to the advent of bitmap screens and pointing devices there was the need of having a forum where the necessary metaphors were investigated to provide more user friendly interactions between man and computers. In Chicago, 1983, at the IEEE Workshop on Languages for Automation the papers [6,7] were presented. There, the meeting of these two research groups put the bases of the first IEEE Symposium on Visual Languages,[2] 1984, Hiroshima, Japan. The research objective was to make the computer accessible to a wider range of people, thanks to the growing diffusion of bitmap-based video technologies. Together with S.K. Chang there were also Ichikawa, Jungert, Levialdi and Tortora. Since then, researchers have been interested in determining whether and how the computer can be easily used to transmit, display, manipulate and retrieve information in terms of images [8–14]. Another important issue has been to evaluate whether the adoption of a visual language provides usability benefits for target users, making users more productive [15].

This paper traces the evolution of visual languages and, more specifically, of iconic languages towards their adoption for supporting mobile software development. Thus, we summarize the main characteristics of iconic languages and then we propose an application of them, the MicroApp iconic language [16–18], enabling an end-user to compose mobile applications using a service-oriented approach. This language has been designed to be properly used directly on the mobile device by exploiting a touch-based interface for composing services represented by icons. We conducted a qualitative evaluation of the proposed approach, named MicroApp Generator [17], aiming at collecting the user perceptions when it is used for generating new mobile applications or reusing existing apps.

The paper is structured as follows: Section 2 introduces the main concepts related to visual languages and briefly summarizes the first steps of the research in the iconic language area, while Section 3 proposes an iconic language for the modeling of mobile applications. Section 4 describes the qualitative evaluation design and reports the obtained results. Finally, Section 5 concludes the paper by addressing also future scenarios for iconic languages.

2. Background

What is a Visual Language (VL)? There exist several definitions with different meanings. A VL is a language making a systematic use of visual expressions (e.g., icons, drawings or gestures) to convey a

* Corresponding author.
E-mail addresses: francese@unisa.it (R. Francese), mrisi@unisa.it (M. Risi), tortora@unisa.it (G. Tortora).

[1] https://en.wikipedia.org/wiki/History_of_the_graphical_user_interface#Xerox_PARC

[2] http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=218

http://dx.doi.org/10.1016/j.jvlc.2016.10.009
Received 21 December 2015; Received in revised form 31 May 2016; Accepted 14 October 2016
Available online 11 November 2016
1045-926X/

meaning in a formal way [8].

While textual information is linear (one-dimensional), visual representation is multi-dimensional.

The objects to be dealt with by a VL can be inherently visual, such as pictures, or inherently non-visual but with imposed visual representation, such as arrays, stacks, and queues. In the latter case, a visual representation is associated with these objects to get a more understandable man–machine interface. For further details see [8].

Visual Programming Languages (VPLs) are VLs used to program visually [19]. They are characterized by programming constructs and rules which are visually represented. Their aim is to enable the users to program by interacting with a computer through visual representations.

VLs are also largely adopted in data visualization in general, and, more specifically in software visualization and algorithm animation. The aim is to support software comprehension by depicting the static structure and dynamic semantics of programs or algorithms, where the system representation is generated by the computer starting from the software system [20]. Different from VPLs, this kind of VLs exploits visual techniques to display data, algorithms and software without visual programming [19].

Diagrams represent information indexed by 2D location [21]. A considerable VL research area consists in formalizing diagrammatic languages which ease humans during reasoning activities and communication [22] and in understanding how humans reason with diagrams [23].

VLs making an extensive use of images and icons are called *iconic languages*. An example of iconic language is the Chinese language, whose characters are not just letters of an alphabet, but often represent whole words or ideas [24]. The composition rules of iconic languages follow the idea behind the composition of Chinese characters, pictorial patterns with finite structures. Complex characters are obtained by composing other characters, by following well defined composition rules, which include horizontal, vertical and surrounding compositions.

In the literature several iconic languages have been proposed for both professional and end-users as *iconic programming systems* and *interfaces* [25,26], because they represent concepts and actions on these concepts.

The icon theory of S.K. Chang exploits icons for automating compilers, e.g., for generating VPL compilers. Other theories explicitly focused on what makes good a visual notation: representing information in a graphical form does not guarantee that it will be worth a thousand of any set of words [4]. One of the most relevant theories on how to use visual notation for communicating is the Bertin's *Semiology of Graphics* [27], which has large application in the modern theory of visual languages, also in the Software Engineering field (see [28] as an example).

According to Chang the term icon denotes a graphic symbol that represents an object (e.g., a document) or an action (e.g., print or delete) representing a computational process. This concept is at the basis of the following definition of *generalized icon* introduced by S.K. Chang [8]:

> An icon X is a pair (Xm, Xi), where Xi is the pictorial part of the icon (i.e., the image which appears on the screen) and Xm is the semantic part, i.e., the meaning of that image.

As an example, an icon depicting a stylized man lying on a bed (pictorial part) represents a Hotel (meaning of this image).

Fig. 1 shows the hierarchical icons classification proposed by S.K. Chang [29]. In general, a *Complex Icon* is a spatial disposition of *Elementary Icons*, which represent both objects and processes (or actions). *Composite Object Icons* are the results of the composition of *Elementary Object Icons*, while *Visual Sentences* are spatial arrangements of *Process Icons* and object icons. Finally, a structured set of related icons constitutes an *Icon System*.

The syntactic structure of an icon language can be specified through a grammar. Tortora and Leoncini proposed a model for the specification of an icon system in terms of *Picture Grammars* and a general purpose icon interpreter, based on attribute grammars [30]. *Picture Grammars* are exploited to describe the composition rules of a complex icon. A picture grammar has the following spatial operators in its terminal alphabet:

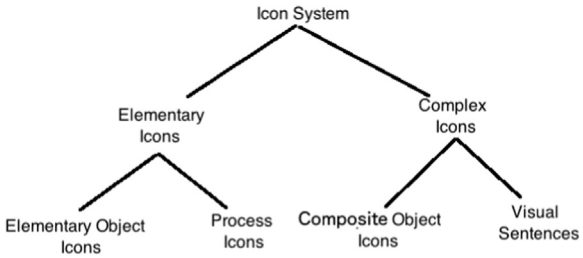

Fig. 1. Icons classification.

- & represents the horizontal concatenation;
- ∧ represents the vertical concatenation;
- + is the overlay operator.

Fig. 2 shows a visual sentence represented by the text string (*char* &*char* & *char*) + *cross*, where the term *char* corresponds to an elementary icon, i.e., the rectangle, while the cross symbol is a process icon corresponding to the action *delete*. The figure means that the three rectangles have to be deleted. Of course, another interpretation can be: "you cannot have three rectangles in a row". This highlights that there may be an ambiguous interpretation of a visual sentence.

An iconic operator applied to two icons generates a new icon. It acts on both the logical and physical parts of the icon. It is important to point out that logical and physical parts of an icon are mutually dependent. Thus, when the image of an icon changes, the meaning associated to it has to be accordingly changed, and vice versa [29].

When designing an iconic language, one of the most relevant factors to consider is the *icon purity* [31]. A pure icon enables to entirely recover the logical part from the physical part, and vice versa. In particular, let MAT(Xm) be the *materializator operator*, which associates to the meaning of an icon the images Xi related to it and DMA(Xi) the *dematerializator operator*, which associates to an image of an icon Xi its meaning.

In general, MAT(Xm) may yield a set of icon images. For example, MAT("Mona-Lisa") may be the original drawing of Mona-Lisa, or a sketch of Mona-Lisa. DMA(Xi) may also yield a set of meanings. An icon is pure only when MAT(Xm) and DMA(Xi) are both singletons. It is obvious that impure icons may cause misunderstanding when used in man–machine interfaces.

IconLisp [32], a sample Visual Programming Language, offered a visual language and a visual environment for Lisp programming. The benefits for programmers were in avoiding the use of parentheses and in the possibility of developing the functionality simply by composing other existing functionalities.

When several VLs have been proposed with different aims and domains, one of the issues was how to recognize the visual sentences of a VL. Thus, other relevant researches focused on the design and generation of an iconic language compilers [25]. The SIL-Icon compiler

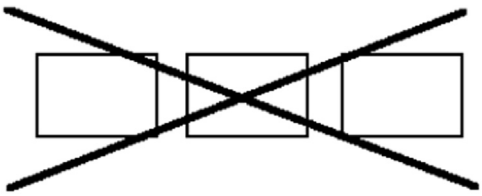

Fig. 2. An example of visual sentence.