A. Heil, I. Moradi, T. Weis; LCARS – The Next Generation Programming Context, CAI '06 Proceedings of the International Workshop in Conjunction with AVI 2006 on Context in Advanced Interfaces, Pages 29 – 31, Venezia, Italy, May 23-26 2006.

LCARS – The Next Generation Programming Context

Andreas Heil
Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart, Germany
andreas.heil@ipvs.uni-stuttgart.de

Iman Moradi
University of Huddersfield
Department of Creative Technologies
Canalside East, HD1 3DH, UK
i.moradi@hud.ac.uk

Torben Weis
Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart, Germany
torben.weis@ipvs.uni-stuttgart.de

ABSTRACT
In this paper, we present a high-level graphical language to develop pervasive applications based on a unique interface design. The language supports a wide range of programming constructs. Its graphical notation is based on the LCARS design, which is appealing to different target groups, based on their specific interests and requirements. We show that users can easily create pervasive applications using an LCARS-based user interface. The first step is to describe the technical context in which the application will execute. Based on this technical context, the UI offers a context-specific set of visual primitives. By composing these visual primitives on the screen, the user can specify the behavior of the application.

Categories and Subject Descriptors
H.5.2 **[Information Interfaces and Presentation (e.g., HCI)]**: User Interfaces – *Graphical user interfaces (GUI), Interaction styles (e.g., commands, menus, forms, direct manipulation)*; H.1.2 **[Models and Principles]:** User/Machine Systems – *Human factors*.

General Terms
Design, Human Factors, Languages.

Keywords
Visual Programming Languages, VRDK, Robots, Model-Driven Software Engineering, Context, Ubiquitous Computing.

1. INTRODUCTION
In the 1980s Michael Okuda designed a new kind of user interface (UI) known as Library Computer Access and Retrieval System (LCARS) [9]. The name originally stands for the main computer onboard of Galaxy-class starships of the television series Star Trek: The Next Generation [2] but it is also used for the colorful interface featured in the television series. Based on this design which is commonly appropriated on many fan websites, a community approach tried to develop and publish a standardized layout and color scheme to investigate if it is possible to assign real functionality to the fictional interface and to use the LCARS in a manner in which it was meant to be utilized in within the series [8]. Inspired by this idea, the Visual Robot Development Kit (VRDK) [5] UI was developed to create an easy accessible interface to program systems related to their current context rather then the specific platform the code has to be developed for. Beside this, we point out a specific feature of the UI that enables you to abstract the appearance of several UI elements by presenting these elements to the user in an appropriate context. This paper is organized as follows. In the next Section we discuss related work. In Section 3 we describe the VRDK user interface and overall features of the system. Section 4 addresses the context-related programming building blocks and their usage. We conclude with a summary of the lessons learned and an overview of future work on this topic.

2. RELATED WORK
Several kinds of visual programming languages exist already. Two that have been around for over 15 years and cater to professionals within science and the arts, are LabVIEW [7] and Max/MSP [3]. LabVIEW provides a graphical programming language, which focuses on the design, control, and testing of embedded systems, industrial monitoring, and a variety of automated tests and measurements. Max/MSP is widely used by performers, composers, and musicians to synthesize sample and process signals in an intuitive visual way. Core graphical elements used in Max/MSP can be modified using different user specified object names thus changing their current context. Both approaches have a tendency to allow users free reign over layout without providing many helpful cues.

3. THE LCARS APPROACH
Inspired by the distinguishable qualities of the LCARS interface elements, the VRDK design introduces several visual concepts for interacting with the application. For a seamless integration of the programming environment, the LCARS design was applied to the entire VRDK application. As we learnt the chosen interface had three elements, which made it very enjoyable to use: In [6] challenge[1], fantasy[2], and curiosity[3] are mentioned as the three building blocks which make the user experience captivating and enjoyable.

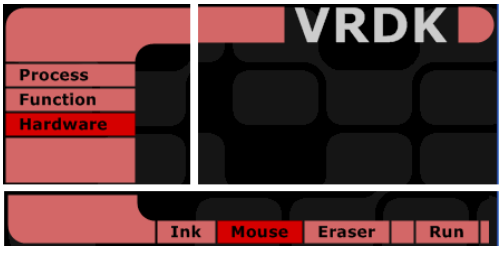

Figure 1. Applied LCARS style

[1] Finding and testing the capabilities of the tool.

[2] The Star Trek Television series associations.

[3] Discovering a new interface.

A. Heil, I. Moradi, T. Weis; LCARS – The Next Generation Programming Context, CAI '06 Proceedings of the International Workshop in Conjunction with AVI 2006 on Context in Advanced Interfaces, Pages 29 – 31, Venezia, Italy, May 23-26 2006.

As depicted in Figure 1, standard UI elements were replaced by LCARS inspired elements. The regions depicting *Process*, *Function* as well as *Hardware* are the equivalents to tabs, while *Ink*, *Mouse*, and *Erase* can be understood and operated as a group of radio buttons. With the same appearance *Run* complies with the functionality of a button. Finally, the image also shows a textbox that contains the text *VRDK*. Users can identify the purpose of each UI element in relation to its particular semantic space [4] without explicit reference to common design conventions.

The first step in developing a new application model is to set up a specific technical context, which has to be composed (cf. Figure 2). Additional entities available for composing a context can be easily created and added as plug-ins.

Figure 2. Composing a programming context

Related to the aforementioned context composition, the user gains access to specific commands and functions. By adding new elements to the technical context, new iconic programming elements are added to the UI. Figure 3 shows an increasing variety of programming elements as new devices and software components extend the technical context.

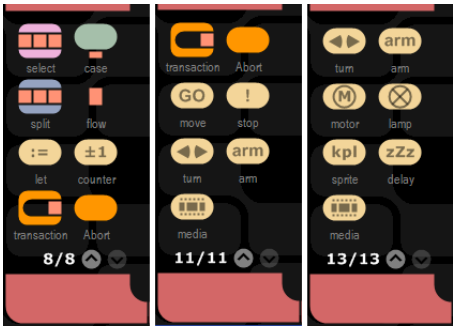

Figure 3. Context related programming elements

Often, non-traditional audiences are the target users of visual programming languages [1]. We can identify several groups of users who may benefit from the system: Young children (6-12 years) benefit from the context-related availability of commands and functionality, focusing on the particular topic of interest. Children (12-16 years) are interested because they are driven by exploration and the learning of new things as well as working with systems, which present a playful challenge. Young People and adults (16-50) as well as elderly people (50 and above) benefit from simple uncluttered interfaces.

4. CONTEXT RELATED VISUAL TOKENS

The visual programming language used in the VRDK is based on a context-free grammar. Each iconic programming element available for programming is in turn a rule or a terminal symbol of this grammar. These iconic elements are henceforth named *visual tokens* [1]. In contrast to many other approaches (cf. Section 2), the VRDK supports all major programming constructs as shown in Figure 4.

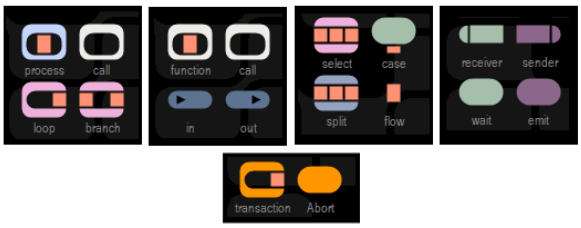

Figure 4. Major programming constructs as visual tokens

General programming tasks can be modeled by the visual tokens without setting a specific technical context. By adding additional devices or software components to a context view, new visual tokens become available within the UI. The color scheme of the visual tokens is predominately chosen to support the user in identifying specific constructs, such as control constructs or statements. By extending the technical context, additional commands and statements become available as shown in Figure 3. Removing entities from the context implicates the removal of related commands and statements from the visual token palette. Therefore, VRDK verifies the model before the removal of any element takes place. Consequently, any inconsistencies will be made apparent to the user. If the specific entity and thus the related visual tokens are used within the model, this conflict must be resolved before the removal is processed as indicated in Figure 5.

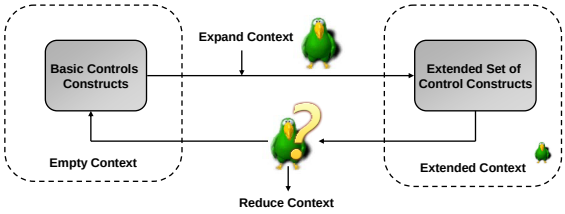

Figure 5. Modifying context

It is important to note that the context can be composed and evaluated without the physical presence of the particular entities used within the context view. The major benefit in this approach lies in the possibility of testing and simulating the application model without any risk or real world consequences.

5. OUTLOOK AND FUTURE WORK

In this paper, we have presented a user interface, inspired by a well-known design from a popular television series. Two different kinds of views on context can be observed: The first instance relates to how the UI abstracts conventional expectations of how UI elements behave. Their usage appears within the direct context of the element and the user can easily identify them as interaction ready buttons, radio buttons, or textboxes. Furthermore, the VRDK allows users to develop graphical models by composing rather than writing source code. As a benefit for the user, they can observe changes within the model immediately. In addition, an international experiment is planned to evaluate the effectiveness of this visual programming language and its adoption by students

A. Heil, I. Moradi, T. Weis; LCARS – The Next Generation Programming Context, CAI '06 Proceedings of the International Workshop in Conjunction with AVI 2006 on Context in Advanced Interfaces, Pages 29 – 31, Venezia, Italy, May 23-26 2006.

who are not formally trained in being programmers who possess none or very little knowledge of programming methodology. A further question to be investigated is the significance of the user's cultural background in their experience of using the system. Also related to this is the question of how being oblivious to the television series[4] may affect the usage of VRDK.

6. REFERENCES

[1] Emerging Technologies. Handbook of Software Engineering and Knowledge Engineering, ed. S.K. Chang. Vol. 2. 2002: World Scientific Publishing

[2] CBS Studio Inc., STARTREK.COM - Website (2006): http://www.startrek.com (06.04.2006).

[3] Cycling '74, Max/MSP - Website (2006): (05.04.2006).

[4] Kaplan, N. and Moulthrop, S.: Where No Mind Has Gone Before: Ontological Design for Virtual Spaces. In European Conference on Hypermedia Technologies. 1994. p. 206-216.

[5] Knoll, M., et al. Scripting your Home. In 2nd International Workshop on Location- and Context-Awareness (LoCA 2006). 2006. Dublin, Ireland.

[6] Malone, T.W.: Heuristics for Designing Enjoyable User Interfaces: Lessons from Computer Games. In Conference on Human Factors in Computing Systems. 1982. Gaithersburg, Maryland, United States: ACM Press. p. 63-68.

[7] National Instruments Corporation: LabVIEW - The software that powers virtual instrumentation - Website (2006): (07.03.2006).

[8] Rossi, C.: LCARS Standards Development Board - Website (2006): http://www.lcarsdeveloper.com (06.04.2006).

[9] Wikipedia: LCARS - Website (2006): http://en.wikipedia.org/wiki/Lcars (06.04.2006).

[4] As far as this is possible.