AUTOMATING A HUMAN FACTORS EVALUATION OF GRAPHICAL USER INTERFACES FOR NASA APPLICATIONS: AN UPDATE ON CHIMES *

Jianping Jiang Elizabeth D. Murphy
Sidney C. Bailin
CTA INCORPORATED, Rockville, MD

Walter F. Truszkowski
NASA-Goddard Space Flight Center, Greenbelt, MD

ABSTRACT

Capturing human factors knowledge about the design of graphical user interfaces (GUIs) and applying this knowledge on-line are the primary objectives of the Computer-Human Interaction Models (CHIMES) project. The current CHIMES prototype is designed to check a GUI's compliance with industry-standard guidelines, general human factors guidelines, and human factors recommendations on color usage. Following the evaluation, CHIMES presents human factors feedback and advice to the GUI designer. The paper describes the approach to modeling human factors guidelines; the system architecture; a new method developed to convert quantitative RGB primaries into qualitative color representations; and the potential for integrating CHIMES with user interface management systems (UIMS). Both the conceptual approach and its implementation are discussed. This paper updates the presentation on CHIMES at the first International Symposium on Ground Data Systems for Spacecraft Control[1].

Key Words: GUI, human factors guidelines, GUI design style guidelines, knowledge representation, color.

1. INTRODUCTION

An effective user interface (UI) is key to the success of any complex, automated system, such as a ground control center for spacecraft operations. Many efforts have been made to assist user-interface designers in building usable visual displays. User-interface prototyping tools provide direct interaction between the designer and the design, as well as reduced programming through automatic code generation[10]. Although building a user interface requires less effort than ever before, many poor designs can still be found because they violate human factors design principles or industry-standard guidelines, such as the specific requirements specified in the *OSF/Motif Style Guide*[9]. Improving poor UI designs requires human factors expertise and style-guidelines knowledge, which are, however, expensive and not readily available during design and development. Automated human-factors-guidelines and style-guidelines compliance checking are promising approaches to the problem[8].

Under the direction of NASA-Goddard Space Flight Center, a series of research and prototyping cycles has produced the current fourth-generation CHIMES methodology and toolset[7]. A previous paper describes the original CHIMES methodology and early prototypes[1]. The current GUI-evaluation prototype includes a style-compliance-checking tool, i.e. an automated critic that checks a design against style rules implemented in the tool's knowledge base. This prototype includes capabilities to evaluate conformance to human factors guidelines and recommendations on color usage. The tool provides feedback within the context of identified instances of non-compliance and advises the GUI designer on how to modify the design.

In the sections that follow, we describe the implementation approach used in the current prototype, the architecture, the implementation of compliance-checking rules, and plans for future research. Color representation in the knowledge base is also described.

*This work was performed at the NASA-Goddard Space Flight Center (Code 522.3) under Contract Number NAS5-30680, with the support of NASA Headquarters (Code O).

2. IMPLEMENTATION APPROACH

Human factors guidelines cover many aspects of a GUI design, from color selection to screen layout. Style guidelines provide more specific requirements. For instance, the *OSF/Motif Style Guide* requires a menubar to be placed at the top left of a screen. Because so many factors are involved, a GUI design's non-compliance with human factors guidelines or style guidelines can take many forms. In most cases, the problems of non-compliance are independent. Sometimes, however, they are inter-connected. For example, using smaller fonts may cause poorer legibility of colored symbols. Thus, checking non-compliance problems in GUI designs demands consideration of many factors and diverse responses from a compliance-checking program. This requires an effective implementation approach to support the complex control of switching between blocks of code.

The conventional programming approach, used by systems written in C or FORTRAN, fails to provide adequate support, because the code in such a system is executed following a predetermined sequence, and a programmer is required to develop the control structure, which is very complex in the case of non-compliance checking. A more appropriate approach is a rule-based system using forward chaining, because its control mechanism automatically executes the right blocks of code based on input data.

A rule-based system has three components: a data memory, a production memory, and an inference engine[3]. Data memory stores the input data and the current state of knowledge during the problem solving process. Production memory stores the rules which constitute the program. The inference engine executes appropriate rules based on the configuration of data memory. A rule has two parts, a conditional part and an action part. The conditional part describes the data memory configuration for which the rule is appropriate, and the action part gives the instruction for changing the data memory. During code execution, rules that are satisfied by the current content of data memory are collected; one of the collected rules is then selected based on some selection strategy; finally, the selected rule is executed.

When forward chaining is used in rule-based systems, rules are executed whose conditional parts are satisfied by the current content of data memory. The rules executed are determined by input data because the content of data memory is a transformatin of input data. This frees programmers from writing complex control code, thus providing better support for system development.

By using forward chaining, a compliance checker can represent GUI designs as facts in the data memory and can model compliance-checking expertise in the form of rules which match non-compliant patterns. When a GUI design violates the guidelines, the facts in the data memory will reflect the violation, which will match a condition of the compliance-checking rules. This leads the inference engine to execute the appropriate program code.

3. CHIMES ARCHITECTURE

The current CHIMES prototype architecture, illustrated in Figure 1, uses a rule-based-system-with-forward-chaining approach to checking a GUI's compliance with human factors guidelines and style guidelines. There are five modules: design acquisition module, compliance-check controller, knowledge base, advice generator, and user interface.

The purpose of the design acquisition module is to acquire GUI design information. It transforms a design into input data for CHIMES to evaluate. This module consists of two submodules, the automatic design capture submodule and the manual design capture submodule. The automatic design capture submodule acquires GUI design information by reading a design description file, such as a TAE[1] resource file. The manual design capture submodule supplements the automatic design capture submodule by capturing additional design information, through queries to the GUI designer.

The primary function of the compliance-check controller is to initiate a CHIMES evaluation of an acquired design. It takes the design information from the design acquisition module, initializes the knowledge base, converts the information into facts, asserts the facts into the knowledge base, activates compliance checking rules, and initiates the evaluation.

The knowledge base is a key component of the CHIMES prototype system. It contains the rules

[1] TAE is a trademark of the National Aeronautics and Space Administration. The Transportable Applications Environment (TAE) Plus is a user interface development and management environment[10].

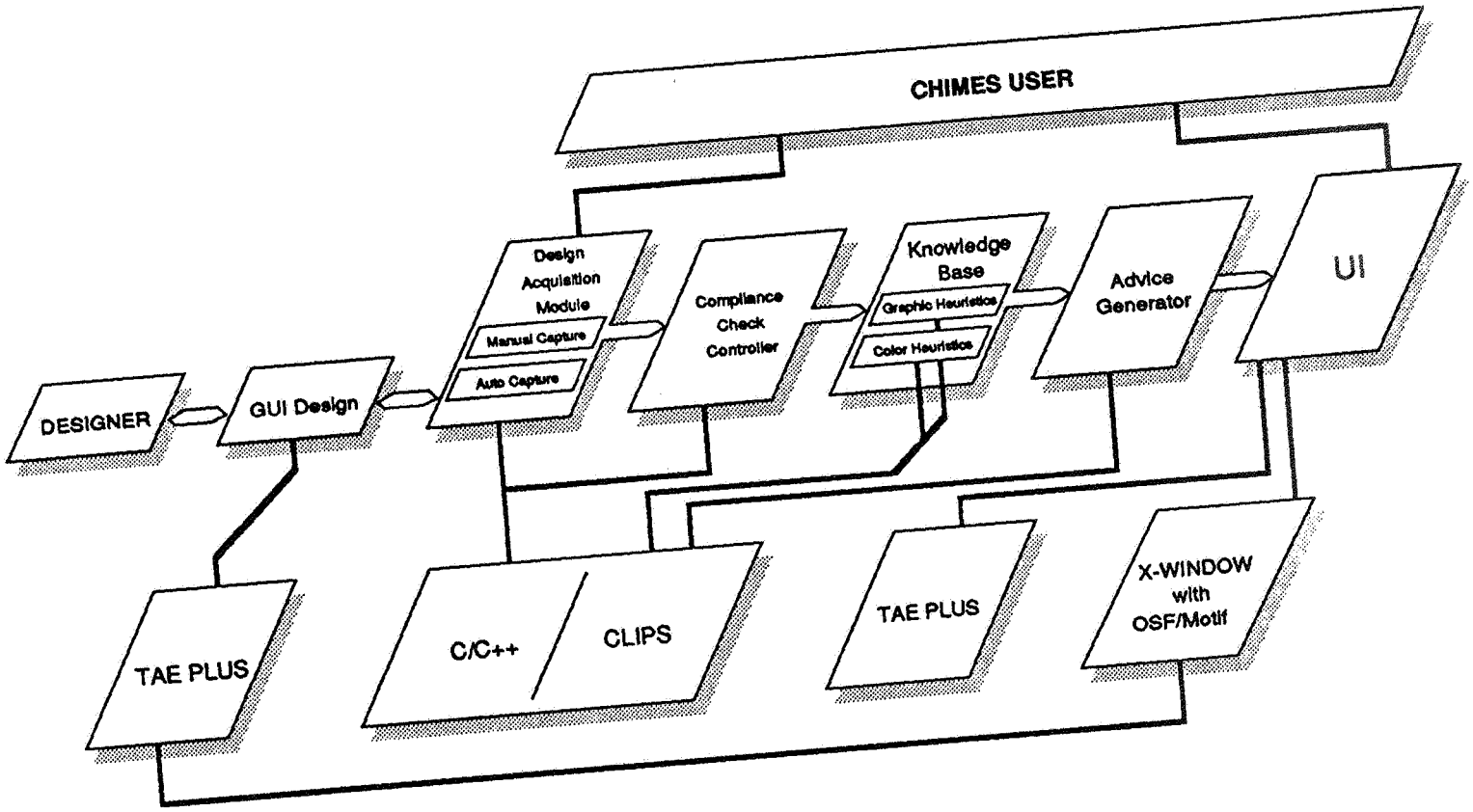

Figure 1: CHIMES Architecture

that model compliance-checking expertise and the facts that represent the user-interface design being evaluated. Upon receiving the evaluation instruction from the compliance-check controller, the knowledge base performs the compliance check and calls the advice generator to generate applicable advice.

The advice generator presents the results of compliance checking to the GUI designer. It prepares advice based on the results of compliance checking. The advice is then displayed in the CHIMES user interface, with an indication of the display element that triggered the advice. This provides advice in the context of the problem.

The user interface allows the GUI designer to initiate compliance checking. It also serves to present the compliance checking results to the GUI designer. For ease of use, all user interactions in the prototype system occur via mouse positioning and clicking.

The knowledge base is implemented by using CLIPS[2][4], which supports development of a rule-based system with forward chaining. The user interface is developed by using TAE Plus with X Window System[3] and OSF/Motif[4]. The design acquisition module, the compliance check controller and the advice generator are written in C and C++.

Interaction of Modules. Compliance checking begins with the design-acquisition module, which currently acquires a user-interface representation from a TAE resource file and transforms it into input data for the CHIMES evaluation. Once the GUI representation has been captured, it is sent to the compliance-check controller, which issues requests for compliance checks to the CHIMES knowledge base. If violations of the guidelines are found, advice is generated on how to improve the design.

4. IMPLEMENTATION OF RULES

Checking a GUI design's compliance with human factors guidelines and style guidelines is performed in the knowledge base, where a design is represented as facts in the data memory, and the compliance-check expertise as rules in the production memory. This section discusses the implementation of compliance-check expertise in the knowledge base.

CHIMES' compliance-checking rules are developed by modeling non-compliance patterns. These patterns take various forms, such as geographic violations; for instance, placing a menubar at the bottom of the screen violates the OSF/Motif style guidelines. Overuse of design elements, for instance, using five fonts in a screen or color mismatchings, for instance, choosing black for the background color and red for the foreground color, can yield poor legibility or visual discomfort[11].

The non-compliance patterns are described in the conditional parts of the rules, which catch the violations. For example, a rule examining menubar position has the following form:

(defrule check-menubar-location
(goal-is check-menubar-location)
(item (name ?name) (type menubar))
(loc (name ?name) (x ?x&:(≠ 0 ?x))
(y ?y&:(≠ 0 ?y)))
⇒
(advice "According to OSF/Motif Style Guide, the menubar should be placed at the top left of the window."))

The conditional part of this rule defines three conditions for the rule to be executed: 1) the knowledge base is checking menubar location; 2) there is a menubar item in the data memory; and 3) the menubar is not located at the upper left corner of the screen. The action part of the rule specifies which piece of advice should be generated if a non-compliant menubar is found. This rule catches any violation of standard menubar placement, which may have the following form in the data memory:

(item (name mb) (type menubar))
(loc (name mb) (x 170) (y 10))

While many of the violation patterns can be represented in the knowledge base directly, color is an exception, because the form of color representation used for electronic display is not appropriate for evaluation by rule-based systems.

Color is produced on an electronic display by additive mixture of three primary colors, red, green

[2]CLIPS (C Language Integrated Production System) is a trademark of the National Aeronautics and Space Administration.

[3]X Window System is a trademark of the Massachusetts Institute of Technology.

[4]OSF/Motif is a trademark of the Open Software Foundation.

and blue. Each primary color is associated with a number to indicate its intensity. For instance, yellow is represented in OSF/Motif as (255, 255, 0), indicating that yellow is the mixture of red and green.

A prominent problem that prevents use of this representation in rule-based systems is the size of the color space, which makes developing rules a formidable task. For example, a color space with each primary color ranging from 0 to 255 consists of more than sixteen million colors. It would require more than sixty trillion rules to cover all the possible color combinations.

To reduce the size of color space, a linguistic color model, the color-naming system (CNS)[2], is used in CHIMES. The CNS model consists of about three hundred color names. Each name contains information on hue, saturation, and lightness. An example of a CNS name is "*very light vivid yellow*," where the saturation modifier *vivid*, and the lightness modifier, *very light*, supply more information on the hue *yellow*.

To use CNS color names in the knowledge base, color representation conversion is needed, because most computer systems use red-green-blue primary colors to represent color. The conversion involves three color models, the red-green-blue (RGB) model[5], the hue-saturation-value (HSV) model[11] and the CNS model. First, a color is translated into the representation of the RGB model, which uses red-green-blue primary colors but with the intensity range between 0 to 1. Second, a color's RGB representation is translated into the HSV representation using a transformation formula[12]. By using the HSV representation, a color's hue is specified in terms of degree on a color ring; saturation and value, or "lightness" which indicates color intensity, are described within the range from 0 to 1. At last, the HSV representation is mapped into a CNS color name by using a mapping matrix. For example, A yellow, represented as (255, 255, 0) in OSF/Motif, can be translated into the RGB representation (1, 1, 0). Then, the RGB representation can be translated into an HSV representation (60, 1, 1), which subsequently can be mapped into a CNS name, *very light vivid yellow*, because the 60th degree hue is mapped to yellow on the color ring, and the high lightness and saturation map to the modifiers, *very light* and *vivid*.

When checking a GUI design's compliance with human factors guidelines on general color usage, color representation conversion is done by the compliance-check controller. It translates all the colors in the design into CNS representation, and stores them in the data memory, where they may trigger compliance-checking rules.

5. FUTURE RESEARCH PLANS

Automated compliance checking is a promising approach to developing a utility in UIMS, such as NASA's TAE Plus. Such a utility can help a GUI designer build more usable user interfaces. It can extract information from the system resources; evaluate the design against guidelines; and provide advice to the GUI designer on how to improve the design. The GUI designer can then use user-interface building capabilities in the UIMS to correct design flaws and produce a better user interface design. The current CHIMES prototype is our first step towards developing such a utility. Future plans include expanding the compliance-checking capability, developing a generic input interface, and creating a style guidelines description language.

The scope of a compliance-checking capability is key to its success. Currently, the CHIMES compliance check is limited to one screen and to the "look," i.e. the visual aspect, of a GUI design. We are going to expand the CHIMES compliance-checking capability so that it can evaluate a design with multiple screens. To lay the foundation for achieving this objective, we plan to develop heuristics on consistency of layout, labeling, and color usage. We also plan to develop heuristics on evaluating the "feel," i.e. the behavioral aspect, of a GUI design, which requires recording a GUI's behavior during execution time.

We plan to expand the CHIMES design-input capability. Currently, CHIMES extracts design data from a TAE resource file. An interface to a standard user interface description language, such as UIL[6], will make CHIMES capable of evaluating GUI designs developed in other UIMS environments.

A style guidelines description language is needed when developing a utility to evaluate GUI designs against various style guidelines. The idea of evaluating GUIs against various guidelines is supported by an existing CHIMES feature, that the designer can change the CHIMES compliance-checking behavior by changing its rules in the knowledge base

without needing to re-compile. With an adequate guidelines description language and a translator that translates guidelines into compliance-checking rules, a compliance-checking utility will be able to check a design's compliance with style guidelines such as those developed for OSF/Motif, Open Look, or other customized toolkits.

References

[1] Bahder, S. A. Murphy, E. D. Sheppard, S. B. & Truszkowski, W. (1990). Developing a user-interface-evaluation tool for space-station-era applications. *Proceedings of the International Symposium on Ground Data Systems for Spacecraft Control* (ESA SP-308, pp. 583-587). Paris: European Space Agency.

[2] Berk, T., Brownston, L., & Kaufman, A. (1982). A new color-naming system for graphics languages. *IEEE Computer Graphics and Applications, 2,* 37-42, 44.

[3] Brownston, L., Farrell. R., Kant. E., & Martin N. (1985). *Programming expert systems in OPS5.* New York: Addison-Wesley.

[4] *CLIPS Reference Manual (Version 5.0),* (1991). Houston, TX: Software Technology Branch, Lyndon B. Johnson Space Center.

[5] Foley, J., van Dam, A., Feiner, S., Hughes, J. (1990). *Computer Graphics.* New York: Addison-Wesley.

[6] Heller, D. (1991). *Motif programming manual.* Sebastopol, CA: O'Reilly & Associates, Inc.

[7] Jiang, J., Murphy, E. D., & Bailin, S. C. (1992). *Comprehensive CHIMES report (Prepared for NASA-Goddard Space Flight Center, Code 522.3).* Rockville, MD: CTA INCORPORATED.

[8] Jiang, J. Murphy, E. D. Bailin, S. C. & Truszkowski, W. (In press). Prototyping a knowledge-based compliance checker for user-interface evaluation in Motif development environments. *Proceedings of the Second Annual World Conference On Motif Application Development and Use.*

[9] Open Software Foundation, (1991). *OSF/Motif style guide (Revision 1.1).* Englewood Cliffs, NJ: Prentice-Hall.

[10] Szczur, M. R. (1992). NASA's TAE Plus: A GUI development tool and application environment. *The X Resource, 2,* 56-77.

[11] Thorell, L. G., & Smith, W. J. (1990). *Using computer color effectively.* Englewood Cliffs, NJ: Prentice-Hall.

[12] Travis, D. (1991). *Effective color displays: Theory and practice.* New York: Academic Press.