Automatically generated User-Interfaces

Ken Lee
Department of Electrical and Computer Engineering
University of Auckland
Private Bag 92019
Auckland, New Zealand
clee207@ec.auckland.ac.nz

ABSTRACT
Developing user interface is often complex and time-consuming, and it is unscalable for the human programmers to create interfaces for each type of devices (PCs, mobile phones and PDAS) and every kind of users. These problems can be corrected with automatic generation of user interfaces. This article reviews four different approaches and generation processes of automatic user interface generation, and findings from different studies. It also looks at the viability of automatic generation of user interfaces. There are four approaches to automatic generation of user interfaces this paper describes: 1) using communicative acts, 2) model-based approach, particularly declarative model-based approach, 3) use of interface design tools, 4) adaptation. Each approach has different generation process which is explained in main section. One of the reviewed studies demonstrated the viability of automatic user interface generation by examining the Personal Universal Controller (PUC) system. Future works are also discusses in this article.

Author Keywords
automatic user interface generation, approach, generation process, viability

ACM Classification Keywords
D.2.2 Approaches and generation process: User interfaces – automatic generation. H.5.2. User Interfaces: Theory and methods.

INTRODUCTION
Research of automatically generating user interfaces has been carried out for more than two decades. There are a number of studies done and works produced in the past. There are several motivations for developing automatic generation of user interface:

1. The increasing diversity of computing devices providing user interface requires multiple user interfaces to be constructed for each application.
2. For certain devices, especially office appliances and consumer electronics, it is economical for manufacturers to include many complex functions but expensive to provide a high-quality user interface
3. There are many users with different backgrounds, goals, and capabilities using today's user interfaces, and each may benefit if his or her interfaces are specifically designed to take individual needs into account. It is impractical for human designers to create a different interface for each individual user, but an automatic interface generator can easily do this.

Therefore, it is necessary to develop automatic user interface generation. There are several approaches and generation processes that have been considered. Using communicative acts [1] is one of the most recent studies in this area. Model-based approach has been considered for quite sometime but it is still in its early stage. Adaptation approach uses adaptive algorithm to generate user interface but its concept it too complicated.

Nichols et al. [6] conducted a user study to demonstrate the usability of interfaces automatically generated by the Personal Universal Controller (PUC) system (figure 7). The results of the study show that the PUC can automatically generate interfaces which are more usable and provide personal consistency.

The remainder of this paper is organized in the following manner. First, it states the general problems of developing user interface and the need for automatic user interface generations. Then it presents four different approaches and generation processes, and what was found in each reviewed studies. At the end of this paper, it describes a user study conducted by Nichols et al.[6] to demonstrate the viability of automatic generation of user interfaces.

PROBLEM
The most common problems of user interfaces are: 1) implementation is expensive and difficult (time & money), 2) different user interfaces offer inconsistent modes of interaction, 3) most of the interfaces are primitive [1,2,3,4]. In developing software applications, an average of 48% of the coding is devoted to the user interface, and about 50% of the implementation time is committed to implement the user interface, i.e. as user interfaces become easier to use, they become harder to create [2]. These problems raise need for automatic user interface generation.

APPROACHES

Communicative Acts
Falb et al. [1] used communicative acts, derived from speech act theory, as an approach to generate user interfaces for multiple devices fully automatically. Computer scientists have found that speech act theory is very useful to

describe interaction and communication, since speech act theory provides a clean and formal view of communication. The use of communicative acts in high-level models of user interfaces allows their creating with less technical knowledge, since such models are easier to provide than user interface code in a usual programming language.

Falb et al. [1] used a meta-model to define high-level user interface model (figure 1). It captures three main concepts used for modelling as well as their relations:

1. the intention capture by a communicative act
2. the propositional content modelled by use of an ontology language
3. the set of interaction sequences modelled with a finite state machine (figure 2) – where each state can have multiple ingoing and outgoing transitions representing segments of the interactions sequences

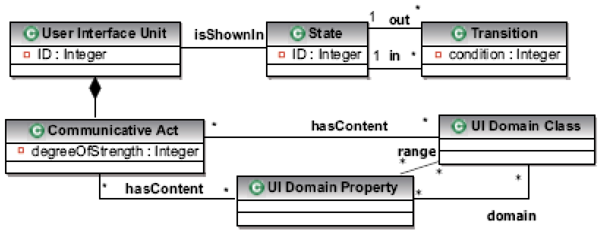

Figure 1. The meta-model of high-level UI models in UML [1]

Figure 2 shows a small selected part of the state machine consisting of four states. A specification of a high-level UI model according to this meta-model provides the essence of a user interface to be generated.

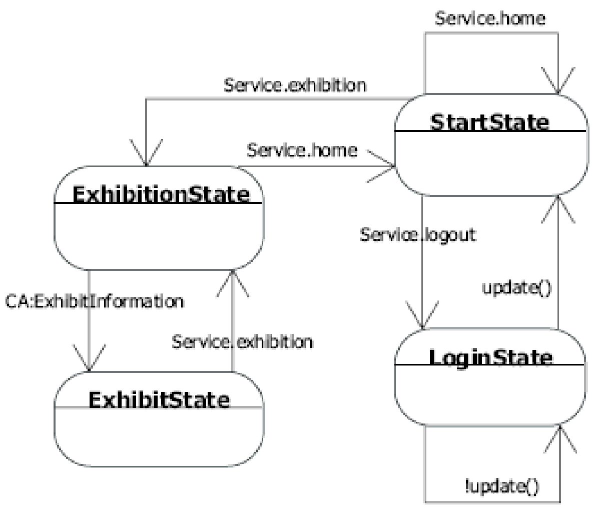

Figure 2. Example of a state machine [1]

Model-based approach

Two of the reviewed studies used model-based approach to guide the generation of the user interfaces [2,3]. There are several model based user interface software tools and the common property (figure 3) of all these tools is that the desired user interface is automatically generated from a specification represented by declarative models [3].

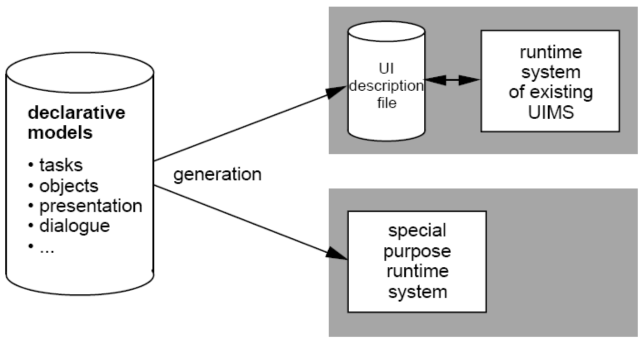

Figure 3. Model-based user interface generation [3]

The model-based approach offers a number of potential benefits over traditional methods of developing user interfaces, e.g., powerful design and runtime tools, support for early conceptual design, consistency and reusability, iterative development, integrated development of user interface and application core.

Schlungbaum and Elwert [2] applied model-based approach in two projects; 1) The Personal Universal Controller (PUC) project applied model-based concepts to automatically generate remote control interfaces for all of the computerized appliances, 2) The Rich Human-Agent Interaction (RHAI) project built a system that allows intelligent agents to communicate with the user. In both project, a high-level, usable, concise XML-based language was designed and rules for generating user interfaces from this language were developed.

Use of interface design tools

Pizano et al. [4] Created an automatic screen layout generator and described a prototype that combines the specification tool and the layout generator with a code generator that produces calls to the GUI toolkit that materialize the interface.

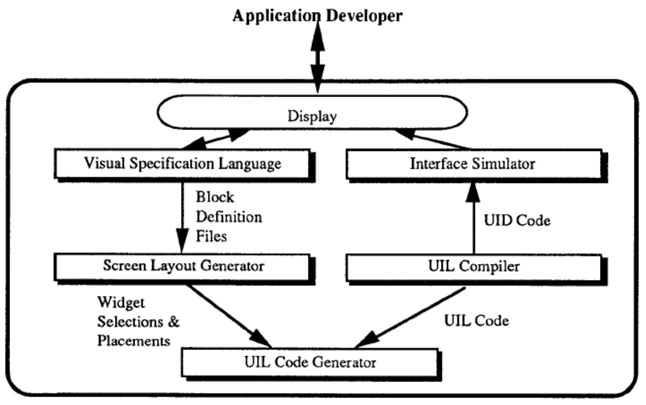

Figure 4. Prototype Architecture [4]

Adaptation

The Adaptive algorithm described in Pizano et al. [5] supplements an existing model-based interface development environment. The adaptive algorithm has three operators for altering the decision tree. The one that reduces the greatest number of errors is selected. The operators are as follows:

1. Change the recommended interators for a given leaf of the tree.

2. Alter the boundary conditions for a branch.
3. Add a branch, and then set the output of the new leaves.

GENERATION PROCESS

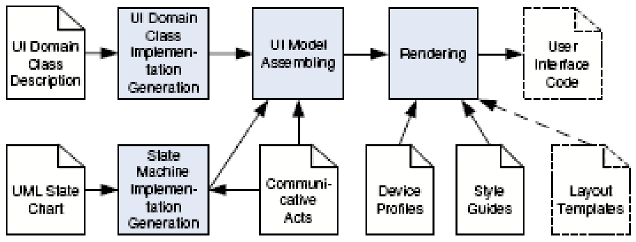

Figure 5. Conceptual Architecture [1]

Figure 6 describes the used generation process that can be divided into four steps as followed.

1. Generation of the UI domain class implementations together with their binding to the actual application functionality.
2. Generation of the finite state machine implementation.
3. Assembly of the UI domain information and the communicative acts according to each state.
4. Rendering of the concrete user interface based on the complete interaction design model

The first three steps transforms the specification into code by applying a typical code generation process that uses an instantiation of meta-model, describe in approach section, as input and applies templates on them. At the fourth step, the rendering process is based on the complete interaction design model and is guided by device profiles, user preferences, application=specific style guides and some heuristics.

The PUC project mentioned in Nichols and Faulring [2] generates a user interface automatically from a functional model that is downloaded from the appliance to the user's device. PUC project uses its own XML-based language and rules to generate user interfaces on different devices such as PocketPCs, Microsoft Smartphones and desktop computers.

The process of user interface generation in Schlungbaum and Elwert [3] consists of four basic steps.

1. High-level dialogue generation: identify all windows of the desired user interface, specify the navigation structure among these windows in the interface, and assign interface objects to each window.
2. Layout generation: Each abstract interaction object is assigned to a concrete Interaction Object (CIO) and all CIOs are placed on their corresponding windows by a layout algorithm that observes interface design guidelines.
3. Low-level dialogue generation: deals with the user interface behaviour on the CIO-level, e.g., disabling of application actions if there no selected object.
4. Layout and design revision: used for participatory design steps on which the end user of the desired user interface is involved.

In Pizano et al. [4], a prototype automatic GUI generator system (figure 4) is used to generate user interfaces. From a developer point of view the automatic GUI generation involves the following steps:

1. invoking the visual specification tool
2. selecting the target database and loading its schema
3. using the schema editor to generate the ASD
4. instructing the system to interpret the ASD and generate the GUI
5. reviewing the resulting interface

In Eisenstein and Puerta[5], a range of tool is provided in order to handle each stage in the interface development cycle.

1. A knowledge elicitation system called U-TEL helps the user of the interface develop models of the interface's data and task structures.
2. The interface designer uses model editors to create relations between the more abstract elements in the data and task structures and the more concrete elements that describe the actual look and feel of the interface.
3. MOBILE, a layout tool that can be configured to reflect the decisions made at previous stages, is provided.

TIMM, The Interface Model Mapper, is used to assist designers in the generation of mapping between various formal elements at different levels of abstraction. A decision tree (figure 5) is used to perform the automatic mappings. A decision tree defines a procedure for classifying cases into groups based on discriminants.

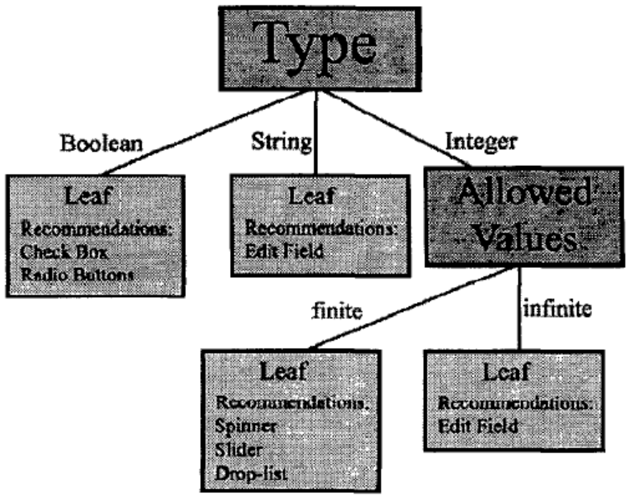
Type
Boolean
String
Integer
Leaf Recommendations: Check Box Radio Buttons
Leaf Recommendations: Edit Field
Allowed Values
finite
infinite
Leaf Recommendations: Spinner Slider Drop-list
Leaf Recommendations: Edit Field

Figure 6. A simple Decision Tree for Interactor Selection [5]

FINDINGS

Falb et al. [1] performed subjective evaluation. They found that with regard to the generated user interface itself, a usability problem has been noted with some embedding. They also found some issues with inherent to the device: a PDA has a small screen requiring both hands for it use and it is not ideally suited for elderly people for several reasons. In general, the usability of the user interfaces generated was assessed informally as good.

An adaptive system for automated user interface design will benefit both designers and interface-design researchers [5]. Designers will benefit in at least three ways as following:

1. User-interface design software will adapt to accommodate their stylistic preferences. In the case of individual idiosyncrasies, designers can trust that the software will take their preferences into account. Where there are whole schools of thought on design-e.g., within a single software company-adapted versions of the interface design software can be distributed.
2. Designers will find it easier to explain their stylistic preferences to others, since the adaptive algorithm will extract a formal description of that style.
3. Technological developments in user-interface design can be accommodated by existing design software without the need for updates or patches. If, for example, a new user-interface widget is introduced, the automatic design algorithm can learn to handle it by observing the designer's behavior

Researchers, who can discover new information about the way designers make decisions, will also benefit from adaptation by observing the results of using the adaptive algorithm. Adaptation will serve as a formal methodology that will help researchers to develop and refine general aspects of a theory of user interface design.

VIABILITY

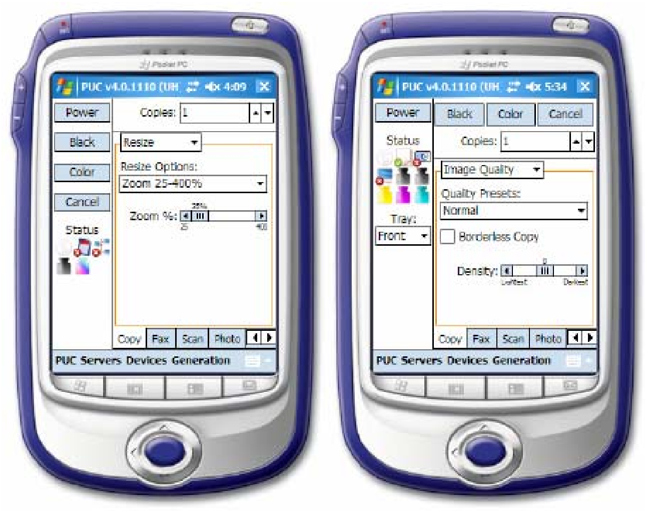

Figure 7. PocketPC interfaces generated by the PUC [6]

Nichols et al. [6] presented a user study that examines the usability of interfaces automatically generated by the Personal Universal Controller (PUC) system (figure 7). The study was carried out as follows:

1. showing that automatic generation can improve usability by moving interfaces that are constrained by cost and poor interaction primitives to another device with better interactive capabilities: subjects were twice as fast and four times as successful at completing tasks with automatically generated interfaces on a PocketPC device as with the actual appliance interfaces.
2. showing that an automatic generator can improve usability by automatically ensuring that new interfaces are generated to be consistent with users' previous experience: subjects were also twice as fast using interfaces consistent with their experiences as compared to normally generated interfaces.

The results show that:

1. users perform faster using the PUC and Uniform interfaces as compared to the printers' built-in interfaces.
2. users perform faster using the Uniform interfaces as compared to the PU interfaces.
3. PUC can improve usability by moving appliance interfaces to another platform with improved interaction primitives

CONCLUSIONS

Developing user interfaces is often hard, error-prone and costly. It is also very difficult to develop interfaces for different devices and users. These problems motivate the creation of automatic user interface generation. The study in this area has been carried out for nearly two decades and there are a number of approaches (communicative acts, model-based approach, use of interface design tools, adaptive approach) and generation processes developed. A user test was conducted by Nichols et al.[6] to show the viability of generation of user interface automatically.

FUTURE WORK

Nichols and Faulring [2] addressed three challenges for future user interface tools that incorporate automatic generation.

1. finding more domains where automatic generation can be applied successfully.
2. Improving modelling languages, which can be made easier to author within a specific domain.
3. Finding novel ways of using models from multiple sources to create customised user interfaces.

Pizano et al. [4] discussed the next step in the project as to development of an automatic code generator capable of producing the call-backs needed to convert the GUIs currently created with the prototype (figure 4) into complete applications.

Eisenstein and Puerta [5] talked about three possible future works.

1. the prospect of incorporating user advice to improve the applicability of the adaptation algorithm
2. Applying the methodology described in Eisenstein and Puerta [5] to other aspects of model-based user-interface design: dialog layout and application structure.
3. Applying the methodology to a variety of design problems outside of user interface design.

REFERENCES

1. Falb, J., Popp, R., Rock, T., Jelinek, H., Arnautovic, E., Kaindl, H. (2007). fully-automatic generation of user interfaces for multiple devices from a high-level model based on communicative acts. *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference* (26-26) Waikoloa, HI: IEEE

2. Nichols, J., Faulring, A., Automatic Interface Generation and Future User Interface Tools. In Proc. CHI2005, ACM Press (2005).

3. Schlungbaum, E., Elwert, T(1996)., Automatic User Interface Generation from Declarative Models. *Proceedings CADUI'96: Computer-Aided Design of User Interfaces*, 3-18.

4. Pizano, A., Shirota, Y., Iizawa. A. (1993). Automatic generation of graphical user interfaces for interactive database applications. *Proceedings of the second international conference on Information and knowledge management* (pp 344-355). Washington DC, United States: ACM

5. Eisenstein. J., Puerta. A. (2000). Adaptation in automated user-interface design. *Proceedings of the 5th international conference on Intelligent user interfaces.* (pp 74-81). New Orleans, Louisiana, United States: ACM

6. Nichols. J., Chau. D.H., Myers. B.A (2007). Demonstrating the Viability of Automatically Generated User Interfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems.* (pp 1283-1292). San Jose, California, USA: ACM