E-Assist II: A platform to design and evaluate soft-keyboards

Bruno Merlin
Universidade Federal do Pará
Tv. Pe. Antônio Franco, 2617
68400-000, Cametá, PA, Brasil
brunomerlin@ufpa.br

Mathieu Raynal
IRIT – ELIPSE team
University of Toulouse
31062 Toulouse cedex, France
mathieu.raynal@irit.fr

Heleno Fülber
Universidade Federal do Pará
Tv. Pe. Antônio Franco, 2617
68400-000, Cametá, PA, Brasil
fulber@ufpa.br

CHI'12, May 5–10, 2012, Austin, Texas, USA.
Workshop on Designing and Evaluating Text Entry Methods

Abstract
E-Assist II is a design and evaluation platform to help researchers and clinicians create new soft keyboards and to evaluate new or existing soft ones. The platform proposes an SDK and a simple XML language to develop complex soft keyboards. It also provides a set of tools to perform theoretical and experimental evaluations.

Keywords
Soft Keyboard, character prediction, entry speed, accuracy

ACM Classification Keywords
H5.2. Information interfaces and presentation (e.g., HCI): User Interfaces – evaluation/methodology.

General Terms
Design, Experimentation

Introduction
Soft keyboards were initially designed to help upper-limb motor impaired users to interact with computers. However, a new generation of touch screen mobile devices generalized and popularized their use. Research in the field of soft keyboards has been very active during the past years and has generated a high number of new alternatives to the traditional mini-QWERTY soft keyboard.

Some years ago, soft keyboards were designed as simple key layouts that imitated physical keyboards. The use of touch screens enabled new interaction techniques based on gestures or continuous strokes (e.g., MessagEase [5] and ShapeWriter [13]). At the same time, soft keyboards have become complex interactive systems. For example, the keyboard layout can be dynamically modified according to the input context [3]. More commonly, keyboards with completion lists provide additional keys that display the most probable word entries based on the previous input characters.

Although soft keyboards are used by HCI researchers, they are also used by ergonomists, clinicians studying motor or mental impairment, and psychologists studying with reading and learning disabilities. Like HCI researchers, clinicians want to evaluate soft keyboards and their usages. However, their special requirements are rarely considered when designing new soft keyboards. Thus, they could also benefit from tools to design and implement new soft keyboards.

Despite differences in soft keyboard designs, they all share several common features and functions. Our primary purpose was to identify the common elements between every keyboard, to implement them as a software library, and to propose a simplified XML-based language and interface. In particular, this language must allow users to design complex soft keyboards without programming skills. Our secondary aim was to build a complete design and evaluation environment for PC, mobile phone, and internet use.

We introduce a rapid state-of-the-art tool for the design and evaluation of soft keyboards. We then present the main features of our design and evaluation platform: E-Assist II.

Related work

In the literature, instrumentation of keyboard design, experimental evaluation, and theoretical evaluation are all tackled independently.

Instrumentation of theoretical evaluation, such as TnToolkit [1] or the use of finite automata [8], targets a short subset of specific keyboards. Keyboard design utilities, such as SoKeyTo [10], tackle the geometric aspects of the keyboard, but they do not manage interaction design or dynamic aspects.

Several works tackle the instrumentation of experimentation. In addition to E-Assiste [7] (the basis of the present work), tools exist to organize experimentation, collect the results, and perform statistical analysis of the result [3], [12]. However, these platforms do not enable experiment management via the internet.

E-Assist II

The aim of the platform E-Assist II was to create an environment for the design of keyboards and their evaluation using theoretical (i.e., predictive models) or experimental methods. It was implemented in Java with the library IntNovate (www.intnovate.org) to work on different platforms, including mobile devices and web browsers.

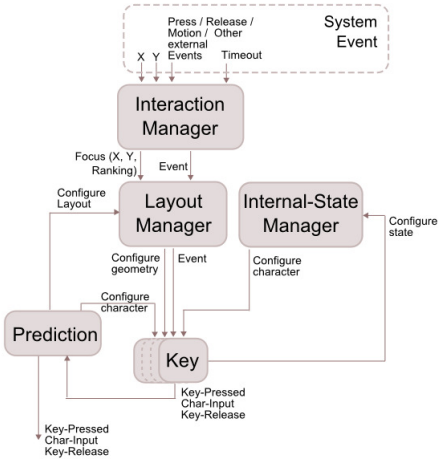

Figure 1: Keyboard model

Keyspec: a language for keyboard design

The platform is based on a keyboard specification language: keyspec. We extract a generic model for keyboard behavior from the study of existing keyboards (cf. Fig. 1). Every element of the model has been implemented as a high configurable component. An XML language enables users to instantiate, configure, and aggregate the components.

The language proposes several predefined interaction techniques. New interactions can be implemented by combining existing interactions.

Dynamic changes of the keyboard are managed by the specification of mathematical constraints relative to prediction system results, pointer device position, last key pressed position, key geometry, and user variables. To manage complex dynamic changes that cannot be specified by constraints (such as spreadkey [3]), the language enables overwriting component feature by integrating Java code.

Although some keyboards such as Dasher [11] or Virhkey [1] cannot be handled by our language, this first iteration of our model and language enabled us to handle the large majority of keyboards found in the literature. Examples at: www.intnovate.org/keyboards.

Keyboards described with keyspec are implemented as Java applications and may be integrated into E-Assist II, accessibility tools for desktop computers and mobile phones, or into a Java swing panel (JPanel).

Theoretical evaluation

The theoretical evaluations are based on motor and cognitive models such as Fitts' law, Hick Hyman law, etc. They consist of measuring the time to search for and target a desired key. The evaluations depend on language-specific character frequency and co-occurrence.

To perform these theoretical evaluations, keyspec provides a tag to specify mathematical relationships (models) to calculate the time to look for and input a character. By default, we use the upper/lower bounds model [9]. Thus, the model may be adapted for different interaction techniques. Moreover, different models can be used to represent access to different characters (e.g., if a key represents more than one character).

A simulation tool performs evaluations as a function of the model specified. The simulation tool can calculate the input time with three measurement strategies: the time to input every digraph weighted by the frequency of the digraph in a language; the time to input a text; the time to input a dictionary weighted by the frequency of the word in a language. The last two strategies evaluate dynamic keyboard and/or prediction system efficiency.

Experimental evaluations

E-Assist II provides an environment to organize, perform, and analyze experimentation of soft keyboards.

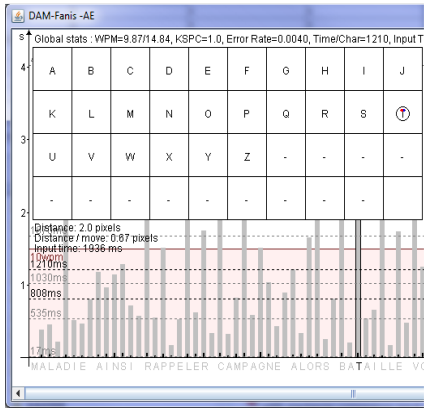

Figure 2: Exercise analysis

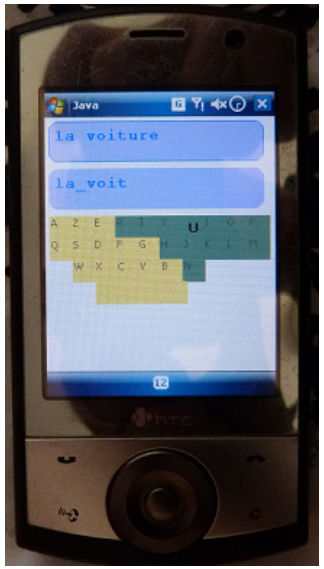

Figure 3: TinyEAssist

Experimentation protocols are described in XML. This allows users to easily manage different groups of participants, each with specific characteristics (such as age range or impairment). It also facilitates organization of instructions to participants, experiment tasks, and delays between exercises. The experimentation may be multi-session.

The analysis tools enable users to visualize the main statistical results (time and error averages, standard deviations, and variances) and generate data for other statistic software (e.g., Statistica). In addition, it enables users to thoroughly analyze what occurred during every exercise (cf. Figure 2).

Mobile & Inline version

TinyEAssist, a light version of E-Assist II, allows experimentation with mobile devices (cf. Figure 3). Also, the experimentation may be done over the internet to facilitate studies involving many sessions, many users, or motor impaired users. See www.intnovate.org/expes.

Conclusion and future work

E-Assist II is a powerful platform to design and evaluate keyboards. However, its efficient use still requires some developer skills because the keyboards and the experimentation protocol are generated in an XML language. We are currently developing a user interface for practitioners with no programming skills.

References

[1] Castellucci, S. J., & MacKenzie, I. S. (2009). TnToolkit: A design and analysis tool for ambiguous, QWERTY, and on-screen keypads. In *Proceedings of* EICS 2009, pp. 55-60 New York: ACM

[2] Martin, B. VirHKey: a VIRtual Hyperbolic KEYboard with gesture interaction and visual feedback for mobile devices, In Proc. of MobileHCI '05, Salzburg, Austria.

[3] Martin, B., Isokoski, P., Jayet, F., and Schang, T. 2009. Performance of finger-operated soft keyboard with and without offset zoom on the pressed key. In *Proceedings of* Mobility '09. ACM, New York, NY, 1-8.

[4] Merlin B. and Raynal, M. Evaluation of SpreadKey system with motor impaired users. In *Proc.* ICCHP'10.

[5] Nesbat, S. B. (2003). A System for Fast, Full-Text Entry for Small Electronic Devices Proceedings of ICMI 2003 (ACM-sponsored), Vancouver.

[6] Raynal, M. and Vigouroux, N. KeyGlasses: Semitransparent keys to optimize text input on virtual keyboard. In Proc. AAATE 2005, IOS Press (2005).

[7] Raynal, M., Maubert, S., Vigouroux, N., Vella, F., Magnien, L. E-ASSISTE: A Platform Allowing Evaluation of Text Input Systems, UAHCI'05, Las Vegas (U.S.A).

[8] Sandnes, F. E. (2005). Evaluating mobile text entry strategies with finite state automata. In *Proceedings of* MobileHCI '05, vol. 111. ACM, New York, NY, 115-121.

[9] Soukoreff, R.W., MacKenzie, I.S. Theoretical upper and lower bounds on typing speeds using a stylus and soft keyboard. *Behaviour & Information Tech.*, 1995.

[10] Vella, F., Vigouroux, N., Truillet, P. (2004). Environnement de conception de clavier virtuel: SOKEYTO. In Proceedings of IHM 2004. Caen, France.

[11] Ward, J. D., Blackwell, A. F., MacKay, D. J. C. Dasher- a data entry interface using continuous gesture and language models. In Proc. of UIST'00, 2000.

[12] Wobbrock, J. O. and Myers, B. A. 2006a. Analyzing the Input Stream for Character-Level Errors in Unconstrained Text Entry Evaluations. Transactions on Computer-Human Interaction 13, 4, 458–489.

[13] Zhai, S., Kristensson, P., Gong, P., Greiner, M., Peng, S. A., Liu, L. M., and Dunnigan, A. 2009. Shapewriter on the iphone: from the laboratory to the real world. In Proceedings of CHI '09. ACM, New York.