An Editor for the Rapid Prototyping of EXPRESS-G Models

R. Zhao, W. Mueller, H.-J. Kaufmann, Th. Kern, F. Buijs
Cadlab, Fuerstenalle 11, 33102 Paderborn, Germany

1 Introduction

The new generation of "notepad" computers as well as electronic whiteboards open a new dimension for designing user interfaces for engineering software. The essential component that makes such computers attractive is the so-called "paper-like" interface in which the user can use handsketches and work in a similar way as (s)he does with paper and pen. See Figure 1 for a typical HW configuration.

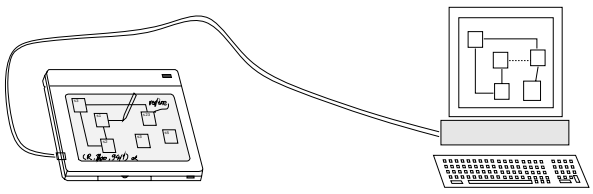

Figure 1: An Integrated Environment with Notepad and Graphical Workstation

A gesture-based interface is one in which freehand drawings are used to indicate commands, scope, and parameters. Such freehand drawings are called gestures. The main advantage of using gestures is that a single gesture combines both the command name and required parameters. Gestural interfaces are appropriate for pen-based computers because a gesture can be easily sketched with an electronic pen. Gestures may be used for common operations across many application domains, for example, a "X"-Gesture for the "delete" command [6]. For EXPRESS–G an additional set of application dependent gestures has be defined.

2 A Gesture-based EXPRESS-G Editor

We have developed an object-oriented software architecture, Handi (HANdsketch-based DIagram editing), for reducing the efforts required by building of gesture-based diagram editors [2]. Handi provides concepts and mechanisms for on-line recognizing handsketches and for specifying gestures as editing commands. The basic idea of Handi is to encapsulate

common characteristics of handsketch-based diagram editors into classes by using object-oriented methodology. One of the key issue of Handi is to build Handi on top of a general editor framework by re-using the general graphical editing functionality. Based on Handi, the key design issues for building a handsketch-based editor are gesture specification and gesture recognition [2]. One problem in building the handsketch-based EXPRESS-G editor is that there are many symbols which look very similar. The entities are all variations of rectangles and the relations are lines with different shapes (solid, dashed, thin, and thick). It is hard to draw such symbols with an electronic pen, specially drawing dashed lines. Further, the recognition of such symbols is difficult. We have defined a gesture set, which can be used to edit EXPRESS-G diagrams. In this gesture set, we use handwritten characters as gestures for creating simple data type symbols. For example, use handwritten "S" for creating a string type, "B" for boolean, etc. See Figure 2 for the EXPRESS-G gesture set. Note here, that we have chosen not to support optional attributes at the gesture level since we haven't found a third intuitive gesture for dashed lines. Nevertheless, this is not a limitation for the EXPRESS-G tool since required attributes can be changed to optional ones in a further refinement when using the non–gestural editor.

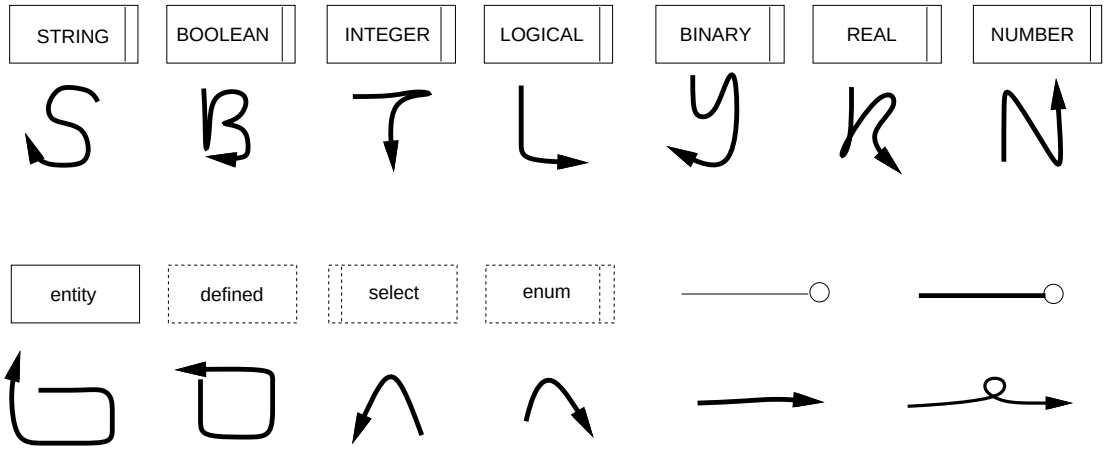

Figure 2: Gestures for creating EXPRESS-G Diagrams

By varying the drawing directions, a handdrawn rectangle can be used as the gestures for creating an entity object or a defined type. One of the characteristics of the EXPRESS-G language is that there are many variants of rectangle symbols used for different data types.

Alternatively, the gestural interface can be combined with the menu interfaces (typed gesture specifications). A typed gesture specifies the basic symbol class, a menu allows the user to make further qualifications. The EXPRESS-G diagram is a box-line based diagram language. Thus, for this alternative approach, it is natural to define two different gestures,

a rectangle gesture for all rectangle like symbols and a line gesture for all "connection like" gestures. Then, the symbol class "box" can be further qualified by "entity". There are two variants of typed gesture specifications (cf. Figure 3). On the one hand, the user may sketch a rectangle when (s)he wants to create a data type symbol. The Handi system can recognize this and maps an option menu for making selections. We call this post-typing since the type of the rectangle gesture is specified after the gesture is drawn. In case that the user creates several symbols of the same type one after another, it is perturbing to make the same selection each time in the same menu. For pre-typing, the user firstly selects the type of the symbol and then sketches the gesture.

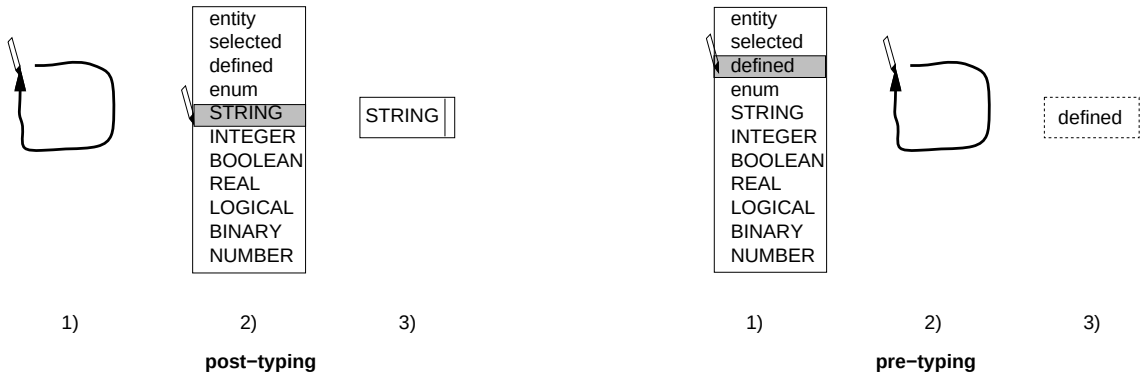

Figure 3: Typed gesture specification

3 Implementation

Currently, Handi has two C++ implementations, one on the top of Unidraw [3] and one on top of our editor toolkit EOS, both provide a comprehensive object-oriented editor framework. The latter one is available for X11R4/5 and MS Windows. We have an EXPRESS-G editor with full functionality on a PC and on a graphical workstation, the EXPRESS-G editor EXPREME. For more information on EXPREME please contact expreme@cadlab.de. Due to our common editor framework EOS it possible that the same Editor with different user interfaces can be implemented both on notepad computers and graphical workstations. In order to make use of the editor on a normal notebook computer the implementation supports gestures drawn by a mouse. The handsketch-based EXPRESS-G editor is based on the same data structures as our full EXPRESS-G editor EXPREME. Therefore, the notepad version can be used for a first conceptual sketching. The sketched model can then be loaded into the full graphical EXPRESS-G editor on a workstation for adding details and for final modifications without any further problems.

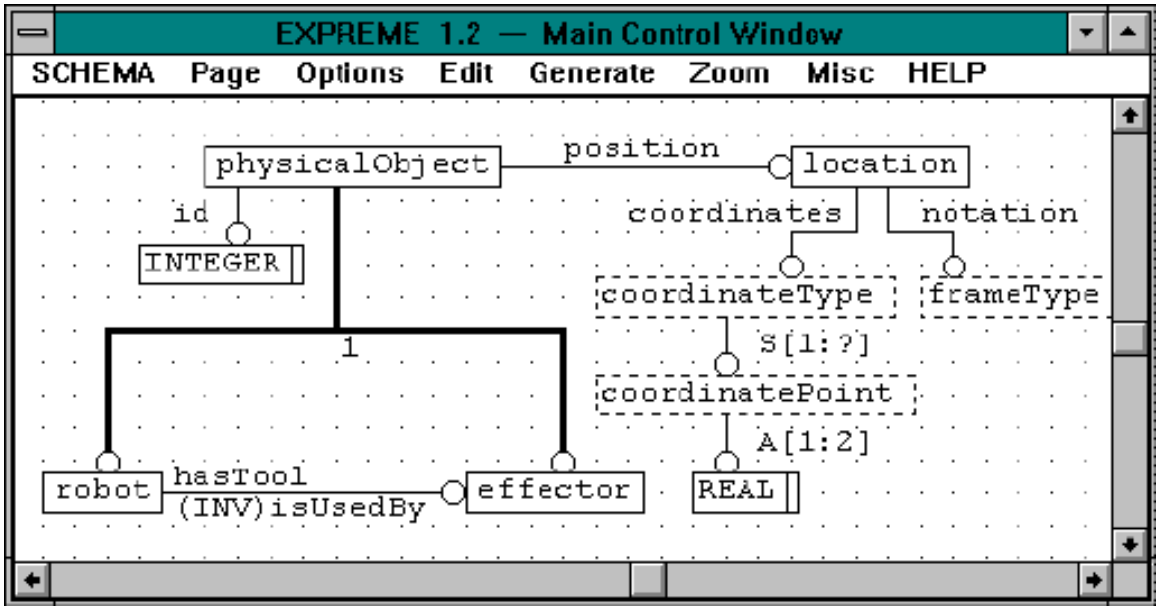

Figure 4: EXPREME V 1.2 Screen Dump

4 Conclusion

Our experiences of integrating gestural interfaces into the EXPRESS/STEP engineering environment indicates a feasible way for using pen-based computers in engineering environments in order to provide tools for the rapid prototyping of EXPRESS-G models. We believe that the user interface has to be adapted to the application and the working environment. For conceptual sketching the gestural interface is a reasonable alternative input technique, but for other tasks the conventional graphical user interface may be more appropriate. Therefore, the key point is how to combine different user interfaces and systems into one environment.

References

[1] ISO/TC184/SC4, Geneve, Switzerland. *EXPRESS Language Reference Manual - ISO/IS 10303-11*, December, 1994.

[2] Rui Zhao. Incremental recognition in gesture-based and syntax-directed diagram editors. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (InterCHI'93)*, pages 95–100, Amsterdam, 1993.

[3] John M. Vlissides and Mark A. Linton. Unidraw: A framework for building domain-specific graphical editors. *ACM Transactions on Information Systems*, 8(3):237–268, July 1990.

[4] Hermann-Josef Kaufmann, Thomas Kern, and Rui Zhao. Detailed functional specification EOS 1.0. Technical report, Cadlab, BT-HCI, 1994.

[5] P. Morrel-Samuels. Clarifying the distinction between lexical and gestural commands. *Internal Journal of Man-Machine Studies*, 32:581–590, 1990.

[6] Dean Rubine. Specifying gestures by example. *ACM SIGGRAPH'91, Computer Graphics*, 25(4), 1991.

Biography

Rui Zhao. Dr. Zhao received his diploma in computer science from the University of Dortmund in 1988. Since then, he works at Cadlab. He received the Ph.D. degree from the Paderborn University in 1992. Since 1992, he is the project leader of the Cadlab group working on Human Computer Interaction. Special interests are handsketch–based editing and graphical user interfaces. Dr. Zhao co–authored a German book on X11 and Motif. He recived the 1993 International software award for graphical data processing for his implementation of Handi.

Wolfgang Mueller. Mr. Mueller received his diploma in computer science from the Paderborn University of Paderborn in 1989. Since 1989, he is with Cadlab. Special interests are EXPRESS, process modeling, and general hardware design. Mr. Mueller authored various papers on EXPRESS and EXPRESS–P.

Thomas Kern. Mr. Kern received his diploma in electrical engineering from the FH Darmstadt in 1985. In 1985, he joined Nixdorf Computer AG. Since 1988, he is with Cadlab. Special interests are graphical user interfaces and user interface management systems. Mr. Kern co–authored a German book on X11 and Motif.

Herman–Josef Kaufmann. Dr. Kaufmann received his diploma in computer science from Paderborn University in 1986. Since 1986, he is with Cadlab. He received the Ph.D. degree from the Paderborn University in 1994. Special interests are graphical editors and graphical user interfaces. Dr. Kaufmann co–authored a German book on X11 and Motif.

Frank Buijs. Dr. Buijs received his M.S. degree in computer science from the University of Twente in 1988. Since 1988, he works at Cadlab. From 1988 until 1992 he worked on his Ph.D. research. He received the Ph.D. degree from the Paderborn University in 1994. Since 1992, he is the project leader of the Cadlab group working on STEP and EXPRESS.