DRAFTER

Cécile Paris* and Keith Vander Linden†
Information Technology Research Institute
University of Brighton
Brighton BN2 4AT, UK
email: {clp,knvl}@itri.brighton.ac.uk
Web Page: http://www.itri.brighton.ac.uk/projects/drafter

DRAFTER is an interactive tool designed to assist technical authors in the production of English and French end-user manuals for software systems. Unlike current generation systems, which aim at the automated production of instructions and thus keep the authors out of the loop, Drafter is a support tool intended to be integrated in the technical author's working environment, hopefully automating some of the more tedious aspects of the authors' tasks.

As with any generation system, Drafter requires a semantic knowledge base from which text can be generated. While Drafter obtains as much as it can of this knowledge base automatically from external sources, it also allows the authors to specify the portions that cannot be acquired automatically, and provides for a parallel development of knowledge base and natural language text.

The Drafter architecture is based on a user requirements analysis (Power et al., 1994). As shown in Figure 1, the system contains two main modules:

- **A developer's tool**: This allows technical authors to specify formally the procedures necessary for the user to achieve their goals, thus supporting user-oriented instructions. It also allows them to control the drafting process.
- **The automated drafter**: This comprises two major components: the text planner (or strategic planner) and the tactical generator. The text planner determines the content and structure of the text, and the tactical generator performs the realization of the sentences. The result is English and French drafts of the instructions for the procedures defined so far by the author using the developer's tool.

Underlying the processing components is a *Domain knowledge base*, which is the main repository of information about the domain. This knowledge base is implemented in Loom (MacGregor, 1988).

Walkthrough of the System

In this demo, we illustrate how a technical author would work with DRAFTER. Our example involves defining the procedure for saving a new file in a Microsoft Word-like text editor, and then to generate text for that procedure. It is easiest for the technical writer if the process starts by defining the interface to be documented with some interface building tool. To show the feasibility of this approach, we implemented parts of the text editor's functions in VisualWorks (VisualWorks, 1994), a widely available interface design environment.

Drafter then has facilities for reading the interface definition produced by VisualWorks in Smalltalk, and finding all the objects relevant for the generation of the instructions. It can also infer some of the actions involved in using these

*Starting this Fall, Dr. Paris' address will be CSIRO, Division of Information Technology, Sydney Laboratory, Building E6B, Macquarie University Campus, North Ryde, Sydney, NSW 2113, Australia.

†After September 1, Dr. Vander Linden's address will be Department of Mathematics and Computer Science, Calvin College, Grand Rapids, MI 49546, USA, kvlinden@calvin.edu.

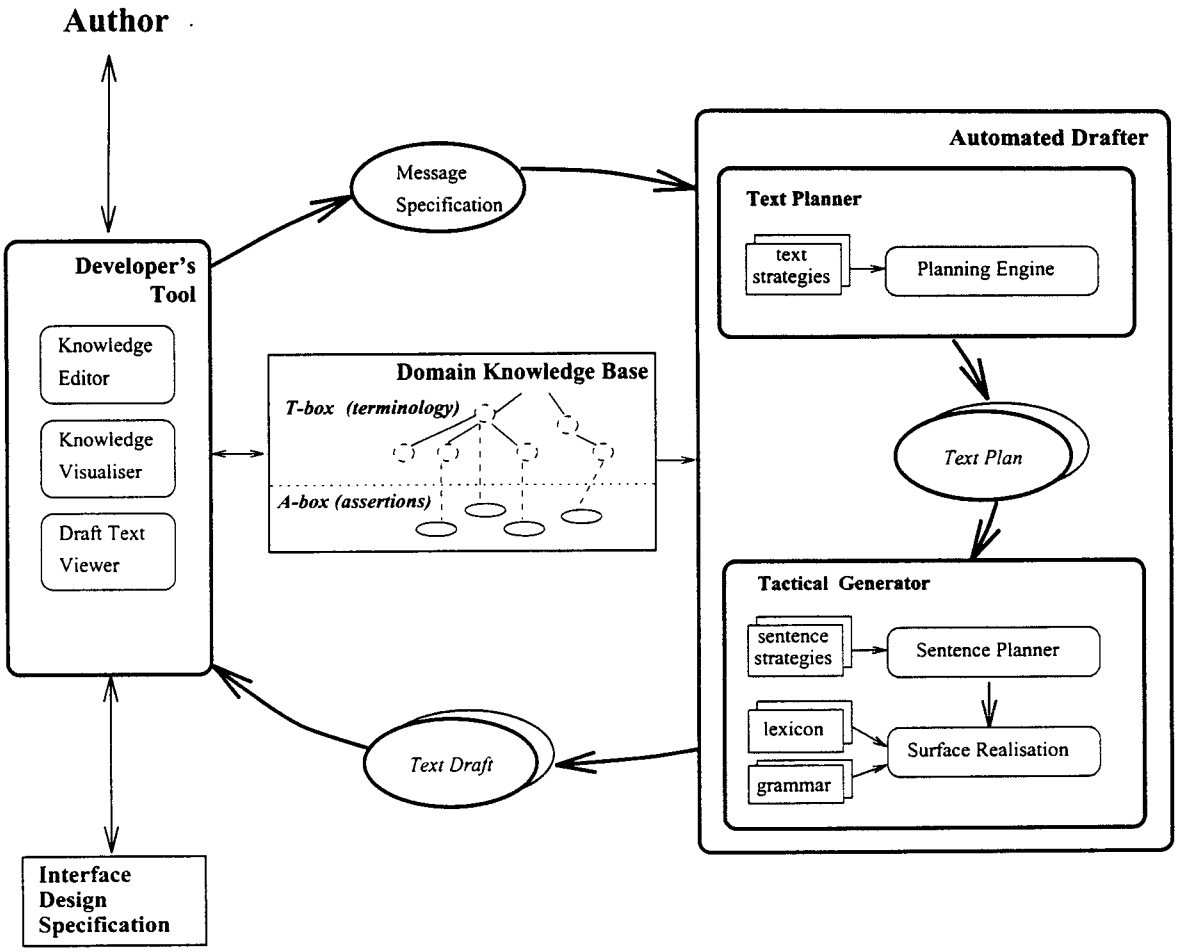

Figure 1: Block Diagram of the Architecture

objects. It uses this information to define a set of object and action entities in the Drafter knowledge base for use in text generation. These actions and objects can then be used by the technical author as building blocks to specify tasks (Paris and Vander Linden, 1996a). Clearly, however, the entities acquired automatically are not all that is needed to document the interface properly. Because of this (and because of the potential for the user to be without a supported interface design tool like VisualWorks), Drafter provides a manual definition facility. This facility is based on an English pseudo-text grammar (Paris and Vander Linden, 1996b), which allows the author to use a relative simple pseudo-text to specify a complex configuration of action and object entities and the relations between them.

Once the action nodes of the graph have been created, or perhaps while they are being created, the author has the ability to link them together using a set of predefined procedural relations: goal, precondition, sub-action, side-effect, warning, and cancellation. This is done in the workspace, with a graphical outlining mechanism. This mechanism allows authors to drag actions from the ACTIONS pane and drop them on the various procedural relation slots in the workspace pane, or, alternatively, to create new actions to fill the slots. The result is a procedural hierarchy such as the one shown in outline form in Figure 2.

In this screen, the WORKSPACE pane contains the procedure being documented in an outline format. The outer box represents the main user goal of saving a document, a goal which is achieved by executing all the actions inside the box. It contains a single method specifying a cancellation

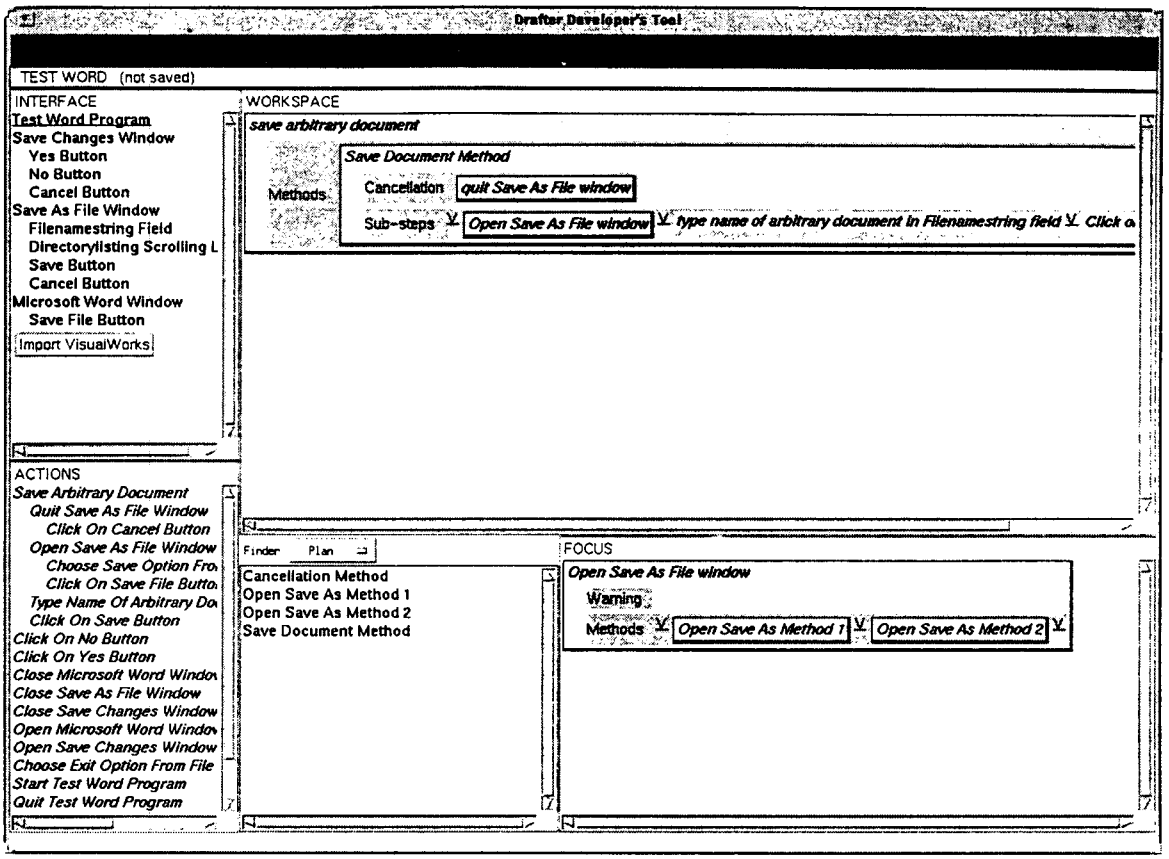

Figure 2: Procedural hierarchy for Saving a file

action (i.e., that the Save-As File window may be closed by performing a particular method), and a set of sub-steps (i.e., opening the Save-As File window, typing the name of the file and clicking the save button).

Once a procedure is defined, the Automated Drafter takes the procedure specified with the Developer's Tool and produces text expressing that procedure. The text generation in Drafter is supported by the reuse (and extension) of three pre-existing tools:

- The Moore and Paris Text Planner (Moore and Paris, 1993);
- The IMAGENE Sentence Planner (Vander Linden and Martin, 1995);
- The KPML tactical generator (Bateman, 1996).

The first two tools operate in sequence, planning, respectively, the high-level rhetorical structure of the text and the low-level grammatical details of the sentences. When this is finished, the KPML generator is called, once for each sentence, to produce the actual text. The text is produced in English and in French. We are largely using the Nigel grammar for English (Mann, 1985), but are developing within the KPML enviroment a grammar for French. One of the texts produced for the Save a File procedure is shown in Figure 3.

Acknowledgements

DRAFTER also involves two industrial partners: Integral Solutions Limited (ISL), a software company specialising in artificial intelligence products, and Praetorius Ltd., a leading translation and technical writing consultancy specialising in software documentation. Other members of the project include Anthony Hartley, Markus Fischer, Lyn Pemberton, Richard Power and Donia Scott.

English:

To Save a Document
1. Open the Save As File window, by choosing the Save option on the file menu or by clicking on the Save File button.
2. Type a name in the Filenamestring field.
3. Click on the Save button.
You can quit the Save As dialog box by clicking on the Cancel button.

French:

Enregistrement d'un document
1. Ouvrir la fenêtre Save As File, en choisissant l'option Save sur le menu File ou en cliquant sur le bouton Save File.
2. Introduire un titre dans la zone de texte Filenamestring.
3. Cliquer sur le bouton Save.
Vous pouvez quitter la fenêtre Save As File en cliquant sur le bouton Cancel.

Figure 3: Generated English and French Drafts

We are also grateful to Jon Barber who assisted in the implementation of this system.

This work is partially supported by the Engineering and Physical Sciences Research Council (EPSRC) Grant J19221, by BC/DAAD ARC Project 293, by the Commission of the European Union Grant LRE-62009, and by the ONR grant N00014-96-1-0465.

References

Bateman, J. A. (1996). KPML Development Environment – multilingual linguistic resources development and sentence generation. Technical report, Institut für Integrierte Publikations- und Informationssysteme (IPSI), GMD, Darmstadt. Release 0.9.

MacGregor, R. (1988). A Deductive Pattern Matcher. In *Proceedings of the 1988 Conference on Artificial Intelligence*, St Paul, MN. American Association of Artificial Intelligence.

Mann, W. C. (1985). An introduction to the Nigel text generation grammar. In Benson, J. D., Freedle, R. O., and Greaves, W. S., editors, *Systemic Perspectives on Discourse*, volume 1, pages 84–95. Ablex.

Moore, J. D. and Paris, C. L. (1993). Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651–694.

Paris, C. and Vander Linden, K. (1996a). Building Knowledge Bases for the Generation of Software Documentation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, Copenhagen, Denmark.

Paris, C. and Vander Linden, K. (1996b). Drafter: An interactive support tool for writing multilingual instructions. *IEEE Computer*. to appear.

Power, R., Pemberton, L., Hartley, A., and Gorman, L. (1994). User requirements analysis. Technical report, ITRI. WP2 Deliverable, Drafter Project IED4/1/5827, financed by the Engineering and Physical Sciences Research Council (EPSRC) Grant J19221.

Vander Linden, K. and Martin, J. H. (1995). Expressing local rhetorical relations in instructional text: A case-study of the purpose relation. *Computational Linguistics*, 21(1):29–57.

VisualWorks (1994). *The VisualWorks Documentation*. ParcPlace Systems, Inc., 999 E. Arques Avenue, Sunnyvale, CA 94086-4593.