Interactive Multimedia Explanation for Equipment Maintenance and Repair

Kathleen McKeown and Steven Feiner

Department of Computer Science
450 Computer Science Building
Columbia University
New York, N.Y. 10027

Introduction

COMET (COordinated Multimedia Explanation Testbed) is an experimental system that generates interactive multimedia explanations of how to operate, maintain, and repair equipment. Our research stresses the dynamic generation of the content and form of all material presented, addressing issues in the generation of text and graphics, and in coordinating text and graphics in an integrated presentation.

COMET contains a static knowledge base describing objects and plans for maintenance and repair, and a dynamic knowledge source for diagnosing failures. A menu interface allows users to request explanations of specific procedures and to specify failure symptoms that will invoke a diagnostic component. The diagnostic component can ask the user to carry out procedures that COMET will explain if requested. In contrast to hypermedia systems that present previously authored material, COMET has underlying models of the user and context that allow each aspect of the explanation generated to be based on the current situation.

In this paper we discuss recent progress on COMET, including the development of an interface for user input, the integration of its individual modules into a working system, and further results in our work on the media coordinator, the text generator, and the graphics generator.

System Overview

COMET consists of the major components illustrated in Fig. 1. On receiving a request for an explanation, the *content planner* uses text plans, or *schemas*, to determine which information should be included from the underlying *knowledge sources* in the explanation. COMET uses three different knowledge sources: a static representation of the domain encoded in LOOM [11], a rule-base learned over time [2], and a detailed geometric knowledge base necessary for the generation of graphics [12]. The content planner produces the full content for the explanation, represented as a hierarchy of logical forms (LFs) [1], which are passed to the *media coordinator*. The media coordinator refines the LFs by adding directives indicating which portions are to be produced by each of a set of media-specific generation systems. The *text generator* and

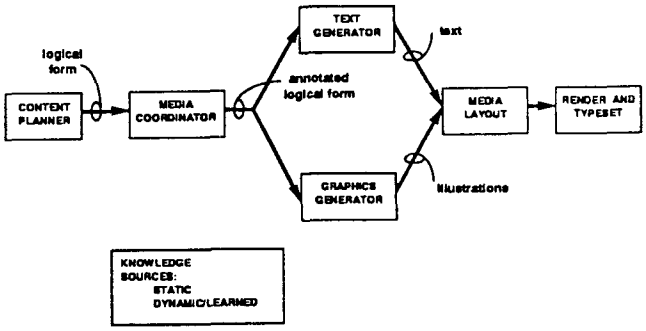

Figure 1: COMET system architecture.

graphics generator each process the same LFs, producing fragments of text and graphics that are keyed to the LFs they instantiate. This output is combined by the *media layout* component, which formats the final presentation for the low-level *rendering and typesetting* software. Much of our work on COMET has been done in a maintenance and repair domain for the US Army AN/PRC-119 portable radio receiver-transmitter [3].

Currently, the system runs in parallel on five Sun and HP machines, one for each of COMET's most computation-intensive modules, which communicate through pipes. A user interacts with COMET through an X11 menu interface, using menus that are created on the fly by the system. At the highest level, the user can choose to request an explanation for an explicit repair procedure directly or can specify that troubleshooting help is needed. When help is requested, the underlying diagnostic system is invoked and the user is asked to specify symptoms of the failure from the menu shown in Fig. 2. In the course of

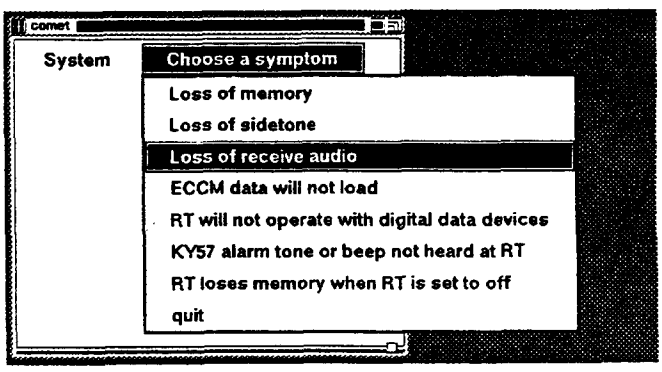

Figure 2: Menu of symptoms.

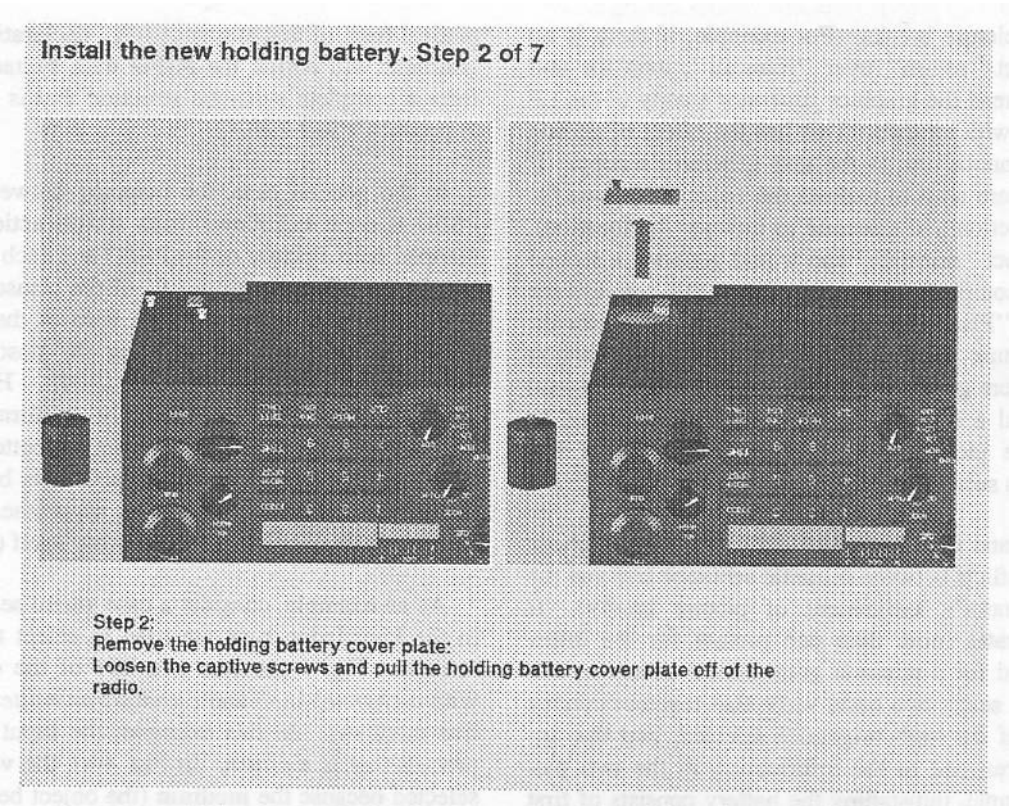

Figure 3: One display from an explanation generated by COMET.

diagnosing the failure, COMET will ask the user to carry out certain troubleshooting procedures. For example, if the user indicates a loss of memory in the radio, COMET will generate a multiple-step test procedure. Each step is shown sequentially on the display. The user can request an explanation of any step, or can move forward or backward in the generated explanation, by using the menu interface. Figure 3 shows one display from an explanation generated by COMET.

Media Coordination

In previous work [6] we focused on three features of our media coordinator: the use of a common content description language by each media-specific generator, allowing goals and information to be mapped to media-specific resources; the ability to make a fine-grained division of information between media; and the ability for information expressed in one medium only to influence the realization of information in the other. In this paper, we describe our recent advances in coordinating sentence breaks with picture breaks.

Informal experiments that we carried out when designing the media coordinator indicated that our subjects strongly prefer sentence breaks to coincide with picture breaks [10]. While more than one sentence may appear with a single picture, there was a strong objection to sentences that run across picture boundaries. For example, in Fig. 3, users would prefer a sentence break to correspond to the two pictures: "Loosen the captive screws." and "Pull the holding battery cover plate off of the radio." Coordinating sentence and picture breaks requires bidirectional interaction between the text and graphics generators since graphical constraints on picture size may sometimes force delimitation of sentences, while grammatical constraints on sentence construction may sometimes control picture size. Our implementation of sentence-picture coordination involves three stages of processing.

In the first stage of processing, text and graphics generators separately annotate their own copies of the LF to indicate minimal sentence and picture break locations. In our current implementation, when the verb for the sentence is selected, the text generator annotates the LF to indicate the grammatical sentence with the smallest number of constituents that can be formed. The lexicon contains the required set of inherent roles for each verb; these are the case roles that must be present to form a grammatical sentence. For example, when the verb "reinstall" is selected, there are two required inherent case roles, the **agent** and the **medium** (note that the agent can be omitted in imperative sentences). Thus, the sentence "Reinstall the primary battery." is perfectly grammatical. However, if the verb "return" is selected, there are three required inherent case roles: **agent**, **medium**, and **to-location**. Thus, while the sentence "Return the primary battery to the radio." is acceptable, "Return the primary battery." is not in this context. The text generator will annotate the LF corresponding to these two sentences differently when the verb is selected. If "reinstall" is selected, the attributes **agent** and **medium** are each annotated with an attribute indicating that it is required. If "return" is selected, the **to-loc** role is also annotated.

In the second stage of processing, if the text generator has a choice of verbs, it will check the graphics generator's

placement of picture breaks. For example, if there is no reason to select "return" over "reinstall", then the text generator will read the graphics generator's copy of the LF by unifying it with its own. This has the effect of adding the graphics annotations to the text generator's copy. If two pictures were used to express the action (e.g., one for the installing action and a second to indicate the location), text would select "reinstall" and would generate a second sentence to accompany the second picture that conveys the location (e.g., "Place it on the radio socket."). However, if a single picture expressing both the installation action and location were generated, then the verb "return" would be selected and a single sentence would be generated to accompany the picture. We are in the process of implementing this second stage.

In the third and final stage, the text generator will check if there are conflicts between minimal sentence size and the graphics generator's assignment of picture breaks. If graphics generates more than one picture for the information required for a minimal grammatical sentence, text will attempt to select two basic verbs that together convey the meaning of the verb originally selected, and that individually correspond to the information in the two pictures. For example, reinstalling the battery consists of first placing it on the radio and then snapping some latches. If each of these steps is portrayed in a separate picture, then text can select the verbs "place" and "select" to convey the compositional meaning of "reinstall" and generate two separate sentences. This stage is also currently under development.

Note that after text and graphics are generated with coordinated breaks, it will be necessary to lay them out so that relationships between corresponding material in different media are clearly visible. Although COMET's current media layout component does not take these relationships into account, we have begun to design a new one that will, building on our previous work on automated layout [7].

Text Generation

One focus in the text generation component has been on selection of appropriate vocabulary for the explanation. We have developed a framework for lexical choice using the Functional Unification Formalism (FUF) [9, 4, 5]. In addition, we have identified how previous discourse and the underlying knowledge sources influence lexical choice and implemented these influences as part of the lexical chooser.

The lexical chooser is part of the text surface generator. It receives its input from the media coordinator and passes its output to the surface generator, which contains COMET's grammar and constructs the grammatical structure of the sentence. As output, the lexical chooser produces a list of partially specified functional descriptions (PSFDs) that are passed as input to the surface generator. Thus, a PSFD is basically a lexicalized LF (using the special feature *lex*) that, in addition, specifies the overall grammatical form of that utterance (e.g., declarative). COMET's grammar will enrich the PSFD with syntactic features to form a complete syntactic structure that is then linearized to produce a sentence.

In the general case, the mapping between a LF and a PSFD is done as follows: each simple action in the LF is mapped onto a clause of the PSFD and each object description in the LF onto a nominal[1] of the clause. The **process** of the action is mapped onto a verb of the clause. Both mappings are made by unifying the description with a *Functional Unification Lexicon (FUL)*. However, while unification in FUF is normally performed top-down, unification with a FUL is performed bottom-up, starting with the most embedded sub-LFs. This is because the lexicalizations of the process roles sometimes constrain the possible lexicalizations of the process itself (i.e., the verb).

As an example, consider a case where semantic features in the knowledge base are used to select the verb of the sentence. Fig. 4 presents two LFs of the concept *c-turn*, with *c-channel-knob* and *c-radio-transmitter* as the respective mediums. In this example, the input LF contains a process that is a *c-turn*. In Fig. 4(a), the verb "to set" is selected because the **medium** (the object being turned) has discrete settings, as is the case for the *c-channel-knob*. In Fig. 4(b), the medium does not have discrete settings, as is the case for the *c-radio-transmitter*, and the verb "to turn" is selected.

For each example, the lexicon is first accessed to lexicalize the object concepts embedded in the roles of the top-level LF: *c-channel-knob* by "channel knob", *c-radio-transmitter* by "radio", *c-position-1* by 'position 1' and *c-front-panel* by "front-panel". It is then accessed again to lexicalize the process concept of the top-level LF: *c-turn* by "to set" in Fig. 4(a), where *c-channel-knob* is the medium and by "to turn" in Fig. 4(b), where *c-radio-transmitter* is the medium. In selecting the verb, the lexical chooser invokes a function that accesses the knowledge base to check whether the medium is an instance of a discrete knob or not. If it is, the verb "to set" is chosen. Otherwise, "to turn" is chosen.

As illustrated in Fig. 5, this lexical choice is implemented by using a special feature of FUF termed CONTROL in the FUL entry for the concept *c-turn*. It allows invocation of an arbitrary LISP predicate during the unification process. Only if this predicate is satisfied will unification of the FD containing the CONTROL pair succeed. In this example, CONTROL is used to have FUL directly query the knowledge base for additional information about the medium of *c-turn*.

COMET's lexical chooser can also choose between words based on context. For example, it will choose the verb "reinstall" or "return" in place of "install" when it

[1] i.e. noun phrase, pronoun or proper noun.

```
((process-concept c-turn)
 (process-type action)
 (mood non-finite)
 (speech-act directive)
 (roles
  ((medium
    ((object-concept c-channel-knob)
     (quantification
      ((definite yes)  (countable yes)
       (ref-obj 1)  (ref-set 1)))
      (ref-mode description)))
   (to-loc
    ((object-concept c-position-1)
     (ref-mode name)))))))
```

(a) LF of "Set the channel knob to position 1."

```
((process-concept c-turn)
 (process-type action)
 (mood non-finite)
 (speech-act directive)
 (roles
  ((medium
    ((object-concept
      c-radio-transmitter)
     (quantification
      ((definite yes) (countable yes)
       (ref-obj 1) (ref-set 1)))
      (ref-mode description)))
    (on-loc
     ((object-concept c-front-panel)
      (quantification
       ((definite yes) (countable yes)
        (ref-obj singular)
        (ref-set singular)))
       (ref-mode description)))))))
```

(b) LF of "Turn the radio onto the front panel."

Figure 4: Two LFs of the same concept with different role values.

instructs the user to install an object that it has previously instructed the user to remove. For each action that has an inverse action, COMET checks whether it has already instructed the user to perform the inverse action in the current explanation. If so, it will select a verb reflecting the inverse. Consider the partial set of instructions for troubleshooting loss of memory in Fig. 6. With no previous discourse, COMET selects the verb "install" to describe the installation for the holding battery. However, after it has instructed the user to "remove" the primary battery and "pull" the battery box away from the radio, COMET selects the verbs "reinstall" and "return" to lexicalize the same installation process.

The use of the unification algorithm for lexical choice is a novel approach that allows for the integration of various types of constraints in a uniform formalism. For example, in COMET, the choice of verb for a process has been constrained simultaneously by its location in the domain hierarchy, by the semantic features of its role, and by the contextual features of the previous discourse. FUF also

```
(((process-type action)
  (ALT
   (((process-concept c-turn)
;; Is medium a type of discrete-knob?
     (ALT
 ;; here the FUL invokes the
 ;; knowledge base, LOOM
      (((control
         ((member  |c|discrete-knob
           (loom::superconcepts
            ( ^ roles
              medium
              object-concept))]
  ;; if |c|discrete-knob is a
  ;; superconcept select 'set'
       (verb
        ((lex 'set')
         (voice-class non-middle)
         (transitive-class transitive)
            )))
  ;;else select 'turn'
      ((verb
        ((lex 'turn')
         (on-loc-prep 'onto')
         (voice-class non-middle)
         (transitive-class transitive)
                                    ]
```

Figure 5: Part of the FUL encoding the choice between two verbs

provides a modular and declarative lexicon that is easily extensible, and ultimately will allow for extensive interaction between the lexical chooser and grammar through a uniform formalism.

Graphics Generation

Work on graphics generation in COMET has concentrated on the development of an approach for generating technical illustrations of 3D objects, embodied in the rule-based graphics generator IBIS (Intent-Based Illustration System) [12]. As in COMET's text generation component, all material is created on the fly, making it possible for the explanation to be customized to the individual user and situation.

Each of IBIS's illustrations is created by an *illustrator*, which designs its illustration to fulfill a set of communicative goals derived from the LF that is presented to it. The illustrator realizes these goals by creating an *illustration* that includes a set of objects to be depicted and their at-

Install the new holding battery.

...

Remove the primary battery: ... **pull** the battery box away from the radio.

...

Reinstall the primary battery: *Return* the primary battery to the radio, **reinstall** the battery box, and snap the latches.

Figure 6: Influence of previous discourse on verb choice

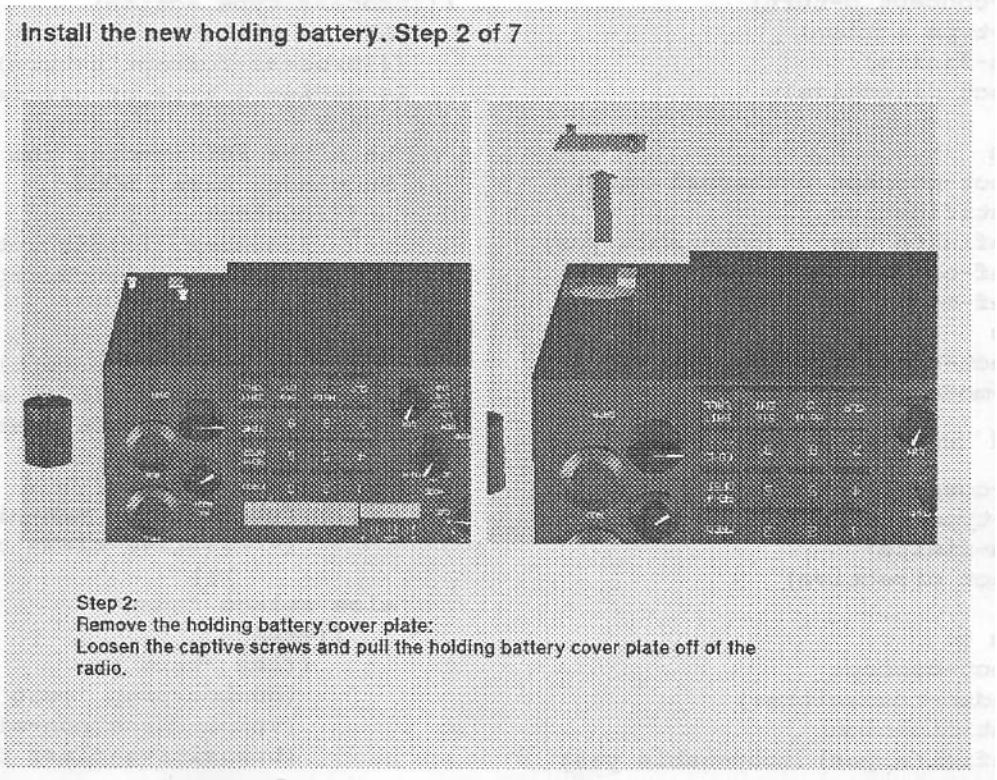

Figure 7: Older version of Fig. 3 without constraints from previous picture generation.

tributes, a lighting specification that indicates how objects are lit, and a viewing specification that indicates how the 3D objects are to be projected onto the 2D display. In designing an illustration, the illustrator relies on a set of rules that form an *illustration style*.

By default, IBIS attempts to express the contents of a logical form in a single illustration. There are many situations, however, in which this cannot be accomplished. For example, an illustration may need to show two objects that are not simultaneously visible from the same viewpoint. Alternatively, two objects to be included may be visible, but may be of sufficiently different size or distance from the viewpoint that showing one in its entirety may necessitate showing the other at too small a size for it to be legible. The objects to be depicted may even include the same object at different points in time. In all of these cases, IBIS can generate *composite illustrations*, much as COMET's text generator can create compound sentences. A composite illustration contains nested subpictures whose objects, lighting specification, or viewing specification may differ. Each subpicture is generated by an illustrator that is spawned by the parent picture's illustrator, and that is given a subset of the parent illustrator's goals to fulfill [12].

IBIS's rules have recently been expanded to deal with certain cases in which an illustration's design should be influenced by previously generated illustrations. One example of this is the incorporation of constraints from previously selected viewing specifications. When two pictures are displayed in spatial or temporal sequence, small changes in viewing specification can be disconcerting, and may appear to be the result of accidental, rather than intentional, camera movement. For example, cinematographers often use a rule of thumb that a change of viewing specification corresponding to less than a 30° rotation about the object of interest is too small [8]. When generating an illustration, IBIS takes into account the viewing specification used in previous illustrations to avoid small changes. Otherwise, attempts to optimize each viewing specification for the individual illustration's goals would result in a picture whose "locally optimal" viewing specification would not be as effective in context of those pictures already generated.

IBIS designed the illustrations in Fig. 3, taking into account the viewing specification of the left illustration when generating the right illustration. In contrast, Fig. 7 includes an earlier version of the right illustration, created with a rule base that does not incorporate these constraints on the viewing specification. Note how the locally optimized subpictures of Fig. 7 look somewhat inconsistent when viewed next to each other. Although IBIS currently completes the processing of each LF before starting on the next, it could use lookahead, as well as lookbehind, to delay making certain decisions until additional information about succeeding illustrations is known. For example, this would allow an illustration's viewing specification to be based on the contents of those illustrations that follow it, as well as those that precede it, maximizing the number of illustrations for which the same viewing specification could be used effectively.

IBIS currently generates each illustration from scratch. We are currently redesigning its picture generation ap-

proach so that it can incrementally modify a design when small changes are made to the goals that an illustration must satisfy. For example, if the viewing specification is partially specified as an input communicative goal, two illustrations' sets of communicative goals may differ only in their viewing specifications. Since IBIS runs on a machine that can render a 3D shaded image in a fraction of a second, if an illustration's specification can be incrementally regenerated fast enough, we can make possible simple user controlled animation. For example, the user could move the camera around a set of objects to view them from different positions, while IBIS maintained constraints such as legibility and visibility of designated objects.

Summary

In this paper, we described our most recent advances in COMET. These included the integration of individual components and the addition of a menu-based user interface, yielding a fully operational testbed. In the media coordinator, we have made progress towards the coordination of picture and sentence breaks. In the text generator, we focused on the problem of lexical choice, developing a framework for lexical choice using the Functional Unification Formalism and implementing influences from previous discourse and the underlying knowledge sources on lexical choice. In the graphics generator, we implemented constraints from previous (pictorial) discourse, and began work on incremental regeneration of illustrations.

Acknowledgements

This work is supported in part by the Defense Advanced Research Projects Agency under Contract N00039-84-C-0165, the Hewlett-Packard Company under its AI University Grants Program, the Office of Naval Research under Contract N00014-82-K-0256, the National Science Foundation under Grant IRT-84-51438, and the New York State Center for Advanced Technology under Contract NYSSTF-CAT(88)-5.

The development of COMET is an ongoing group effort and has benefited from the contributions of Cliff Beshers (menu interface), Andrea Danyluk (learned rule base), Michael Elhadad (FUF), David Fox (text formatting component for media layout), Laura Gabbe (static knowledge base and content planner), Jong Lim (static knowledge base and content planner), Jacques Robin (lexical chooser), Doree Seligmann (IBIS), Tony Weida (static knowledge base), Matt Kamerman (user model), and Christine Lombardi and Yumiko Fukumoto (media coordinator).

References

1. Allen, J. *Natural Language Understanding*. Benjamin Cummings Publishing Company, Inc., Menlo Park, CA, 1987.

2. Danyluk, A. Finding New Rules for Incomplete Theories: Explicit biases for induction with contextual information. Proceedings of the Sixth International Workshop on Machine Learning, Ithaca, N.Y., June, 1989.

3. Department of the Army. *TM 11-5820-890-20-1 Technical Manual: Unit Maintenance for Radio Sets AN/PRC-119, . . .* Headquarters, Department of the Army, June, 1986.

4. Elhadad, M. Extended Functional Unification ProGrammars. Columbia University, New York, NY, 1989.

5. Elhadad, M. Types in Functional Unification Grammars. Proceedings of the 28th meeting of the Association for Computational Linguistics, Pittsburgh, Pa, June, 1990.

6. Feiner, S.K. and K.R. McKeown. Coordinating Text and Graphics in Explanation Generation. Proc. AAAI 90, Boston, MA, July 29–August 3, 1990.

7. Feiner, S. A Grid-Based Approach to Automating Display Layout. Proc. Graphics Interface '88, Edmonton, June, 1988, pp. 192-197. (Palo Alto: Morgan Kaufmann, 1988).

8. Karp, P. and Feiner, S. Issues in the automated generation of animated presentations. Proc. Graphics Interface '90, Halifax, Canada, May 14-18, 1990, pp. 39-48.

9. Kay, M. Functional Grammar. Proceedings of the 5th meeting of the Berkeley Linguistics Society, Berkeley Linguistics Society, 1979.

10. Lombardi, C. Experiments for determining the assignment of information to media in COMET. Columbia University, New York, NY.

11. MacGregor, Robert and David Brill. LOOM Reference Manual. USC-ISI, Marina del Rey, CA, 1989.

12. Seligmann, D.D., and Feiner, S. Specifying Composite Illustrations with Communicative Goals. Proc. ACM Symposium on User Interface Software and Technology, Williamsburg, VA, November 13-15, 1989, pp. 1-9.