"Instructing by doing:" Interactive graphics- and knowledge-based generation of instructional text

Manfred Stede and Holmer Hemsen

Technische Universität Berlin, FB Informatik, Projektgruppe KIT,
Franklinstr. 28/29, 10587 Berlin/Germany,
{stede|hemsen}@cs.tu-berlin.de

Abstract. Much research has been conducted on applying natural language generation to the creation of technical documentation. A critical issue for such applications is supplying the input: How does the technical writer interact with a system to produce representations that can be processed by a generator? In this paper, we explore the possibility of interaction with a virtual reality as a means to produce the kernel of such representations; this needs, however, to be augmented with other techniques in order to account for those portions of instructional text that do not directly relate to physical actions.

1 Introduction and related work

Automatically generating instructional text (e.g., as a part of technical manuals) has become a popular application for text generation, as it offers a number of distinct advantages: text can be produced in multiple languages; regular updates can be created without manual re-writing and re-translating; existing data and knowledge sources can possibly be integrated into the document production process; last but not least, the language found in technical documents is typically not too complicated to impair their automatic generation. The critical issue, however, is in supplying the input to such a system: In what way does the human (co-) author of the technical document interact with the system to produce the desired text in an effective manner? Previous research has suggested menu-based interfaces (e.g., in TECHDOC [Rösner, Stede 1994] and DRAFTER [Hartley, Paris 1997]) and incremental text-template filling (WYSIWYM, [Power, Scott 1998]). In this paper, we explore a new option: interactive manipulation in a 3D graphical environment. While mixing text and graphics in the instruction generation *output* has been realized in several systems (e.g., in WIP/PPP [André 1997] or VISDOK [Hartmann et al. 1998]), graphics has to our knowledge not yet been applied on the *input* side.

The idea of our approach is that the 'author' manipulates objects on the screen using the mouse (for the time being), puts them together to create composite objects, etc. A symbolic knowledge base monitors the graphical activities and classifies them as conceptual 'actions'. In an aggregation step, individual

actions are joined to form complex action representations. These are the input to the verbalization component, which maps the conceptual representations first to sentence-semantic specifications, and then to linguistic utterances. — This scenario is not unlike those of systems producing descriptions of image data, such as NAOS [Novak 1987] or SOCCER [André et al. 1988]. They also employ domain knowledge to identify elementary actions and "chunk" them into linguistic descriptions. In contrast to these systems, however, our input is the concrete manipulation data; extracting relevant changes in image data is thus not a primary concern. As another point of contrast, we are interested in multilingual output and, eventually, in combining graphics-input with other modes of user interaction in the production of instructional text.

At present, we have implemented a first prototype intended as "proof of concept", which thus illustrates the basic functionality. Its architecture is described in section 2. We illustrate the approach with our implemented pilot application, a construction kit, in section 3. Specifically, we describe in detail the events leading to the construction of an axle; this would in a more complete implementation be a part of building some kind of vehicle. In section 4, we discuss the advantages and disadvantages of our approach, explore possibilities for extending it to more complicated texts, and hint at some directions for practical applications of the approach. In particular, we suggest to fuse our approach with the WYSIWYM method in order to account for passages of instructional text that go beyond descriptions of physical activities, and thus are not immediately amenable to graphical input. Ultimately, we therefore view our approach as one part of an "author's workbench", where interactive graphics can help producing the raw text that needs to be further processed with appropriate tools.

2 System architecture

The architecture of the prototype is shown in figure 1. While the user manipulates objects in the 3D environment, a stream of *elementary events* is produced and written to a file. The events are in the format of assertions in the description logic LOOM [MacGregor 1991]. When the user initiates text generation, a segmentation module reads the event-file, performs aggregations and maps it to a sequence of action representations in the format of 'SitSpecs' [Stede 1999]. This process is driven by the LOOM knowledge base (KB), which holds the knowledge about the level of abstraction desired for verbalizing the activities. The SitSpecs are converted to sentence-semantic specifications (SemSpecs) using the MOOSE module [Stede 1999], and SemSpecs are finally turned into English or German sentences by the KPML generator [Bateman 1997].

2.1 Interactive graphics

The graphics module is implemented in Java-3D [Sowizral et al. 1998]. With the mouse, the user can

1. enter a new object into the world, choosing it from a menu,

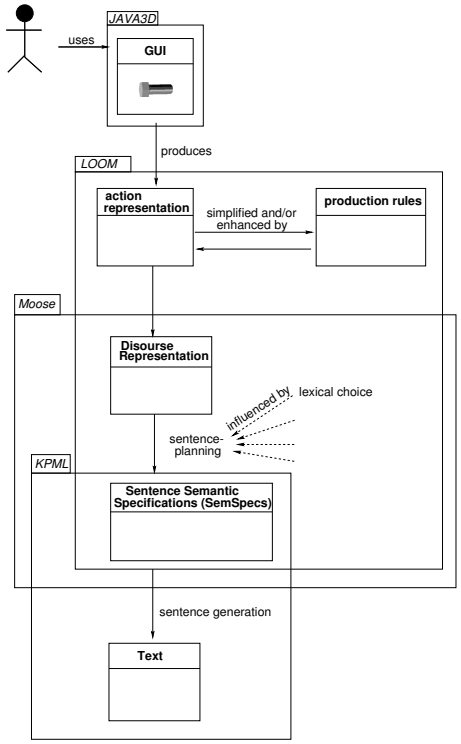

Fig. 1. System architecture

2. move an object to a new location,
3. turn an object around,
4. connect an object to another one.

The set of objects that can be introduced to the world (1) is determined by a menu, and thus the type of each new object in the world is fixed. This provides the link to the Loom KB: For a new object, a LOOM instance of the respective type is created.

Items (2) and (4) go beyond the level of straightforward graphical manipulation: (2) needs to detect collisions, i.e., ensure that an object is not moved "through" some other object. (4) needs to check whether the two objects involved can indeed be connected, i.e., whether they are nut and bolt or some other suitable pair. We decided to handle both tasks by the same mechanism. For a start, Java-3D offers a "built-in" collision detection, which notices a topological overlap between two objects (or, alternatively, their bounding boxes). Whenever this condition is triggered, we perform a deeper analysis of the topological relationship between the two objects in order to determine whether the user is likely to intend a connection between them. The conditions depend on the specific pair; for instance, if the tip of a screw collides with a wheel close

to the hole in the middle, and the angle between wheel and screw is close to 90 degrees, we surmise that the user intends to move the wheel over the screw. As soon as the collision as well as the additional conditions have been detected, the user is prompted to either confirm or reject the connection (in case the collision was not on purpose). Upon confirmation, the system completes this move and arranges the parts in their final position. For illustration of the results of the graphical manipulations, see figure 3, which will be explained in section 3.

2.2 Knowledge base

A key idea in our approach is to realize a close connection between the graphical representation of the "world" on the one hand and a symbolic representation of this world within a description logic on the other hand, and to exploit the power of automatic concept classification. For these purposes, we use the LOOM language and classifier [MacGregor 1991].

The terminological part of the knowledge base (Tbox) holds the knowledge about the various types of objects and their properties, and the possibilities for connecting them: Nuts can be connected to bolts, liquids can be put into a container, etc. The assertional part (Abox) is a symbolic representation of the state of the world and the changes that occur; there is an instance for each object in the world, and the connections between objects are modelled via LOOM relations. These are explicitly asserted when the user performs a connect-event in the 3D world. Importantly, as a result of a new connection-role, the LOOM classifier can automatically determine new type information. An example involving the classification of an assembled axle follows below. In other scenarios, re-classification can also occur when some other attribute of an object changes, e.g., when the user flips a switch or turns a knob.

In analogy, a sequence of actions can be automatically classified as a "meaningful" macro-action. For instance, the sequence of elementary actions shown in figure 2 can be classified as a 'connect' macro action with the connector being the screw and the connectee the ring; the concept definition of the macro action contains a sequence of elementary actions of appropriate types (which abstract from the topological details that are not relevant for the classification).

While some aggregations can be performed by the classifier automatically, others do not lend themselves to being formulated as a complex concept; for these cases we use LOOM production rules to trigger additional inferences.

2.3 Text generation

Supported by the automatic classifier as just described, the first step of the text generation process consists of "chunking" the sequence of elementary actions into a text plan, which involves the well-known task of aggregation (e.g., [Dalianis 1996]). At the moment, we are using only a fairly simple aggregation module that is geared to the scenario of our pilot application, to be explained in the next section.

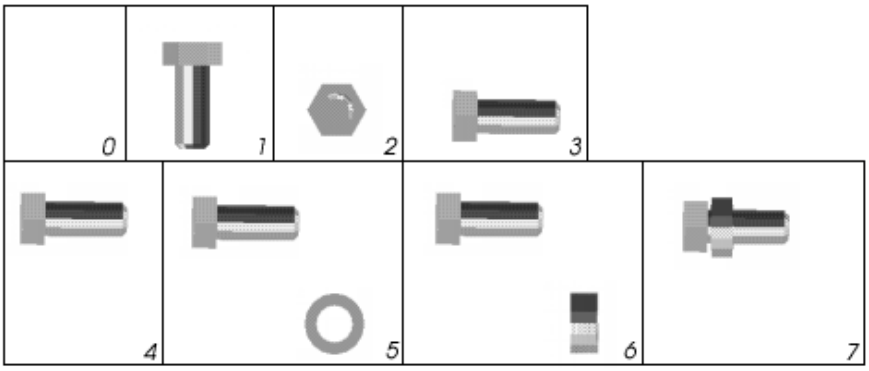

Fig. 2. Sequence of elementary actions in the Java-3D world

The text plan is also represented in LOOM and follows the format of 'SitSpecs' as used in the MOOSE generator [Stede 1999]. MOOSE converts the SitSpec into a sequence of sentence-semantic specifications; these are language-specific, lexicalized structures that are in the final step converted to linguistic utterances (in either English or German) by KPML [Bateman 1997]. MOOSE, originally a single-sentence generator, is currently being upgraded to produce complete paragraphs of text; thus it will accomplish sentence planning tasks such as determining sentence boundaries and structure, and choice of referring expressions. The input to the system can therefore be either an individual SitSpec, or a rhetorical tree (in the spirit of RST [Mann, Thompson 1988]). As the following section will show, however, the sentence planner at present is still in a very preliminary stage.

3 Example: Constructing an axle

Our pilot implementation of the framework (documented in [Hemsen 2000]) deals with a 'construction kit' with a set of parts that can be assembled into various mechanical objects. (This scenario was also used by [Wachsmuth, Jung 1996]). In the following, we describe the example of construcing a vehicle axle, composed of two wheels, rings and screws, and a cube holding them together.

In the Java-3D world, the user enters the various objects and moves them close together so that the system can infer the intended connections. The sequence of elementary actions shown in figure 2 is one possible beginning of the activity. Abstracting from the movement events, the initial sequence of elementary actions is as follows. Two objects are introduced, which prompts the creation of two Loom instances (recall that their type is associated with the menu options):

```
(createm 'HexaScrew@11d1c62 'hexagon-headed)
(createm 'Ring@1778fcd 'ring)
```

When the ring has been moved to the screw, and the system detects that the preconditions for a connection are fulfilled, the corresponding event instance is created, together with two location states, which are linked to the event as pre-state and post-state, respectively:

```
(createm 'connect1 'event)
(createm 'location-state2 'loc-state)
(createm 'location-state3 'loc-state)
(tellm (has-locst-locatum location-state2 Ring@1778fcd))
(tellm (has-locst-location location-state2 'somewhere))
(tellm (has-locst-locatum location-state3 Ring@1778fcd))
(tellm (has-locst-location location-state3 HexaScrew@11d1c62))
(tellm (has-locst-localizer location-state3 'onto))
(tellm (has-ev-activity connect1 indefact2))
(tellm (has-ev-pre-state connect1 location-state2))
(tellm (has-ev-post-state connect1 location-state3))
```

This process continues until the axle is complete, which the LOOM classifier notices automatically. Here is the definition of the concept:

```
(defconcept axle
    :is (:and cube-with-parts
              (:exactly 2 has-connectee-part)
              (:all has-connectee-part axle-part)
              (:satisfies (?y) (Sum (has-connectee-pos ?Y) 7))))
```

'Axle-part' is in the same way defined as a screw with a ring and wheel connected to it. The 'satisfies' clause in the concept ensures that the two axle-parts are indeed mounted to opposite sides of the cube (otherwise, all the parts would be there and connected, but not to the effect of a functioning axle). The resulting object in Java-3D is shown in figure 3.

The text planning module, in charge of building a rhetorical graph structure, consults the knowledge base to determine that the concept 'axle' is a sub-concept of 'integral-part', which denotes an integral constituent of some higher-level entity (here, some vehicle). Accordingly, it infers that constructing the axle was indeed the purpose of the action sequence, and thus constructs a structure that can be abbreviated as follows:

```
(PURPOSE (has-nucleus (construct-axle ...))
         (has-satellite (SEQUENCE (take screw ...)
                                  (take ring ...)
                                  (put ring screw ...))))
```

Using only one aggregation rule (skip the second 'take' action), a straightforward English version of the text produced by our system is

> "Take a hexagon bolt, and put a ring onto the hexagon bolt, and put a wheel onto the hexagon bolt, and fasten the hexagon bolt to a thread-cube, and take a hexagon bolt, and put a ring onto the hexagon bolt, and put a wheel onto the hexagon bolt, and fasten the hexagon bolt to the thread-cube, in order to construct an axle."

In our ongoing work on the sentence planner, this text is to be improved by various additional aggregation rules. In contrast to "chunking" the elementary events from the input data stream, we are now concerned with aggregation on the *text* level: Sentence boundaries have to be introduced, which requires a different signal of the PURPOSE relation (some appropriate adverbial rather than a conjunction), and referring expressions can be improved. Furthermore, the fact that the two halfs of the axle are assembled in exactly the same way should be reflected in the text (which is to be recognized on the level of action aggregation rather than text aggregation, though). One possible target text incorporating these improvements is:

> "In order to construct an axle, take a hexagon bolt, put first a ring and then a wheel onto the hexagon bolt, and fasten this bolt to a thread-cube. Take another hexagon bolt and again put a ring and a wheel onto it. After that, fasten this bolt to the thread on the opposite side of the thread-cube."

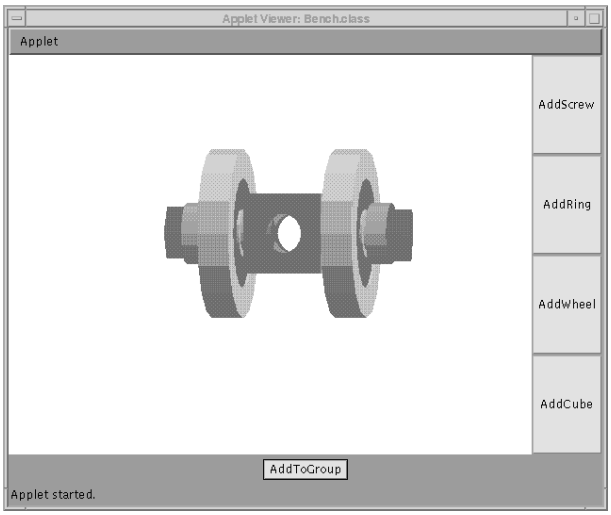

Fig. 3. Screenshot Java-3D: assembled axle

4 Perspectives

4.1 Directions for extensions

The pilot application is merely a first "proof of concept" for the scenario, which can now be enhanced into various directions. As mentioned above, an improved sentence planning module is currently under development. Another step that

needs to be improved for a larger-scale application is constructing the input to the text planner: Recall that at present, we use simple annotations to KB concepts in order to determine whether the creation of some object was made "on purpose" and is to be verbalized as such. In general, though, it is necessary to map the (partly aggregated) stream of elementary actions first to a (pre-verbal and pre-RST) plan structure that explicitly reflects what actions are parts of other actions, and what goals are being followed [Mellish, Evans 1989]. This structure can then be mapped to an RST-inspired text plan, as it was for instance done in TECHDOC [Rösner, Stede 1994].

Improvements can also be envisioned on the side of the graphics input. For example, instruments and tools that are necessary for certain actions can appear as clickable icons, so the user can indicate that it is required for some activity ("remove the wheel with a screwdriver"). Furthermore, the graphical world and the knowledge base should be coupled more closely to the effect that graphics activities by the user are immediately checked by the KB for their possible consequences. For example, some rules of physics can be implemented so that moving a liquid into a container has a different effect than moving it to the outside of a container; this kind of knowledge does not belong to the Java-3D model but to the symbolic knowledge. Also, visible consequences of user's actions (e.g., a light turning on in response to moving a switch) need to be computed in the KB and propagated back to the visual scene.

Java-3D offers the advantage that the applications can be run over the web. The associated disadvantage, however, is that the 3D models as well as the possible modes of interaction are relatively impoverished when compared to state-of-the-art virtual reality environments. On the other hand, 3D web browsers, coupled with more sophisticated input devices (3D mouse, data glove), will soon become available and commonplace. Then, the task of creating verbal protocols of user's activities in the virtual reality will of course be much more complex than in our example presented above, but it offers many applications, not only for producing instructions but also for other purposes.

4.2 The role of the knowledge base

As we have stressed the role of automatic classification in the process leading to verbalizing the user's activities, it is clear that a comprehensive domain model must provide the detailed concept representations enabling these classifications. For our pilot implementation, the domain model was built by hand, but it reused significant portions of the ontology and domain model that were developed earlier for the MOOSE system. Re-usability is indeed a key factor for scaling up the prototype to a practical application: When models are built in such a way that the upper ontology as well as general knowledge about types of technical objects can be carried across domains, the prohibitive costs of manually building domain models can be reduced. Furthermore, it can be expected that the ongoing efforts in standardizing knowledge representation formats and in sharing knowledge bases will lead to the availability of standard modules that can be used as a basis for the knowledge sources required for generation.

4.3 Toward an author's workbench

Focusing now again on instructional text, we notice that extending the graphical environment into a full-fledged virtual reality will, at any rate, cover only one side of the coin. While descriptions of sequences of physical activities are a central ingredient of instructional text, there are additional elements that also need to be accounted for, and that are not easily accomplished with graphical means.

Consider a somewhat more complicated instructional text from a car manual. We have divided it into 'minimal units' and marked them with square brackets.

> [Wait]1 until [the engine is cool]2, then [turn the radiator cap clockwise]3 until [it stops]4. [DO NOT PRESS DOWN WHILE TURNING THE CAP]5. After [any remaining pressure has been relieved]6, [remove the cap]7 by [pressing down]8 and [again turning it counterclockwise]9. [Add enough coolant]10 to [fill the radiator]11, and [reinstall the cap]12. [Be sure to tighten it securely]13. [Fill the reserve tank up to the max mark]14 with [the engine cold]15.

Using the labels of the minimal units, the text can be assigned the following RST analysis (notation: (RELATION NUCLEUS SATELLITE)):

```
(SEQUENCE (UNTIL 1 2)
          (CIRCUMSTANCE (UNTIL 3 4)
                        5)
          (PRECONDITION (PURPOSE (SEQUENCE 8 9)
                                 7)
                   6)
          (PURPOSE 10 11)
          12
          13
          (PRECONDITION 14 15))
```

Several portions of this text cannot be inferred from actions in the graphics environment. First, the UNTIL–relation cannot be read off directly from an action sequence; and in particular, 'waiting' is not an action that is easily demonstrated in a virtual world. Second, both PRECONDITIONs are problematic: noticing that all pressure has been relieved would require a highly sophisticated simulation; the engine being cold might be visualized in some way or another, but the fact that it is a precondition for something else might not. Third, CIRCUMSTANCES typically convey information that *accompanies* an activity and is thus difficult to simulate; here it is even more problematic since it is an instruction *not* to do something. Finally, the "be sure..." sentence represents a cognitive activity rather than a physical one.

To account for such problems, additional mechanisms are needed. While it is not impossible that the user in between actions clicks on some buttons with coherence relations on them, this would only be a partial answer. In general, it seems more reasonable to open up the possibility of linguistic interaction in addition to graphics interaction. Here, fusing our approach with the WYSIWYM method proposed by Power and Scott [1998] seems to be a viable option. The

graphics component would produce the "backbone text" that the user can further augment by clicking on the text rather than on the image. For instance, text segments spanned by a coherence relation can be marked and the relation chosen from a menu, whereupon the system would alter the text to include a signal for the relation. Similarly, new propositions can be inserted, such as cognitive activities or circumstances of actions.

Assuming that our approach is developed into the directions just sketched, it can address a significant problem in the production of technical documentation: the knowledge gap between engineer and technical writer. Nowadays, the technical writer typically receives a more or less precise specification from the engineer and strives to produce a readily understandable text from it. This can require feedback from the engineer, which is not always available, though. The resulting instruction manuals often reflect this problem. When the engineer can through virtual-reality interaction provide a detailled formal representation of the instruction content, the gap may be narrowed: The generation system turns the specification into a clear and unambiguous — yet raw — text, and the technical writer can interactively polish it to the effect that a well-written and understandable text results.

References

[André 1997] E. André. "WIP and PPP: A Comparison of two Multimedia Presentation Systems in Terms of the Standard Reference Model." *Computer Standards and Interfaces* 18(6-7), 1997.

[André et al. 1988] E. André, G. Herzog, T. Rist. "On the Simultaneous Interpretation of Real World Image Sequences and their Natural Language Description: The System SOCCER." In: *Proceedings of the 8th ECAI*, München, 1988.

[Bateman 1997] J. Bateman. "Enabling Technology for Multilingual Natural Language Generation: The KPML Development Environment." In *Journal of Natural Language Engineering*, 3(1), 15-55, 1997.

[Dalianis 1996] H. Dalianis. *Concise Natural Language Generation from Formal Specifications.* Dissertation, The Royal Institute of Technology and Stockholm University, Stockholm, 1996.

[Hartley, Paris 1997] A. Hartley, C. Paris. "Multilingual Document Production: From Support for Translating to Support for Authoring." In: *Machine Translation* 12, pp. 109–128. 1997.

[Hartmann et al. 1998] K. Hartmann, R. Helbing, D. Rösner, T. Strothotte. "Visdok: Ein Ansatz zur interaktiven Nutzung von technischer Dokumentation." In P. Lorenz and B. Preim, editors, *Simulation und Visualisierung '98*, SCS–Society for Computer Simulation Int., pp. 308-321, Erlangen, 1998.

[Hemsen 2000] H. Hemsen. "Generierung mehrsprachiger Instruktionstexte aus der Interaktion mit einer virtuellen Welt." Diploma thesis, Dept. of Computer Science, TU Berlin, April 2000.

[MacGregor 1991] R. MacGregor. "Using a Description Classifier to Enhance Deductive Inference." In: Proc. of the Seventh IEEE Conference on AI Applications, 1991.

[Mann, Thompson 1988] W. Mann, S. Thompson. "Rhetorical structure theory: Towards a functional theory of text organization." In: *TEXT*, 8:243-281, 1988

[Mellish, Evens 1989] C. Mellish, R. Evans. "Natural language generation from plans." In: *Computational Linguistics* 15(4), pp. 233–249, 1989.

[Novak 1987] H.-J. Novak. *Textgenerierung aus visuellen Daten: Beschreibung von Straßenszenen.* Heidelberg: Springer Verlag, 1987.

[Power, Scott 1998] R. Power, D. Scott. "Multilingual authoring using feedback texts." In: Proceedings of COLING-ACL '98, pp. 1053–1059, Montreal, 1998.

[Rösner, Stede 1994] D. Rösner, M. Stede. "Generating multilingual documents from a knowledge base: The TECHDOC project." In: *Proceedings of the International Conference on Computational Linguistics (COLING).* Kyoto, 1994.

[Sowizral et al. 1998] H. Sowizral, K. Rushforth, M. Deering. *The Java 3D API Specification.* Amsterdam: Addison-Wesley Longman, 1998.

[Stede 1999] M. Stede. *Lexical semantics and knowledge representation in multilingual text generation.* Dordrecht/Boston: Kluwer, 1999.

[Wachsmuth, Jung 1996] I. Wachsmuth und B. Jung. "Dynamic Conceptualization in a Mechanical-Object Assembly Environment." *Artificial Intelligence Review* 10, pp. 345–368, 1996.