Simple Natural Language Generation and Intelligent Tutoring Systems

Barbara Di Eugenio, Michael Glass, Michael J. Trolio
Electrical Engineering and Computer Science Department
University of Illinois at Chicago
Chicago, IL, 60607 USA
{bdieugen,mglass,mtrolio}@eecs.uic.edu

Susan Haller
Computer Science Department
University of Wisconsin Parkside
Kenosha, WI 53141, USA
haller@cs.uwp.edu

Abstract

In this paper, we report on our approach to adding Natural Language Generation (NLG) capabilities to ITSs. Our choice has been to apply simple NLG techniques to improve the feedback provided by an existing ITS, specifically, one built within the DIAG framework (Towne 1997). We evaluated the original version of the system and the enhanced one with a between subjects experiment. On the whole, the enhanced system is better than the original one, other than in helping subjects remember the actions they took. Current work includes exploiting more sophisticated NLG techniques but still without delving into full fledged text planning. We are also conducting a constrained data collection, in which students and tutors interact via the ITS.

Introduction

Today, many projects aim at providing ITSs with a full-fledged dialogue interface, e.g. see the work at the CIRCLE center (http://www.pitt.edu/˜circle/), or (Hume *et al.* 1996; Moore, Lemaire, & Rosenbloom 1996; Rosé, Di Eugenio, & Moore 1999; Freedman 1999; Graesser *et al.* 2000). On the contrary, our approach to adding NLG capabilities to an Intelligent Tutoring System falls on the weak side of the divide: we are concentrating on simple sentence planning with no or minimal amounts of text planning. Our choice is partly a development strategy, because we set out to rapidly improve the language feedback provided by an existing ITS shell, partly a desire to evaluate how effective the system can be with a relatively small effort. A similar approach — using simple generation techniques for surface realization in tutoring dialogues — is taken in YAG (McRoy, Channarukul, & Ali 2000). Our results so far suggest that simple NLG can help, but the gains are small enough to suggest that moving to somewhat more sophisticated techniques should be beneficial, even if we still don't intend to develop a full fledged NLG interface.

We take this approach for two reasons. First, we want to understand what can be accomplished by interfacing an NL generator to an ITS taken as a blackbox: can the ITS tutoring strategy be left as is, or is there a point in which the dialogue strategies and the original tutoring strategy are at odds with each other? Second, we are interested in finding out what is the "added value" of an NL interface to an ITS. One way to do so is to compare a system that does not use NL techniques to a version of the same system that uses NL. We are aware of only one other experiment in this direction (Trafton *et al.* 1997), in which subjects gave input to a cartographic system using either NL only, direct manipulation only, or a combination of the two. Subjects were given instructions such as "go to intersection X"; time on task and score on map drawing after the session were recorded. In the NL only condition, subjects performed the poorest on the map drawing task. However, it is not clear which conclusions should be drawn from this work, given that the system they describe does not seem to qualify as a real ITS. In general, the evaluation of NL interfaces to ITSs is an area that needs investigation. ITSs are often evaluated in terms of pre/post-test score, however task performance measures may be appropriate as well. To our knowledge, the only ITSs with an NL interface which has been formally evaluated is CIRCSIM (Evens *et al.* 1993; Kim, Glass, & Evens 2000), but the results of the evaluation are not available yet.

We will first discuss DIAG, the ITS authoring shell we are using. We will then discuss the work we have completed; this comprises the aggregation rules we implemented within EXEMPLARS and the formal evaluation we conducted. We will then discuss some current work on generating more coherent feedback by exploiting more sophisticated NLG techniques, and the data collection we have started, to study how tutors verbalize the information that the ITS wants to communicate.

DIAG

DIAG (Towne 1997) is a shell to build ITSs that teach students to troubleshoot complex artifacts and systems, such home heating and circuitry. DIAG in turn builds on the VIVIDS authoring environment (Munro 1994). VIVIDS based tutors deliver instruction and practice in the context of graphical simulations. Authors build interactive graphical models of complex systems, and build lessons based on these graphical models.

A typical session with a DIAG application presents the student with a series of troubleshooting problems of increasing difficulty. DIAG's tutoring strategy steers the student towards performing the tests that have the greatest potential for reducing uncertainty (Towne 1997). Most of the times, a test consists of the visual observation of an *indicator*. DIAG keeps track of the tests the student performs, and the inferences that could be made from the symptoms shown. The student interacts with the application by testing indicators and trying to infer which faulty part (RU) may cause the detected abnormal states. RU stands for *replaceable unit*, because the only course of action open to the student to fix the problem is to replace faulty components in the graphical simulation. Figure 1 shows one of the graphical views in a DIAG application that teaches how to troubleshoot a home heating system. The subsystem being displayed is the furnace system. Some of its components are indicators (e.g., the gauges labeled Burner Motor RPM and Water Temperature). Others are either replaceable units, or other complex modules that contain indicators and replaceable units, e.g. the Oil Burner. Complex components are in turn zoomable.

At any point, the student can consult the built-in tutor in one of several ways. For example, if the student suspects an RU to be faulty, s/he can ask the tutor to specify the likelihood that this part is the cause of the fault. The tutor will also indicate the state of any indicators that the student has explored and try to imply a correlation, positive or negative, between the states of the indicators to the RU in question. By utilizing the tutor's feedback, the student can deduce relationships among the system parts and continually refine his/her solution.

Language Generation in DIAG

After deciding which content to communicate, the original DIAG system (*DIAG-orig*) uses very simple templates to assemble the text to present to the student. The result is that the feedback that DIAG provides is repetitive, both as a sequence of replies to requests for feedback, and within each verbal feedback. In many cases, the feedback presents a single long list of many parts. This problem is compounded by the fact that most DIAG applications involve complex systems with many parts. Although there are different levels of description in the system model, and hierarchies of objects, the verbal feedback is almost always in terms of individual indicators or units. The top part of Figure 2 shows the reply originally provided by DIAG to a request of information regarding the indicator named "Visual Combustion Check".

We set out to improve on DIAG's feedback mechanism by applying aggregation rules. For example, a long list of parts can be broken down by classifying each of these parts in to one of several smaller lists and then presenting the student with this set of lists. The bottom part of Figure 2 shows our aggregation rules at work. The revised output groups the parts under discussion by the system modules that contain them (Oil Burner and Furnace System), and by the likelihood that a certain RU causes the observed symptoms. Notice how the *Ignitor Assembly* is singled out in the revised answer. Among all mentioned units, it is the only one that cannot cause the symptom. This fact is lost in the original answer.

As our sentence planner, we chose EXEMPLARS (White & Caldwell 1998) over better known systems such as FUF (Elhadad 1993) and Penman (Bateman 1994) because of the complexity and learning curve of the latter two. Efficiency and rapid prototyping are among the reasons we chose EXEMPLARS.

EXEMPLARS is an object-oriented, rule based generator. The rules (called *exemplars*) are similar to schema-like text planning rules because they are meant to capture an exemplary way of achieving a communicative goal in a given communicative context, as determined by the system designer. EXEMPLARS is a hybrid system that mixes template-style and more sophisticated types of text planning. The text planner selects rules by traversing the exemplar specialization hierarchy. The applicability conditions associated with each exemplar are successively evaluated in order to find the most specific exemplar for the current context.

In the enhanced version of the system (*DIAG-NLP*), DIAG passes the information to be communicated to EXEMPLARS (the two systems communicate via a text file). EXEMPLARS performs essentially three tasks:

1. it determines the specific exemplars needed;
2. it adds the chosen exemplars to the sentence planner as a goal;
3. it linearizes and lexicalizes the feedback in its final form, writing it to an external file which is passed back to DIAG for display in the appropriate window.

In *DIAG-NLP*, we concentrated on rules for aggregation, some of which also affect format and layout. Our choices were suggested by the need to relate the language feedback to the hierarchical structure of the physical system. We have two main kinds of rules, description rules and aggregation rules.

Description rules are used when the full description of a part is required, such as whether the part is

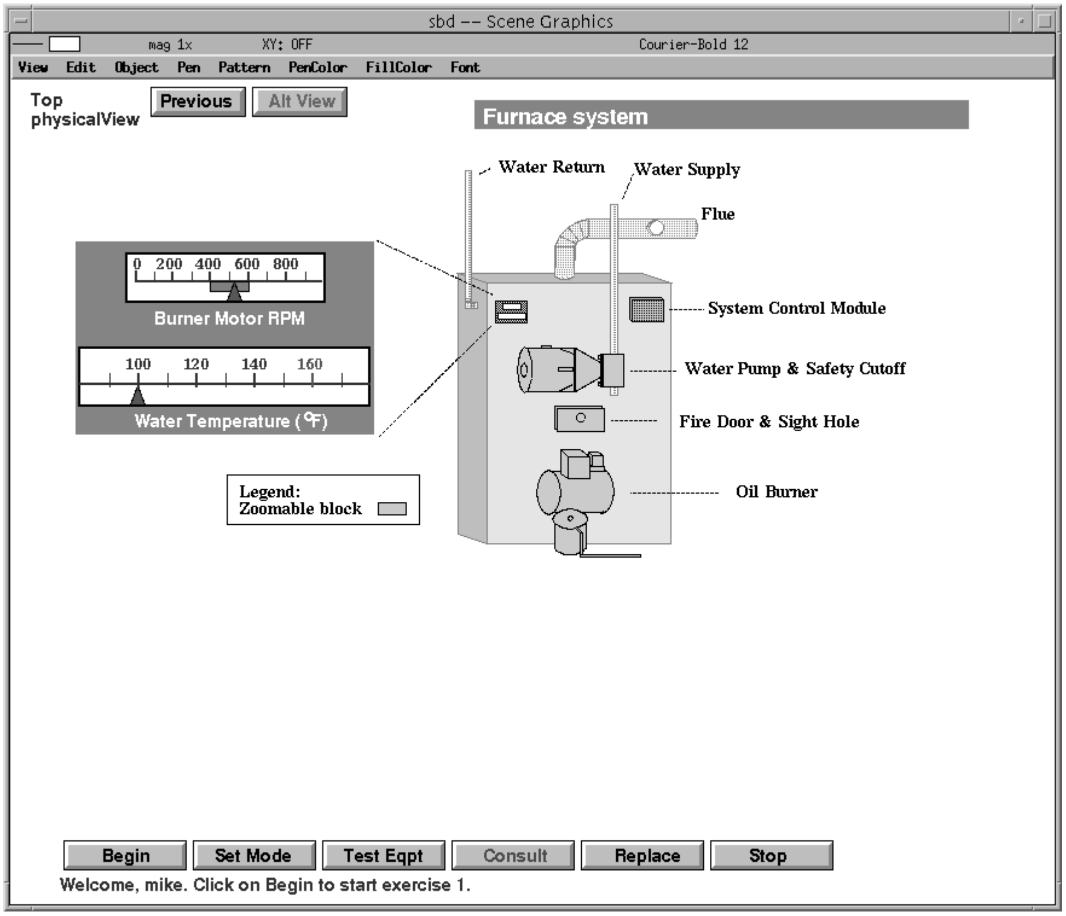


Figure 1: A screen from a DIAG application on home heating

in a normal state, its current reading, and, if abnormal, what the normal state should be (see the first sentence in the bottom part of Figure 2).

The aggregation rules are used to group large lists of parts into smaller lists. They allow composite aggregation, so that nested lists are created. Among our aggregation exemplars are:

- *AggByContainer*: each part within this DIAG application is contained within a larger block, called a system module. The *AggByContainer* rule accepts a list of parts, classifies each part by its containing module, and then creates a set of lists by module;
- *AggByFufer*: it groups replaceable units according to the likelihood of being at fault for a specific symptom;
- *AggByState*: it groups indicators by their normal / abnormal state.

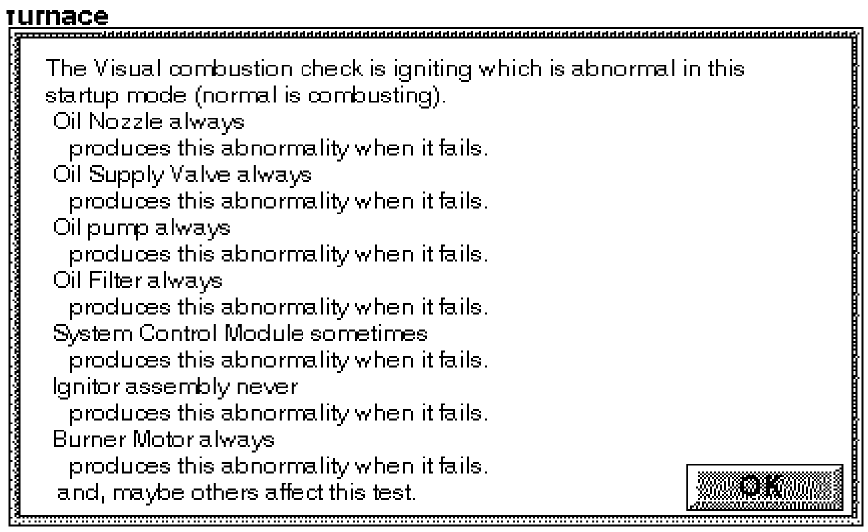
A final exemplar, invoked by the other aggregation rules, deals with formatting, namely, creating vertical lists, spacing, etc.

The most frequent application of the aggregation rules is to group parts according to the system module they belong to, and within each module, to group replaceable units by how likely it is they may cause the observed symptom, as shown in Figure 2.

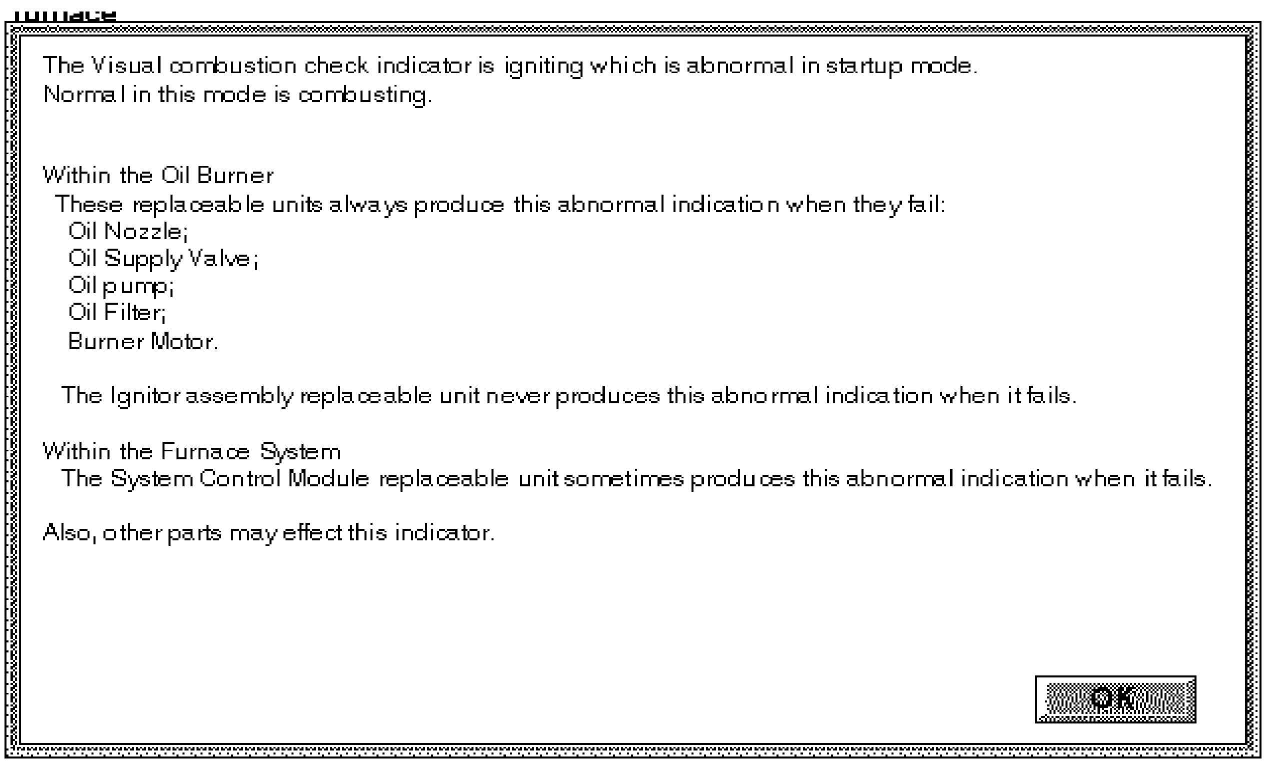
In this version of *DIAG-NLP*, morphology, lexical realization and referring expression generation were all treated ad hoc, i.e., they were directly encoded in the appropriate exemplars.

Experiments

Intuitively, the contrast between the feedback produced by *DIAG-orig* and by *DIAG-NLP* (top and bottom in Figure 2) suggests that even simple aggregation rules dramatically improve the language feedback. To provide a real assessment of this claim, we conducted an empirical evaluation designed as a between-subject study. Both groups interact with the same DIAG application that teaches them to troubleshoot a home-heating system. One group interacts with *DIAG-orig* and the other with *DIAG-*



The Visual combustion check is igniting which is abnormal in this
startup mode (normal is combusting).
Oil Nozzle always
produces this abnormality when it fails.
Oil Supply Valve always
produces this abnormality when it fails.
Oil pump always
produces this abnormality when it fails.
Oil Filter always
produces this abnormality when it fails.
System Control Module sometimes
produces this abnormality when it fails.
Ignitor assembly never
produces this abnormality when it fails.
Burner Motor always
produces this abnormality when it fails.
and, maybe others affect this test.



The Visual combustion check indicator is igniting which is abnormal in startup mode.
Normal in this mode is combusting.

Within the Oil Burner
These replaceable units always produce this abnormal indication when they fail:
Oil Nozzle;
Oil Supply Valve;
Oil pump;
Oil Filter;
Burner Motor.

The Ignitor assembly replaceable unit never produces this abnormal indication when it fails.

Within the Furnace System
The System Control Module replaceable unit sometimes produces this abnormal indication when it fails.

Also, other parts may effect this indicator.

Figure 2: Original (top) and revised (bottom) answers provided by DIAG to the same *Consult Indicator* query

NLP.

Seventeen subjects were tested in each group. Our subject pool comprised 13 undergraduates, 18 graduates, and 3 research staff, all affiliated with our university. Participation in the experiment was restricted to science or engineering majors. Each subject first reads some short material about home heating that we developed. Afterwards, each subject goes through the first problem as a trial run. Each subject then continues through the curriculum on his/her own. The curriculum consists of three problems of increasing difficulty. Subjects are encouraged to interact with DIAG as much as possible. At the end of the experiment, each subject is administered a questionnaire.

Metrics. A detailed log is collected while the subject solves problems. It includes how many problems the subject solved, and, for each problem: total time, and time spent reading feedback; how many and which indicators and RUs the subject consults DIAG about; how many, and which RUs the subject replaces.

Questionnaire. The questionnaire is divided into three parts. The first part tests the subject's understanding of the domain. Because the questions asked are fairly open ended, this part was scored as if grading an essay.

The second part concerns the subjects' recollection of their actions, specifically, of the indicators they consulted the system on and of the RUs they replaced. By taking the log of the subject's actions as the target, we can compute the usual measures of precision and recall. We compute precision as the percentage of correct answers out of the total number of answers the subject gave; whereas recall is the percentage of correct answers they gave with respect to the log of their actions. We also compute the F-measure, $\frac{(\beta^2+1)PR}{\beta^2 P+R}$, that smooths precision and recall off, with $\beta = 1$.

The third part of the questionnaire asks the subject to rate the system's feedback along four dimensions on a scale from 1 to 5 (see Table 3).

Comments on collected measures. As the reviewers of this paper pointed out, almost all the measures we collected, and whose significance is analyzed below, pertain to task performance or user satisfaction, rather than to learning per se — only *Essay score* directly addresses learning. We agree that learning measures should be the ultimate test of the success of the NL interface to the ITS. However we would argue that performance measures are important too: they provide indirect evidence of the effectiveness of the system, including issues of usability. For example, the lower number of indicator consultations in *DIAG-NLP* is evidence in favor of the effectiveness of the aggregated feedback: because the feedback highlights what is important (such as that the Ignitor Assembly can never cause the Visual Combustion check to ignite, see Figure 2), subjects can focus their troubleshooting without asking as many questions of the system. We would argue that an ITS whose NL feedback leads the student more effectively towards the solution of a problem is a better ITS. This holds for usability as well (the four measures in Table 3): presumably, in a real setting, students should be more willing to sit down with a system that they perceive as more friendly and usable than a system that engenders similar learning gains, but is harder to use.

	DIAG-orig	DIAG-NLP
Time	29.8'	28.0'
Feedback Time	6.9'	5.4'
Consultations	30.4	24.2
Indicator consultations	11.4	5.9
RU consultations	19.2	18.1
Parts replaced	3.85	3.33
Essay score	81/100	83/100

Table 1: Performance measures

	DIAG-orig	DIAG-NLP
Indicator Precision	.33	.17
Indicator Recall	.33	.27
Indicator F-measure	.44	.29
RU Precision	.74	.65
RU Recall	.73	.63
RU F-measure	.72	.63

Table 2: Precision / recall

The measures in Table 2 measure something in between learning and performance. One could argue that remembering what you did correlates with learning — e.g., if you remember that to solve a certain problem you checked whether the furnace was combusting (the "Visual Combustion Check" in Figure 2) and that gave you crucial information, you may be able to apply similar knowledge in similar problems. However, it is unlikely that detailed quantitative measures such as those we collected in this experiment are telling in this regard; and in fact, we would be happy to eliminate them, as they actually show an advantage for *DIAG-orig*. However, we collected them because they are relevant to the more general question of the added value of NL interfaces to applications, which we are also interested in.

	DIAG-orig	DIAG-NLP
Usefulness	4.35	4.47
Helped stay on right track	4.35	4.35
Not misleading	4.00	4.12
Conciseness	3.47	3.76
Average score	4.04	4.18

Table 3: Subjective rating of DIAG's feedback

Results. Every student solved all the problems, but differences emerge with respect to other measures. Tables 1, 2, 3 show the results for the cumulative measures across the three problems (measures on individual problems show the same trends).

On the whole, Tables 1 and 3 show a cumulative effect in favor of *DIAG-NLP*, whereas Table 2 does not. Focusing first on Tables 1 and 3, differences on individual measures are not statistically significant; the measure that individually comes closest to statistical significance is *indicator consultations*, which exhibits a non-significant trend in the predicted direction (Mann-Whitney test, U=98, p=0.11). We have discussed individual measures at length in (Di Eugenio & Trolio 2000); here, we provide a different statistical analysis to assess whether the *cumulative* effect of these measures shows that *DIAG-NLP* performs better than *DIAG-orig*.

We consider only independent measures (for example, the total number of consultations in Table 1 is clearly not independent from indicator and RU's consultations, given it is the sum of these two measures). For each measure, we decide whether its value indicates a "success" for *DIAG-NLP*. We are not looking at the magnitude of the difference between the two values of the measure, but simply at the fact that the values differ. Every measure in Table 1 is in *DIAG-NLP* favor, and so is every measure apart from *helped stay on right track* in Table 3 (we consider a tie as a success for *DIAG-orig*). We then ask, what is the probability that the *m* successes for *DIAG-NLP* out of the *n* independent measures are simply due to chance? We can answer via $B(m-1, n, 0.5)$, the binomial cumulative distribution function through $m-1$ for sample size n and probability of success p = 0.5: it gives us the probability that of n random trials, the number of successes will fall between 0 and $m-1$, inclusive. Thus, $1 - B(m-1, n, 0.5)$ gives us the probability that m or more successes out of n are due to chance.

As an example, consider Table 4, in which we combine the independent measures from Tables 1 and 3 and note whether they represent a success for *DIAG-orig* or *DIAG-NLP*. The probability of 8 successes out of 9 measures is p = 0.020 ($1 - B(7, 9, 0.5)$). If we leave *Total Time* out because

	DIAG-orig	DIAG-NLP
Total Time		$\surd$
Indicator consultations		$\surd$
RU consultations		$\surd$
Parts replaced		$\surd$
Essay score		$\surd$
Usefulness		$\surd$
Helped stay on right track	$\surd$	
Not misleading		$\surd$
Conciseness		$\surd$

Table 4: Successes for *DIAG-orig* and *DIAG-NLP*

it may not be an independent measure,[1] the probability of 7 successes out of 8 is p = 0.035 ($1 - B(6, 8, 0.5)$). Finally, if instead of using the four subjective measures we use their average (which constitutes a success for *DIAG-NLP*),[2] we obtain p = 0.016, and if we eliminate time in this last case, we obtain p = 0.031. To conclude, in whatever way we combine these measures, we obtain evidence that the better scores *DIAG-NLP* obtains, albeit individually not statistically significant, cumulatively show that *DIAG-NLP* outperforms *DIAG-orig*.

However, we have not discussed Table 2 yet. This table shows that subjects in *DIAG-orig* remember what they did better than those in *DIAG-NLP*. The measures concerning indicators achieve or show trends towards statistical significance: indicator precision and indicator F-measure are significant (t-test, respectively 2.19, p = 0.04 and 2.51, p = 0.02), and indicator recall is marginally significant (Mann-Whitney, U = 93.5, p = 0.08). All in all, this is a puzzling result, especially because subjects in *DIAG-orig* consult the system on indicators almost twice as many times as the subjects in *DIAG-NLP*, thus we would expect them to have more problems remembering what they did. Perhaps this result can be related to (Kintsch 1998), that shows that high-quality text does not necessarily lead to better performance.

Finally, the reader may wonder what happens to the cumulative effect that shows *DIAG-NLP* better than *DIAG-orig* if we take into account the measures in Table 2 as well. By adding to Table 4 two successes for *DIAG-orig*,[3] we compute the probability of obtaining 8 suceesses out of 11 measures

[1]One could argue the time went down because of the smaller number of consultations.

[2]A reviewer pointed out that including all four measures gives much more weight to the subjective measures than e.g. to the single *essay score* learning measure.

[3]Given their definitions, precision and recall cannot be considered as really independent measures, and certainly the F-measure that combines them is not independent from either of them. So we synthesize Table 2 as two successes for *DIAG-orig*, one for indicators, one for RU's.

by chance. We obtain p = 0.113, which shows a non significant trend in the predicted direction. However, recall that we are being conservative: for example, we counted *help stay on right track* in favor of *DIAG-orig* even if it is a tie; if we count it in favor of *DIAG-NLP,* p goes down to 0.033.

Current and future work

The results of the study we just discussed make us confident that it is not necessary to add a full fledged NL generator to an existing ITS or to change the ITS original tutoring strategies to obtain reasonable results. Better language can be added at a relatively low cost (the implementation took one graduate student six months), and it can be effective.

As a consequence, we are now pursuing two lines of research. The first is to add some more sophisticated NL techniques without plunging into full text planning, because we want to see how far the weak approach can go. Second, we are conducting a constrained data collection to help us discover some empirical foundations on which to base the realization of the facts the ITS intends to communicate.

We now discuss both efforts in more detail.

Focusing and rhetorical relations

In the work done so far, we imposed coherence on the tutor turn by means of aggregation rules. However, the turn could be made more coherent by introducing appropriate referential expressions (generated ad hoc so far), and a few domain or rhetorical relations among the facts to be expressed. For example, the fact that the ignitor assembly never causes the abnormal indication in Figure 2 as opposed to the other parts within the oil burner always causing it, could be given more prominence if the relevant propositions were linked by a *contrast* relation[4] rendered via an appropriate cue phrase, such as *but* ((A) and (B) are used later to refer to the appropriate part of the explanation):

(1) *The visual combustion check indicator is igniting which is abnormal in startup mode. Normal in this mode is combusting.*

(A) Within the oil burner, the oil nozzle, oil supply valve, oil pump, oil filter and burner motor always produce this abnormal indication when they fail. (B) But the ignitor assembly never does.

In ongoing work, we have coupled EXEMPLARS to a knowledge base built via the SNePS representation system (Shapiro & Rapaport 1992). SNePS is a semantic network formalism where each node represents a proposition. In general, it is very difficult to access the knowledge about the physical structure of the system and causal relationships in VIVIDS-based tutors. These types of knowledge are often only indirectly present: they are reflected in how changes to graphical objects affect other objects, but this is not sufficient to generate language. When they are present, they are expressed in a very non-symbolic way. In a sense we need to extract some of this knowledge from the existing tutor and represent it in a usable form for the NL generator — this was done in *DIAG-NLP* by representing the required knowledge via Java classes, as EXEMPLARS is written in Java (SNePS is written in LISP, communication between the different systems is achieved via a Java API).

The need to replicate some of the knowledge present in DIAG may be seen as inconsistent with our earlier claim that we treat the ITS as a blackbox. Actually, we intended that claim to apply only to the tutoring strategies the ITS embodies, not to its underlying knowledge. However, it is certainly true that a full fledged blackbox approach cannot work if the ITS is built without taking into account the knowledge needed for communication. For example, the CIRCSIM tutor embodies domain knowledge at three different levels, because it was found that all three levels are necessary for communication (Khuwaja *et al.* 1992), even if only one level is directly relevant to the material to be mastered.

SNePS make it easy to represent and reason about entire propositions, not just about objects. For example, it is straightforward to represent the various individual propositions that underlie Ex. 1 above, and the causal relations between the failure of the individual parts and the abnormal state of the visual combustion check. Moreover, it is also easy to represent the contrast relation between the two complex propositions (A) and (B). Finally, because propositions are full fledged entities in the representation, they can become part of the discourse model, and be referred to with appropriate referential expressions. In this version of the generator, we implemented the GNOME algorithm to generate referential expressions (Kibble & Power 2000).

This revised version of the generator renders the same facts underlying Figure 2 as shown in Figure 3. The deictic *This* is generated by the GNOME algorithm and is used to refer to the proposition representing the abnormal state of the visual combustion check indicator; this cuts down on some of the repetitiveness of the feedback generated by both *DIAG-orig* and *DIAG-NLP*, cf. Figure 2. However, the indefinite articles in Figure 3 are incorrect (the algorithm we implemented does not take into account the visual context, or the fact that there is only one part with that description). The contrastive particle *but* is not included because we have not yet implemented exemplars to generate cue phrases; however,

[4] We are using relations from Rhetorical Structure Theory (Mann & Thompson 1988).

```
A visual combustion check indicator is igniting in startup mode.
The visual combustion check indicator igniting in startup mode is abnormal.
Within the furnace system,
This is sometimes caused when a system control module replaceable unit is inop-
erative.
Within the oil burner,
This is never caused when an ignitor assembly replaceable unit is inoperative.
This is sometimes caused when a burner motor, oil filter, oil supply valve, or
oil nozzle is inoperative.
```

Figure 3: Adding a bit more of sophistication to the generator

as soon as we do so, it will be straightforward to generate it, as the appropriate rhetorical relation is included in the SNePS representation of the message to be conveyed.

First observations of human consulting

The aggregation rules we implemented in EXEMPLARS appear to be plausible, but they have no empirical foundation. To understand how a human tutor may verbalize a collection of facts, we are collecting tutoring dialogues between a student interacting with the same DIAG application we have previously discussed and a human tutor. In this experiment the tutor and the student are in different rooms, sharing images of the same DIAG tutoring screen. When the student exercises the consult function the tutor sees the information that DIAG would use in generating its advice — exactly the same information that DIAG gives to EXEMPLARS in *DIAG-NLP*. The tutor then types a response that substitutes for DIAG's response. Although we cannot constrain the tutor to provide feedback that includes all and only the facts that DIAG would have communicated at that specific moment, we can still see the effects of how the tutor uses the information provided by DIAG. As of this writing, we have preliminary observations of several human tutors, consisting of about 200 human responses to DIAG consult requests, in over 20 sessions.

The two most striking patterns we observe in the human-generated advice are 1) they often eschew syntactic aggregation of part lists and instead describe or name functional aggregations of parts, and 2) they give advice on the problem-solving process, either directly or indirectly. In the following examples, the pairs of utterances show two tutors independently describing the same assemblages of parts and giving similar problem-solving advice:

1. Referring to oil nozzle, supply valve, pump, filter, etc:
   a) "…check the other items on the fuel line" [Tutor 1]
   b) "…follow the path of the oil flow" [Tutor 2]
2. Referring to all the burner parts:
   a) "What are the parts that control the combustion?" [Tutor 1]
   b) "…consider the units that are involved with heating the water" [Tutor 2]

The assemblages we see in the human discourse are not necessarily represented in the training documentation or the functional diagrams on the DIAG screen; it appears the tutors are constructing them. In general the assemblages seem to be fixed collections. But the tutor sometimes constructs an impromptu subset according to the discourse context, as in "the valve is open, so you have to check the point below the filter," which appears to be a reference to the parts in the fuel line "below" the filter.

The problem-solving advice generally conforms to the patterns of "point-to" and "convey-information" hints observed by (Hume *et al.* 1996).

Some of the other phenomena we have observed:

- In contrast with DIAG, tutors less often mention parts that *cannot* be causing the problem (e.g., the ignitor assembly in Figure 2), except when the student consults precisely on those parts.
- Tutors frequently introduce devices for inter-turn coherence. For example, two adjacent turns were introduced by "not a good choice" and "better choice," respectively. Another turn was introduced by "the question is now," indicating the reasoning was in some way following from the previous turn.
- The human tutors occasionally justify a statement, frequently by appealing to causal reasoning. For example, one tutor wrote "The oil filter is normally clean. A dirty and clogged oil filter *blocks the flow of oil* and should be replaced" (emphasis added). By contrast, DIAG merely states whether a broken oil filter can cause the problem, without interpolated explanation.

As our experiments with human tutors continue, we should be able to produce a more complete catalog of language and discourse phenomena. Of particular interest, given our emphasis on aggregation, is the parts assemblages the tutors use, especially how they are described and when they are invoked, and how to organize the knowledge the system needs in order to replicate human tutors.

Acknowledgements. This work is supported by grants N00014-99-1-0930 and N00014-00-1-0640 from the Office of Naval Research, Cognitive, Neural and Biomolecular S&T Division. We are grateful to CoGenTex Inc., in particular to Mike White, for making EXEMPLARS available to us; and to Michael Scott for suggesting the binomial cumulative distribution.

References

Bateman, J. A. 1994. KPML: The KOMET-Penman (Multilingual) Development Environment. Technical report, Institut für Integrierte Publikations- und Informationssysteme (IPSI), GMD, Darmstadt. Release 0.6.

Di Eugenio, B., and Trolio, M. J. 2000. Can simple sentence planning improve the interaction between learners and an intelligent tutoring system? In *Building Dialogue Systems for Tutorial Applications (AAAI Fall Symposium)*. American Association for Artificial Intelligence.

Elhadad, M. 1993. FUF: the universal unifier – user manual version 5.2. Technical Report CUCS-038-91, Columbia University.

Evens, M. W.; Spitkovsky, J.; Boyle, P.; Michael, J. A.; and Rovick, A. A. 1993. Synthesizing tutorial dialogues. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, 137–140. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Freedman, R. K. 1999. Atlas: a plan manager for mixed-initiative, multimodal dialogue. In *AAAI99 Workshop on Mixed-Initiative Intelligence*. Orlando, FL: American Association for Artificial Intelligence.

Graesser, A. C.; Wiemer-Hastings, K.; Wiemer-Hastings, P.; Kreuz, R.; and the Tutoring Research Group. 2000. Autotutor: A simulation of a human tutor. *Journal of Cognitive Systems Research*.

Hume, G. D.; Michael, J. A.; Rovick, A. A.; and Evens, M. W. 1996. Hinting as a tactic in one-on-one tutoring. *Journal of the Learning Sciences* 5(1):23–47.

Khuwaja, R. A.; Evens, M. W.; Rovick, A. A.; and Michael, J. A. 1992. Knowledge representation for an intelligent tutoring system base on a multilevel causal model. In Frasson, C.; Gauthier, G.; and G.I.McCalla., eds., *Intelligent Tutoring Systems, Second International Conference*.

Kibble, R., and Power, R. 2000. Nominal generation in GNOME and ICONOCLAST. Technical report, Information Technology Research Institute, University of Brighton, Brighton, UK.

Kim, J. H.; Glass, M.; and Evens, M. W. 2000. Learning use of discourse markers in tutorial dialogue for an intelligent tutoring system. In *COGSCI 2000, Proceedings of the 22nd Annual Meeting of the Cognitive Science Society*.

Kintsch, W. 1998. *Comprehension. A paradigm for cognition*. Cambridge University Press.

Mann, W. C., and Thompson, S. 1988. Rhetorical Structure Theory: toward a Functional Theory of Text Organization. *Text* 8(3):243–281.

McRoy, S. W.; Channarukul, S.; and Ali, S. 2000. Text realization for dialog. In *Building Dialogue Systems for Tutorial Applications (AAAI Fall Symposium)*. American Association for Artificial Intelligence.

Moore, J. D.; Lemaire, B.; and Rosenbloom, J. A. 1996. Discourse generation for instructional applications: Identifying and exploiting relevant prior explanations. *Journal of the Learning Sciences* 5(1):49–94.

Munro, A. 1994. Authoring interactive graphical models. In de Jong, T.; Towne, D. M.; and Spada, H., eds., *The Use of Computer Models for Explication, Analysis and Experiential Learning*. Springer Verlag.

Rosé, C. P.; Di Eugenio, B.; and Moore, J. D. 1999. A dialogue based tutoring system for basic electricity and electronics. In *AI-ED 99, Proceedings of the 9th International Conference on Artificial Intelligence in Education*.

Shapiro, S., and Rapaport, W. 1992. The SNePS Family. *Computers and Mathematics with Applications, Special Issue on Semantic Networks in Artificial Intelligence, Part 1* 23(2–5).

Towne, D. M. 1997. Approximate reasoning techniques for intelligent diagnostic instruction. *International Journal of Artificial Intelligence in Education*.

Trafton, J. G.; Wauchope, K.; Raymond, P.; Deubner, B.; Stroup, J.; and Marsch, E. 1997. How natural is natural language for intelligent tutoring systems? In *Proceedings of the Annual Conference of the Cognitive Science Society*.

White, M., and Caldwell, T. 1998. Exemplars: A practical, extensible framework for dynamic text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, 266–275.