AI Research in Natural Language Generation

Prof. David D. McDonald, Principal Investigator [1]
University of Massachusetts at Amherst

Final Report

Reporting work done for the **Rome Air Development Center** under the Post-Doctoral Program: subcontract F302602-81-C-0169 to Syracuse University, task I-7-4398; subcontract F30602-81-C-0193, task I-7-4410 to the Southeastern Center for Electrical Engineering Education (SCEEE).

RADC Task Engineer: Capt. R. L. Russel, Jr. RADC/IRDT

Time period: nominally June 15, 1987 to September 30, 1987, with a no-cost extension (Syracuse only) through February 29, 1988.

Summary

Paraphrasing the task description that authorized this effort, the objective was to extend RADC capabilities in natural language generation by studying the interfacing of the natural language generation linguistic realization component Mumble-86 to the expert mission planning system KRS.

The primary element of this effort was the development of an in-depth understanding of the technical issues involved in such an interface. This element was very successful: a general interface framework was designed and implemented based on the experience with KRS (see ref. 2), and several RADC personnel were thoroughly initiated in the design principles as well as general issues in the use of Mumble-86. A secondary element, the

[1] Present position and address: Senior Scientist, MAD Intelligent Systems, Text Group. 55 Wheeler St., Cambridge, MA 02138. Phone: (617) 492-1982, 661-4514.

actual installation of a joint Mumble/KRS system, received relatively little emphasis due to the delayed initiation of the funding for the contract.

Technical Progress Achieved

1. Background

Effective presentation of data and explanation of reasoning is a crucial element of any knowledge-based system's performance if it is to be accepted by its intended user community. This communication is most likely to succeed if it is done in the medium that its users are already accustomed to, which for a majority of situations is a natural language, e.g. English.[2] Computers have been presenting English messages to their users practically since they were first used, but almost without exception these messages have been "canned", i.e. complete, predefined, stored strings of characters that were directly embedded in the code of the programs and printed when the corresponding subroutine was called.

While canned text is adequate and even preferable for some applications (e.g. error messages), by its nature it cannot be adapted to new situations except by painstaking programmer effort. In addition, a design based on canned text is easily overwhelmed by the demands posed by today's knowledge-based systems, which can have dozens of completely different communications needs and are continually constructing new objects and events whose descriptions cannot be canned ahead of time. For these reasons, knowledge-based systems require dynamic, online text generation if they are to have the flexibility and naturalness that their users need.

Over the course of the past decade, natural language generation systems, typically running in experimental settings independent of any practical systems, have achieved a significant competence in the low-level, linguistic aspects of text generation. Furthermore, enough forays have been made into the little understood high-level aspects: coherence, style, perspective, planning, etc., that we are able to carefully define the issues and the degrees of difficulty involved in their research and development. The salient problem now is to determine the requirements and priorities on existing generation systems and their near-term extensions so that they be used with actual, independently developed knowledge-based systems, while at the same time looking forward towards the long-term problems that will cap the level of performance our near and mid -term projects can achieve.

[2] For best effect, expert system presentations should involve all available and customary media, for example maps, graphs, tables, spoken as well as printed output, etc. Our effort, however, was concerned strictly with printed presentations (i.e. displayed on a VDU) of dynamically generated English text.

2. Specific focus

The goal of this effort was to learn what is required in order to extend the natural language generation and explanation capabilities of a conventional, knowledge-based expert system by interfacing it to a highly capable independent language generation component. The system chosen for the work was the "Knowledge Replaying System" (KRS) previously developed for RADC by the Mitre Corporation (Dawson et al. 1987). The generation component was Mumble-86, which was contemporaneously being completed at the University of Massachusetts at Amherst under funding from DARPA.

The KRS system, as installed at RADC when this effort commenced, had a minimal generation system already in place. This system used the so-called "direct replacement" algorithm (see McDonald 1986) along with an ad-hoc notation for associating strings of words with units in KRS's conceptual model. Though difficult to extend, that system was capable of producing texts like the one below. (The KRS data structure that it was generated from--a rule--is given just below it.) The purpose of the text was to facilitate editing the rule on-line.

text produced by KRS

By TARGET-AIRCRAFT-2: There is a severe conflict between the target and the aircraft since:
1: DATA: The target of OCA1002 is BE50318
2: DATA: Part of BE50318 is BE50318-SEARCH-RADAR
3: INHERITANCE: BE50318-SEARCH-RADAR is active
4: DATA: The aircraft of OCA1002 is F-111E and F-111E is not a F-4G.

the rule that was the source for the text

```
(constraint-state
  ((tbelow  ((aircraft  (tgtacp1)) (target (tgtacp1)) ))
    (tnewhere nil)
    (toldabove ( (aircraft  (tgtacp1)) (target (tgtacp1)) ))
    (tnewhere nil)
   ...
   (((target-aircraft-2
                      (data (target OCA1002 BE50318))
                      (data (powa BE50318 BE50318-SEARCH-RADAR))
                      (inheritance (is-a BE50318-SEARCH-RADAR
                                        ELECTRONICS))
                      (data (aircraft OCA1002 F-511E)
                            ((NOTEQ F-111E 'F-4G)))
           ...)
```

We took this text and its corresponding rule, and the few others like it in KRS, as our starting point. They are representative of the state of the art when a sophisticated generation component is not used. In the course of our work we identified the problems such texts exemplify, and determined near, middle, and long-term recommendations for how these problems can be rectified.

3. Problems identified

As a text, this example and the others like it have two kinds of problems. The most obvious is that they are unnatural -- a person giving the same information would speak with markedly greater fluency. The second, more subtle, problem is that they are also potentially misleading.

Fluency, per se, is fairly easy to improve once one can take advantage of a generation component with a sophisticated model of English grammar. The text of the example was unnatural because it consisted solely of short, uniformly constructed, single clause sentences. Any generator with a model of clause combining principles and a syntactic description of each of the individual clauses (which a direct replacement generator almost never has) can combine the short sentences into a long one, or can find alternative realizations for some of the information, say as adjectives or descriptive references, thereby eliminating some clauses altogether.

The rub comes when the generator tries to decide just which of the dozens of combination possibilities it should use, and this is where the problem of being potentially misleading comes in. We need to appreciate that people are not accustomed to being addressed by computers. Unless they are regular computer users, they will assume that the machine intended to communicate the same information that a person would had he uttered the same remark. This information includes facts about the relative importance of the remark's different facets and whether they are new or something the user is presumed to already know.[3] Whenever an audience reads a text--more so when they hear one, they ascribe this kind of contextual information to the text whether it was intended or not. Consequently, the more important the propositional information that a computer system is trying to covey, the more important it is that this sort of contextual information is appreciated and factored into the text generation process.

We can illustrate this by looking at some of the different ways one can more fluently express the first part of the information in the example rule. For example:

The target has an active search radar.

This version couches the information as a property of the target. An alternative could be to phrase it in terms of the status of the radar unit; this would be especially appropriate if the user already knew that the target had a radar protecting it.

The target's search radar is active.

A fluent rendering of the entire rule might then be

The target's search radar is active, but the mission aircraft do not include any F4Gs.

An alternative rendering, with the same information but the opposite focus,[4] would be

The mission aircraft do not include any F4Gs, but the target has an active search radar.

---

[3] These are just two aspects of the sort of "non-propositional" information that people communicate when they speak which happen to be relevant to this example; there are many others.

[4] We understand from the people we worked with at RADC that the standard intent of the rule is for the person planning the mission to change what aircraft are used. This fits the focus of the first version of the text: the "given" information is the choice of target, and the rule points out a consequence of that choice that the person hadn't anticipated in choosing the aircraft to be used. The given/new organization of the text matches the if/then pattern of the rule. In a different situation, however, the same rule could be driven in the opposite direction: Suppose that F4Gs are simply not available for the mission, that KRS knows this, and that its intent is to suggest that the person select an alternative target that is not protected by a radar-guided anti-aircraft battery. In this case the second version of the text has the correct structure.

3.1 Implications of these problems

These sorts of remarks are well within the state of the art of today's generation components. Given only the level of experience that several RADC/COES personnel acquired while at UMass for a week during the summer of 1987, a "specialist-based" interface to Mumble-86 could be developed for the rule model specifically used by KRS in a small amount of time (see "short term recommendations" below). While this would be an improvement in the naturalness of KRS's output, it would be a less than satisfactory solution in two respects:

(1) It would be a special-case solution. To replicate the effort for another system we would have to start over and write comparable specialists and contextual heuristics essentially from scratch.[5] This problem is addressed in the middle-term recommendations.

(2) However good one was able to make the text per se, the information conveyed by a rendering of the literal information in the example rule is an inadequate explanation of the problem for an inexperienced user, and might well mislead an experienced one.[6] This problem stems from a simple lack of the relevant

[5] In a case such as this there are three kinds of information that must be supplied. One is the mapping from the primitive units of the conceptual representation to suitable English phrases, e.g. the term `POWA` ("part of") is linked to *part of* or *has*. The second is the procedures and rationales for combining the units into larger phrases, e.g. *the target's radar* or *the target has a radar*. The third, and the most difficult to come by, is the criteria by which alternatives in mapping or combination are to be adjudicated. This involves linguistic analysis to determine what rhetorical and connotative effects each of the alternatives can have, and also a study of the specific task and the particulars of the system's conceptual model and intentional states in order to determine what conditions in the system should direct the use and choice of these effects.

[6] The F4G aircraft carries special munitions and equipment designed specifically to neutralize the radar facilities of hostile anti-aircraft batteries. This explains their significance to a mission against a target, like the one in the example, that is defended by radar-directed anti-aircraft platforms. The cryptic designation of the search radar in the KRS rule as "electronics" was used by the rule writer to signify that it was the kind of object that emitted radiation that could be tracked back by the F4G's special munitions.

This kind of explanatory information is not given in the rule, and consequently cannot possibly be passed on for the benefit of an inexperienced user of KRS who does not already know about the special properties of F4G aircraft. By the same token, to an experienced user, the information that the search radar at the target "is active" is superfluous: every radar has the kinds of properties that make it susceptible to an F4G. Since a person will always presume that the information they are being told must serve some purpose, the experienced user is likely to look for some interpretation of "active" that would make its information significant, perhaps that the anti-aircraft batteries in question were mobile and that intelligence reports had indicated that they were recently in the area of the target; the actual KRS rule, however, implies nothing of the kind.

information in KRS's conceptual model, and is addressed in the long-term recommendations.

4. Recommendations

Based on the work done in the course of this effort, we can make specific and generic recommendations to guide future R&D efforts in natural language generation. They are grouped into short, medium, and long -term, i.e. roughly estimated as through calendar years 1991, 1994, and 1999 respectively.

The criteria determining how much real calendar time these periods will actually correspond to are firstly how much effort is actually put into generation by coordinated teams within the intellectual community (if, for example, the Darpa community shifts its natural language resources predominantly to speech research, the time-frames will lengthen). Secondly, how much of the effort is devoted to open-ended research versus engineering specific solutions on the basis of the existing technology (the more research, the shorter the time-frames). And thirdly how much academic research on generator occurs (given the luxury of Ph.D. research, new graduates will in all likelihood have tackled problems that are years away for externally-funded research teams). Given the enormous variation possible in how these factors actually work out, the recommendations should be understood as a sequence of staged phases rather than a rigid set of time-frames.

4.1 Short-term recommendations

A generation system can be said to consist of three parts: (1) A linguistic component, with a knowledge of grammar and of how texts may be formed and produced; (2) a planning component, making decisions about what information should be included in the text, how it should be organized, and the perspective, register, and style with which it should be expressed; and (3) an underlying applications program of some sort, the source of the goals the other components are to achieve and of the conceptual model of the application's knowledge and state that the other components work from.

In any practical generation project, the underlying program -- something like KRS -- is usually a given. The linguistic component -- say Mumble-86 -- can be taken off the shelf with perhaps only minor extensions to its grammar to handle new constructions. The planning component, however, is presently going to be specific to each pair of components: planners for new pairs will largely have to be built from scratch.[7] This is because the likelihood of being able to use an "off the shelf" text planning program developed elsewhere for applications systems or realization components even slightly different than one's own is extremely low; sometimes even to adapt its specific architectural principles may be difficult. As a consequence, any new group will invariably be developing the bulk of the text planner on its own. This is likely to be the case for the short term, i.e. the next two to three years, assuming the research community follows its present trends.

The question is what research path will most quickly turn this situation around.[8] Here we see five guidelines that would do well to be adopted by new projects initiated during this period.

-- Be circumspect in the use of schemas (in the sense of McKeown 1985). A schema imposes an externally derived organization on a text (usually of paragraph length), an organization whose particulars are not motivated by the underlying applications program. Their use is a crutch that gives the impression of greater understanding of what it is saying than the application actually has---a serious problem once applications are used interactively because of the likelihood that people will read in an unintended emphasis and perspective as described above. A safer tack is to stick with some variation on direct

7 Text planning is in its infancy as a research problem. (For an overview of the issues in text planning, one should look at the proceedings of the recent workshop on the subject held at the AAAI meeting in August of 1988 and available from the AAAI office.) By contrast, the linguistic issues in generation, because they have received more attention and were able to draw on work on grammars done in other fields, can now be approached with considerable sophistication. Drawing on the general principles and the wealth of experience that have been gained from earlier linguistic realization components, a group newly into generation could build another such component from scratch with a high probability of success (though their effort would be better spent elsewhere). Text planners are a much trickier enterprise.

8 The emphasis here is deliberately on research rather than development. Development only makes sense when the theoretical and technical underpinnings have been established by prior research and the intended applications demand the skill level that the research has achieved. Neither is the case today. Given what is still a very low level of sophistication in the modeling and interpersonal goal-setting capabilities of today's application computer programs, any effort to generate texts from the sources they can realistically provide will be best served by direct replacement generation techniques and canned format statements--getting the programs to motivate anything more sophisticated would be squeezing water from a stone. By the same token, if one were working from an experimental application that might well be able to supply the motivations, the generation technology available in the crucial area of text planning, as noted, is too immature to be the basis of large-scale, generalized development efforts.

replacement as the basis of text structure above the clause level, since this will force the structure to at least indirectly reflect the actual data structures of the application and thus be accessible to any discourse-tracking facilities that the application has (see ongoing work by Beverly Woolf's tutoring group at UMass). In the near term, the inevitable monotony and occasional awkwardness in the large scale structure of the text that this leads to will remind users of the application's limitations.

-- Simplify the development effort by adopting an architecture based on "specialists", hand tailored treatments for each class of object or event the application needs to realize. This is the design used by the programs thusfar most successful at varying their descriptions to fit the circumstances (see especially Hovy 1988). It has the obvious weakness of lack of generality---quite acceptable in the near term---but has the advantage of freeing the experimenter to look for the rationales for generation decisions wherever they might be found, while not requiring these rationales to be applicable across the board. At this point what the field needs is experiments on as broad a front as possible, and the choice of architecture should support this.

-- Presume that all of the rules and code that are developed will be thrown away when the follow-on project is begun. The goal of work during this period is to explore the breadth of problems at issue in text planning. Exploratory work yields prototypes, not polished systems, and any attempt at polish will invariably come at the expense of the breath of the issues examined. The result would be a brittle system, with a restricted architecture that can not accommodate the extensions necessary as further issues become understood.

-- Choose the applications domain with great care. In a research context, the purpose of the application is to challenge the research while not presenting problems that are too great to be surmounted with a year's work. Adopting a domain simply because there is an available expert system is a mistake: the application's domain model and reasoning must be rich enough to support the judgements the generator will be making. The application's authors must be on ready call to answer questions about the intent behind particular primitives and expressions. The very best situation is when the application is still under development and changes can be made in response to the generation researcher's needs.

-- Do not have the goal of creating a "text planning formalism". Without tested general principles to guide one's work, and with only a very few initial experimental systems to take as examples, it is premature to adopt any single discipline on how text planning knowledge is to be encoded analogous to the rigorously defined grammar formalisms now being used in linguistic realization components.

4.2 Medium-term recommendation

By the middle term, there should be a well-established inventory of problems to be solved if text planning is to be done to human standards. There should also be a significant body of experience in how these problems can be approached, though it is unlikely to be consistent when taken as a whole. At this point we should be able to attempt general theories of text planning that address coherent subsets of the problem and could be applied across the board to any applications programs that meet certain minimal standards.

In developing any general theory, it is crucial to make a careful choice of where to draw the line between what will fall within the bounds of the theory and what will be outside it and thus potentially dealt with as a special case. This applies to the reference knowledge to be used as well as to processing and control. Two methodological recommendations are in order for any medium term effort:

(1) That the line be drawn at the boundary of the text planner and the underlying application (i.e. all applications should be able to use the same text planning machinery).

(2) Furthermore, that the generator should support its own, complete, application-independent semantic model of the world.[9] The interface from the application to the generator would then consist of a mapping from the application's knowledge base to this model, plus an indication of the communications goals to be achieved.

Some aspects of the world model will always have to be supplied from within the generator in any event: an application has no motivation to include, for instance, the fact that the pilots that fly the strike missions are male while their targets have no sex and are treated as neuter. Gender is nevertheless grammatically required information that must be supplied whenever an English generator uses a pronoun, so that information must come from the generator. This is not an isolated case. It is normal for today's applications not to represent such things as the temporal information needed to specify

[9] More precisely, the generator should support a model of how the world is characterized in natural language by speakers of English.

tense and aspect, summary information that records common properties of its individuals (*All of the targets ...*), or perspective data that would allow simple choices between lexical pairs like *go*/*come* or *buy*/*sell*, to mention just a few instances that we have explored.

Given that any typical application's output to a generator is always going to have to be augmented, it is a small step, architecturally, to doing this in all cases. Rather than work from the conceptual categories of the application directly, the generator views them indirectly via their correspondences in its own model of the world. This eliminates a perpetual problem in generation projects whereby every change in underlying conceptual representations (some times even every change in the programmers preparing the domain model!) forces adjustments in the internal algorithms and decision criteria of the generator.

Overall, the goal of this methodology is to permit theories of text planning to be developed in the way we see as most likely to yield coherent, extensible results: namely by working backwards from the surface, observable aspects of language. A natural language is, as the term implies, a system organized around the needs and cognitive faculties of human beings. We have no idea what people's real systems for thought and intention are, and thus cannot know what the only known-to-be-effective text planning system starts from. Nevertheless, we do know where the human generation system stops (i.e. with texts of the sort people produce), and can therefore make reasonable assumptions about where a text planner that we would develop should stop given a well-motivated design for the realization component that it uses.

The point of developing a complete, application-independent semantic world model within the generator/text-planner is to take this methodology of working backwards from known data about as far as it can logically go, i.e. to define what a text planner/generator would start from in an ideal system. The independent model would in effect emulate what a theory expects as the organizing semantic categories and compositional principles of thought/language in people, i.e. its conjectures about the principles behind the underlying representation used in the "applications programs" (so to speak) that drive the human generation process. By basing the judgements of the generator/text planner on a set of linguistically motivated set of categories (object types) and compositional principles (rather than the idiosyncrasies of applications developed without language in mind), one has the best chance of developing a coherent, comprehensive theory. Using such a generator then becomes a matter of establishing a mapping from the world model of the application to the independent world model within the generator, and having all of the application's directives to the generator translated according to this metric before they are acted upon.

This methodology was initiated in the "Cicero" component of our project for explaining legal reasoning (McDonald & Pustejovsky 1987), but not followed through to a fully working system because of the departure of personnel. Another design with essentially identical motivations is the uniform "upper structure" employed in the interface required by the Penman generator at ISI -- a uniform set of topmost concepts in the highest (most abstract) layers of the taxonomic lattice that defines the underlying application's world model; see Matthiessen 1987.

4.3 Long-term recommendation

While local, specific representational deficiencies can be accommodated by extended modeling within the generator, we cannot escape the fact that a generator cannot say things that its application doesn't know.[10] Consequently, in the long term, the limitations on the quality of the text that can be generated will come from the application rather than the generator. There will be limitations for two principal reasons:

(1) missing information that the human user would deem relevant,

(2) failure to represent the context in which information appears.

The second of these is easiest to address. Application programs, because they are not concerned with the details of communication with people and do not have to commit their information to words, can and possibly should remain ignorant of the subtleties of perspective and connotational that pervade human communication. But they must provide the generator with the information it needs to cope with distinctions based on these subtleties, even if they have no need to represent it explicitly themselves.

To take a simple example, commercial transactions involving money are described by people in terms like *buy* or *sell*, the choice depending--in a not well understood way--on the perspective from which the transaction is being reported. However, very few

[10] At this juncture it is worth pointing to a generation technique reduced to practice at APEX systems (Cambridge, Mass.) by Bill Woods, which is an excellent way to allow a system to "generate" extensive fluent texts even when the explicit knowledge of the application is severely impoverished. What one does is have human authors prepare full texts of paragraph or longer length as appropriate to the subject the program is to talk about. The texts are then parameterized using a simple interactive interface in order to allow amounts, dates, names, etc. to be treated as variables that are substituted for at runtime when the parameterized text is used. If the only things to be substituted are simple NPs and names then the technique will work strikingly well; it will fail as soon as the text to be substituted for the variables requires a grammatical adjustment according to the textual context in which it appears, for instance attempting to substitute an event realized as a clause. APEX used the technique to prepare customized financial reports for the clients of its financial planning program.

applications would have any independent motivation for representing such transactions as anything other than a single event (i.e. one complex data object) which lists the identity of the buyer and seller as instantiations of its parameters.

When presented with this transaction object and told to produce a sentence from it, a generator is forced to arbitrarily commit to one or the other perspective (*buy* or *sell*). It doesn't have to be this way. An application only has to provide a consistent, explicit representation of why it is interested in the transaction event just now, and for it to be possible to probe that "something" to determine which was more significant for its present purposes, the buyer or the seller (a notion referred to as "salience").[11] If this can be provided there is no need for the application to change its local representation (e.g. to move to a redundant model where each of the perspectives is spelled out as a distinct representation of the event with its own data object).

To say that the designers of an application "only" have to make certain provisions may be to presume a great deal. There are scientific problems in asking for an explicit representation of an item's relative salience (significance, centrality to the processing), as there are almost no theories of what salience consists of or how it could be manifest.

The ultimately more important problem, however, is not scientific but social. What we need is a change in the attitudes of the people in the knowledge representation community who are preparing the applications. Applications can presently perform most of their assigned jobs without recourse to a generator, and at the present time it is undeniable that supporting a generator's representational needs is an additional burden that application designers and programmers only take on altruistically or out of curiosity. Unfortunately altruism is hard to sustain in the face of project deadlines.

Frankly, the only way to change this state of affairs is to make it worth the programmer's while. This will not be easy: Even though it is possible to see many direct advantages of an integrated generation facility to the designers and programmers of an application's world model (for instance in semi-automatic documentation, easily read reports and process traces, explanations of procedures, and the like), these are not available today and will not be any time soon unless a concerted effort is made. In the meantime, however, there is a methodological step that can be taken that will greatly expedite the process and, indirectly, improve the situation with the first limitation, the lack of information a human user would deem relevant.

[11] For a discussion of what this comes to in the context of an appointment manager application, see McDonald 1988.

We should set up as a long-term goal the implementation of the principle that **anything that an application can represent it should be able to say**. Put another way, this goal requires all application/generator interfaces to be prepared in such a way that every atomic, declarative data structure in the application would be acceptable to the generator as a text source that would lead to an English phrase expressing the information it represented.

Achieving this requires some flexibility in the generator so that the range of data types it is prepared to select is suitably broad, but primarily it requires diligence on the part of the applications programmer. Every generator provides some sort of user interface by which programmers indicate those linguistic correspondences of their data structures that cannot be determined by rule (e.g. proper names, idioms, noun-noun compounds, other arbitrarily organized phrases, and all the single term open-class words)

To make the principle work, the builders of the application's world model must avail themselves of this interface every time they define a new term in their model. Obviously the generator interface must be made friendly enough that they will not find this an onerous thing to do. The point of this is to continually make the designers/programmers aware of the consequences of their choices for what the generator is able to do. When the correspondence-specifying interface between the world model and the generator's lexicon is easy enough to use that these non-computational linguists will really use it, they will be able to see first hand how good or bad the fit is between the information they intended their new data structures to embody and the information actually conveyed by the texts those data structures engender. The hope is that this experience will make it apparent to them what they would have to do to improve the fit, and the immediate feedback will encourage them to actually do it.

In conclusion, if one wants computer programs to be able to use natural language with the same understanding of what they are doing as people have (thereby eliminating any possibility of pathological misunderstandings), then there is no alternative to coordinating the construction of the model that drives the generation with the construction of the one that actually drives the application.

Chronology of the effort

1. Getting the contracts in place

The initial discussions that led to this effort took place in the spring and summer of 1986, and revolved around a White Paper: "Directions for Research on Natural Language Generation". By June of 1986 a draft proposal was in place and efforts were underway within RADC to secure funding for it.

In November of 1986 it was determined that the post-doctoral program was the appropriate funding vehicle and a budget was roughed out for it based on a February 1st start date. It was presumed that about $35,000 was available and that the overhead rate for going through the post-doctoral program was going to yield $30.5k for direct costs. This turned out to be incorrect, as the initial assumption that UMass overhead requirements could be side-stepped through the use of the post-doctoral program was not sustained. Final, accurate budgets were prepared in mid-February after the initiating paperwork became available from the two subcontractors that RADC used to manage the funds. The ultimate amounts that were put to direct costs were roughly $4.5k for salary and travel by the principal investigator (SCEEE), and $16k for salary and travel by the student research assistants (Syracuse). The reason why the disbursement was divided across the two subcontractors was never made clear.

After February, the UMass Office of Grants and Contracts and the two subcontractors proceeded to take a small eternity to actually process the paperwork. It took until mid-June 1987 for the Syracuse contract to be finalized and the UMass internal account set up, at which point the work commenced. (The SCEEE contract required an additional two months to finalize, but travel funds were taken from the Syracuse contract to cover P.I. travel during the interim.)

2. The work that was performed

The first work directly under the contract was a familiarization meeting at RADC by Professor McDonald and his senior graduate student, Marie Meteer, in late June. On this one day trip a current copy of Mumble-86 was installed on COES' Symbolics Lispmachines; we were thoroughly briefed by Sharon Walter on the relevant parts of the KRS rule system including hardcopies of much of the code to study back at UMass; and we collectively made plans for the rest of the summer's work.

The first half of the month of July was spent at conferences on the West Coast. During this time the groundwork was laid for what eventually became the "Water to Wine" paper presented the following February at the Applied Computational Linguistics meeting. This work was largely a reaction to the weakness and inflexibility of the KRS rule representation as a text source. It is the primary research result of the project, and establishes a framework for a pragmatic approach to near and middle term development projects that must generate from conventionally designed applications programs. A copy of the paper is included with this report.

During the second half of July, a two day tutorial was prepared presenting the technical issues and state of the art in natural language generation, with special attention to the generation projects supported by DARPA. This tutorial was presented at RADC at the end of the month to an audience averaging about a dozen people.

In August the tables were turned: Sharon Walter spent a week at UMass getting experience in preparing an interface to Mumble-86 with the assistance of the students in the Mumble Development Group. (Doug White was also on campus for that week reviewing a contract, and spent some of his time working on Mumble.)

In September of 1988, as expected, Professor McDonald left the University of Massachusetts, spending the rest of the grant period effectively on sabbatical. The final month of direct contract work (September) was primarily spent finishing the documentation technical report for Mumble-86. The other significant activity that month was the RADC workshop on natural language processing (9/21-23) organized by Sharon Walter, which McDonald attended. McDonald had profitable discussions with RADC personnel Sharon Walter, Doug White, Bob Russel, and Nort Fowler, as well as with the other invited participants.

An important item coming out of this meeting for Professor McDonald was the fact that there have never been evaluation standards established for research in natural language generation---it has never even been established what it is one would measure! This has led to discussions by McDonald and Marie Meteer (now at BBN) with Sharon Walter urging that RADC find a way to support a special workshop chartered to develop the technical basis for establishing generation standards.[12]

After Professor McDonald's departure from everyday contact with UMass, day to day supervision of the three graduate research assistants who continued to be funded on the Syracuse no-cost extension of the contract was turned over to Dr. Beverly Woolf, who is herself supported by RADC/IRDT funding in a joint project with SUNY Buffalo under the auspices of the AI Consortium. Anticipating that the three students would eventually come under the support of that effort, we gradually shifted the emphasis of their work in the course of the semester from support for the Mumble-86 program to establishing the requirements for the new effort.

During the next six months McDonald made a number of colloquium presentations based in part on research performed as part of this contract. The places and dates are listed below.

[12] A first cut at this workshop took place as part of the "Workshop on the Evaluation of Natural Language Processing Systems" December 8,9 1988 in Wayne, PA. However the limited time available and the mixed focus meant that the participants barely had time to scratch the surface of the problem. A workshop specifically addressing the evaluation of generation systems is highly recommended.

In February Professor McDonald visited Sharon Walter and Doug White for a one-day briefing on the work that had been done on the contract during the fall. He presented a composite of the colloquium talks he had given on RADC-based work, including the "Water to Wine" paper that was then presented by Marie Meteer the following week at the meeting on Applied Natural Language Processing in Austin Texas. Also discussed was the model developed by McDonald and Meteer for an "orchestration" component to act within the later stages of a generation text planning system, and their model of "semantic templates" for systematizing the procedure of preparing input realization specifications for Mumble-86 from selected program-internal data structures. This briefing was the last work done under the support of the contract.

3. Talks, colloquia, etc. presenting this work:

Applied Expert Systems (APEX), October 27, 1987.

University of Montreal Linguistics Dept., October 29, 1987.

ATT Bell Laboratories, December 8, 1987.

NYU Computer Science Department, December 9, 1987.

2d Conference on Applied Natural Language Processing (ACL), February 9, 1988.

University of Buffalo Computer Science Dept., March 10, 1988.

General Electric R&D Center, Schenectady NY, April 14, 1988.

4. Papers published/prepared referencing this contract:

1. Marie Meteer, David McDonald, Scott Anderson, David Forster, Linda Gay, Alison Huettner & Penelope Sibun, **Mumble-86: Design and Implementation**, TR #87-87 Dept. Computer & Information Science, UMass., September 1987.
2. David McDonald & Marie Meteer (Vaughan), "From water to wine: generating natural language texts from today's applications programs", 1988 conference on Applied Natural Language Processing (**ACL**), Univ. of Texas at Austin, February 9-12, to appear; available as TR #87-51 Dept. Computer & Information Science, UMass., August 1987.
3. Penelope Sibun, Alison Huettner & David McDonald , "Directing the Generation of Living Space Descriptions", **COLING-88**.

4. Marie Meteer & David McDonald, "The Orchestration Process in Text Planning for Generation", failed submission to the ACL annual meeting; available as a manuscript from the authors, 9pgs.

Additional references cited

Hovy, Eduard (1988) **Generating Language under Pragmatic Constraints**, Lawrence Erlbaum Associates, Hillsdale, N.J.

Matthiessen, Christian (1987) "Notes on the organization of the environment of a text generation grammar" in Kempen (ed.) **Natural Language Generation**, Martinus Nijhoff, Dordrecht, The Netherlands, pp. 253-278; distributed by Kluwer Academic, Higham Mass.

McDonald, David (1986) "Natural Language Generation", in Shapiro (ed.) **The Encyclopedia of Artificial Intelligence**, John Wiley & Sons, in press, pp.642-655.

McDonald, David & James Pustejovsky "The Councelor Project at the University of Massachusetts", **Computational Linguistics** 12(2), Summer 1986, Finite String Newsletter, pp. 139-141.

McDonald, David (1988) "On the Place of Words in the Generation Process", Fourth International Workshop on Natural Language Generation, Catalina Island (Information Sciences Institute/USC), July 18-21, 1988; publication of the proceedings as a book is anticipated.

McKeown, Kathleen (1985) **Text generation: Using discourse strategies and focus constraints to generate natural language text**, Cambridge University Press, Cambridge, England.