FINAL REPORT
AND
SOURCE CODE

Deliverable No.: B004 & B005

Prepared by:

JAYCOR

Prepared for:

Naval Research Laboratory
Washington, DC 20375-5000

In Response to:

Contract #N00014-85-C-2444

12 May 1988

Issues in the TERSE Project - A Final Report

M. Kathryn Di Benigno

JAYCOR

ABSTRACT

In this report, several important issues relating to the TERSE project are discussed. This report constitutes the deliverable B004 of contract N00014-86-C-2444. The description for this deliverable is a *final report describing work done during this phase.* Detailed descriptions of the work done for other deliverables during this phase have already been provided. This opportunity is being taken to deliver some work and opinions that have been formulated not only during this phase, but during the entire life of the contract. These are the less tangible results of our work that we believe are of importance to all parties that have been involved in this effort. In this report, we address some of the concerns about transitioning this project to the *real world* and also we discuss in detail, some problems that we foresee with the past approach to knowledge representation.

Issues in the TERSE Project - A Final Report

M. Kathryn Di Benigno

JAYCOR

1. Introduction to the CASREP Message Domain

Naval ships are required to send out a CASREP report on every piece of equipment on board which experiences a failure that can not be corrected for a period of time exceeding 24 hours. The messages can be about equipment as strategic as the main engines and as trivial as doors to and from compartments. Our test set of messages deal with failures related only to *starting air compressors* (used to start the main propulsion engines).

The Navy is faced with several problems concerning these messages: The first problem is the shear number of messages coming in. The second problem is to sort out time-critical information from non-time critical information. Another requirement, is to forward relevant information to different organizations. Because CASREP reports contain formatted sections in addition to the free text section, some information detection and dissemination can be done fairly easily with computers when the relevant information is contained in the formatted part. However, the task of processing the information contained in the narrative portions of the messages is a much more difficult one and is not currently being handled automatically. Since this information has been deemed important enough to include in the report, the need to use that information exists.

Our goal has been to parse the text contained in the narrative portion of the messages and map the information transmitted in that section into a computer-based representation. After this mapping is accomplished, specialized application programs can access the knowledge base containing the information and perform their specific tasks. This paper deals with knowledge representation as it relates to the problems of representing information communicated in natural language text. This work is part of the TERSE project which is an effort to understand and make use of information communicated in free text portions of Navy CASREP messages.

2. Representing Written Text

Information conveyed in natural language text can be analyzed at several different levels. There is the discourse level at which we see things like plots developing. There is the assertional level at which basic actions and relationships are described. The discourse level provides the contextual environment needed to interpret subsequent assertions. In text, there is no need to supply a complete description of the state of the world in order to make a new assertion. That information is *remembered* from the previous information from the discourse or otherwise it can be inferred with knowledge about the domain of discourse.

The most primitive level of natural language where complete thoughts can still be conveyed is the *assertional level*. At this level, actors and objects are related to some

event, action, or state. Assertions can be found in nominalized verbs, adjectives, entire sentences or clauses. Assertions have the property of predicating something about the actors and objects related by the assertion. We will call the element which is indicative of an event, state, or action, the *predicate*. The actors and objects that are associated with the predicate are called its arguments. This paper focuses on the representation of assertions.

Processing of the entire discourse is of equal importance but without a good way to represent the main elements of the discourse, it would be premature to address the most of the issues involved at this level. However, to perform a complete analysis and to effect an understanding of the individual predicates, it is necessary to look at the entire context of the discourse. We do not wish to diminish the importance of devising a natural language system that takes into account the entire discourse by only addressing the representation issues at the assertional level in this paper. It is simply the case that this is our starting point and any future work be directed towards integrating this representation into a holistic discourse representation.

2.1. Knowledge Required for the Assertional Level

In the representation of predicates, it necessary to enumerate all the arguments that are central to the semantic definition of that predicate. It is also necessary to include information that tells us from what field of objects in the domain (from what semantic class) each argument can come. An equally important type of information is the semantic role information. This information is the equivalent to the "procedure" element of the equation "knowledge representation = structure + procedure". The procedures making up the semantic role information are what enable the application portion a system to "know" what it knows about the "meaning" of that particular predicate.

There are some general purpose semantic roles that are considered universally interesting. For example, roles dealing with such concepts as intentionality, causality, and instrumentality would be useful to almost any application program. It is interesting to have a concept of event-initiators, entities changed because of the event, entities used to effect the event and other concepts needed for the particular application. These concepts represent the rudiments of understanding. This brief discussion about representing assertions is presented here just to provide the reader with some sense of where our interests lie. This topic will be discussed in more detail in the following sections analyzing the current approach to knowledge representation.

3. The CASREP Implementation

In the Summary project, an approach to representation called **Information Formatting** was adopted. The initial concept that was conceived by Zellig Harris during the late 50's[1]. He developed a method called **distributional analysis** for deriving equivalence classes from natural language discourses. The classes that were derived were referred to as **semantic categories**. Harris preformed this work in an effort to seek "... some global structure characterizing the whole discourse, or large sections of it". Some of the structure that he was seeking was found in patterns of these semantic categories. These patterns were found to vary from one subset of language to another. These subsets of language were named **sublanguages**. Harris believed that through characterizing the structure of a sublanguage, some degree of semantics from that sublanguage would

be discovered.

Theoretically, distributional analysis does not make use of any domain specific knowledge to discover these classes of words. The constituents are classified entirely on their syntactic characteristics and their surrounding environment. This approach gave Harris a methodical way of describing characteristics of narrow domains of discourse that had never been described (at least in this manner) before.

During the 70's, Harris' work was expanded on by Naomi Sager[2] at NYU. She concentrated much of her effort on discovering and employing these "textual regularities" that are characteristic of sublanguages. She developed a process called **information formatting** in which texts were mapped from their raw form into tables. Each of the semantic categories developed for the particular sublanguage is a column in the table. Each line of the table is called a single **format**, except for special lines that perform the function of connecting two lines of the table. These lines are called **connectives**. They introduce a tree-like structure that takes only two arguments. The arguments can either be a single line format or another connective. One of the most important aspects of this work is that the process of mapping natural language text into these formatted tables could be performed by a computer program.

3.1. Characteristics of the Message Domain

The CASREP message domain constitutes a complicated sublanguage. CASREPs range in complexity from simple to complex, and from almost entirely grammatically correct to ill-formed.

TestA 1

Starting air regulating valve failed. Unable to consistently start NR 1B gas turbine. Valve parts excessively corroded.

TestB 7

SAC had local monitoring capability for lube oil pressure only, due to the recent failure of the sac lube oil pressure transducer. Prior to engagement it was reported that SAC LO pressure dropped to zero. No metallic particles in LO filters. Borescope investigation revealed a broken tooth on the hub ring gear. It is likely the LO pump has sheared. The LO pressure and alarm capability is a necessity for operation.
SITREP 002: Drive shaft for SAC was manufactured locally. S/F reinstalled old SAC utilizing new drive shaft. On testing of SAC lube oil pressure could not be adjusted above 35 PSIG. Replacement SAC will be required. The original drive shaft, when installed, was packed utilizing 60 grams of grease, when removed, on failure of SAC, the drive shaft was dry and showed signs of extensive heat stress.

CASREPs can contain a great amount of discussion concerning the actions of the participants in discovering and attempting to correct the equipment failure. The previous two

messages are from the SAC CASREP message set. It is apparent from the second message that this "sublanguage" is not that constrained.

3.2. The CASREP Information Formats

An initial set of semantic categories and semantic patterns was developed in the early stages of this project. After some work was begun on the application end of the Summary system, it was determined that the representation was lacking in its ability to accurately and consistently represent information contained in the texts. This problem was addressed in several different ways. First, several new semantic categories were added. The display below shows the original categories and the categories which were added are shown below.

semantic categories	**example members**
ADMIN	*forward, report, expedite*
FUNC	*rotate, operate, engagement*
INVEST	*check, isolate, troubleshooting*
MSG	*CASREP, message*
ORG	*ship, CCS, technician*
PART	*sac, shaft, gear*
PROCURE	*deliver, purchase, reorder*
PROCESS	*manufacture*
PROP	*clearance, pressure, capability*
REPAIR	*repair, adjust, overhaul*
STASK	*arrival, assignment, transit*
STATUS	*fail, malfunction, loss*

new semantic categories	**example members**
ALARM	*alarm*
AREA	*internal, tip*
PARTLOC	*come*
PIECE	*particle, chunk, material*
REQUIRE	*require*
SUBSTANCE	*metal*

In the revision, connectives were broken down into several different subclasses of connectives. In the original implementation all connectives were classified under a single heading of CONN. The following display shows the new subheadings of CONN.

new connectives

CONJOINED, TIME-CONJ, RELATION, SUB-CONJ,
EMBEDDED, REL-CLAUSE.

In the original implementation, a semantic category could be modified by several different types of modifiers as shown below.

old category modifiers

TIME, LOC, QUANT, EFFORT(man-hours), NEG, MODAL

In the revision, the EFFORT modifier was dropped as there were no occurrences of its use in the equipment failure messages. The TIME modifier was removed from the category level and a block of TIME modifiers was added to modify a format as a whole. This is more appropriate since formats should be assertional elements and time information is more appropriately attached to an assertion than to a single word. The following display shows all the fields of the TIME block record which modifies an entire format.

TIME modifier block

TIME
- verbtense
- change
- beg
- end

EVENT-TIME
- TIME-PREP1
  - prep
  - time-unit
  - reference-point
- TIME-PREP2
  - prep
  - time-unit
  - reference-point

3.3. The CASREP Semantic Patterns

Eight semantic patterns were initially developed for a set of CASREP messages. Six of these patterns are shown below. The two that are not shown were specific to the *electronic equipment* messages and not applicable to the *starting air compressor* messages.

Pattern Number	Pattern Name	Pattern
1	General Administration	*ORG PART* (**ADMIN PROCURE**)
2	Repair	*ORG PART STATUS* **REPAIR**
3	Investigation	*ORG PART STATUS* **INVEST**
4	Part State	*PART FUNC PROP* **STATUS**
5	Assistance	*ORG* **ASSIST**
6	Ship Task	*ORG* **STASK**

Pattern number one contains a parenthesized element that indicates that either an ADMIN word or a PROCURE word co-occurs with the other pattern elements. Both will not be present at once. This suggests that there are two different patterns: one in

which PROCURE occurs and one in which ADMIN occurs. The entry that contains two parenthesized elements indicates that the pattern comes in two flavors where one of the two elements is present. The bold-print items are required to identify the pattern. It is generally the case that all the items in a single pattern can be present at one time in an assertion of that type.

4. Problems with the Current Approach

4.1. Missing Semantic Patterns and Unformatted Information

In the design of information formats for the CASREP messages, there has been a digression from the idea that there exist patterns for every type of information in the sublanguage. Our CASREP patterns cover only the portion of patterns which represent *simple* assertions where the assertion does not contain more than one item of the same semantic category. All other patterns which represent assertions with more than one item classified under a single semantic category and assertions that take other assertions as arguments, have not been described and are missing from our master list of patterns for this sublanguage. Consider the following example from the CASREP domain:

Drive shaft sheared all internal gear teeth from drive adapter hub

LINE#	CONNECTIVE	PART	AREA
	RELATION		
1	*shear*		
2		*drive shaft*	
3		*gear teeth*	*internal*

No pattern developed in this implementation can accommodate two PART entries. In the CASREP implementation, the verb *shear* was treated as a connective and its two PART arguments were put into separate formats. This treatment of the problem introduced another problem because it yielded formats (for each of the arguments) that were non-assertional.

Actually, there are no patterns to handle any connectives. Therefore, any time we have been forced to use a connective (as in the previous and following examples), there has been no pattern to describe that information. The following example shows how predicates that take other predicates as arguments were handled. Basically, the high-level predicate was treated as a connective. The arguments to that connective were then mapped into separate formats. Because the arguments were other predicates and thus assertional themselves, no problem of non-assertional formats was created in handling this particular situation in this way, but still there was no pattern to describe this occurrence in the sublanguage.

Reduced capability of NR4 SAC restricts ships operation.

LINE#	CONNECTIVE	FUNC	PROP	STATUS	PART
	RELATION				
1	*restrict*				
2			*capability*	*reduced*	*sac*
3		*operation*			

From these examples, other implementation-related problems can be seen. In the first example, the prepositional phrase *from drive adapter hub* was not formatted at all. In the second example, the word *ships* was not formated. A significant amount of information from the original text may not be formatted. This is especially true of prepositional phrases. While it is important for a knowledge representation to surpress irrelevant detail, the information which was omitted was often important information used in understanding what was written in the message.

4.2. Inherent Problems with the Current Approach

Although several of the problems mentioned above are related to oversights in the implementation of the CASREP information formats, the most central problem is related to a particular type of information that is not provided through distributional analysis. However, recognizing the need for this missing information requires a shift in the theoretical perspective imposed in the LSP system. In our discussion of this problem, the reader will recognize the markings of a theory of linguistics called **case frame theory**. The theory of linguistics employed by the LSP system (**immediate constituent analysis**) and case frame theory are rather diverse theories. **Immediate constituent theory** analyzes natural language in terms of consituency relationships. **Case frame theory** looks at natural language constituents in terms of their *functionality*. It is this concept of functionality that is not reflected in the current CASREP implementation.

4.3. What Does Distributional Analysis Give You?

To perform distributional analysis on a sample discourse, a set of transformations must be performed where sentences and nominals are normalized into Subject-Verb-Object forms. These transformations and others are described in Harris[1]. Once the sample is completely normalized, a procedure that is also outlined in Harris' paper is applied where equivalence classes of constituents are generated.

The semantic classes that are generated *may* not be as semantically appealing as one might hope. It is possible to find classes of nouns, for example, that seem to have in common only their syntactic position in the test data. One phenomenon that can account for this is the transformation of different deep level cases into the same syntactic position. For example, in each of the following sentences with the same verb, the surface syntactic subject is a different deep structure argument to the verb.

Mary cut the cake.
The knife cut the cake.
The cake cut easily.

This is potentially a problem with using the results of distributional analysis "as is". One must be careful that an occurrence of this type does not cause confusion in the generated noun classes. However, if enough different uses of the nouns involved in this kind of ambiguity can be found in the data, there *may* exist justification for separating them from the same class. It is also not likely that this kind of variability in subject-fronting will be present in a narrow sublanguage although it does occur to a small degree in the CASREPs.

Sac engaged.
We engaged the sac.

example

This problem may arise because the normalized form from which all the analysis is performed is not equivalent to a *semantic normal form*. The set of transformations described by Harris do not take into account these deep structure arguments and therefore the "normalized" SVO structures can still reflect variability in syntactic structure.[1]

Semantic patterns are groups of semantic categories that occur in some regularity in a sublanguage. Because of the above-mentioned problem with semantic categories, care must be taken when looking at the "semantic patterns" which arise in a sublanguage. That is, when one decides that certain sample sentences exhibit the same pattern, the possibility of differing semantic roles in the same syntactic position must be considered.

4.4. What Distributional Analysis Does Not Give You

Semantic patterns found through the process of distributional analysis, do not include the **semantic role** information that describes the *function* of its basic elements. The best that can be hoped for, is that the analysis can provide you with skeletal structures that have unnamed slots but have semantic restrictions on what can be filled into those slots. The semantic restrictions on the slots are provided by the semantic class information. The number of slots that are contained in the semantic pattern corresponds to the number of arguments that are found to occur with the predicate in the sublanguage. Therefore, distributional analysis yielding semantic patterns gives us the term "structure" in the equation "knowledge representation = structure + procedures". The **semantic role** part of the equation represented by the "procedure" term is not provided by this analysis. The representation called *information formats* also, does not give any hint that such information should exist.

5. An Enhanced Representation

In the following sections, an enhanced representation will be discussed. The enhancements being proposed would provide: a more complete specification of the patterns in the sublanguage, a consistent way of handling nested assertional arguments, an explicit identification of predicating elements, and the possibility of being able to combine some semantic and syntactic information in a uniform way.

5.1. A Predicate Oriented Representation

In the enhanced representation we will need to represent the predicates as a unified units that contains the following information or is easily integrated with other knowledge sources containing the following information. In these units we need 1) semantic type information for predicate arguments 2) semantic role information to describe the

[1] Harris does mention that cases will arise where the classes that are justified distributionally, will not fit semantically and that the person performing the distributional analysis can break a class up to give it a more palatable semantic flavor.

relationship of the argument to the predicate and 3) a way to identify the actual predicate from its arguments. It should be noted that if we accomplished only number 1, we would essentially have semantic patterns. But as pointed out earlier, semantic role information is essential and making the predicate central and distinguished is an important point because it makes other aspects of the understanding process easier to deal with. We envision the implementation of this knowledge representation in frame-like[3] units with slots and even possibly with procedures attached to the units and/or the slots. The typical inheritance behavior of a frame system is also assumed.

5.1.1. Semantic Role Information Revisited

In our discussion on Information Formatting and Distributional Analysis, it was shown that this *semantic role* information is what was lacking in our representation. We stated that this information is critical to a knowledge representation and, distributional analysis did not provide that information. As far as we know, there exists no automatable procedure to derive this *semantic role* information. However, we can use distributional analysis to help isolate the arguments to a predicate. But from that point on, we must resort to using our own knowledge even though we do not understand that knowledge about the meaning of the predicate and also our knowledge about the requirements of the application.

To obtain this semantic role information, we take a look at the arguments that distributional analysis has highlighted. With the ultimate application in mind, we decide what knowledge sources would make use of that particular argument. At this point, we have roughly assigned an interpretation to that argument. We assign some name that is descriptive of that relationship to the slot. This is just a tool to help us remember what interpretation we wanted to use. It is the implementation of the knowledge sources which interpret this particular argument that really embody the *nature* of the role we have assigned to that argument.

5.1.2. Primitive Predicates

After going through the process of determining the role for each predicate argument for every predicate in the domain, it is possible that some predicates will appear to be synonymous to other predicates. This synonymy may be exhibited be a) two or more predicates having the same number of arguments and b) the knowledge sources which define the relationships of the arguments to the predicates being identical. It may not be the case that the predicates forming such a set would be considered synonymous in all other situations or applications, but since all knowledge sources were created with respect to the goals of a particular the application, it is possible that for some applications it is not necessary to differentiate the nuances in meaning between some predicates.

These synonymous predicates form sets of predicates that can be described by a single *primitive predicate.* Within that set, we do not need to differentiate meaning. Therefore, in our representation, a frame typically represents a predicate described at an application-specific primitive level and it is likely that more than one predicate in the domain will map into that frame thus being considered essentially synonymous.

5.1.3. The Relation of One Predicate Primitive to Another

Once we have decided what predicate primitives exist in our subdomain, other relationships (besides synonymy) between predicates can be established. For instance, there could be an hierarchy of the predicates where more specific concepts are found under more general concepts. We could thus exploit the inheritance mechanism in a frame system to implement this relationship and others in an efficient way. The ways that predicates should be related to one another depends on the needs dictated by the application.

5.1.4. Improvements Over CASREP Information Formats

The slots of the frames will represent the arguments to the predicate. Because our slots are defined in terms of a relationship to the predicate (semantic role), there will be no difficulties with predicates that take two or more arguments of the same semantic class as was a problem with CASREP patterns. In this situation, each argument although possibly of the same semantic class, is serving in a different semantic role with respect to the predicate. Thus, there will be separate slots to accommodate each of the arguments with no problems created by the fact that two different slots can be filled with arguments into different semantic classes. The semantic class information would be stored as a type restriction on the slot, not as a slot itself as was the case with CASREP patterns.

This representation will not differentiate between predicates taking simple elements as arguments and predicates taking other predicates as arguments. **Each** argument to a predicate, regardless of its complexity, will be filled into a unique slot. This does not cause any concern because it is the **role** relationship that is being stressed in this predicate oriented representation, not the logical complexity of the actual argument. This eliminates the problems caused by representing some predicates as formats and others as connectives in the CASREP implementation.

By including the appropriate knowledge in our representation and by choosing a representation which makes the inclusion of that knowledge straight-forward, we have enabled ourselves to define ALL of the necessary predicate frames for any domain. This representation can accommodate some of the information needed in all phases of the understanding process (both syntactic and semantic phases). The lexicon of predicates can be defined in terms of these structures and this structure could be filled in during parsing and then shipped off to the application component for further processing.

So, as an improvement over LSP Information Formatting approach, we have provided a datastructure that can be used during all phases of processing and that demands all predicate-related information be defined in one place. Collecting this information in one place seems like a anti-climatic benefit, but it is one of the most profound benefits to be gained in this representation. Serious problems of keeping up with all this information in LSP have caused the integrity of the information contained in the LSP system to be compromised. In LSP, predicate type information can be found in at least the lexicon, restrictions, lists, and computed attribute procedures thus making it more difficult to maintain and modify.

While this representation may not be adequate as an ultimate representation for every application, it will at least serve as a good intermediate representation. It is powerful in that everything can be represented and it is versatile in that it is not domain

dependent in any way. In the following section, we will discuss how this approach could be combined with a theory of linguistics from which even greater potential benefits might be derived.

5.2. Applying Case Frame Theory

5.2.1. Role Abstractions and Case

Linguists have attempted to isolate a set of abstract roles from which ALL predicates can be defined. They call this level of description the *deep structure.* From a deep structure, linguists hoped to be able to describe a set of transformations that could account (at least) for variations in where certain arguments can appear in surface structures. The set of abstract roles that can be found for all predicates in the world should be language independent and only their manifestation in a particular language would differ. Charles Fillmore is credited with developing this theory which is called **case frame theory.**

Fillmore presented an initial set of role relationships called **cases** in the first famous paper on the topic[4]. In a later paper[5], he revised that list. Many different researchers including Grimes[6] , Chafe[7] and Schank[8] have proposed different sets of cases as those that would suffice to describe deep structure and also a set of transformations on those cases which explains why different predicates order their arguments differently in their surface syntactic manifestation. Unfortunately, no generally accepted set has ever been developed.

5.2.2. The Benefits of Case Theory

Because the case frame is thought to be deep level and language independent, it seems intuitive that the relationship between predicates and their cases would somehow encompass some level of semantic information about the predicate. Therefore, the great hope for a universal theory was further encouraged because this syntactic approach seems to somehow explain a certain level of semantics, as well. Bridging at least some of the gap between syntax and semantics could have significant ramifications on the design of natural language systems. This aspect of case frame theory is what has attracted our attention to case frames.

By "making up" our own roles, we have not precluded the possibility of classifying our made-up roles in terms of the abstract roles. When we created our roles, we were not generating more or less roles than a predicate has, we were just defining the interpretation of those roles in an application specific way. So, assuming that there is such a thing as a basic set of *cases* (roles) and given enough information about how to identify those cases, we should be able to superimpose the *abstract cases* on our *application-specific cases.*

Given that we can perform this superimposition, we can exploit the current state of research in case theory and define some syntactic transformations which can account for some of the syntactic variability in surface structures. The most important aspect of using case theory is that we can associate these transformational rules with particular *case frames* in a very modular and organized manner. Specific transformational rules can be included in the definition of the predicate's frame. Therefore, once we have identified the predicating element in an syntactic unit, we can quit trying all the possible

permutations of its arguments and only try to parse its arguments through the transformations that are part of its definition. At the same time we are trying to discover the syntactic structure, we can use semantic type checking on the candidate arguments in order to rule out bad analyses on semantic grounds.

5.3. A Prototype Knowledge Representation System

Appendix A includes some examples of some of the predicate primitives that exist in the starting air compressor CASREP domain. These frames were derived by hand but, certainly an attempt to be methodical was made. We believe that these frames would be an improvement over information formats for reasons previously discussed. This set is not to be considered complete or absolute. The process of deriving a good set of frames is a difficult one and depends greatly on the goals of the application. These frames were developed with no particular application in mind, just with an idea of what the major types of information contained in these messages are. Several examples from our test data will be presented with each predicate class to give the reader an idea of what predicates would be members of the class. In the specification of the frame, the first line is the name of the frame, the following lines are the slots of the frame. On each slot line, the first element is the role name. The second element, printed in capital letters, is the semantic class restrictions on arguments filling that slot. Some semantic class restrictions are noun classes but others will be frame names when an predicate can serve as an argument to another predicate. Since we have not developed this representation for a particular application, no knowledge sources are provided to show how to interpret the roles assigned to particular predicates. We did do some guessing based on our experience with the previous TERSE system application and this is reflected in our choice of slot names in the following examples.

6. Transitioning the Terse System to Production

In this section, we will present a high-level view of the problem-domain that we have been working on and also a broad overview of the approach that has been taken. A evaluation of where the project currently is and how far away it is from being usable for performing the real task that it was designed for will be presented. This discussion assumes that the basic functionality of TERSE would remain the same as what has been described in this paper. However, the second portion of this section will examine the the current goals of TERSE and make some suggestions regarding further work with a system like TERSE.

6.1. The Real World Problem

Naval ships are required to send out a CASREP report on every piece of equipment on board that experiences a failure that will not be corrected for a period of time exceeding 24 hours. The messages can be about equipment as strategic as the main engines and as trivial as doors to and from compartments. The failure of some pieces of equipment can degrade the ship's readiness rating while other failures do not effect the rating.

The Navy is faced with several problems concerning these messages. The first problem is the shear number of messages coming in. The second problem is to sort out time critical information from non-time critical information. Some of these tasks can be handled using fairly simple human or computerized approaches, especially when the important data is contained in the formatted portions of the reports. However, some of the important information is contained in sections of the message written in free text. This information is much more complicated to deal with than the information contained in the formatted sections.

The approach currently employed by NAVSEA, who handles some subset of these CASREPs, has been to have contracted personnel read the messages and extract up to 60 characters of the *most important* information from the paragraph. The rationale behind this approach seems to have been to reduce the shear volume of the paragraph text. We assume that after this reduction is made, the extracts are then entered into a computer database or collected in a report. We have been told that this information is then used in failure trend analysis.

6.2. The Automated Approach

The overall approach to the task of automatically processing CASREP messages can be divided into two phases. The first phase involves natural language analysis. The second phase is an application specific phase where particular kinds of information would be extracted and processed.

In the first phase, the messages must be read into the computer. Information that is contained in the formatted sections of the report can be more or less directly mapped into a computer database. The information that is contained in the free text paragraphs must be analyzed by the natural language understanding system and normalized into a more appropriate form in order to be input into a database. This process is a very complicated process which involves a great amount of syntactic and semantic processing. The output of this phase is the input for the application phase.

The second phase of the system is the application phase. The application phase is where the information from the report is put to some use. These uses can involve forwarding all or some of the information to other parties, performing statistical analyses for failure trend analysis, etc.... This phase does *not* involve any natural language analysis. It is assumed that all of the necessary processing was performed in the first phase and that this phase receives a consistent and much more explicit, computer processable representation of that information.

6.3. The Currently Abilities of LSP and TERSE

The natural language processing system that was used for this project is the LSP (Linguistic String Parser) from New York University. A grammar and lexicon was developed especially for this application. This system parses natural language sentences and maps them into database entries called *information formats.*

Processing is done in two phases. In the first phase, a sentence is parsed and some semantic information is applied to help choose among ambiguous analyses. Currently, the system is able to parse all of the sentences (172 number of sentences) contained in 46 messages. However, 18 out of the 172 sentences (10.5%) have been edited by hand in order to get a parse from the system. So, we can report an 89.5% success rate in parsing.

In the second phase of processing, the syntactic parse is mapped into one or more information formats. This phase is conceptually easy once a correct parse is obtained. However, the LSP system is very difficult to program. This is the point at which a great amount of error is introduced in our current implementation. While we do not have statistics to report, we are not overly concerned because to fix this problem is really just a matter of reprogramming the formatting component.

6.4. Upgrading to the Real World

The difference between the real world and the sample world that we have been analyzing would be the breadth of the types of equipment discussed in the messages. Theoretically, the parsing system should become stable in terms of the addition of new grammar rules. The lexicon would have to be expanded to include all new words used to discuss failures from different domains. The outlook seems to be fair on this phase of the system.

In considering the consequences of expanding the knowledge-based, application system to a broad range of CASREPs, we are faced with a serious problem. Although many of the rules contained in the domain-independent section of the knowledge bases will work well as long as the basic language of the CASREPs does not change drastically, the domain model does not enjoy such broad application. The rules used in the domain model to make inferences will probably apply to any other mechanical domain, but the actual domain model (that portion of the system containing the component definitions) will have to be updated to include **ALL** pieces of equipment that could be reported in a CASREP.

The task of constructing a domain model for literally every piece of equipment on a Navy ship is a very extensive and costly task. There are thousands of pieces of equipment on a ship. Even with the task of encoding a large set of equipment models, we are

faced with the problem of keeping up with new equipment and keeping up with upgrades on the existing equipment. The Starting Air Compressor for which we have a model, is only one small piece of equipment in comparison to all the other equipment on board and it has taken approximately 150 knowledge base units to represent. Perhaps in the future, equipment manufacturers could be required to supply a computer representation of their equipment. Even if these giant knowledge bases were made available, we would then be faced with the problems of efficiently accessing such large amounts of information in a reasonable amount of time.

6.5. Before We Send THIS System to Production

In considering the possibility of transitioning the current system, we must look at the difficulties involved, but we must also make sure that the system that we spend much time and expense on transitioning constitutes the best approach to the real world problem. The following section analyzes the approach that has been taken by NAVSEA for handling these reports and then asks some serious questions about whether we should be designing an automated system to do the same thing.

6.5.1. Hypothesis About Motivations Behind this Approach

The contractors at NAVSEA have been directed to read CASREPs and produce a 60 character (maximum) extract reflecting the *most important* information in the textual part of the message. We have met with NAVSEA and their contractors and the following discussion includes some information that was given to us from those interviews and some inferences that we have drawn, ourselves.

The first "burning question" is "Why throw away the text that was not included in the 60 character extraction?". Our hypothesis is that when this operation was implemented, computer technology was in its *second generation.* This generation was characterized by very limited and expensive primary storage (main memory) and very limited, expensive, and slow secondary storage (compared to what is available today). Database retrieval systems were very inefficient at handling the variable length records that would be required to hold the paragraph information in these reports. Even if the text could have been formatted into these databases, there were no means for retrieving the contents because these fields could not be "keyed" like traditional fields in a database record. The system would have had to resort to exhaustive character by character examinations of all texts records contained in the database to even retrieve a record containing a particular word. All in all, technology in both software and hardware made it prohibitive to really do anything useful with this portion of the messages on the computer. It seems that NAVSEA's answer to this problem was to reduce the text to a more manageable and predictable size, but the problems of accessing the reduced natural language text in these traditional database systems still remained. We doubt that any automated procedure uses these fields to this day.

It seems that the NAVSEA people that we interviewed were tasked with extracting information relevant to failure trend analysis. They were really reducing the text to a reasonable quantity with the goal of retaining information important to performing failure trend analysis. Although the "extracts" are ultimately useful for other humans to look over, the process of failure trend analysis has no hope of being automated using this data since it is still essentially, raw natural language text (just a bit shorter).

In general, it seems that the entire approach adopted by NAVSEA was centered around an inability to deal with natural language texts. The task that they were performing was geared towards ridding themselves of this text. Our entire project has been using and developing technology for handling this precise problem of natural language texts. With the ability to handle this natural language text, is there still a need to "reduce" the text?

6.5.2. The Real Needs

Since the text in the CASREP reports tends to be very telegraphic and only one paragraph in length, it seems that the authors have already gone through the trouble of reducing what they wanted to write. We assume that all of what is written is important, otherwise it wouldn't be included. If this were not the case, then it would be possible to limit the message writers to only a 60 character text field. We assume that if there were a way to get all of the information out of the paragraph into a database then it would be done. We also assume that if there were a way to automatically perform some of the information processing needs on the computer (failure trend analysis etc....) that this would also be done.

The addition of the kind of technology that we are experimenting with drastically changes what is possible in terms of automatically processing these messages. The rules of the game change. We believe that this particular application is a good example of the need to rethink the approaches to old problems when new technology is introduced.

The goals of this application were an attempt to mimic what has been previously done by the NAVSEA contractors. Our goal was to extract that information that the NAVSEA people said was important, but with a computer program. We accepted the raw natural language paragraphs as our input, processed them, and then returned an *extract* which represented what was considered most important. However, we need to look at the forest instead of the trees and evaluate what contributions we have made to this real world problem. We went through an unnecessary process that was done by people only because their computers lacked the ability that we are now attempting to provide (the ability to process natural language inputs). So much work has gone into processing the natural language that simply outputting an extract seens like a terrible waste. It seems much more worthwhile to go "the whole 9 yards" and attempt to tackle some of the real information processing tasks such as failure trend analysis.

Improving the direction of the application portion of the system does not mean that the work that went into it was for naught. It does point out a very important aspect of introducing new technology: it is sometimes necessary and **beneficial** to change the approach to a problem in order to reap the greatest benefits from new technology. We propose that more emphasis be placed on producing very good natural language system that can produce good database entries that applications can access. Statistically we have a pretty good natural language system right now, but there are serious shortcomings in the database entries that it sends as output (see section on knowledge representation).

In terms of the work spent on the TERSE application, some portion of it could be transferred over to a system that might actually attempt to preform failure trend analysis. It will still be necessary to know what is considered important for that particular application. However, we must stress the need to re-evaluate that criteria with

respect to the prospect of using a totally automated system. Some of the rules contained in the TERSE system might be useful in a failure trend analysis application system, but most are oriented towards the task of extracting information. Even with this new approach, we are still faced with the cost and expense of large domain models. We suggest that after redesigning the knowledge representation being output by the natural language system that further re-evaluation be done on exactly how much these potentially expensive domain models will improve an application like failure trend analysis. Further investigation should be done in this area.

Appendix A

BEHAVIOR-FUNC		
	mechanical-processor	EQUIP
	human-operator	ORG

examples of BEHAVIOR-FUNC predicates

a4.1.3 *engine* **jacks over**
b14.1.2 *starting air compressor* **engaged**
b16.1.4 *lube oil pump* **seized**
b12.1.1 **engagement** *of sac*

DAMAGE		
	damager	EQUIP
	damagee	EQUIP

examples of DAMAGE predicates

b17.1.1 *gear housing* **cracked**
b16.1.5 *driven gear was* **sheared**
a24.1.1 **wiped** *bearings*

ATTRIBUTIVE		
	described	*any-noun-or-predicate*

examples of ATTRIBUTIVE predicates

b7.1.11 *the drive shaft was* **dry**
a25.1.3 **faulty** *high speed rotating assembly*
a31.1.1b **normal** *sac lo pressure*

BE-LOC		
	located	(PIECE MATERIAL PHYSICAL-FEATURE)
	location	(EQUIP SUBST)
	orientation	??

examples of BE-LOC predicates

a5.1.3 *metal particles* **(be)** *in strainer*
b17.1.2 *large crack* **(be)** *in gear housing on aft end*

POSSESS		
	possessor	(EQUIP SUBST)
	possessed	(PHYSICAL-FEATURE EQUIP PROPERTY CAPABILITY ATTRIB

examples of POSSESS predicates

srep30 *the drive shaft* **has** *a drilled passage*
b14.1.4 *lube oil* **has** *burnt odor*
srep18 *s/f* **does not have** *capability to t/s further*

CAPABILITY		
	do-er	(EQUIP ORG)
	goal	*any-goal-oriented-predicate-frame*

examples of CAPABILITY predicates

srep18 *capability to t/s*
a1.1.2 *unable to consistently start nr 1b gas turbine*
b7.1.1 *local monitoring capability*

FAIL		
	processor	(EQUIP ORG PROPERTY)
	goal	*any-goal-oriented-predicate-frame*

examples of FAIL predicates

a1.1.1	*starting air regulating valve failed*
b13.1.1	*oil pressure failed*
srep8	*sac fails to maintain lube oil pressure*

MAINTAIN		
	agent	(ORG EQUIP)
	object	PROPERTY
	recipient	EQUIP

examples of MAINTAIN predicates

srep3	*(unable to) maintain minimum oil pressure*
srep18	*sac (fails to) maintain lube oil pressure*
a5.1.1	*(unable to) maintain lo pressure to sac*

CHANGE-QUANTITY		
	agent	ORG
	object	PROPERTY
	total-amount	(QUANT MEASURE-POINT)
	starting-point	(QUANT MEASURE-POINT)
	ending-point	(QUANT MEASURE-POINT)

examples of CHANGE-QUANTITY predicates

a9.1.2	*lo pressure was dropping*
a21.1.1a	*nr 4 sac oil pressure dropped below alarm point of 65 psig*
b7.1.9	*lube oil pressure could not be adjusted above 35 psig*

References

1. Zellig S. Harris, *Discourse Analysis Reprints,* Mouton & Co., The Hague, 1963.
2. Naomi Sager, *Natural Language Information Processing,* p. 214, 216, 219, Addison-Wesley Publishing Co., Reading, Mass., 1981.
3. Minsky, "A Framework for Representing Knowledge," *The Psychology of Computer Vision,* pp. 211-277, McGraw-Hill, New York, 1975.
4. Charles J. Fillmore, "The Case for Case," *Universals in Linguistic Theory,* pp. 1-90, Holt, Rinehart and Winston, Chicago, 1968.
5. Charles J. Fillmore, *The Case for Case Reopened.*
6. Joseph E. Grimes, *The Thread of Discourse,* Mouton, The Hague, 1975.
7. Wallace L. Chafe, *Meaning and Structure of Language,* University of Chicago Press, Chicago, 1970.
8. Roger C. Schank, *Conceptual Information Processing,* North Holland, Amsterdam, 1975.