SWAN: AN EXPERT SYSTEM WITH NATURAL LANGUAGE INTERFACE FOR TACTICAL AIR CAPABILITY ASSESSMENT

Robert M. Simmons
Systems Research and Applications Corporation
2000 North 15th Street
Arlington, Virginia 22203

ABSTRACT

SWAN is an expert system and natural language interface for assessing the war-fighting capability of Air Force units in central Europe. The expert system is an object-oriented knowledge-based simulation with an alternate worlds facility for performing "what if" excursions. Responses from the system take the form of generated text, tables, or graphs. The natural language interface is an expert system in its own right, with a knowledge base and rules which understand how to access external databases, models, or expert systems. The distinguishing feature of the Air Force expert system is its use of meta-knowledge to generate explanations in the frame- and procedure-based environment.

INTRODUCTION

The goal of the SWAN project is to demonstrate the feasibility of artificial intelligence technology to assess tactical air capability for the Air Force. SWAN deals with four airbases in central Europe, the aircraft and squadrons assigned to those airbases, munitions and other resources required for sortie generation, and the missions flown by the aircraft. Notable factors outside the scope of SWAN include weather, targets, and enemy capability. Air Force units are tasked to fly missions via the Air Tasking Order (ATO). A unit's ability to execute the ATO depends on several factors: weather, resource limitations, aircraft availability, and others. Based on its knowledge about aircraft, munitions, and missions, SWAN determines the ability of a unit to execute its tasking. SWAN identifies any factors which limit a unit's capability and provides facilities for relaxing constraints to improve projected capability, and for tightening constraints to perform sensitivity analysis.

Figure 1 shows the high-level SWAN modules. The SWAN front-end accepts English questions covering a wide range of queries: general domain knowledge, tasking, capability, limiting factors and "what if" excursions. After understanding a question, the SWAN natural language interface sends a semantic representation of the question to the expert system for processing. When the expert system returns an answer, SWAN constructs a semantic representation of the answer for subsequent generation.

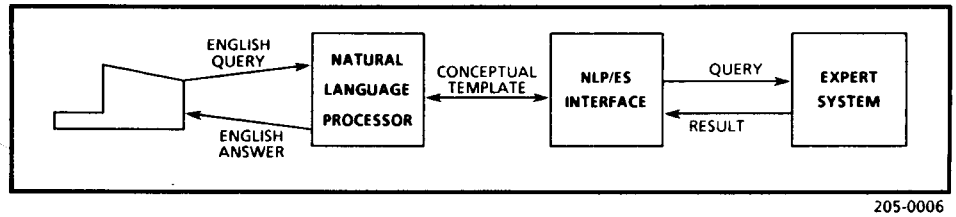

Figure 1. SWAN modules.

This paper describes our approach to two important aspects of SWAN: explanation and natural language query of the expert system. There is a wide spectrum of approaches to explanation. Simple rule-based systems explain by tracing the rules that led to a conclusion. There is little control over

the level of detail in the explanation; some rules may be at a high conceptual level while others perform low-level calculations. At the other end of the spectrum are explanations from human experts. They are concise, to the point, and tailored to the listener's expertise. SWAN uses meta-knowledge to produce explanations derived from the expert's view of the domain, not from a trace of the expert system's calculations. Our approach produces explanations based on general knowledge of the domain and specific data from the user's question and the expert system. The other focus of this paper is natural language query of an expert system. Such query is difficult because of a recurring disconnect between the way a user phrases a question and the relatively rigid way the knowledge is stored. We describe a knowledge-based approach that gives SWAN the flexibility it needs to understand indirect queries. A typical question in our domain is "What mission does Hahn airbase fly?", difficult in light of the fact that airbases do not fly missions; it is the aircraft at those bases that fly. Our approach frees the user from having to know how (even where) the knowledge is stored.

The next two sections of the paper present general descriptions of the expert system and natural language processor. The remainder of the paper describes in detail the explanation facility of the expert system and the natural language interface strategy for handling indirect queries.

The SWAN Expert System

SWAN's expert system includes an extensive domain knowledge base and LISP procedures which model the sortie generation process. The knowledge base is frame-based, developed in KEE. The sortie generation model runs in LISP on top of KEE. Various procedures calculate the capability to execute tasking, determine limiting factors, and modify assumptions or constraints to permit recalculation. Modification of assumptions or constraints results in the generation of an "alternate world" and recalculation within that world. The use of alternate worlds gives the user a powerful capability to investigate possible remedies to factors which limit capability. SWAN has the ability to move between worlds by referring to the assumptions that make the world unique. The user can compare results across worlds. SWAN can generate explanations about specific results and its domain in general; this is significant considering the absence of classical expert system rules. This capability is discussed in detail in the section "EXPLANATION IN A NON-RULE-BASED ENVIRONMENT."

The Natural Language Interface

SWAN's natural language interface combines syntactic parsing with semantic analysis to produce a "deep structure" semantic representation of a question. This deep structure representation is used to communicate with the expert system. The natural language interface has five processing phases: (1) preprocessing; (2) parsing; (3) semantic analysis; (4) interface processing; and (5) answer generation. Preprocessing includes a spelling checker and morphological analyzer. SWAN's parser is an adaptation of the DIAMOND parser and DIAGRAM grammar from SRI's TEAM system [1, 6].

DIAMOND often generates multiple parses for a question, due to the lack of semantics which specify how sentence constituents should be syntactically attached. SWAN must determine the deep structure of the question from among those several parses. SWAN's parser accepts sentence fragments such as noun phrases, prepositional phrases, and verb phrases. SWAN recognizes such elliptical questions and processes them during the semantic understanding phase. The parser uses a lexicon of approximately 5,000 words. The lexicon contains syntactic information for the parser as well as semantic pointers into the natural language interface's knowledge base. We emphasized Air Force jargon to help SWAN understand the everyday language of users in this domain.

EXPLANATION IN A NON-RULE-BASED ENVIRONMENT

Explanations from an expert system are necessary to instill confidence and to facilitate testing of the system. The level of detail is important -- explanations must be complete enough to account for all

system behavior, general enough to be meaningful to the user, and specific enough to clearly relate to the question at hand [2]. There are several types of explanation that SWAN must handle:

1. Justification for basic facts in the knowledge base;
2. Explanation of results; and
3. Explanation of processes.

Explanations of the first type must refer to the original knowledge source (expert or document) or to supporting logic. The second type of explanation comes from questions such as "How did you get that answer?" The third type comes from questions such as "How do you determine the best munition to use?" or "Why do you need that information?"

In a non-rule-based environment, explanations are difficult to generate. A typical SWAN computation might pass control from a LISP function to a knowledge base query. Access to a frame in the knowledge base may trigger a daemon which transfers control to another LISP function which performs additional knowledge base access, and so on. There is no homogeneous thread of logic as in traditional rule-based systems such as MYCIN [3].

The approach to explanation in SWAN is based on meta-knowledge. This work is similar in spirit to the XPLAIN system [4] and the EES project [5], except that SWAN does not automate the development of an expert system with a program writer as in EES. SWAN maintains a meta-knowledge network of "common sense" domain knowledge about the sortie generation domain and the expert system itself. "Common sense" in this context is relative to the domain; the knowledge consists of general information about objects in the domain and their relationships. There are three types of network entities: objects, events, and scripts. Events are relationships between two objects. SWAN employs a minimal set of relationship (or link) types for defining events. The primitive relationships in the network and their associated semantics are as follows:

1. PRODUCE. X causes an increase in the quantity of Y.
2. CONSUME. X causes a decrease in the quantity of Y.
3. ALTER. X alters the state of Y. The semantics of this relationship are further refined by specifying the degree to which X alters Y and the manner in which X alters Y.
4. DETERMINE. X is necessary and sufficient for Y.
5. REQUIRE. X requires Y in order to be a causal agent.
6. USE. X uses Y as an instrument.

We have been able to encode the bulk of the general domain knowledge using PRODUCE, CONSUME, and ALTER; the network is referred to as the PCA network. Figure 2 shows a representative segment of the network. Our use of a minimal set of links permits the construction of general rules regarding cause/effect relationships between events. For example, SWAN's PCA network defines the following events:

E1. AIRCRAFT -- produce--> SORTIES
E2. AIRCRAFT -- consume--> FUEL

We have a general rule capable of determining that event E1 may REQUIRE event E2. The rule basically states that producers must consume in order to produce.

Besides having rules which can infer relationships between events, SWAN permits the explicit linking of events using a number of higher-level cause and effect relationships:

PREVENT/CAUSE/PERMIT
CANCEL/CURE/NEGATE
COMPETE/SUBSTITUTE

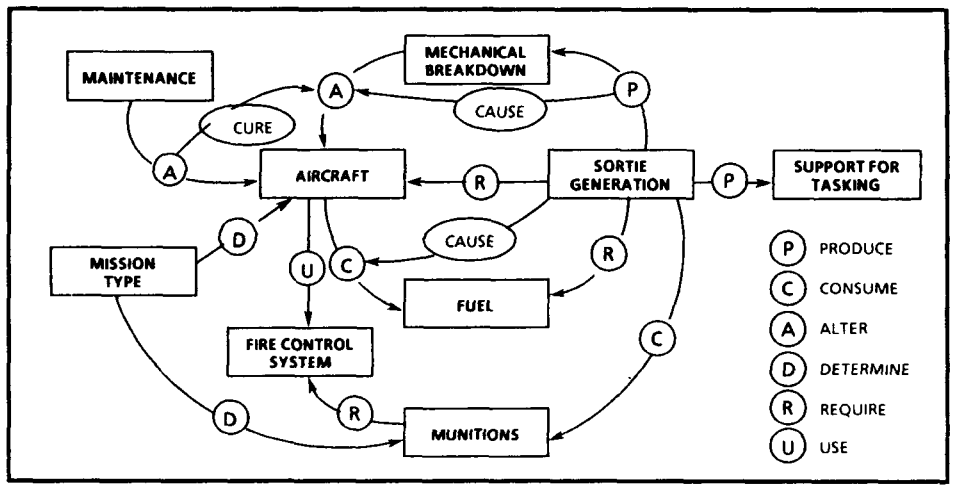

Figure 2. Part of the SWAN PCA network.

PRECEED/FOLLOW/OCCUR-WITH

For example, events E1 and E2, above, might be connected explicitly with a PERMIT relationship:

E3. E2 --permits--> E1

Explicit representation of inter-event relationships is more efficient, but not necessary. In fact, the PCA network is capable of recording new links after inferring relationships between events. Events E1 through E3 and the PCA rules allow SWAN to generate answers to questions such as "What happens if I run out of fuel?" and "How can I get more sorties?" The third type of PCA entity is the script. Scripts organize events and provide an abstraction mechanism for dealing with complex activities at varying levels of detail. Scripts and events can be named, effectively making them objects able to participate in events. SWAN has scripts for refueling, for sortie generation, for maintenance, and others.

Using the PCA network, SWAN generates explanations by relying on knowledge of general processes and relationships. SWAN explains its results by knowing how a process WOULD generate the results. Human experts often explain their results by reworking a problem, going into detail when necessary, to trace the development of the result. This means that "How would you...?" and "How did you...?" questions can be answered identically except for the level of specificity in the answer. SWAN answers the first type of question by traversing the PCA network, looking for appropriate scripts and factors which influence events in the scripts. To explain results from a computation, SWAN uses the same process but refers to information from the question and information from the knowledge base to generate specific answers. For example, the PCA network has the following events regarding aircraft and munitions:

E4. AIRCRAFT --use--> FIRE-CONTROL-SYSTEM
E5. MUNITIONS --require--> FIRE-CONTROL-SYSTEM

Consider the question "Why can't F-16s carry Sparrow missiles?" From its taxonomy, SWAN knows that the F-16 is an AIRCRAFT and the Sparrow missile is a MUNITION. SWAN searches the PCA network for a path between AIRCRAFT and MUNITION, retrieving the information in E4 and E5. This is sufficient to generate a general explanation:

AIRCRAFT USE FIRE CONTROL SYSTEMS REQUIRED BY MUNITIONS

To produce the specific explanation, SWAN substitutes information from the question into the general explanation:

F-16S DO NOT USE THE FIRE CONTROL SYSTEMS REQUIRED BY SPARROWS

Notice that the negation assumption in the original question leads SWAN to generate a negative answer.

The PCA network also supports general "What if...?" and "How can...?" questions through a network traversal strategy that resembles forward and backward rule chaining. In the forward direction, SWAN examines the links from an object to determine the effects of an increase or decrease in that object. For example, for the question "What happens if we fly more sorties?", SWAN would examine links involving the PCA node SORTIES and produce the following information:

INCREASING SORTIES WOULD PRODUCE MORE TASKING SUPPORT
INCREASING SORTIES WOULD CONSUME MORE FUEL
INCREASING SORTIES WOULD CONSUME MORE MUNITIONS
INCREASING SORTIES WOULD PRODUCE MORE MECHANICAL BREAKDOWN

The above explanation points out that flying more sorties is good, but it has a cost associated with it. To answer goal-directed questions such as "How can I get more aircraft?", SWAN looks for events leading into the PCA node AIRCRAFT that produce the specified effect (increase). SWAN's answer to the question would be:

THERE ARE 3 POSSIBLE WAYS TO INCREASE AIRCRAFT:
YOU CAN INCREASE WARGOER RISK OR INCREASE MC RATE TO INCREASE AIRCRAFT STATUS WHICH AUGMENTS AIRCRAFT INVENTORY.
YOU CAN CHANGE 4102 PLAN TO INCREASE AUGMENTATION WHICH INCREASES AIRCRAFT INVENTORY.
YOU CAN DECREASE ENEMY AIR DEFENSE TO DECREASE HOSTILE ATTRITION WHICH DESTROYS AIRCRAFT.

The first recommendation says that you should either fly with aircraft that are broken (accepting risk) or improve the maintenance rate (MC is Mission Capable) for the aircraft. The second states that you can try to alter the resupply plan (4102) that provides for additional aircraft from the United States in wartime. The third recommendation says that you should try to offset the effects of attrition, in which the enemy shoots down your aircraft. Notice that the last recommendation is wrong; we have improved SWAN's knowledge so that it understands that stopping the consumption of X is not the same as producing X.

NATURAL LANGUAGE QUERY OF THE EXPERT SYSTEM

There are two basic problems which SWAN's natural language interface solves. First, SWAN must determine the semantics of a question. Semantically-equivalent questions can be phrased any number of ways. Even for a single question, however, the presence of multiple parses aggravates the problem of semantic interpretation. Second, SWAN must get the expert system to answer the question. A user's perspective may differ considerably from that of the expert system; SWAN has the flexibility to translate between one perspective and the other.

To determine the semantics of a question from the parse tree, SWAN relies on a bottom-up approach that is largely immune to confusion from multiple parses. Semantic information associated with the leaves of the parse tree identify the basic meaning of individual words. To understand the meaning of constituent phrases, SWAN manages a process of semantic combination and attachment at each node in the parse tree. Verb nodes activate "conceptual templates" that represent domain activities or relationships. These templates form the framework for a question's deep structure. Most other leaf categories (noun, adjective, adverb, determiner) become "semantic objects" that represent domain objects or attributes. Some semantic objects can combine with each other to produce different objects. In other cases, semantic objects attach to other semantic objects (e.g.articles/adjectives to nouns) or to conceptual templates.

SWAN interprets the combination of semantic objects as a query of the expert system's knowledge base. For example, consider the phrases:

aircraft at Hahn	(prepositional modifier)
Hahn aircraft	(noun-noun modifier)
Hahn's aircraft	(possessive)

Each phrase refers to the "F-16" aircraft, the type of aircraft based at Hahn. SWAN semantically collapses these phrases by combining the semantic objects [AIRBASE HAHN] and [VEHICLE AIRCRAFT] to produce [AIRCRAFT F-16]. This results from the generation of an expert system query to determine the AIRCRAFT of HAHN. The order of combination is not important because SWAN determines that AIRCRAFT is more general than HAHN (general class versus proper name). Therefore, SWAN acts as if AIRCRAFT is an attribute of the domain object HAHN.

The natural language interface uses a knowledge base to control semantic combination and attachment. To avoid confusion between the knowledge base of the natural language interface and that of the expert system, further references to a knowledge base will be NLI-KB or ES-KB, respectively. NLI-KB covers a broader domain than ES-KB, though at less depth. Information in NLI-KB includes a CLASS/SUBCLASS taxonomy, attachment restrictions, and expert system query information. Both NLI-KB and ES-KB are frame-based, containing a network of objects and attributes. NLI-KB contains knowledge about the information in ES-KB. For each pair of domain object classes that are meaningful to query, the NLI-KB contains a list of slot names that define a path through ES-KB. For example, in NLI-KB, the AIRCRAFT frame has an AIRBASE attribute. We can determine the AIRCRAFT of a specific AIRBASE by generating a query from information in the AIRBASE attribute of the NLI-KB frame below:

```
(AIRCRAFT
   ...
   (AIRBASE (INVENTORY) (FIGHTER-SQUADRON))
   ...)
```

The information above can be viewed functionally as

(INVENTORY (FIGHTER-SQUADRON AIRBASE)) = = > AIRCRAFT

The AIRCRAFT of an AIRBASE is not found in a slot of an AIRBASE frame in ES-KB; we find it by looking in the INVENTORY of the FIGHTER-SQUADRON of the AIRBASE. This representation provides a powerful notation for handling indirect expert system queries. With the addition of special "slots" in NLI-KB we can also handle queries that might seem natural to the user but foreign to the expert system. As an example, consider the question "What missions does Hahn airbase fly?" Taken literally, the answer might be "Hahn does not fly missions. Aircraft fly missions." One of SWAN's

design goals, however, is to answer questions the way the user thinks of them. To handle this indirect query, the NLI-KB information

```
(MISSION
   ...
   (AIRBASE (MISSION) (DETOUR AIRCRAFT))
   ...)
```

specifies the special slot (DETOUR AIRCRAFT). The MISSION of an AIRBASE can be found by "detouring" to the AIRCRAFT of the AIRBASE, then querying the MISSION of those AIRCRAFT. Thus SWAN understands the question as if the user had asked "What missions do the aircraft at Hahn fly?" Another special slot (SELECT ...) permits filtering of attribute values according to a user-specified function. We use this to deal with questions requiring the first value, the average value, the maximum value, the "best" value, and so on.

Verbs in a question activate SWAN's conceptual templates. A small number of semantic "predicates" represent all verbs in the lexicon. Each predicate has a number of conceptual templates defined in NLI-KB; each template represents a different sense or use of the predicate. For example, the predicate USE has several templates, including:

```
(USE (AIRCRAFT MUNITION MISSION))
(USE (AIRCRAFT FUEL))
(USE (AIRBASE RESOURCE TIME-UNIT))
```

The top template corresponds to any question about an aircraft using munitions for a mission. The bottom template deals with questions about consumption of resources at an airbase over a period of time. Semantic objects from the parse tree attach to the slots in each template. When processing of the parse tree is complete, SWAN selects the template that most completely matches and sends it to the expert system. Many of these templates require knowledge base queries while others require calculations by the expert system. The interface module shown in figure 1 accepts the conceptual template and formulates queries or calculations for the expert system. After receiving the answer from the expert system, the interface modifies the question template to become an answer template ready for generation.

CONCLUSION

A natural language interface is a useful and feasible means of communication with an expert system. SWAN's natural language interface uses a knowledge-based process of parse tree interpretation to understand a user's questions and interface with the expert system. Our approach allows SWAN to handle queries the way the user asks them. The expert system generates explanations using meta-knowledge that understands general processes and relationships in the domain. This approach is motivated by the absence of rules in the expert system which permit explanations based on traces of forward or backward chaining rules. SWAN's meta-knowledge network defines objects, events, and scripts that capture general expertise about the sortie generation domain. General rules about production and consumption provide the capability to reason about the meta-knowledge in the PCA network and derive new relationships between objects or events.

ACKNOWLEDGEMENTS

The SWAN project team is Sherman Greenstein, Dr. Mary Dee Harris, Dr. Hatte Blejer, David Reel, William Brooks, and Robert Simmons. I would like to thank Hatte Blejer, Bill Brooks, Sherman Greenstein, and Helen Simmons for their comments on this paper.

SWAN is being developed as part of the AI Tactical C2 Demonstration Project, Department of the Air Force contract number 85X-54277C.

REFERENCES

1. Winograd, T., LANGUAGE AS A COGNITIVE PROCESS (SYNTAX), Addison-Wesley, 1983, pp. 381-382.
2. Davis, R., and Buchanan, B. "Meta-level knowledge: Overview and Applications," IJCAI 5, pp. 920-928.
3. Shoftliffe, E. H., COMPUTER-BASED MEDICAL CONSULTANTS: MYCIN, American Elsevier, New York, 1976.
4. Swartout, W., "XPLAIN: A system for creating and explaining expert consulting systems," ARTIFICIAL INTELLIGENCE, 21, 3 September, 1983, pp. 285-325.
5. Neches, R., Swartout, W., Moore, J., "Explainable (and maintainable) expert systems," IJCAI 9, pp. 382-389.
6. Robinson, J., "Extending Grammars to new Domains," RR-83-123, USC/ISI, Marina del Rey, CA, January, 1984.