Dankel

136886 N93-17506

p 6

GATOR: Requirements Capturing of Telephony Features

Douglas D. Dankel II
ddd@cis.ufl.edu

Wayne Walker
ww0@cis.ufl.edu

Mark Schmalz
msz@mosquito.cis.ufl.edu

E301 CSE, C.I.S.
University of Florida
Gainesville, FL 32611
(904) 392-1387 (Office)
(904) 392-1220 (FAX)

1. Introduction

During the past twenty years the telecommunications industry has become increasingly dependent upon software-controlled switching systems. The software of these systems automates the billing of long distance calls, supports direct dialing of overseas calls, and provides features (e.g., call waiting, call forwarding) that many people consider essential components of everyday life. While telephony software has become both very large and complex in function and structure, the methods of software description have changed little over the past two decades. All existing characteristics and features as well as any modifications or additions to this software are described through natural language requirements specification documents.

These documents present a real dilemma to both the developers and customers. While these documents are essential to describe of the functionality of telephony features, the ambiguity and uncertainty inherent within natural language often leads to misinterpretations which can severely impact the resulting implementation of the functionality, the user acceptance of these features, and/or the development cycle.

We are developing a natural language-based, requirements gathering system called GATOR (for the GATherer Of Requirements) that assists in the development of more accurate and complete specifications of new telephony features. GATOR interacts with a feature designer who describes a new feature, set of features, or capability to be implemented. The system aids this individual in the specification process by asking for clarifications when potential ambiguities are present, by identifying potential conflicts with other existing features, and by presenting its understanding of the feature to the designer. Through user interaction with a model of the existing telephony feature set, GATOR constructs a formal representation of the new, "to be implemented" feature. Ultimately GATOR will produce a requirements document and will maintain an internal representation of this feature to aid in future design and specification.

This paper consists of three sections that describe (1) the structure of GATOR, (2) POND, GATOR's internal knowledge representation language, and (3) current research issues.

2. The Structure of GATOR

GATOR consists of three major components, illustrated in Figure 1:

1. **The User Interface** (consists of the Parser, Lexical & Grammatical Knowledge Base, Predicate Generator, and Response Generator) accepts natural language requirements descriptions and reports its understanding of these requirements to the user. Additionally, the User Interface answers user queries regarding the system's understanding of a feature and requests clarification of input which may be ambiguous or may contain recognizable errors.

2. **The Command Interpreter** (consists of the Interpreter) receives information and commands from the user interface, and issues queries and update instructions to the Knowledge Base/Data Base. This information specifies actions to be taken by the telephone switching circuits and software (e.g. "The Directory Number is always transmitted to the terminating office as a part of the Initial Address Message."), provides structural/organizational knowledge (e.g., "Calling Number Delivery Blocking (CNDB) is a CLASS feature."), or describes actions for displaying information (e.g., "Display a call with CNDB and Three-way Calling (3WC)."),

locating information within the representation (e.g., "What are the parts of a call?"), creating new knowledge that must be stored (e.g., "After the access code is entered, it is checked for validity."), or modifying existing knowledge (e.g., "The check for CNDB validity is made after the access code is verified as a valid code.").

3. **The Knowledge/Data Base** is a repository of information about the general structure of a call, existing features, and the new feature being defined. It consists the three levels, described in the next section.

3. Knowledge/Data Base

The Knowledge/Data Base contains specific knowledge of the components of a call and all existing features. It was built using POND [DANK92] (the Pantological Organization of New Delineations), a knowledge representation structure based on the family of KL-ONE languages [BRAC85, BRAC89, WOOD90]. While most of the KL-ONE languages divide knowledge into two partitions, called the Terminological Box or TBox and the Assertional Box or ABox, POND consists of three distinct knowledge levels as shown in Figure 2:

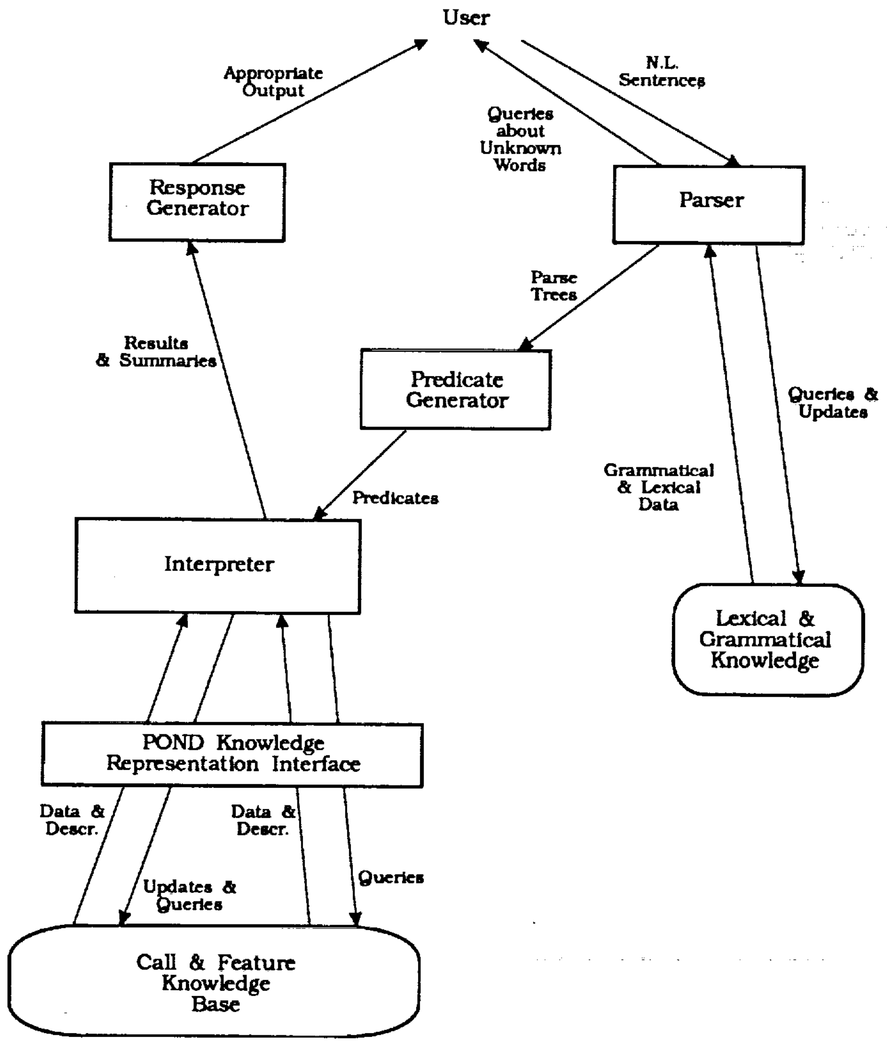

Figure 1. Internal System View

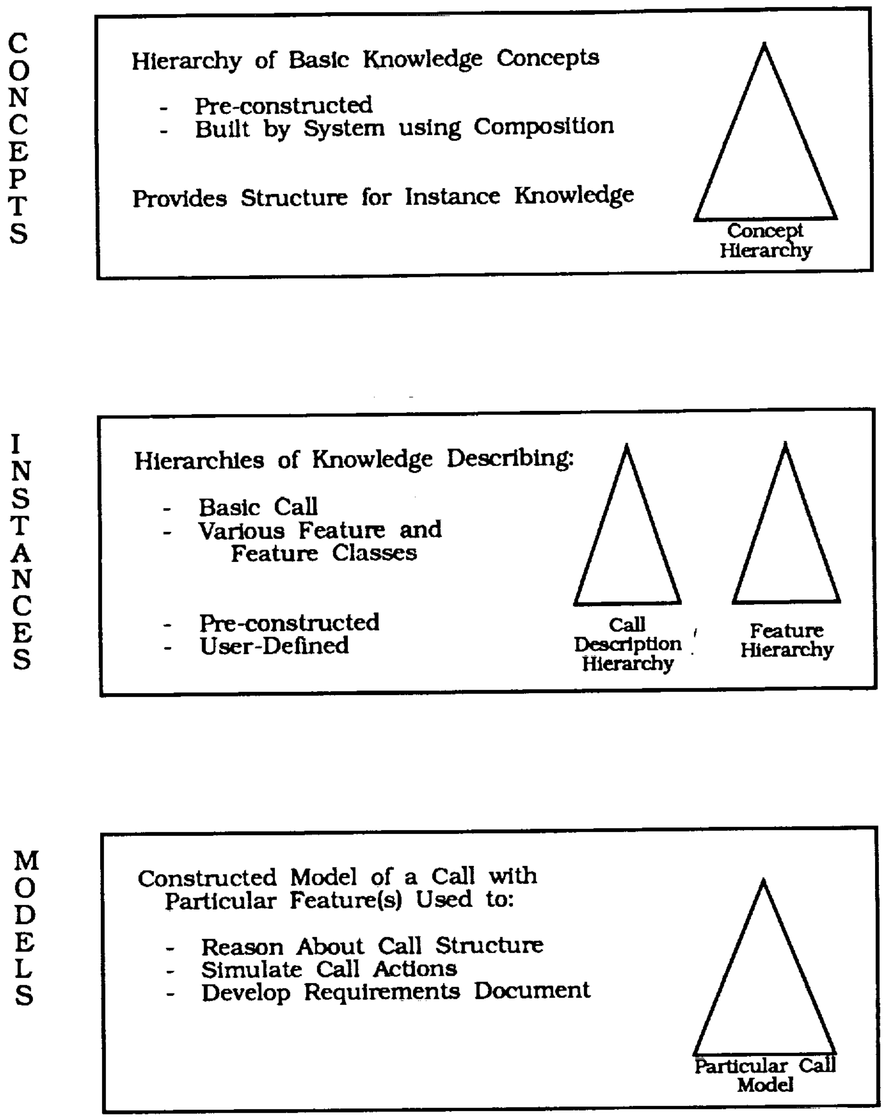
CONCEPTS
Hierarchy of Basic Knowledge Concepts
- Pre-constructed
- Built by System using Composition
Provides Structure for Instance Knowledge
Concept Hierarchy
INSTANCES
Hierarchies of Knowledge Describing:
- Basic Call
- Various Feature and Feature Classes
- Pre-constructed
- User-Defined
Call Description Hierarchy
Feature Hierarchy
MODELS
Constructed Model of a Call with Particular Feature(s) Used to:
- Reason About Call Structure
- Simulate Call Actions
- Develop Requirements Document
Particular Call Model

Figure 2. A Conceptual Diagram of the Knowledge Levels within the Call & Feature Knowledge Base

1. **Concepts.** High-level conceptual knowledge used to structure all of the knowledge within the knowledge base.
2. **Instances.** Specific descriptions of the components of a call, existing features, and the dynamic specification of the feature under definition.
3. **Models.** A constructed model of a particular call containing specific features.

A short description of each of these components follows.

3.1. Concepts

Knowledge on the Concept Level provides a structure for the knowledge on the Instance and Model Levels. Conceptual knowledge includes definitions of:

1. The concepts that represent telephone call and feature components. Each concept contains several slots (i.e., *:features*) that define the type and number of permitted values. Concepts can additionally include references (i.e., *:ako*) to other concepts on which they are based and applicable constraints (i.e., *:annotation*).
2. Special slots, or attributes, of the concepts, which define a set of restricted values or define relationships between concepts. For example, the *category* slot defines a restricted set of allowable slot values, while the *children* and

parent slots define relationships between slots.

3. Temporal relations [ALLE85] required for specifying temporal ordering within instances and models.

3.2. Instances

The Concept Level defines knowledge fundamental to instances on the Instance Level. A particular concept associates with each instance providing a structure and certain internal values for the instance. Instance knowledge includes descriptions of call and feature components. For example, a *call* initially decomposes into the logical components (instances) of *go-off-hook*, *make-call*, and *disconnect-call*. Each of these components is, in turn, further decomposed on the Instance Level. The temporal relationships associated with each instance define the instance's location to the other instances.

Instance Level knowledge also includes descriptions of the various telephony features. Each feature, such as Three-Way Calling (3WC) and Calling Number Delivery Blocking (CNDB), decomposes into a structure similar to the decomposition of a *call* shown in Figure 3. These decompositions detail the individual operator actions that enable each feature, resultant system actions, and temporal relationships between feature components and *call* components.

Besides modeling individual features, the feature descriptions contain restrictions and special interactions between features. For example, since the features of 3WC and CNDB are compatible but interact, the interaction must be specified. See Figure 4.

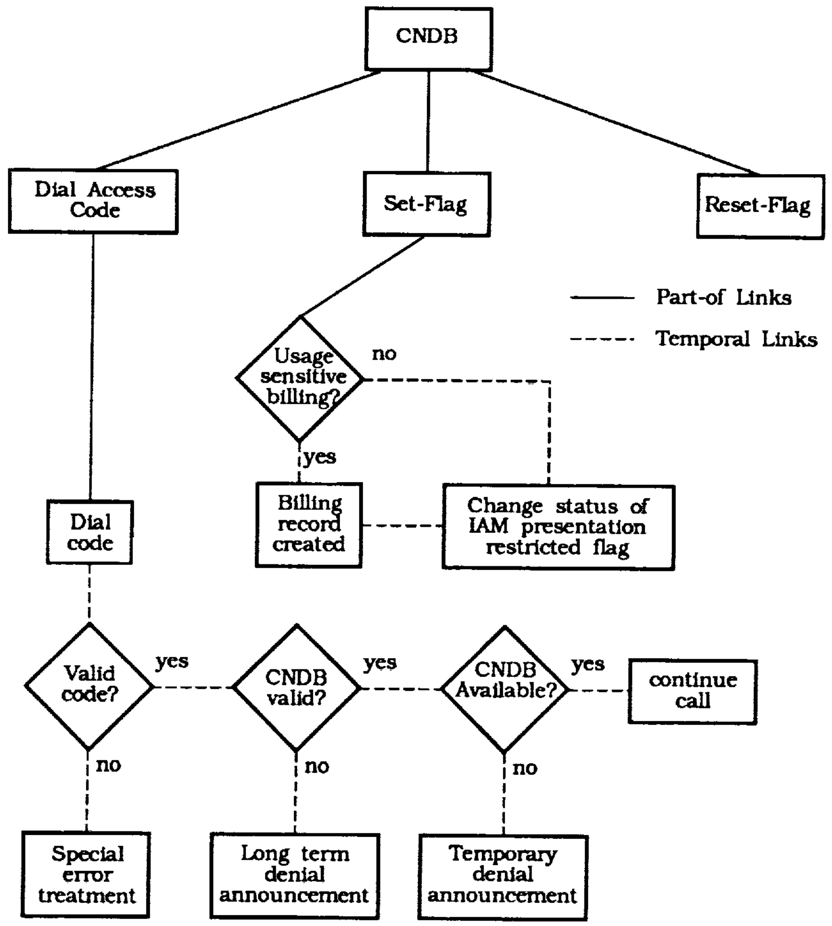

Figure 3. Decomposition of *CNDB*

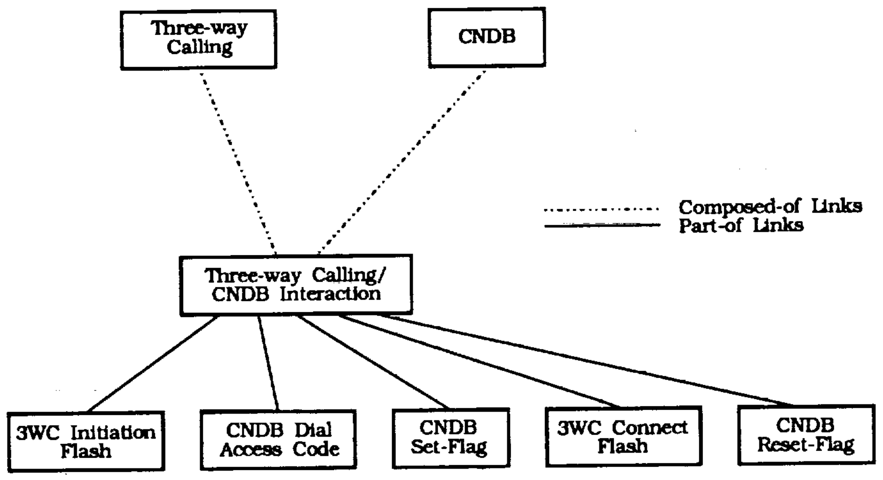

Figure 4. Feature Interaction with the Feature Hierarchy

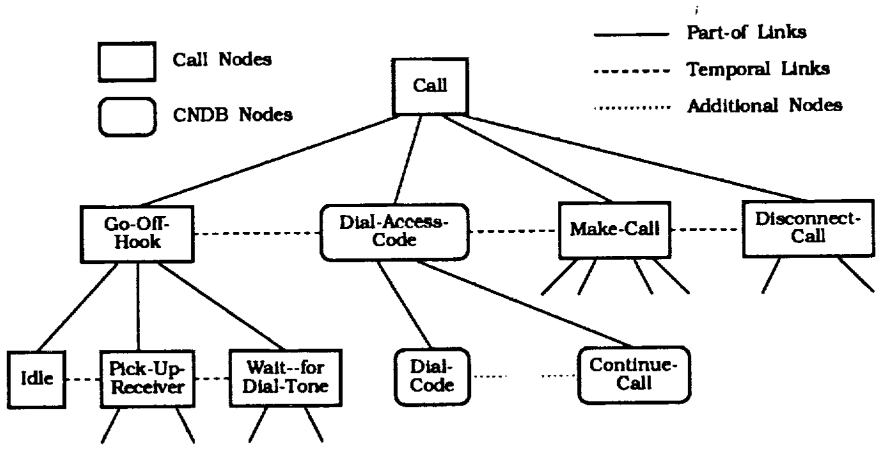

Figure 5. Model Representation of a *Call* with *CNDB*

3.3. Models

The Model Level facilitates the building of a description of a particular telephone call exhibiting specific features. While the knowledge on the Concept Level changes very little (due to the operation of Composition [DANK92]) and the primary goal of GATOR is for the user to build Instance Level knowledge of some new feature, the Model Level is significantly more dynamic.

While operating GATOR, the user can request a description of a telephone call which exhibits a particular feature or set of features. The system examines its Instance Level knowledge and retrieves the appropriate instances representing a general call and the set of features of interest to the user. These instances combine to develop a model of the requested call type as shown in Figure 5. The instance ordering depends upon any explicit or implicit interactions and dependencies that exist between features and the feature specification order.

Upon completion, models are presented to the user. The display of a model allows the user to verify that knowledge represented on the Instance Level is correct and complete. Errors detected generally result from incomplete or incorrect specifications on the Instance Level. Each must be identified and corrected by the user. After locating and correcting an error, the user can verify that appropriate corrections were made by creating another model and examining its revised structure.

4. Status and Plans

Our current research in automated requirements gathering includes:

1. **Improvements and extensions to POND.** POND, as originally constructed, provides a rich environment for specifying set/member and part/sub-part relations. While the system currently includes the ability to specify temporal information, it does not provide a unified temporal reasoning component or subsumption, each of which require further definition and incorporation.

2. **Expansion of the knowledge base to include additional feature knowledge.** Currently, the feature knowledge base consists of a limited set of telephony features. This knowledge base needs to be significantly expanded to provide an adequate environment for developing new features, specifying feature interactions, testing model building, and expanding the natural language interaction.

3. **Development of a Specification Document Generator.** Once feature knowledge has been captured within GATOR's knowledge base, it must be made more accessible to developers, implementors, and customers. An output generation system is currently under design with which the user will be able to produce a feature requirements specification document.

4. **Extension of the Natural Language Capabilities.** The current system is limited in the range of input which is can process. We are expanding the syntactic and semantic capability of the system to more closely model the range of language used by designers when they describe a feature's structure.

While our research has concentrated on telephony, our approach is applicable to a wide range of domains. An initial examination of telephony features has shown that GATOR can capture 80 to 90 percent of the functional requirements of a feature contained in a typical specification document. We expect that the use of such an automated tool, in this and other domains, will significantly reduce ambiguities and uncertainties within specification documents, thereby decreasing development time and expense.

5. References

[ALLE85] Allen, J., *Maintaining Knowledge about Temporal Intervals*, in **Reading in Knowledge Representation**, edited by R. J. Brachman and H. J.. Levesque, Morgan Kaufman, Los Altos, CA, pp. 509 - 521, 1985.

[BRAC85] Brachman, R. J. and J. G. Schmolze, *An Overview of the KL-ONE Knowledge Representation System*, **Cognitive Science**, Vol. 9, No. 2, pp. 171 - 216, 1985.

[BRAC89] Brachman, R. J., A. Borgida, D. L. McGuinness, and L. Alperin Resnick, *The CLASSIC Knowledge Representation System, or, KL-ONE: The Next Generation*, **Workshop on Formal Aspects of Semantic Networks**, Santa Catalina Island, CA, 1989.

[DANK92] Dankel, D. D., W. Walker, and M. Schmalz, *POND: A Knowledge Representation Language which Facilitates Requirements Capturing*, Working Paper submitted to the **12th International Avignon Conference**, 1992.

[WOOD90] Woods, W. and J. Schmolze, *The KL-ONE Family*, TR-20-90, Center for Research in Computing Technology, Harvard University, Cambridge, MA, 1990.