NAVAL POSTGRADUATE SCHOOL
Monterey, California

DTIC
ELECTE
JAN 2 4 1996
S
D
G

THESIS

THE OBJECT-ORIENTED DATABASE AND PROCESSING OF ELECTRONIC WARFARE DATA

by

J. J. Lee
and
Thomas D. McKenna

March 1996

Thesis Advisor: David K. Hsiao
Co-Advisor: C. Thomas Wu

Approved for public release; distribution is unlimited.

19960122 075

DTIC QUALITY INSPECTED 1

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time reviewing instructions, searching existing data sources gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget. Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1996	Master's Thesis

4. TITLE AND SUBTITLE
The Object-Oriented Database and Processing of Electronic Warfare Data

5. FUNDING NUMBERS

6. AUTHOR(S)
J. J. Lee
Thomas D. McKenna

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/ MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Electronic Warfare Integrated Reprogramming (EWIR) database is the primary Department of Defense source for technical parametric performance data on noncommunications emitters. It has been identified by the National Air Intelligence Center as difficult to use in its current hierarchical database form. There are two problems addressed by this thesis. First, is an object-oriented EWIR database a superior method for managing complex electronic warfare data collections? Second, is the prototype Object-Oriented Interface (O-OI) developed at the Laboratory for Database System Research in the Naval Postgraduate School capable of supporting a complex object-oriented database such as EWIR?

(continued on back)

14. SUBJECT TERMS	15. NUMBER OF PAGES
Object-Oriented Database; Electronic Warfare Integrated Reprogramming; Multimodel and Multilingual Database	101
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

To answer these questions, a *subset* of the EWIR Object-Oriented Specification developed in a separate thesis is implemented on the O-OI. Using the O-OI Data Definition Language, the object-oriented EWIR database schema and its associated record data are stipulated and loaded to create the live database. Using the O-OI Data Manipulation Language, nine EWIR transactions are elaborated and executed.

The first result of this thesis is the O-OI performs as specified, but requires additional data manipulation and logical control functions to handle complex databases. The minimum additional functions include *Insert*, *Delete*, and *If-then-else*. The inheritance feature also requires a generalization-to-specialization data retrieval capability. The second result of this thesis is the straightforward data manipulation capability of the object-oriented version of the EWIR database. The object-oriented specification more accurately captures data relationships. The inheritance, path, and object comparison features streamline the linkage of related data, thus simplifying ad hoc query construction.

Approved for public release; **distribution is unlimited**

THE OBJECT-ORIENTED DATABASE AND PROCESSING OF ELECTRONIC WARFARE DATA

J. J. Lee
Lieutenant Colonel, Taiwan Army
B.S.E., Chung-Cheng Institute of Technology, 1981

Thomas D. McKenna
Lieutenant Commander, United States Navy
B.S., University of South Carolina, 1980

Accesion For
NTIS CRA&I [X]
DTIC TAB [ ]
Unannounced [ ]
Justification .....
By .....
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

March 1996

Authors: _____
J. J. Lee

_____
Thomas D. McKenna

Approved by: _____
David K. Hsiao, Thesis Advisor

_____
C. Thomas Wu, Co-Advisor

_____
Ted Lewis, Chairman
Department of Computer Science

ABSTRACT

The Electronic Warfare Integrated Reprogramming (EWIR) database is the primary Department of Defense source for technical parametric performance data on non-communications emitters. It has been identified by the National Air Intelligence Center as difficult to use in its current hierarchical database form. There are two problems addressed by this thesis. First, is an object-oriented EWIR database a superior method for managing complex electronic warfare data collections? Second, is the prototype Object-Oriented Interface (O-OI) developed at the Laboratory for Database System Research in the Naval Postgraduate School capable of supporting a complex object-oriented database such as EWIR?

To answer these questions, a *subset* of the EWIR Object-Oriented Specification developed in a separate thesis is implemented on the O-OI. Using the O-OI Data Definition Language, the object-oriented EWIR database schema and its associated record data are stipulated and loaded to create the live database. Using the O-OI Data Manipulation Language, nine EWIR transactions are elaborated and executed.

The first result of this thesis is the O-OI performs as specified, but requires additional data manipulation and logical control functions to handle complex databases. The minimum additional functions include *Insert*, *Delete*, and *If-then-else*. The inheritance feature also requires a generalization-to-specialization data retrieval capability. The second result of this thesis is the straightforward data manipulation capability of the object-oriented version of the EWIR database. The object-oriented specification more accurately captures data relationships. The inheritance, path, and object comparison features streamline the linkage of related data, thus simplifying ad hoc query construction.

TABLE OF CONTENTS

I. INTRODUCTION 1

II. M^{2}DBMS BACKGROUND 3
A. AN OVERVIEW OF THE M2DBMS 3
B. THE DATA DEFINITION IN THE OBJECT-ORIENTED INTERFACE. 5
1. The Class Specification 6
2. The Inheritance Specification. 7
3. The Cover Relationship 8
4. Set Relationships 8
C. THE DATA MAINIPULATION IN THE OBJECT-ORIENTED INTERFACE 9
1. Object Identifiers 11
2. Object-Oriented Operations 11
3. The Query Constructs 12
D. LIMITATIONS AND ADVANTAGES 12

III. THE NEW EWIR DATABASE SPECIFICATION 15
A. SOME BACKGROUND ON EWIR 15
B. THE EWIR DATA MODEL: THE OLD HIERARCHICAL VERSUS THE NEW OBJECT-ORIENTED 16
C. THE OBJECT-ORIENTED IMPLEMENTATION OF EWIR DATA 18
D. THE NEW EWIR OBJECT-ORIENTED DATABASE 21
1. A Top-Level View of the New EWIR Database 21
2. The Antenna Data 22
3. The EWIR Signal Data 24
4. The EWIR Receiver Data 25
5. The EWIR WARM Data 25
6. The EWIR Parametric Data 28

IV. THE IMPLEMENTATION OF THE EWIR DATABASE 31
A. THE DATA DEFINITION OF THE EWIR DATABASE 31
1. The Schema Listing 33
2. The Data Dictionary 33
3. The Template File 33
4. The Descriptor File 36
B. THE EWIR DATABASE RECORD DATA 39
1. A Proposed EWIR Record Template 40
2. The Mass Load Record File (EWIROODB.r) 40

V. THE MANIPULATION OF EWIR DATA 45
A. THE GOALS OF THE EWIR TRANSACTIONS 45
B. THE EWIR TRANSACTIONS 45
1. An Attribute Look-up (Query_1) 47
2. A Single-level Inheritance and its 1:1 Relationship (Query_2) 48
3. Multi-level 1:1 Relationships (Query_3) 49
4. A Two-level Inheritance (Query_4) 52
5. A Five-level Inheritance and One-level 1:1 Relationship (Query_5) 54
6. The Retrieval of the Emitter ELNOT (Query_6) 55
7. The Retrieval of Parametric Data Given an ELNOT (Query_7) 56
8. Multiple Logical Operators and Changing Data (Query_8) 59
9. The Data Retrieval From Multiple Subsections of EWIR (Query_9) 60

VI. CONCLUSIONS 63
A. THE THESIS METHODOLOGY SUMMARY 63
B. THE EVALUATION OF THE OBJECT-ORIENTED INTERFACE 64
C. THE EVALUATION OF THE OBJECT-ORIENTED EWIR DATABASE 65

APPENDIX A. THE EWIR SPECIFICATION 67

APPENDIX B. THE NEW EWIR DATABASE DATA DICTIONARY 75

APPENDIX C. THE EWIR TEMPLATE FILE 81

LIST OF REFERENCES 87

INITIAL DISTRIBUTION LIST 89

ACKNOWLEDGEMENTS

We would like to thank Dr. David K. Hsiao and Dr. C. Thomas Wu for their time and advice during the research and writing of this thesis. Without their knowledge and expertise this thesis would not have been possible.

We would also like to thank CDR Bruce Badgett, LTjg Aykut Kutlusan, LTjg Erhan Senocak, LT Kevin Coyne, and Army Captain T.W. Kwon. Their technical assistance was pivotal in the success of this thesis.

I. INTRODUCTION

The objective of our thesis is to research into the application of object-oriented database management for the Electronic Warfare Integrated Reprogramming (EWIR) data. This research is driven by the following factors:

- Object-oriented database management is the latest database technology.
- The effectiveness of object-oriented data management has not been realized on any electronic warfare data.
- DOD representatives have requested our database reengineering research with object-oriented methodology and technology on the EWIR data.

In conducting this research, the following *goals* are achieved:

- Determine whether the object-oriented EWIR database is more effective, efficient, and intuitive for managing complex electronic warfare data collections than conventional database systems.
- Evaluate the new object-oriented-data-model-and-data-language interface (for brief, the *Object-Oriented Interface*) of the Multimodel and Multilingual Database System (M^2DBMS) and its capability in supporting the object-oriented EWIR database.

To achieve the above, this thesis consists of the specification, implementation, processing, and evaluation of a proposed object-oriented EWIR database (for brief, the *new EWIR database*). The new EWIR database is derived from the object-oriented specification of the existing EWIR database [Ref. 1]. The EWIR evaluation includes a sequence of object-oriented queries through the Object-Oriented Interface of the M^2DBMS.

This thesis is the second of a two-thesis project designed to transform the existing EWIR data into a working object-oriented database. The first thesis [Ref. 1] is responsible for the conceptual design of the object-oriented specification of the existing EWIR database. The conceptual design transforms the EWIR's existing hierarchical and flat-file data collections into an object-oriented database specification. This second thesis uses the conceptual design of Ref. 1 to create an implementable subset of the EWIR database on the M^2DBMS. The Object-Oriented Data Definition Language (O-ODDL) is used to specify

and create the EWIR object-oriented database [Ref. 2]. The Object-Oriented Data Manipulation Language (O-ODML) is used to specify and write a series of object-oriented queries on the newly created object-oriented EWIR database for processing [Ref. 3].

In Chapter II of this thesis, an overview of the M^2DBMS is given. In Chapter III, an examination of the existing EWIR database and the new EWIR database implemented in this thesis is presented. The implemented subset is in terms of a new object-oriented specification. In Chapter IV, an examination of the new EWIR database schema in terms of the O-ODDL is detailed. The new EWIR database record data is also specified. In Chapter V, we cover the O-ODML specification of nine transactions, their processing purpose, and an analysis of the returned results. In Chapter VI we have concluding remarks.

II. M^{2}DBMS BACKGROUND

In this chapter, the necessary system software, language features, and supporting architecture are expounded. Our new object-oriented EWIR database is implemented on the Object-Oriented Interface of the M^{2}DBMS [Ref. 4]. To this end, Section A will provide an overview of the M^{2}DBMS.

The first step in creating the EWIR database on the Object-Oriented Interface involves EWIR data definition. Thus, the Object-Oriented Interface data definition *process* is examined in Section B as well as the DDL constructs. There must also exist a Data Manipulation Language (DML) to process the data. Data manipulation in a database is done via *queries*. How the Object-Oriented Interface processes a query and the DML constructs are the subjects of Section C.

Because M^{2}DBMS and its associated Object-Oriented Interface are research projects, some limitations have been designed into the system. Section D will summarize those limitations plus list some of the system advantages.

A. AN OVERVIEW OF THE M^{2}DBMS

A major goal of this thesis is to evaluate the performance of the object-oriented interface of the M^{2}DBMS with a complex data set, namely EWIR. An understanding of the unique approach that M^{2}DBMS brings to database systems design is important for two reasons. First, this system offers a distinct paradigm shift for accessing information across heterogeneous multi-databases. Second, any subsequent discussion on the design and implementation features of the Object-Oriented Interface must first be placed in the context of its host database system.

The definition of a multi-database system includes the support of heterogeneous databases, each of which is based on a different data model. The M^{2}DBMS differs in that it is not a collection of separate data models and database languages. Instead, the M^{2}DBMS organizes all of its heterogeneous databases on a *single* data model and data

language. As depicted in Figure 1, this common data model and data language is called the *database kernel*. The M^{2}DBMS kernel uses the *attribute-value pair* as the atomic unit of

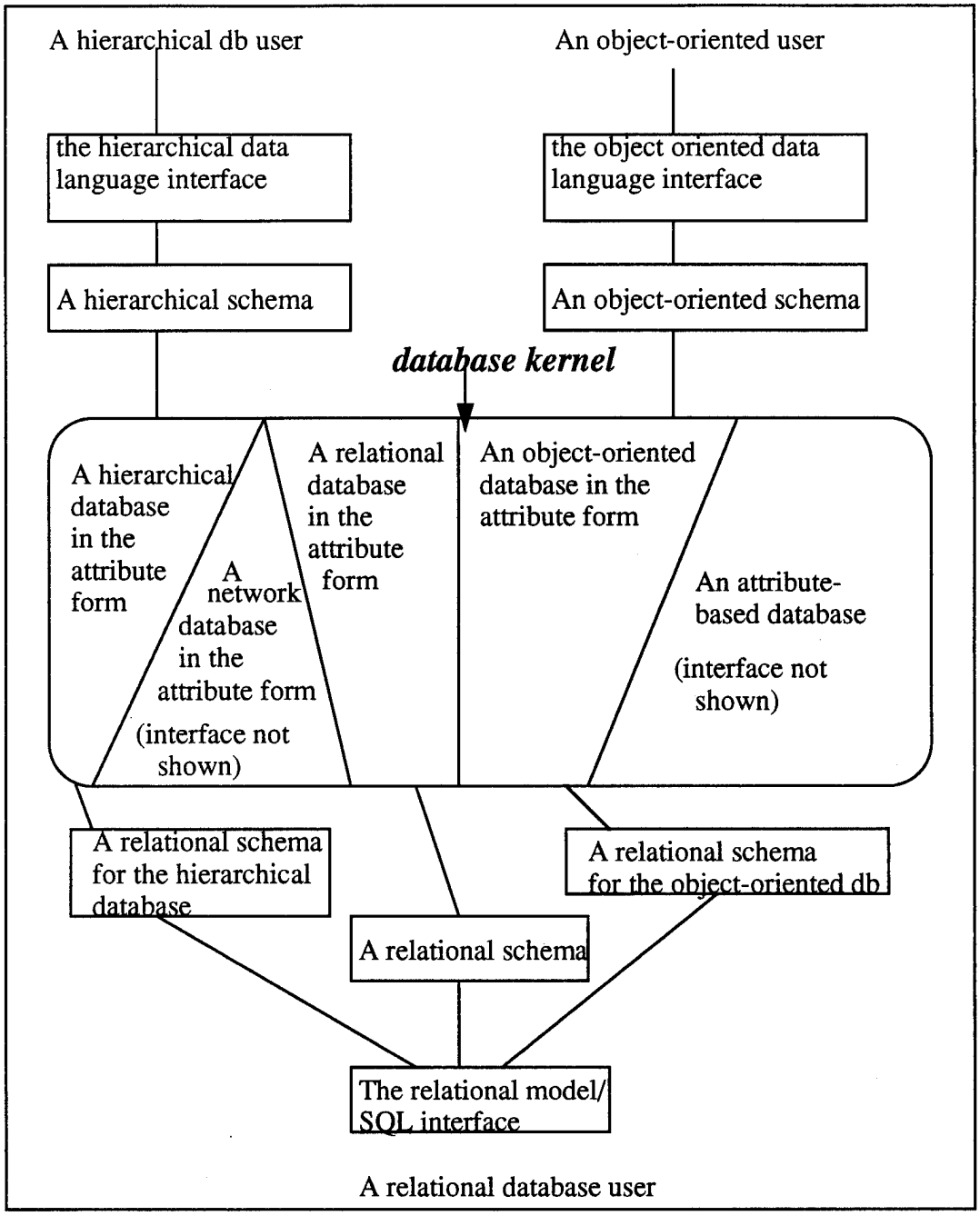


Figure 1. Data Sharing in the M^{2}DBMS - An Example of the Relational User Accessing Object-Oriented Data Relationally.

data. The attribute-value pair consists of a system generated object identifier (attribute) and its associated value. To distinguish one kernelized database from the other within M^2DBMS, we use their corresponding *schemas*. A *database schema* is a database definition of the attributes and the relations in the database. The M^2DBMS contains a relational schema, hierarchial schema, network schema, functional schema, and the newly completed object-oriented schema.

The database user needs no knowledge of how the underlying system translates the data to a kernel format. In addition, the kernel system and schemas of the M^2DBMS allow translation to occur *between* heterogeneous databases within the system. For example, a relational user can access information from an object-oriented database in a relational query language. *The relational schema for an object-oriented database* will translate the object-oriented data into its relational equivalent for display. Also in Figure 1 is a conceptual diagram of how heterogeneous database data is shared.

B. THE DATA DEFINITION IN THE OBJECT-ORIENTED INTERFACE

Because all physical data resides in the kernel, the object-oriented data model (O-ODM) must be mapped to its Attribute-based Data Model (ABDM) equivalent. A mapping is a translation from one data representation to another. The mapping from object-oriented to ABDM is as follows: The Object-Oriented Data Definition Language (O-ODDL) accepts as its input the specification of an object-oriented database schema [Ref. 5]. The O-ODDL compiler checks the correctness of the specification via a *scanner* and *parser*. The scanner checks the O-ODDL statement syntax while the parser checks ensures the statement follows the grammatical rules of the language. The compiler also generates a *data dictionary* for use by the query constructor. The compiler *descriptor* and *template* files are the object-oriented specification in an equivalent kernel database. In Figure 2 is a diagram of the data definition process.

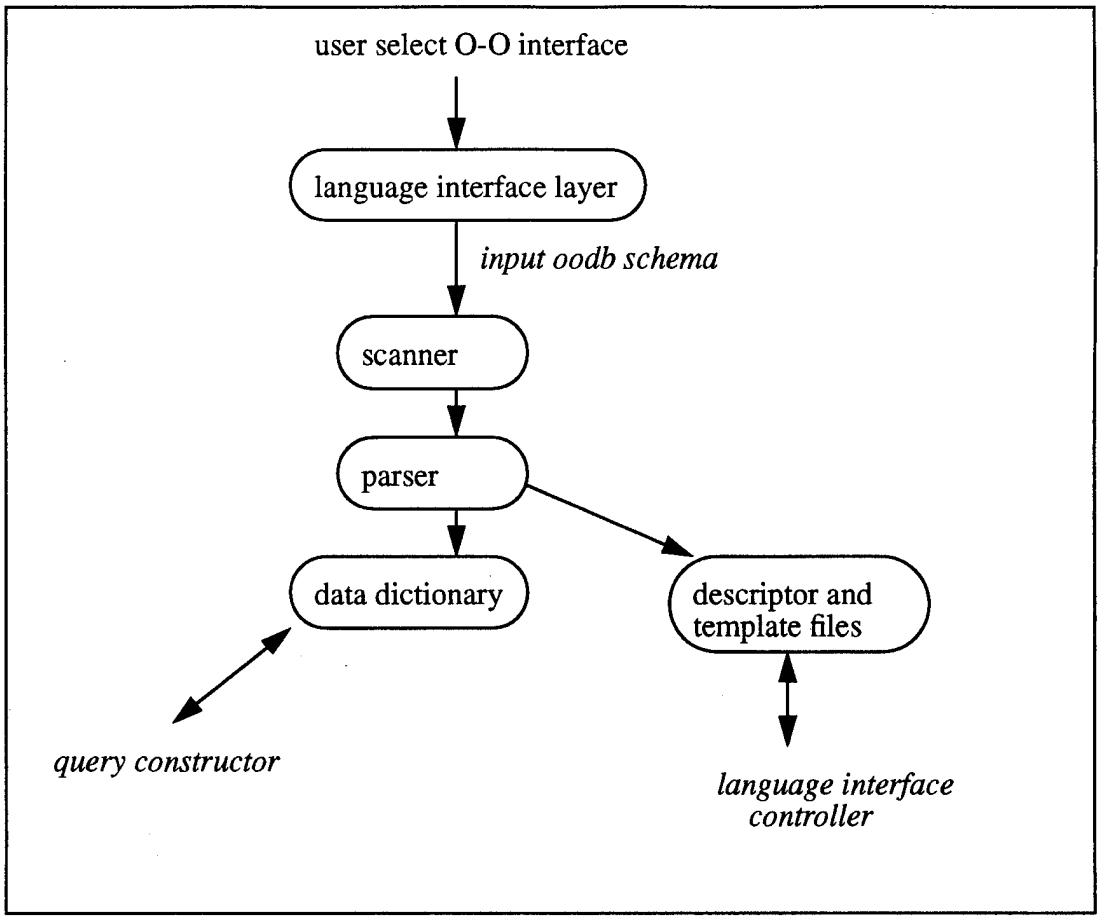


Figure 2. The Data Definition Process.

The O-ODDL provides the constructs for creating a new database schema. The following O-ODDL specifications are used by the Object-Oriented Interface to create a schema for an object-oriented database.

1. The Class Specification

A class is the grouping of objects which share common attributes and methods. Attributes are data types which provide value and meaning to a class. A method is a procedure within a class used to manipulate information. In Figure 3 is the O-ODDL template for the specification of a class. Methods have not been implemented on the Object-Ori-

ented Interface and are omitted from the class specification.

```
Class Class_name{
  attribute_type_1   attribute_name_1;
      .
  attribute_type_m   attribute_name_m;
  };
```

Figure 3. The Specification of a Class.

Attribute_type represents the domain of the attribute. The domain may be of a simple type, such as an integer or character, or a complex type, such as another class. The *attribute_name* is the user assigned name of the attribute and functions as the placeholder for its value.

2. The Inheritance Specification

Inheritance is the receipt of all attributes and methods from another class. Inheritance is described as a hierarchy. In this hierarchy, a subclass inherits from the superclass. The subclass has attributes in addition to those which were inherited. Thus, the subclass is a specialization of the superclass. The inheritance specification is listed in Figure 4. *Class_name1* is the subclass (specialization) of *Class_name2* (generalization). The attributes in this specification are unique to *Class_name1* and not part of *Class_name2*.

```
Class Class_name1 :Inherit Class_name2{
  attribute_type_1 attribute_name_1;
      .
      .
  attribute_type_m attribute_name_m;
  };
```

Figure 4. The Inheritance Specification.

3. The Cover Relationship

Covering defines a mapping relationship between an object in a class (the cover class) to a set of objects in a second class (the member class). The covering specification is depicted in Figure 5. In this instance, *Class_name1* is the cover class and *Class_name2* is the member class. This means an object of class *Class_name1* maps to a subset of objects from class *Class_name2*.

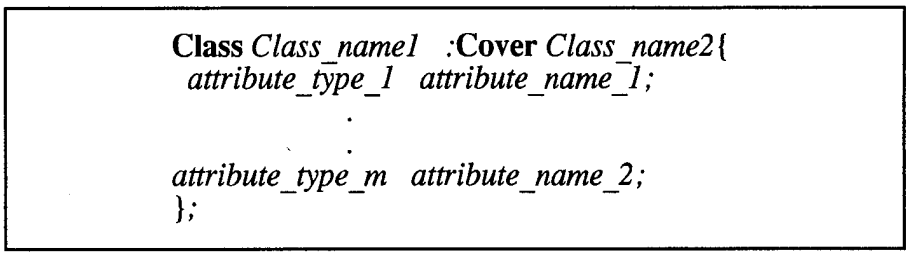


Figure 5. The Cover Specification.

4. Set Relationships

The set relationship allows an attribute of a class to be more than just a singleton value. *Set_of* forms a one-to-many (1:N) relationship. *Inverse_of* is the complement of *set_of* and allows a many-to-many (M:N) relationship. Figure 6 specifies the *set_of* and *inverse_of* constructs allowing the O-ODM to represent 1:N and M:N relationships of two object classes.

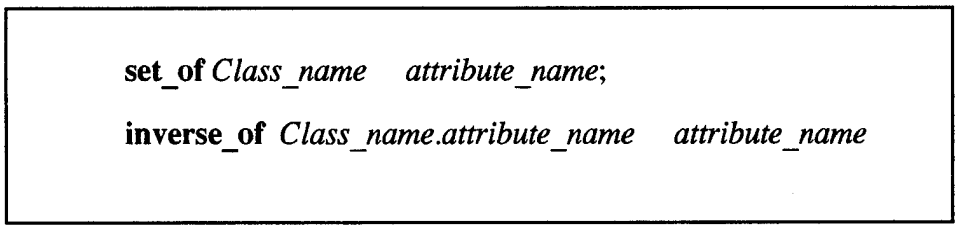


Figure 6. The Specification for Set_of and Inverse_of.

C. THE DATA MANIPULATION IN THE OBJECT-ORIENTED INTERFACE

A query uses DML constructs to manipulate data in a database. The O-ODML compiler delineates the language constructs used to process and execute high-level queries [Ref. 6]. A high-level query language specifies *what* data is to be manipulated rather than *how* to manipulate the data.

In Figure 7 is a diagram of the steps involved in O-ODML compiler processing. First, a query expressed in a high-level query language must be scanned, parsed and validated. The *scanner* identifies the language components (tokens) in the text of the query while the *parser* checks the query syntax to ensure it is formulated in accordance with the syntactical rules of the query language. The scanner builds a *symbol table* composed of a variable name, variable type, and class name. The symbol table information is required for later use by the *query constructor.*

The parser completes an *intermediate table.* The intermediate table stores the pertinent information from the query. Pertinent information includes the data manipulation operation to be performed and any associated arguments. Some query optimization may also be possible using the intermediate table data. Optimization is the reordering and restructuring of query information to minimize the time spent searching for data in the database. Ultimately, the intermediate table information is used by the query constructor.

The query constructor extracts information stored in the Data Dictionary, Symbol Table, and Intermediate Table and produces as output either a *nearly-executable* equivalent transaction or a *well-formed* query. Nearly-executable means some of the ABDML statements require additional information that must be retrieved from the kernel data. Well-formed means all information required to execute the query is immediately available. Retrieval of any missing information is the responsibility of the *Real Time Monitor.*

The *Real Time Monitor* accepts these nearly executable or well-formed ABDML statements from the query constructor and, using the data dictionary, assembles the requisite information to create fully executable queries in the ABDBMS [Ref. 7]. The Real Time Monitor also sends the returned results of the query to a *Kernel Formatting System*

(KFS). The KFS puts the retrieved information in the proper format for output to the user [Ref. 8].

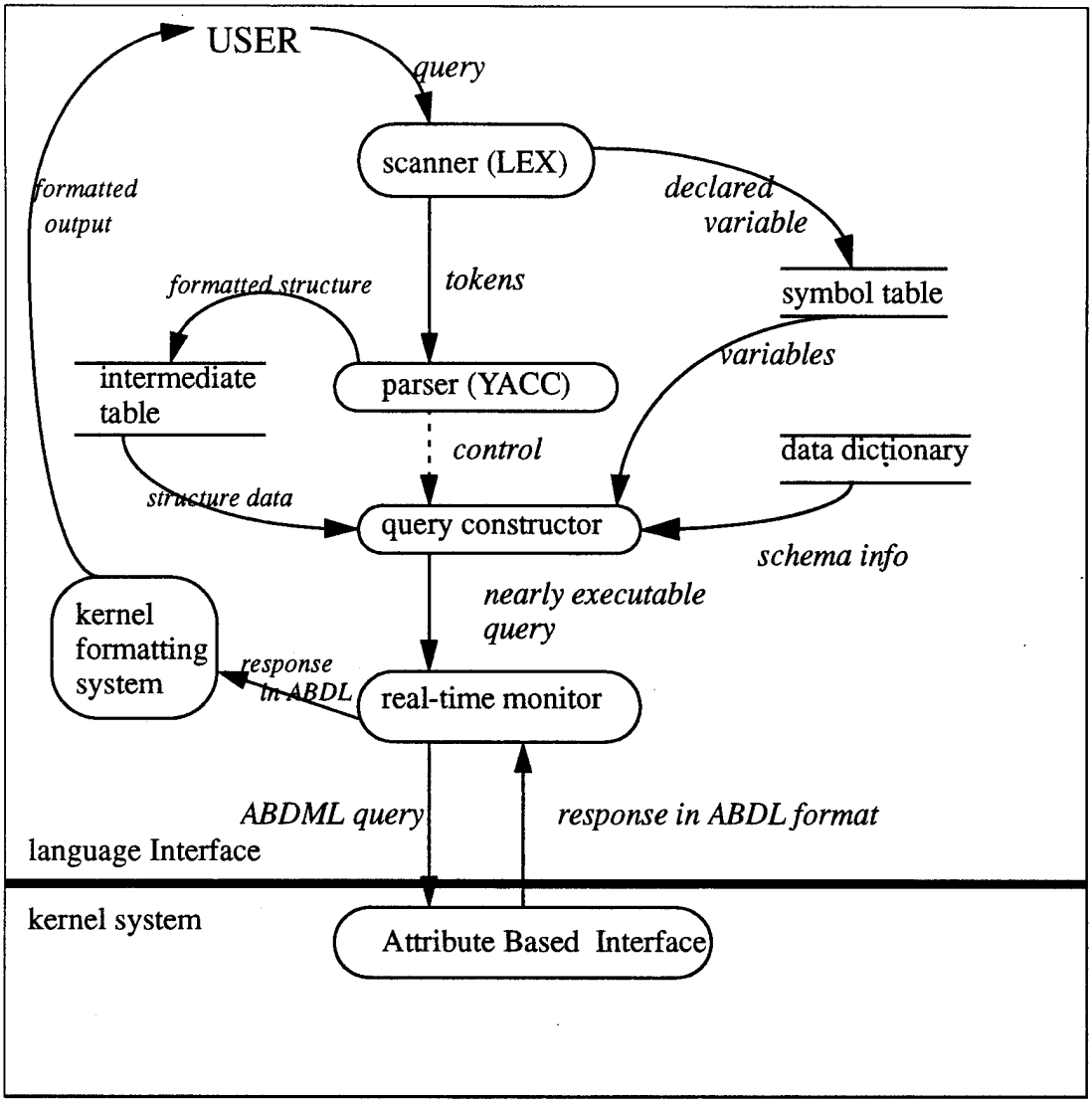


Figure 7. Query processing.

The O-ODML is designed for writing transactions and processing queries which can be executed within the M^2DBMS via the Object-Oriented Interface. Manipulating a database includes such functions as retrieving specific data, updating the database, and

generating reports. Typical manipulations include insertion, deletion, retrieval and modification of the data. In Ref. 3, Appendix A is the full *Object-Oriented DML Users Manual* for the Object-Oriented Interface. Important concepts of the object-oriented DML include:

1. Object Identifiers

Every object has a unique, system generated identifier. Identifiers uniquely label each data object and provide the system a means to identify and perform manipulations on created objects in the database. Identifiers are maintained by the system and are not available to the users.

2. Object-Oriented Operations

Although methods are an integral part of object-orientation, the Object-Oriented Interface does not currently support methods for objects. The addition of method functionality to the Object-Oriented Interface is a topic for future research on this system.

Instead of methods, the Object-Oriented Interface uses object-oriented *operations*. Object-oriented operations are transactions defined for use on an object of the object class. These are not methods because they are not contained within any object class of the database. These operations are generic and not tied to specific data aggregates. In Figure 8 is an example of two operations available as listed in Table 1 of Ref. 3.

OPERATION	SYNTAX	SEMANTICS
Find_one	find_one *class_name*	Searches and returns only the first object from *class_name* that satisfies the expression.
add	add (obj_ref, *attr*)	Adds an attribute, a single value, to a list or set of attributes in a cover relation.

Figure 8. Examples of Operations.

3. The Query Constructs

The Object-Oriented Interface query is a structured collection of declarations and operations in a block format. The object-oriented operations [Ref. 3, Table 1], the DML production rules [Ref. 3, Table 2], and reserved words [Ref. 3, Table 3] provide the functionality needed to construct high-level queries. Each query is compiled by the object-oriented compiler [Ref. 6]. The compiler ensures each query is formatted and structured within strict guidelines to be recognized as a legitimate. The format of a query is divided into five basic parts as illustrated in Figure 9.

	Syntax	Semantics	Example
part 1:	QUERY id IS	Query Heading	QUERY add_radar IS
part 2:	*type* id;	Declarations Part	obj_ref p;
part 3:	BEGIN	Reserved word	BEGIN
part 4:	*statement_list*;	body of *executable statements* and ops	p:= insert Radar; p.freq:= 12345
part 5:	END;	Reserved word	END;

Figure 9. Query Constructs.

D. LIMITATIONS AND ADVANTAGES

The M^2DBMS uses many of the features and constructs of object-oriented programming languages. These features include the concepts of unique objects, object classes, inheritance, encapsulation and covering. Because the M^2DBMS is a research project, it does not have the full range of features and functionality expected in a commercial product. Some current limitations on the system include:

- No methods within a class.
- No floating-point arithmetic.

- Only four logical operators in a single statement.
- Only a subset of the operations listed in Table 1 of Ref. 3 have been implemented. The operations currently available are *find_one*, *find_many*, *display*, *add*, and *contains*. The remaining operations are to be implemented at a later date.

The kernel system must service several different database systems while being limited to an attribute-value pair format. All database functionality, as perceived by the user, is emulated in software above the kernel system. This makes implementation more complex than a simple uni-database system, but the advantage and flexibility achieved in a multimodel and multilingual system more than offsets the more complex implementation.

III. THE NEW EWIR DATABASE SPECIFICATION

As stated in Chapter I, the implementation of the EWIR database using the Object-Oriented Interface is a primary objective in this theses. Thus, it is important to understand the background and structure of the existing EWIR database. In Section A of this chapter, the basic background information on the existing EWIR database is provided. In Section B, a comparison of the existing, non-object-oriented EWIR data model versus the object-oriented specification of the existing EWIR database [Ref. 1] is given. In Section C, implementation issues of the new EWIR database are given. In Section D, there is the object-oriented specification of the new EWIR database.

A. SOME BACKGROUND ON EWIR

The existing EWIR database is the primary Department of Defense approved source for technical parametric and performance data on noncommunications emitters and associated systems. The EWIR system provides an up-to-date and accurate source of information for reprogramming United States Electronic Warfare (EW) Combat Systems. EWIR includes parametric data on radars, jammers, navigational aids, and numerous non-communication electronic emitters.

The EWIR database is chosen as the test database for an object-oriented implementation for several reasons. First, it is a widely used, real-world application with sufficient complexity to adequately evaluate the versatility and utility of the Object-Oriented Interface. Second, the EWIR database is identified by the National Air Intelligence Center (NAIC) as difficult to understand and use in its current form. Third, it possesses properties that make it an acceptable candidate for object-orientation. These properties include easily identifiable objects and relationships between objects.

Users of the present EWIR database are united in their assessment that the current hierarchical data model relies heavily on the users' understanding of the data relationships, many of which are not explicitly depicted in the data model. In addition to the poor data modeling, the EWIR database format is difficult to interpret where codes are not stan-

dardized for all record types. Further, the database is described in terms of the physical storage structure.

B. THE EWIR DATA MODEL: THE OLD HIERARCHICAL VERSUS THE NEW OBJECT-ORIENTED

A detailed analysis and comparison of the two EWIR data models is presented in Ref. 1. This thesis validates the object-oriented model of EWIR data as a vast improvement over the original hierarchical one by creating a live object-oriented EWIR database.

The parametric data associated with electronic emitters in the EWIR is represented as a logical tree. A small part of the EWIR parametric tree is shown in Figure 10. This parametric tree orders a long list logically and hierarchically in a way that proceeds from broad characteristics through levels of successively finer ones. Parametric data exist in subfiles of the tree structure. Subfiles are major groupings, or subtrees, within the parametric tree which contains logically related data.

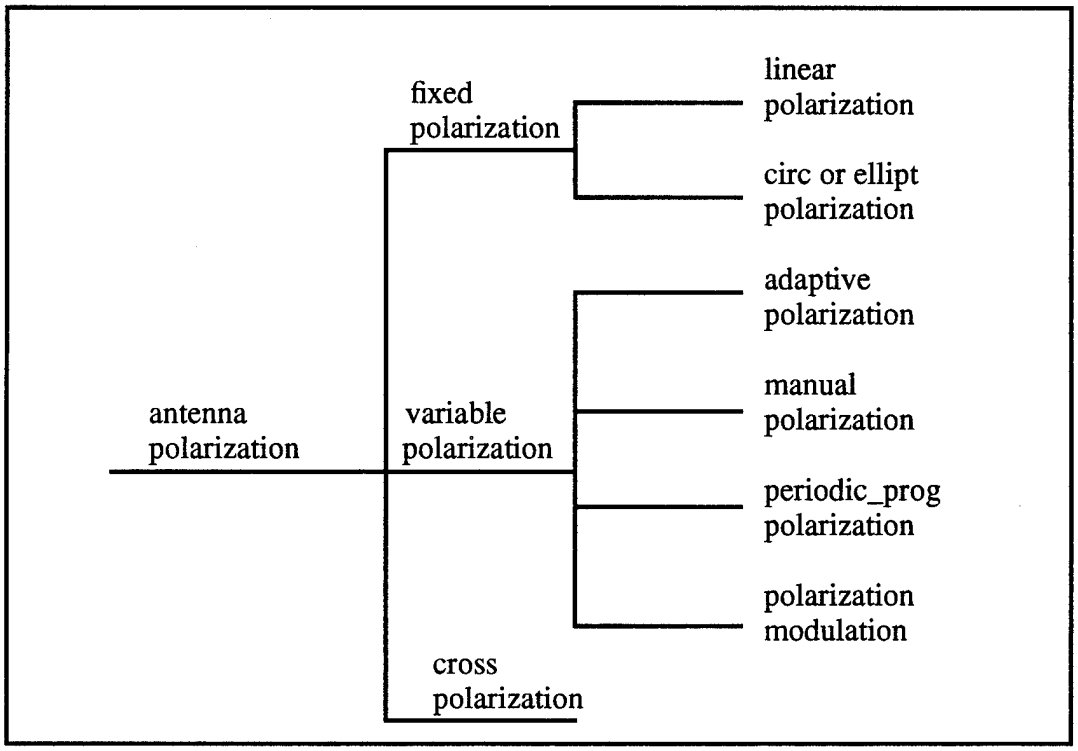


Figure 10. The EWIR Parametric Tree.

The parametric tree of Figure 10 demonstrates how poorly data relationships are modeled. For example, the hierarchical tree structure leads you to believe that antenna polarization is divided into three distinct types: fixed, cross and variable. In reality, antenna polarization can only be fixed or variable. The *cross polarization* is a characteristic or relationship *of all antennas* and not a discrete type of the antenna polarization. Thus, semantics of the cross polarization are not captured in the model. The responsibility for an understanding of the cross polarization is incumbent upon the user.

A second example of shortcomings of the current EWIR data model is the relationship of linear polarization to fixed polarization. Linear polarization is a potential characteristic of *all* antenna polarization types but is depicted as a part of the fixed polarization only. The hierarchical EWIR model again fails to accurately depict this relationship.

The above examples of the current EWIR model show deficiencies in capturing essential data relationship information. Related information is often scattered over many relations or records. A solution to this information gap is the use of an *Object-Oriented Data Model (O-ODM).* The O-ODM supports the modeling of object structures and interrelationships in a natural way. The O-ODM not only supports object structural definitions, but also the modeling of object behaviors and their dynamic constraints.

To illustrate the above, we present in Figure 11 an object-oriented version of the same subset of information as depicted in Figure 10. Notice that the polarization is divided into fixed and variable only. The cross polarization is depicted as an attribute of polarization and not a distinct type of polarization. Also observe that the linear polarization is a specialization type of the antenna polarization and not strictly a subset of the fixed polarization. The object-oriented data model embeds these data relationships into the data model. The responsibility for any knowledge of data relationships is no longer incumbent upon the user. When integrating object-oriented concepts into a database, the end result is a much more intuitive, natural, and powerful database system.

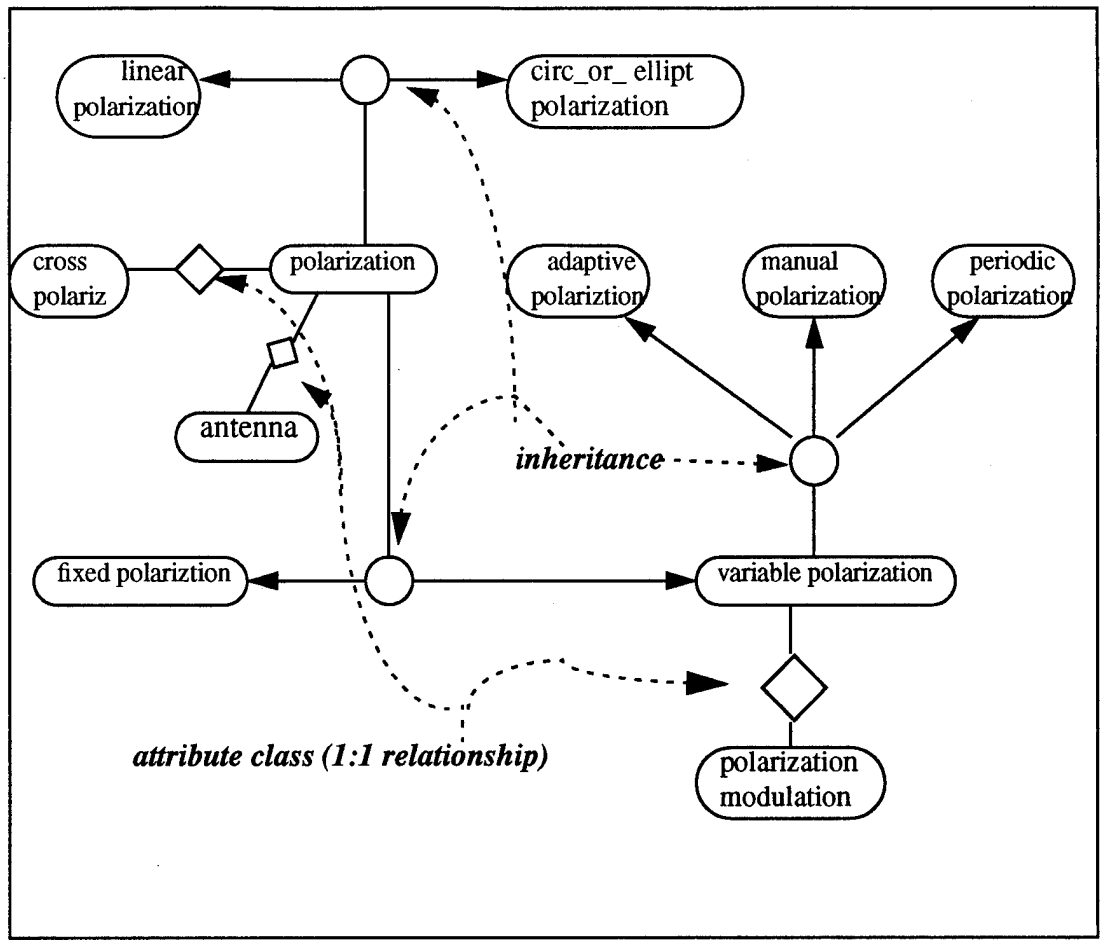


Figure 11. The Object-Oriented Model of the EWIR Tree in Fig. 10.

C. THE OBJECT-ORIENTED IMPLEMENTATION OF EWIR DATA

The exhaustive examination of the entire EWIR database [Ref. 1] and its object-oriented modeling are beyond the scope of this thesis. Here, we derive a subset of the object-oriented specification of the EWIR database [Ref. 1] and implement it on M^2DBMS as a live object-oriented database using the Object-Oriented Interface. The live database is to evaluate and demonstrate the effectiveness of the Object-Oriented Interface and its ability to manage complex object-oriented data relationships.

Approximately 20 percent of the fully specified object-oriented EWIR database [Ref. 1] is implemented in this thesis. This percentage is sufficient to test the Object-Oriented Interface and evaluate the effectiveness of an object-oriented EWIR database system. The thesis objectives are achieved for the following reasons:

- All of the data relationships (inheritance, 1:1, 1:N, M:N) represented in the fully specified object-oriented EWIR database are captured in the implemented subset. This includes the *depth* of inheritance relationships and *levels* of data relationships (i.e.,path or dot notation).
- The scale is smaller, but the complexity of data relationships is not diminished. Essential classes from each major EWIR sub-section are retained to allow for cogent and pertinent data manipulation. Thus, it is possible to evaluate whether or not the Object-Oriented Interface is more effective, efficient, and intuitive for managing the complex data collections of EWIR.

A change to the object-oriented specification of the existing EWIR database [Ref. 1] is required to implement the inheritance relationship. A problem with the current Object-Oriented Interface is that it allows the data retrieval in an inheritance relationship *only* from a specialization class to a generalization class. In other words, if transactions are in a specialization class, they can retrieve information from any generalization class from which it inherits. However, no transaction can retrieve data in an inheritance relationship starting from a generalization class and attempting to retrieve information from a specialization class from which it is inherited.

In Ref. 1, many of the relationships involve multiple levels of inheritance. Many of these inheritance hierarchies have entries at the highest level of the abstraction (generalization). Thus, using the Object-Oriented Interface it is impossible to retrieve data from successive specialization classes. In referring to Figure 12(a), the following is an example of information which cannot be retrieved: "For a given *Transmission_power*, list an attribute of *constant_power*". The current Object-Oriented Interface cannot navigate through the database in the generalization-to-specialization direction. On the other hand, the Object-Oriented Interface can process the following request: "List the *Transmission_power* when *constant_power.data* = xyx". This is specialization-to-generalization and the system can successfully navigate in this direction.

In Figure 12(b), we depict our modification in order to avoid classes that cannot be reached in the Object-Oriented Interface. The solution is the following: Any "leaf" nodes in the object-oriented specification of the existing EWIR database [Ref. 1] which are specialization classes are modified to include a 1:1 relationship with a generalization class. This allows an entry into the lowest-level specialization, thus providing access to data in all of its generalization classes.

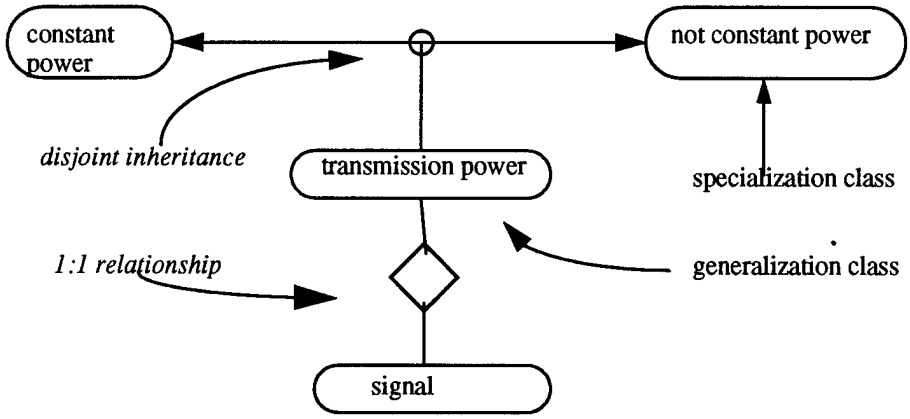


Figure 12(a): The Object-Oriented Specification [Ref. 1].

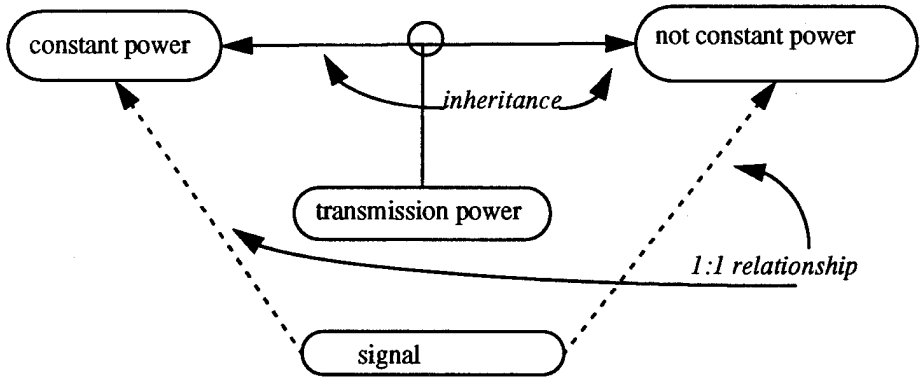


Figure 12(b):The Object-Oriented Interface Implementation.

Figure 12. A Comparison of Object-Oriented Specifications and Object-Oriented Implementation.

D. THE NEW EWIR OBJECT-ORIENTED DATABASE

This section is composed primarily of figures depicting the new EWIR object-oriented specification. In Figure 1 is the key for use in clarifying Figures 14 through 19. To minimize cluttering the figures, individual attributes of each class are omitted. Specific attribute information for each class is provided in Chapter IV. For ease of reference, Appendix A contains the object-oriented specification of the existing EWIR database [Ref. 1].

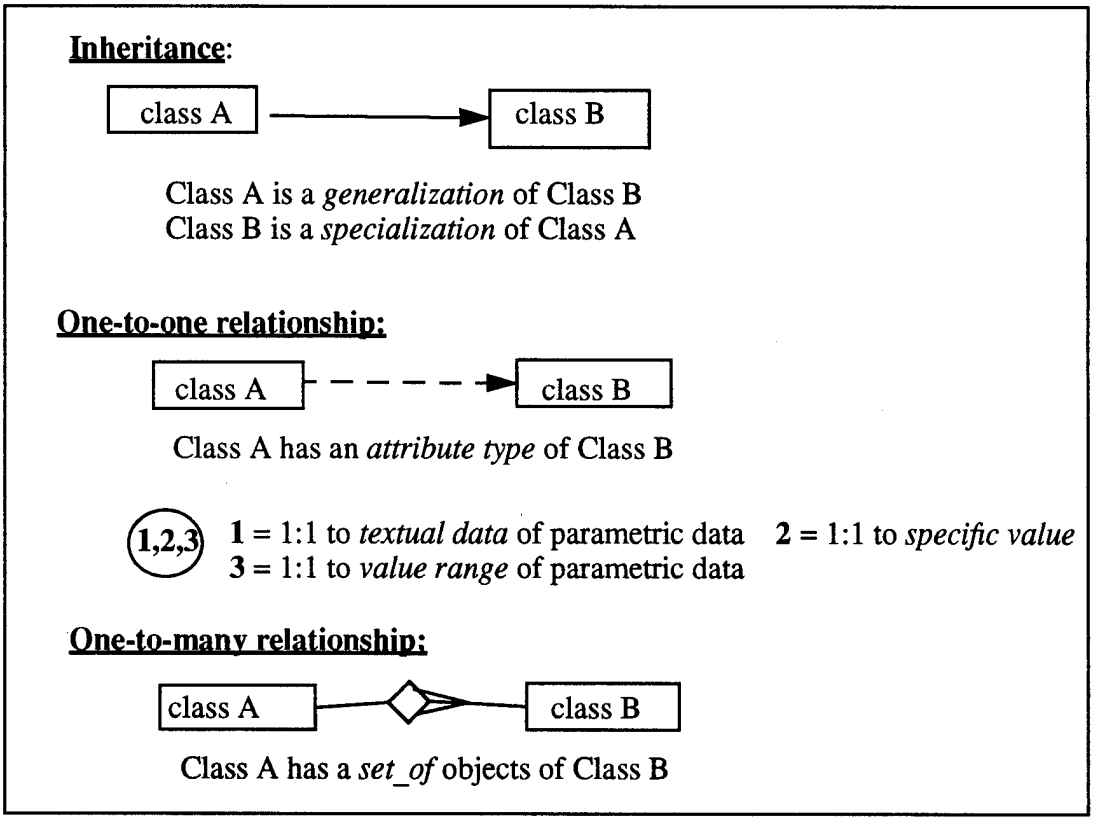


Figure 13. The Data Model Key.

1. A Top-Level View of the New EWIR Database

A top-level view of the new EWIR database shown in Figure 14. When comparing the new EWIR database with the object-oriented specification of the existing EWIR data-

base [Ref. 1], the most notable difference is the density and structuring of the data classes. Subsequent figures illustrate details of each major subdatabase of Figure 14. What is important to note in Figure 14 is that all subdivisions in the object-oriented specification of the existing EWIR database also exist in the new EWIR database. Thus, important data relationships between major subdatabases of EWIR remain intact.

One notable difference between the two data specifications is how the new EWIR database depicts 1:1 relationships between classes in *parametric_data* and classes in all of the other databases. Many of the 1:1 relationships tie parametric data to the leaf nodes of the EWIR database. This tie-in must occur to avoid having the unreachable inheritance specialization classes as discussed in Chapter II.

2. The Antenna Data

The antenna data form a fundamental component of an emitter. One emitter can have many different antenna components, and thus the emitter-antenna relationship is modeled as a 1:N. The antenna has a profound effect on signal characteristics throughout the range of EW activities. The antenna characteristics used in this thesis are but a small representation of the entire spectrum of antenna data.

In Figure 15 is the depiction of the antenna classes in the new EWIR database. The key concepts retained from the original EWIR specification [Ref. 1] are the antenna polarization, antenna radiation patterns, antenna scanning methods, and antenna tracking methods. The classes omitted from the original specification are, for the most part, sibling branches in a hierarchical tree rooted at the antenna class. These sibling branches are typically just variations of inheritance specializations. For example, the original specification [Ref. 1] for the class *Scan* has three inheritance specialization hierarchies - *Mechanical scan, Manual scan,* and *Electronic scan* (See also Appendix A). By implementing only *Mechanical scan* in the new EWIR database, an essential function of *Scan* is retained while eliminating the near-duplicate information. The ability to successfully manipulate data in the *Mechanical_Scan* inheritance hierarchy adequately validates the ability of the Object-Oriented Interface to manipulate all of the branches of *Scan* in the original EWIR specification.

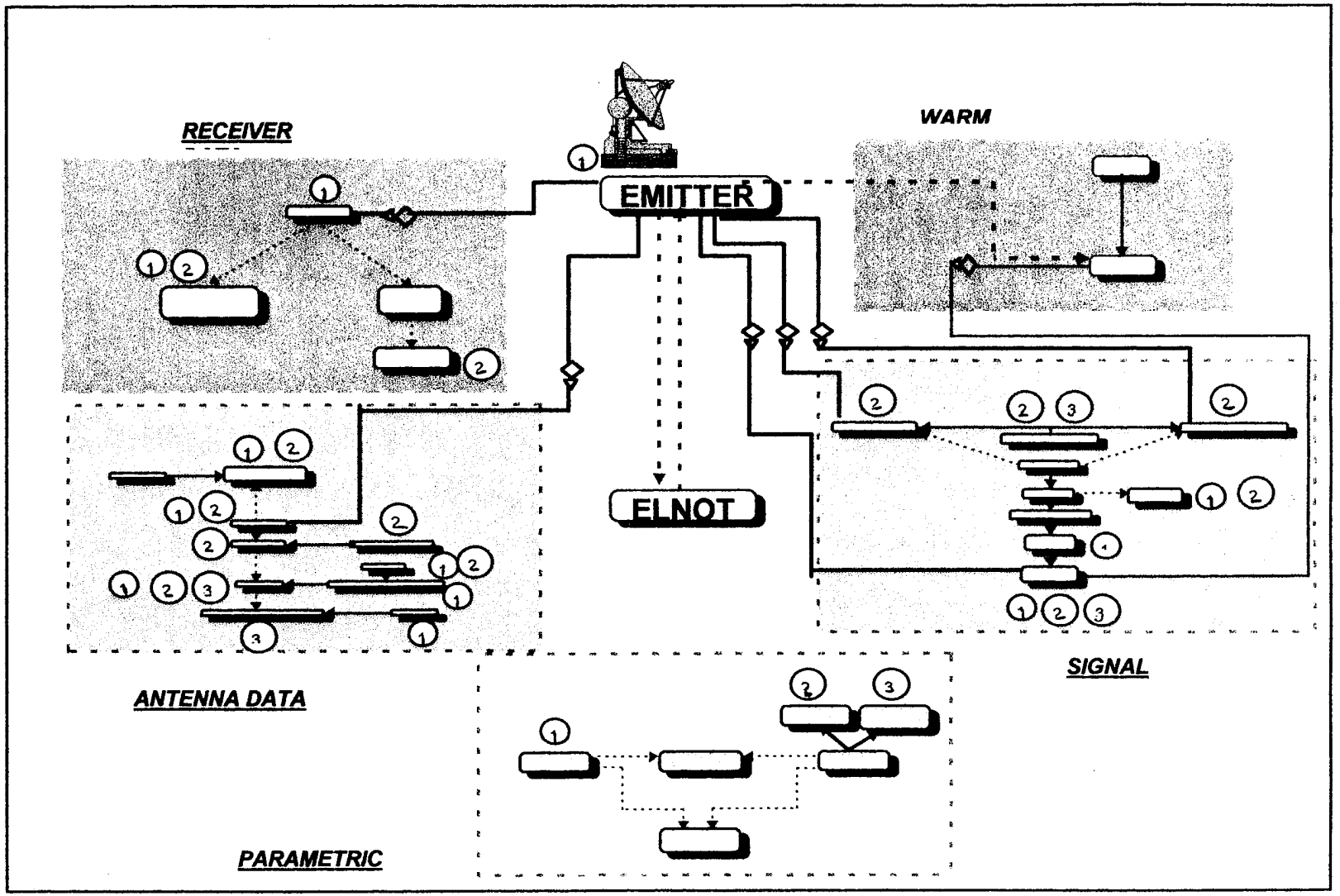


Figure 14. The Top-level View of the New Object -Oriented EWIR Database.

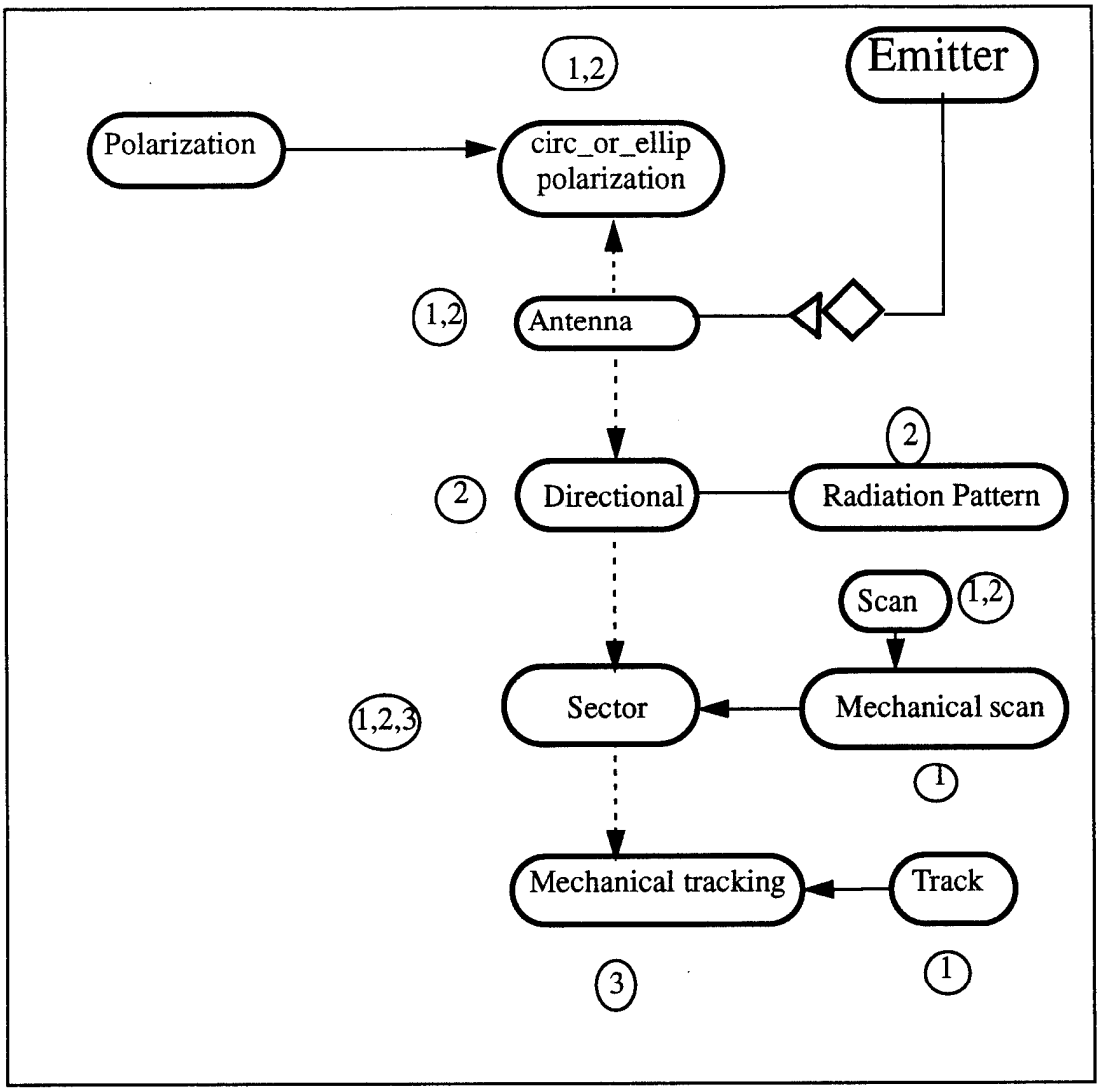


Figure 15. The Antenna Specification.

3. The EWIR Signal Data

The Signal data form the sum and substance of the EWIR database parametric data. The control of the electromagnetic spectrum requires an intimate knowledge of the characteristics of signals produced by adversary emitters. The EWIR parametric data are designed to convey the emitter signal information which forms the "fingerprint" used in

identifying emitters.

In deciding how to reduce signal data to an implementable size, we focus primarily on retaining the most useful signal information. A second priority is to retain some of the complex data relationships. The signal data of the EWIR database contains multiple-level inheritance and 1:1 relationships. By following the pulsed RF inheritance specialization of *Signal*, a four-level inheritance hierarchy is attained (See Appendix A). In Figure 16, this four-level inheritance is integrated into the new EWIR database. A 1:1 relationship between *signal* and both *constant power* and *not-constant power* is shown in Figure 16. This change from the original specification is to avoid having unreachable specialization classes.

Although most of classes of signal data in are not implemented in the new EWIR database, the essential purpose of signal data is retained. The signal RF, power, agility, and structure remain in the new database. The retained classes provide adequate structure and content to evaluate the merits of its object-oriented implementation.

4. The EWIR Receiver Data

The receiver, like the antenna, is a fundamental component of the emitter. One emitter can have many receivers, and the emitter-receiver relationship is therefor modeled as a 1:N. The parameters of a receiver reveal an emitters' capability to evaluate incoming signals. The capability is a major determinant in the emitters overall performance.

The receiver is primarily concerned with the signal processing. Thus, the signal processing classes are included in the new EWIR implementation. The classes and attributes retained are based on the most frequently used data in the receiver subsection. The receiver specification of the new EWIR database is shown in Figure 17 and can be compared with the original object-oriented specification reproduced in Appendix A.

5. The EWIR WARM Data

The meaning of WARM is the War Reserve Mode. Many emitters have special modes to be used only in a time of war. The intended purpose is to create some confusion about EW sources in order to gain an advantage over the enemy. Therefore, any prior

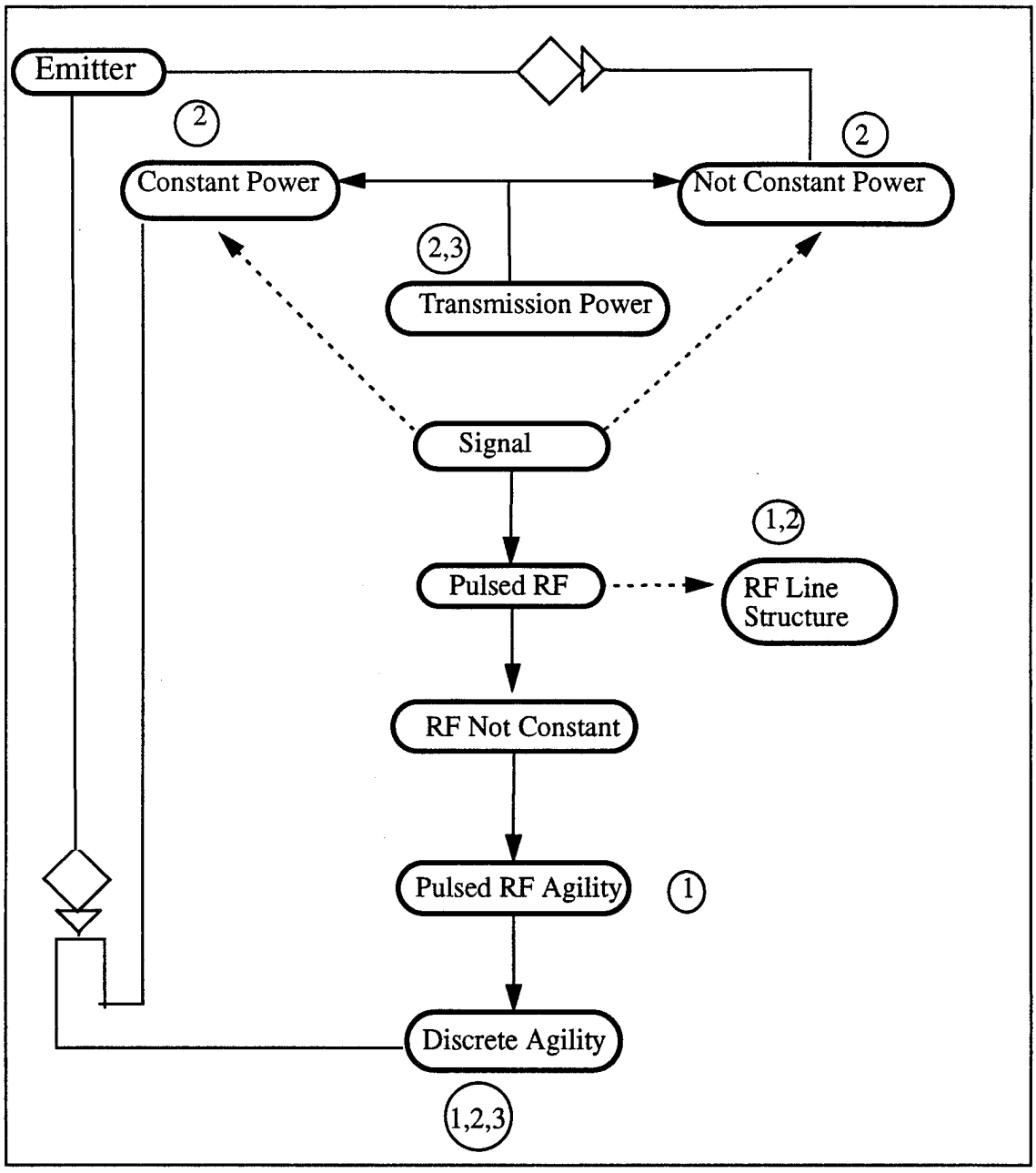


Figure 16. The Signal Data.

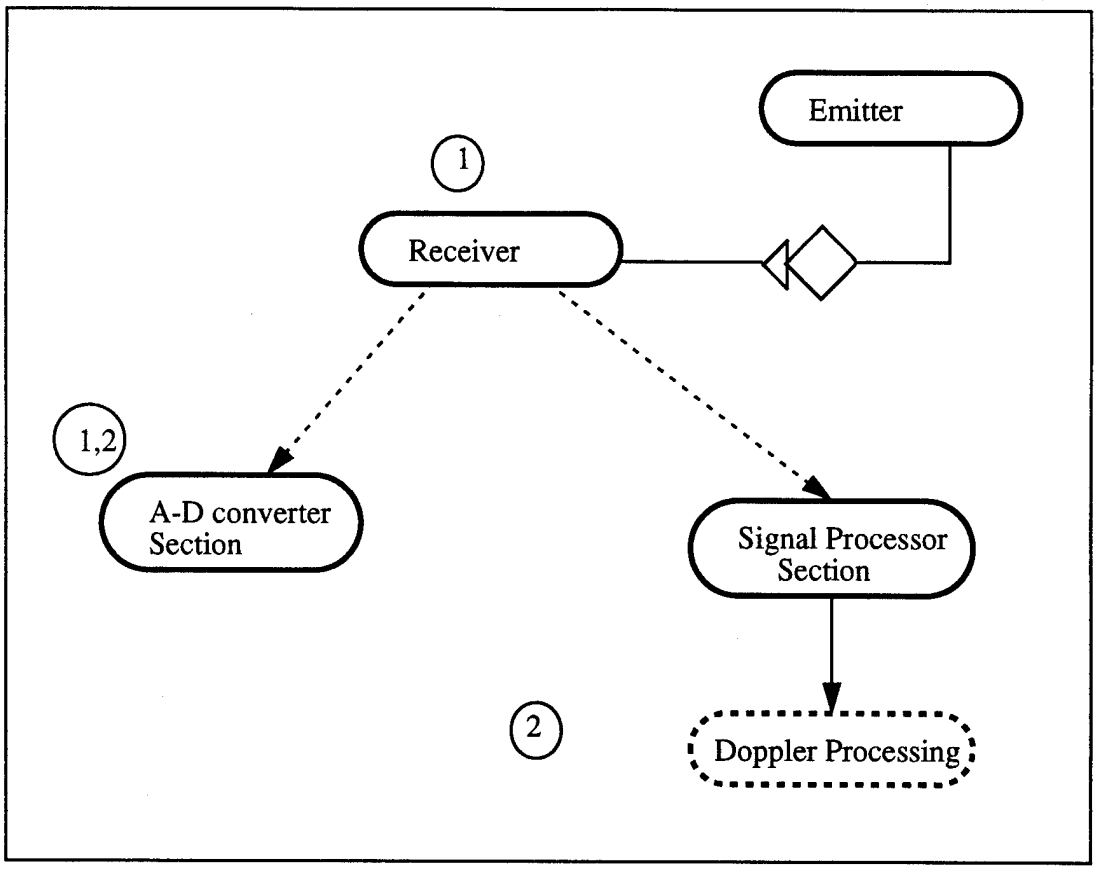


Figure 17. The Receiver Data.

an enemy's war reserve modes is of major importance. The EWIR database tracks available emitter WARM data.

The WARM data [Ref. 1], is a hub and spoke structure (See Appendix A). WARM is the hub and functions as the generalization class for all the spokes (i.e., the specialization classes). The new EWIR database retains only the RF ECCM class as one of the inheritance specializations (See Figure 18). This single specialization adequately represents the purpose and structure of WARM in the new EWIR object-oriented specification.

6. The EWIR Parametric Data

The parametric data classes in the new EWIR database are essentially repositories of logically related EW parametric data. The parametric data classes in this thesis are primarily part of 1:1 relationships with the specialization classes of the major subsections. With only the specialization-to-generalization path available in the Object-Oriented Interface, all of the classes are reachable. The parametric data specification in the new EWIR database is shown in Figure 19.

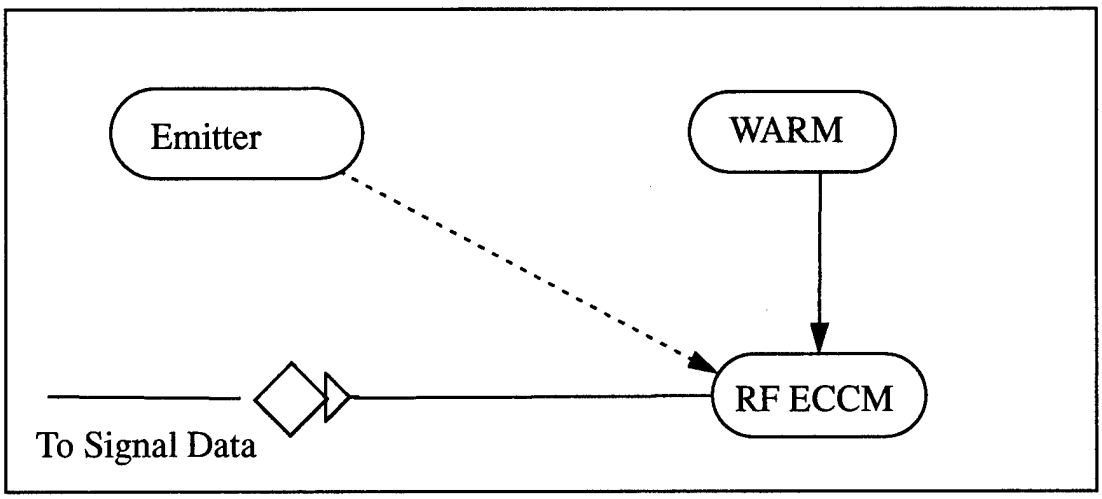


Figure 18. The WARM Data.

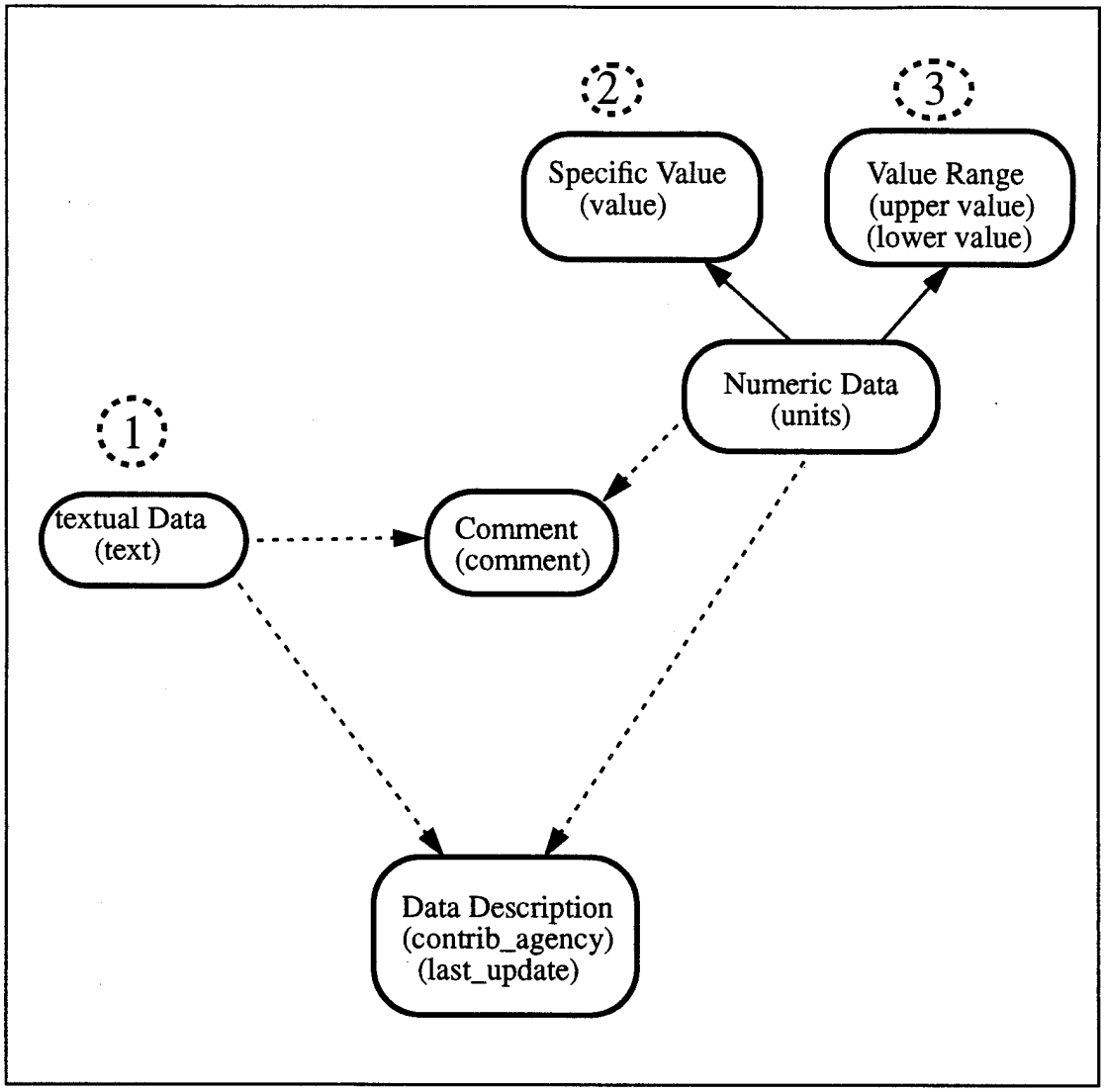


Figure 19. The Parametric Data.

IV. THE IMPLEMENTATION OF THE EWIR DATABASE

In Chapter III the object-oriented specification of the existing EWIR database is reduced to an implementable size. The next step is to translate the new EWIR specification in Chapter III into a database schema usable by the Object-Oriented Interface. The database schema is constructed using the O-ODDL as specified and referenced in Chapter II. In Section A of this chapter, the schema of the new EWIR database is presented. The files created by the Object-Oriented Interface which are derived from the schema are the *Data Dictionary*, the *Descriptor File*, and the *Template File*. These files are also described in Section A.

Once the Object-Oriented Interface has accepted the database schema and created the requisite files, it is ready to accept the record data. In Section B, the process of inserting the EWIR record data into the newly created EWIR database is discussed. This section also includes the record format used by the mass load function of the Object-Oriented Interface. With the loading of the EWIR data into the new EWIR database, the process of creating a live EWIR database is complete. The database is now ready to execute transactions.

A. THE DATA DEFINITION OF THE EWIR DATABASE

Using the new EWIR database specification developed in Chapter III, the object classes to be implemented are easily identified. The Object-Oriented Interface requires strict formatting for the schema definition. In the object-oriented specification of the existing EWIR database [Ref. 1], no formatting restrictions were applied to class and attribute specifications because the specification was not tied specifically to the Object-Oriented Interface. In conforming to the Object-Oriented Interface standards, the following restrictions are applied when translating classes and attributes from the object-oriented specification to the database schema:

- Class names are limited to seven characters.
- Attribute names are limited to fifteen characters.

- No class names or attribute names can be identical.
- All inherited classes must be specified before the class that inherits.
- The second character of attribute name is not an underscore. This restriction is evident only when mass loading the record file onto the backend system. The DDL compiler of the Object-Oriented Interface will allow the underscore. The source or reason for this restriction during the mass load is unknown.

The restriction of class names to seven characters creates two problems. The first problem is not being able to have semantically meaningful class names. In a large, complex database such as EWIR, a seven-character limit is inadequate for users to easily associate class names with their purpose. A second problem is correlating a class name in the new EWIR database schema with the class name used in the EWIR specification in Chapter III. To facilitate translation, Figure 20 is a listing of identical classes in the EWIR schema and the EWIR specification.

Specification	Schema	Specification	Schema
Elnot	*Elnot*	Data_admin_info	*Data_ds*
Comment	*Comment*	Numeric_data	*Num_dat*
Specific_value	*Spec_va*	Value_range	*Val_ran*
Transmission_power	*Trans_p*	Constant_power	*Const_p*
Not_constant_power	*N_con_p*	Signal	*Signal*
Textual_data	*Text_da*	Rf_line_structure	*Rf_l_st*
Pulsed_rf	*Pulsed*	Rf_not_constant	*Rf_n_co*
Pulsed_rf_agility	*Pul_rf*	Discrete_agility	*Disc_ag*
Warm	*Warm*	Rf_eccm	*Rf_eccm*
Polarization	*Polariz*	Circ_or_ellip_polariz	*C_el_po*
Scan	*Scan*	Mechanical_scan	*Mech_sc*
Track	*Track*	Mechanical_tracking	*Mech_tr*
Sector	*Sector*	Radiation_pattern	*Rad_pat*
Directional	*Directi*	Antenna	*Antenna*
Doppler_processing	*Doper_p*	Signal_processing	*Sig_pce*
A_d_converter	*A_d_con*	Receive	*Receiver*
Emitter	*Emitter*		

Figure 20. The Conversions of EWIR Specifications to Schema Class Names.

To this point, the only class *attributes* depicted are those which are themselves classes connected via a 1:1 relationship. For example, an attribute of *Antenna* is *ant_direc* (of type *Directi*) which is another class. In the EWIR schema, a listing of *all* attributes of each class is included. The attributes in the new EWIR schema do not include all of the class attributes delineated in the object-oriented specification of the existing EWIR database [Ref. 1]. The attributes of the new EWIR schema are those which best capture the function of the class. This allows the new EWIR database to retain the essential functionality and characteristics of the existing EWIR yet be of implementable size for the Object-Oriented Interface.

1. The Schema Listing

In Figure 21(a) and 21(b) is a listing of the new EWIR schema file (filename: EWIROODL). In Chapter II, Figures 3 through 6 explain the DDL constructs used in the class specifications on the Object-Oriented Interface:

2. The Data Dictionary

The Object-Oriented Interface accepts the new EWIR schema and creates a data dictionary (filename: EWIROODB.dict). A data dictionary describes the database structure, information, and physical database design such as storage structures, access paths and record sizes. The data dictionary in the Object-Oriented Interface is referenced by the query constructor when performing data manipulation functions.

The data dictionary generated by the Object-Oriented Interface for the EWIR database is listed in Appendix B. In Figure 22, a description of the data dictionary constructs used in the Object-Oriented Interface is given. Detailed information on the Object-Oriented Interface Data Dictionary can be found in Ref. 5.

3. The Template File

The Object-Oriented Interface also creates a template file (filename: EWIROODB.t). The purpose of the template file is to provide the specification of the

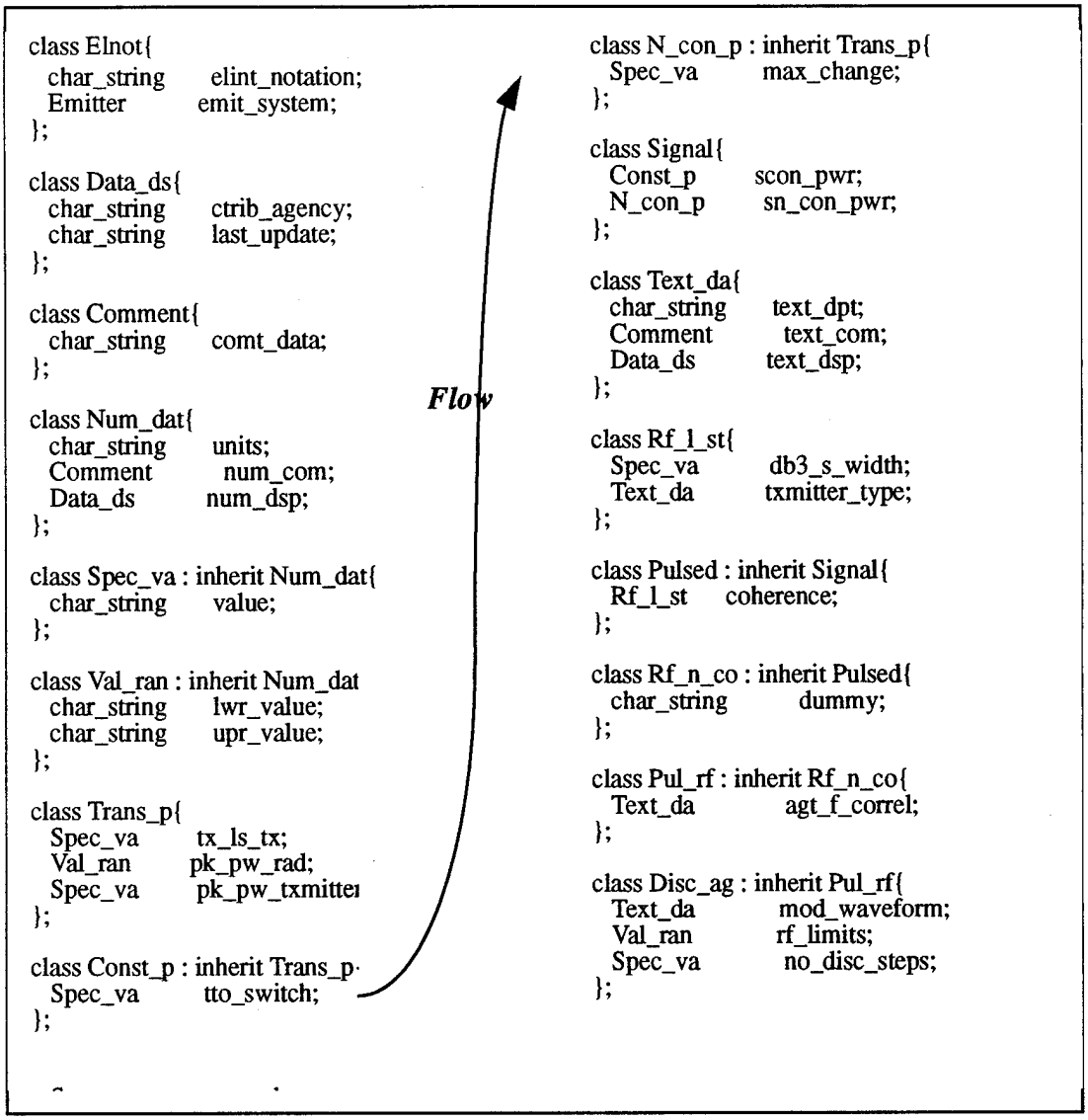


Figure 21(a). The EWIR Schema Specification.

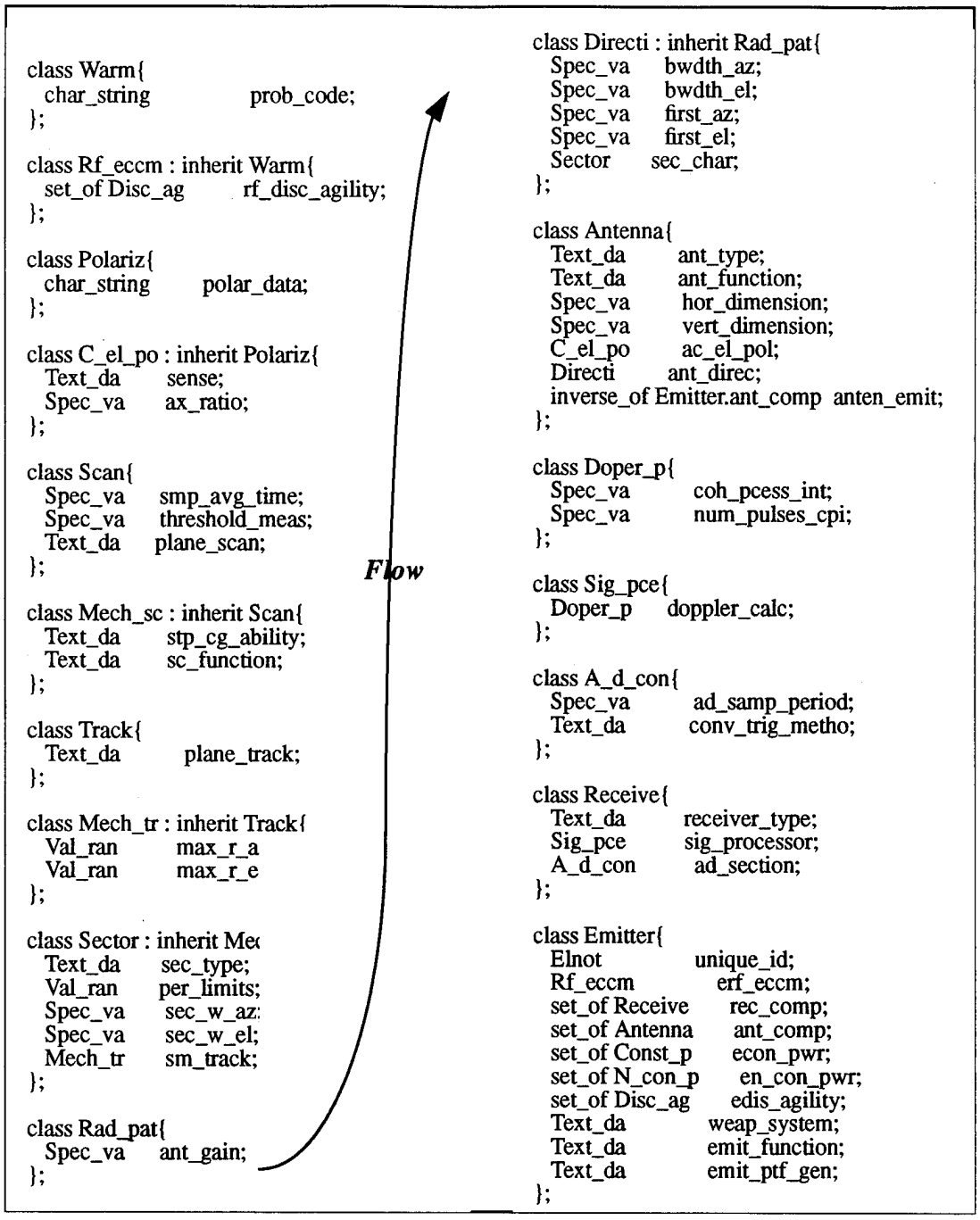


Figure 21(b). The EWIR Schema Specification.

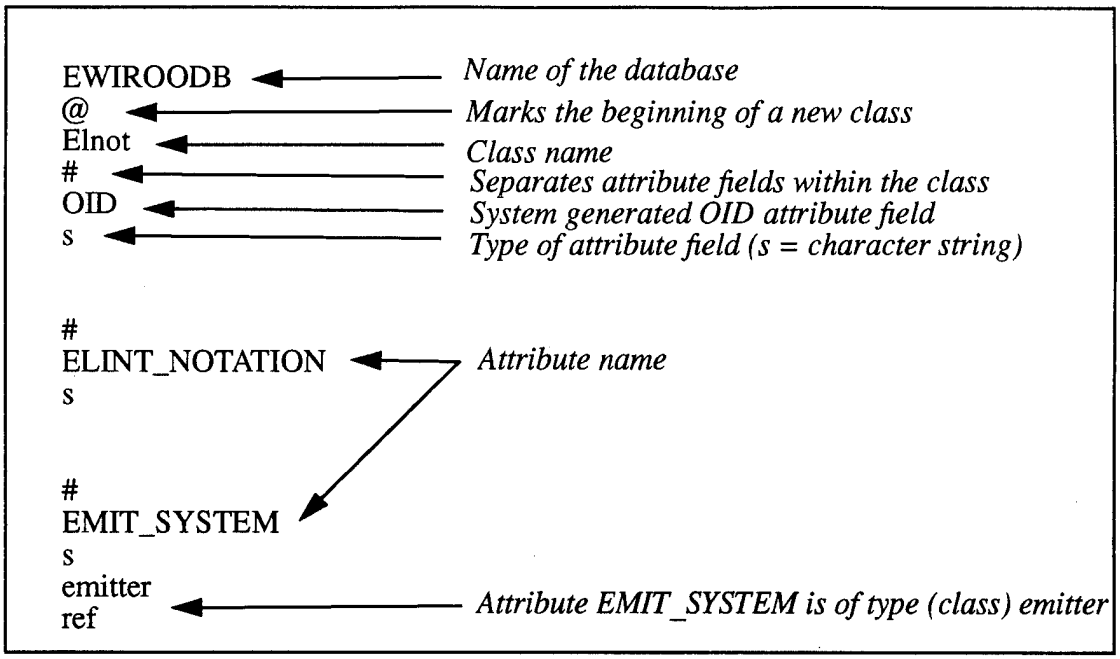


Figure 22. The Data Dictionary Format.

object-oriented EWIR database in an equivalent kernel database. The template creates the attribute-value pair used by the kernel system. In Appendix C, a complete listing of the new EWIR database template file is provided. The format of the template file used in the Object-Oriented Interface is provided in Figure 23. For more information on the Object-Oriented Interface template file, see Ref. 6.

4. The Descriptor File

The Object-Oriented Interface creates an EWIR descriptor file (filename: EWIROODB.d). The descriptor file is used by the Language Interface Controller for communication with the kernel system. The descriptor file provides the kernel system a listing of all database classes including those not specified by the user. For example, the *set_of* relationship in the database schema results in a new class generated by the Object-Oriented Interface. Specifically, the class *Emitter* has an attribute of type *set_of receivers*. In Figure 24, the new system generated class *Receiver_emitter* is listed. *Receiver_emitter*

was not a user-defined class in the EWIR schema. The entire EWIR descriptor file is listed in Figure 24. See Ref. 5 for more information on the Object-Oriented Interface descriptor file.

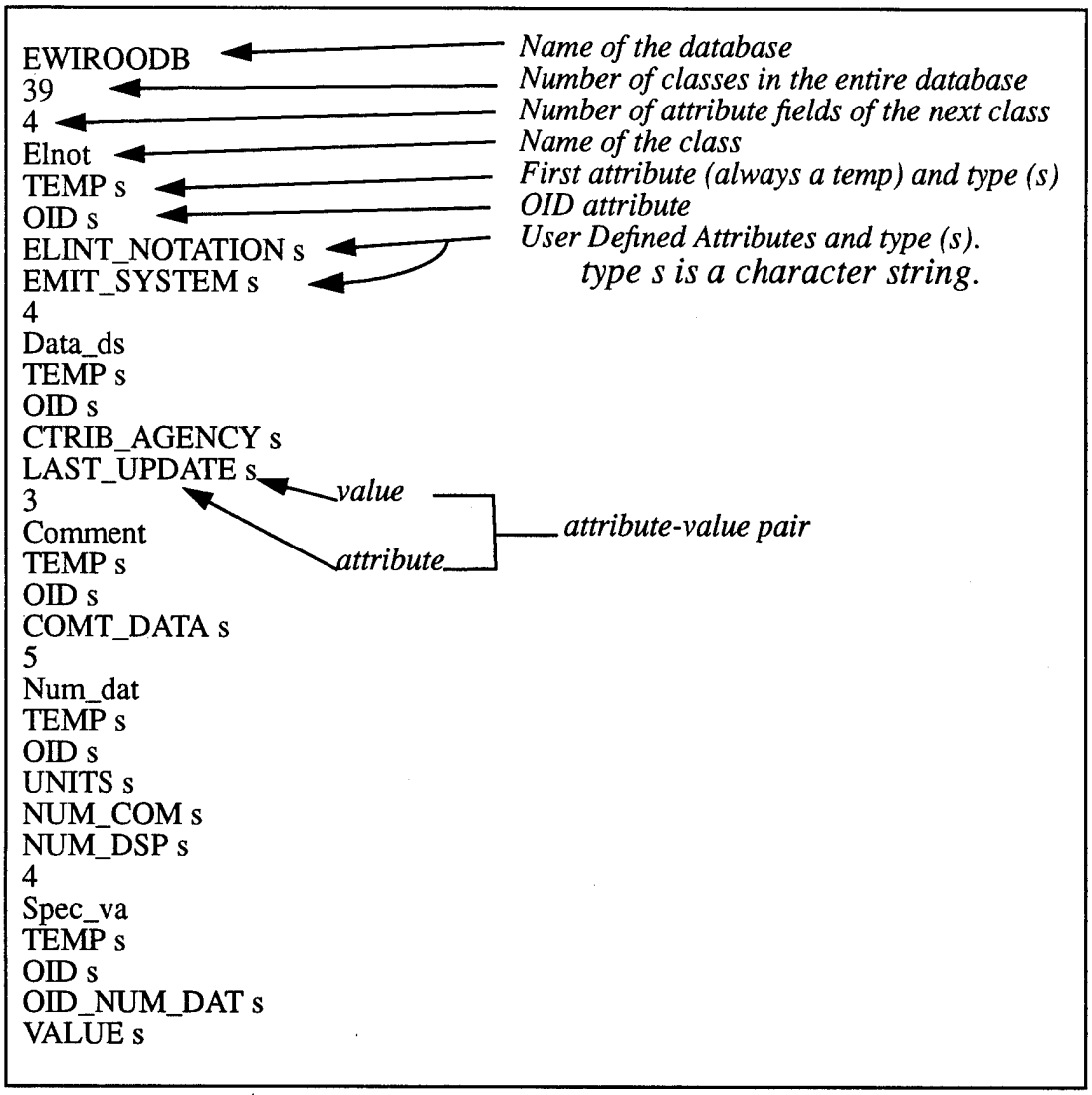


Figure 23. The EWIR Template File Format.

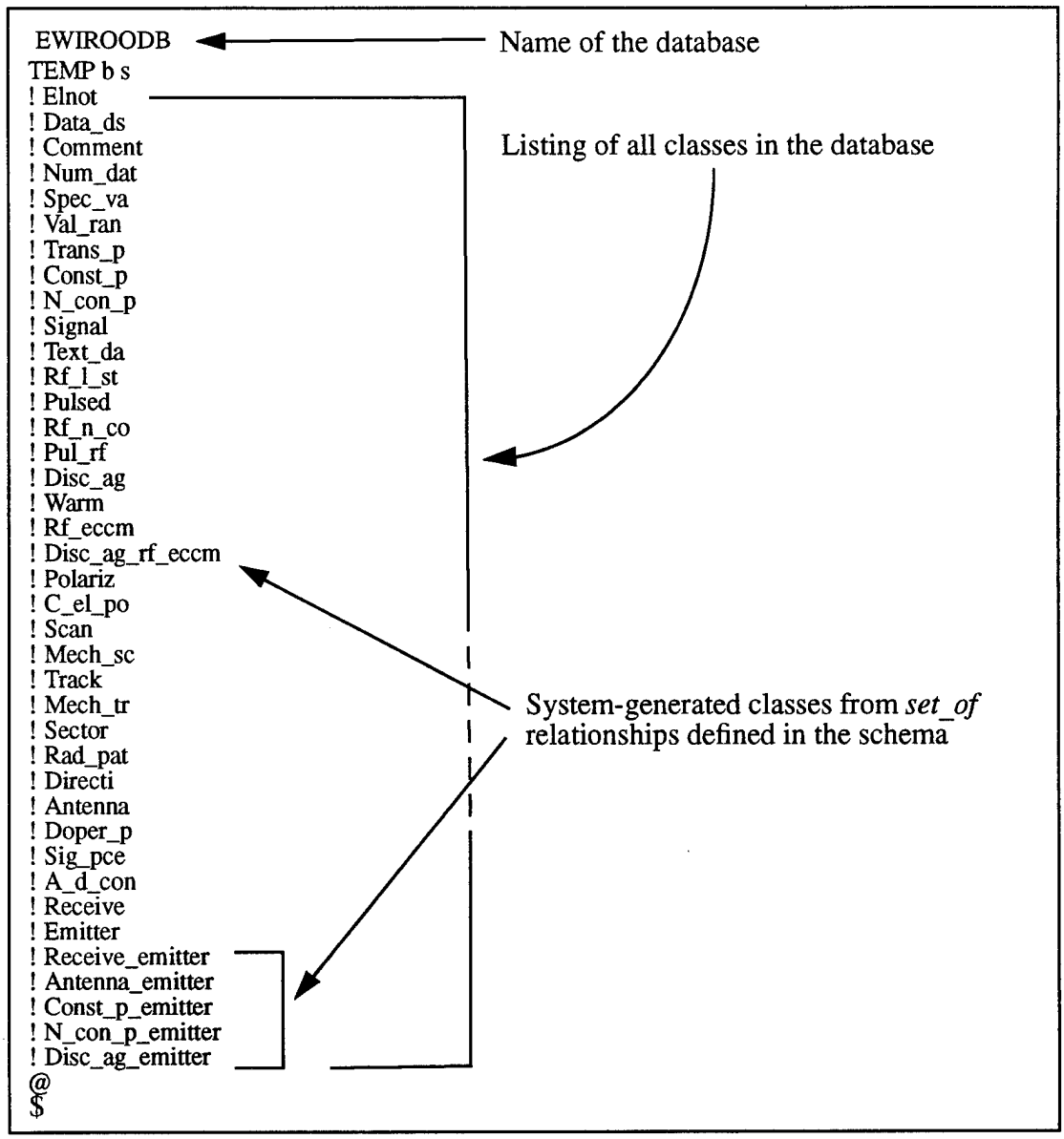


Figure 24. The EWIR Descriptor File.

B. THE EWIR DATABASE RECORD DATA

The *Insert* function of the Object-Oriented Interface has not been implemented, and thus a record-by-record insertion of data is not available. Instead, the *mass load* function is used to load all record data. The EWIR record file (filename: EWIROODB.r) is created by the user to mass load all of the EWIR record data into the EWIR database. When using the mass load option, the user must adhere to the following conventions when specifying the record data:

- Classes occur in the same order as the schema file (EWIROODL).
- Attributes are in the same order as listed in the schema.
- The @ symbol is used to separate classes.
- The user generates the OID's.
- If a class inherits from another class, then the OID format of the generalization class is used by the specialization class followed by "na". Thus a specialization class record would have the following format:

  < OID of generalization> na attribute1 attribute2.....attributeN

- An illustration of how to translate the schema specification into the record data format is illustrated in Figure 25:

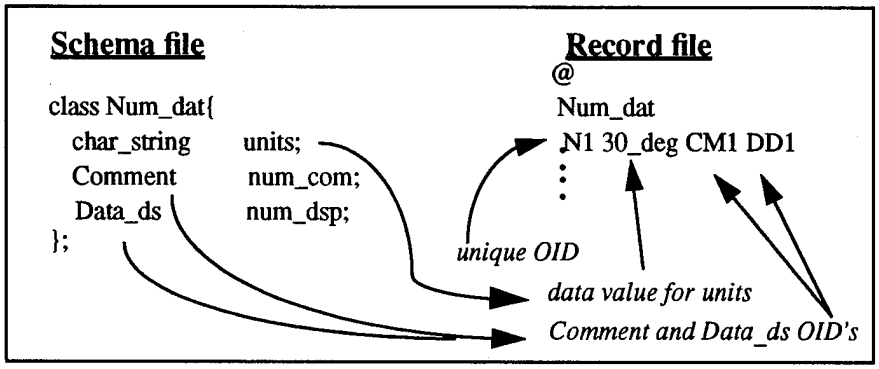


Figure 25. The Record File Format.

1. A Proposed EWIR Record Template

When entering data into a database, the database user typically has an *interface* designed to collect related record information. The database interface can be of many different types. User-friendly interfaces may include a menu-based interface, a graphical interface, or a forms-based interface. A logical choice for inserting data into the EWIR database is a forms-based interface. A forms-based interface allows the user to insert data into an *Emitter* template form which collects logically related information and then inserts the data into the database.

A sample forms-based interface for the new EWIR database is provided in Figure 26. The user-provided EWIR data is in italics. This sample database insertion form is very large and packed with information about a single emitter. The nature of the EWIR database is such that large amounts of information is logically related to a single emitter. Because the mass load function is the only available method for insertion of data, this sample template is not actually used in this implementation. It is presented in this thesis to illustrate the complexity of a single object "emitter". The template also provides a logical collection of emitter data for testing the accuracy of transactions on the sample emitter. When the Object-Oriented Interface implementation is expanded to include the insert function, a well designed and user-friendly interface can be developed.

2. The Mass Load Record File (EWIROODB.r)

In a commercial database system, the user will never be concerned with the physical storage structures of the database. The insert function will handle the data storage details. For the purposes of this thesis, it is important to show the EWIR record data used by the mass load function because it must be created by the user. The EWIR mass load record is listed in Figure 27(a) and 27(b). The logically related record of Figure 26 is integrated into the mass load record file as well as generic test data. *Please note that all record data names (i.e.,names of radars) and values (parameters) are not actual real-world EWIR database names and values*. Creating and testing the EWIR database on the Object-Oriented Interface is an unclassified research project and thus precludes the use of real

EWIR data. The mass loading of the EWIROODB file completes the creation of the new EWIR database.

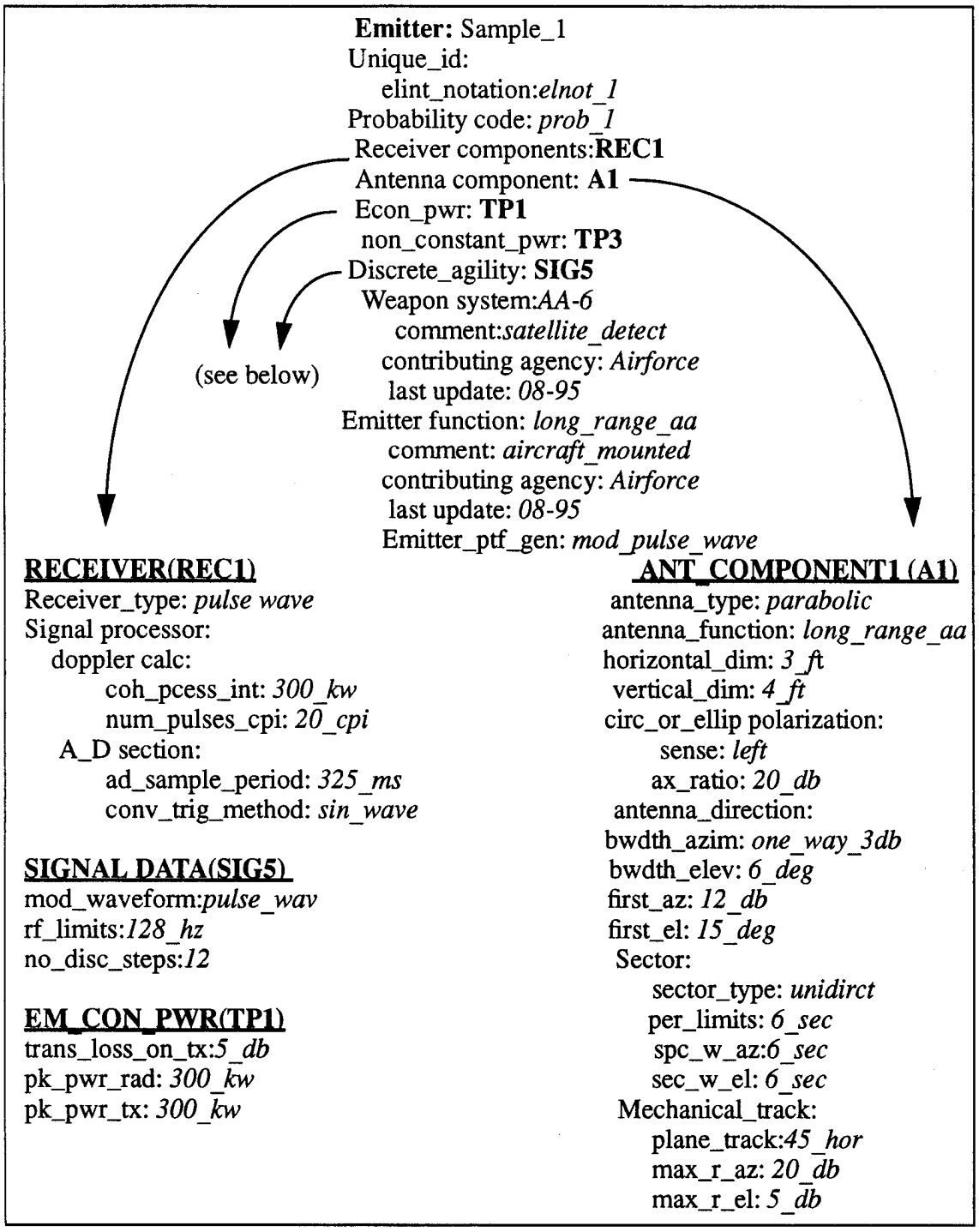


Figure 26. A Sample EWIR Record Template.

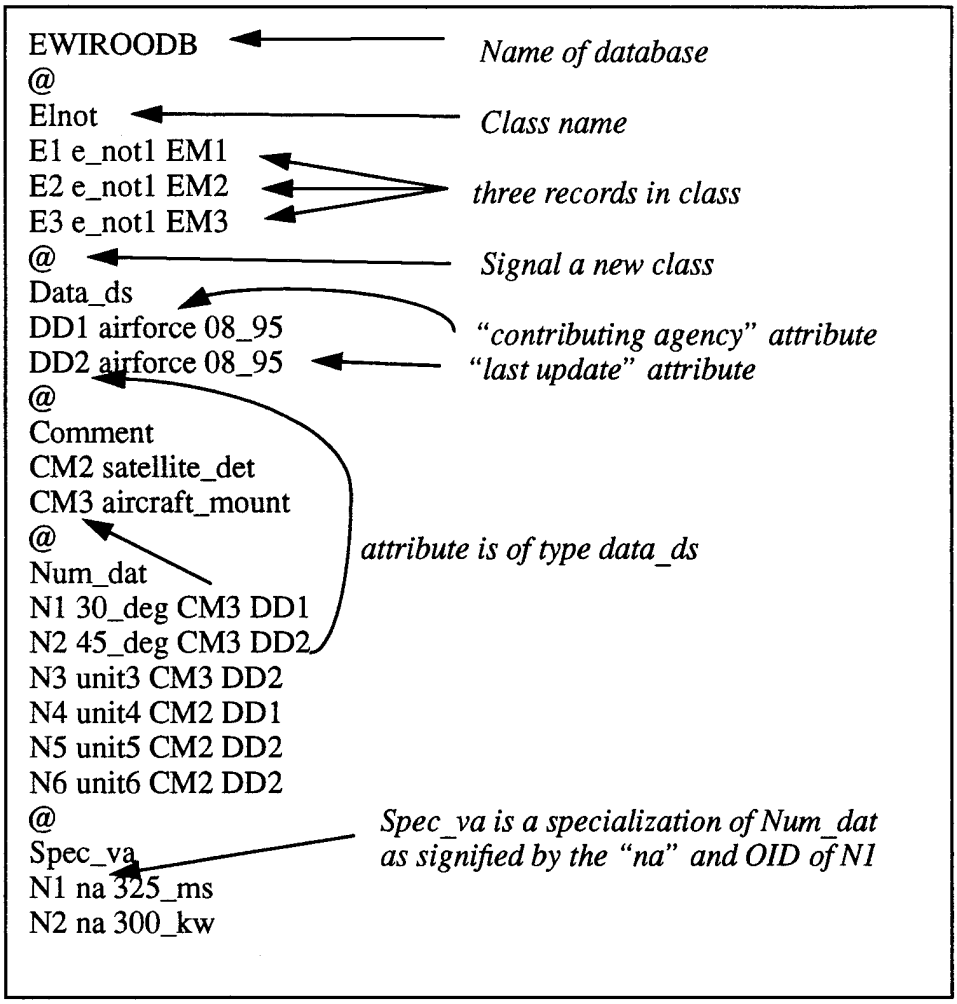


Figure 27(a). The EWIR Record File.

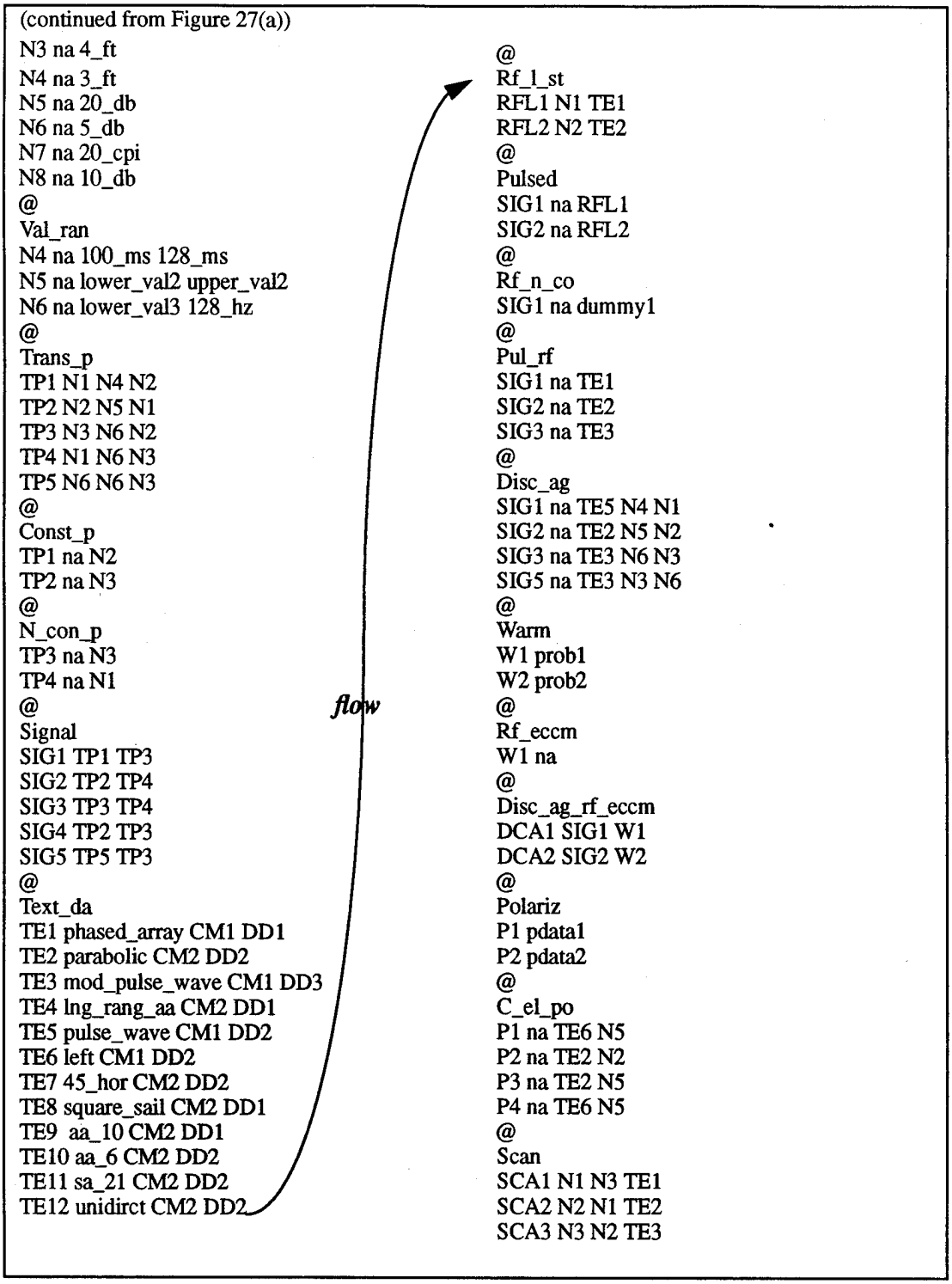


Figure 27 (b). The EWIR Record Data.

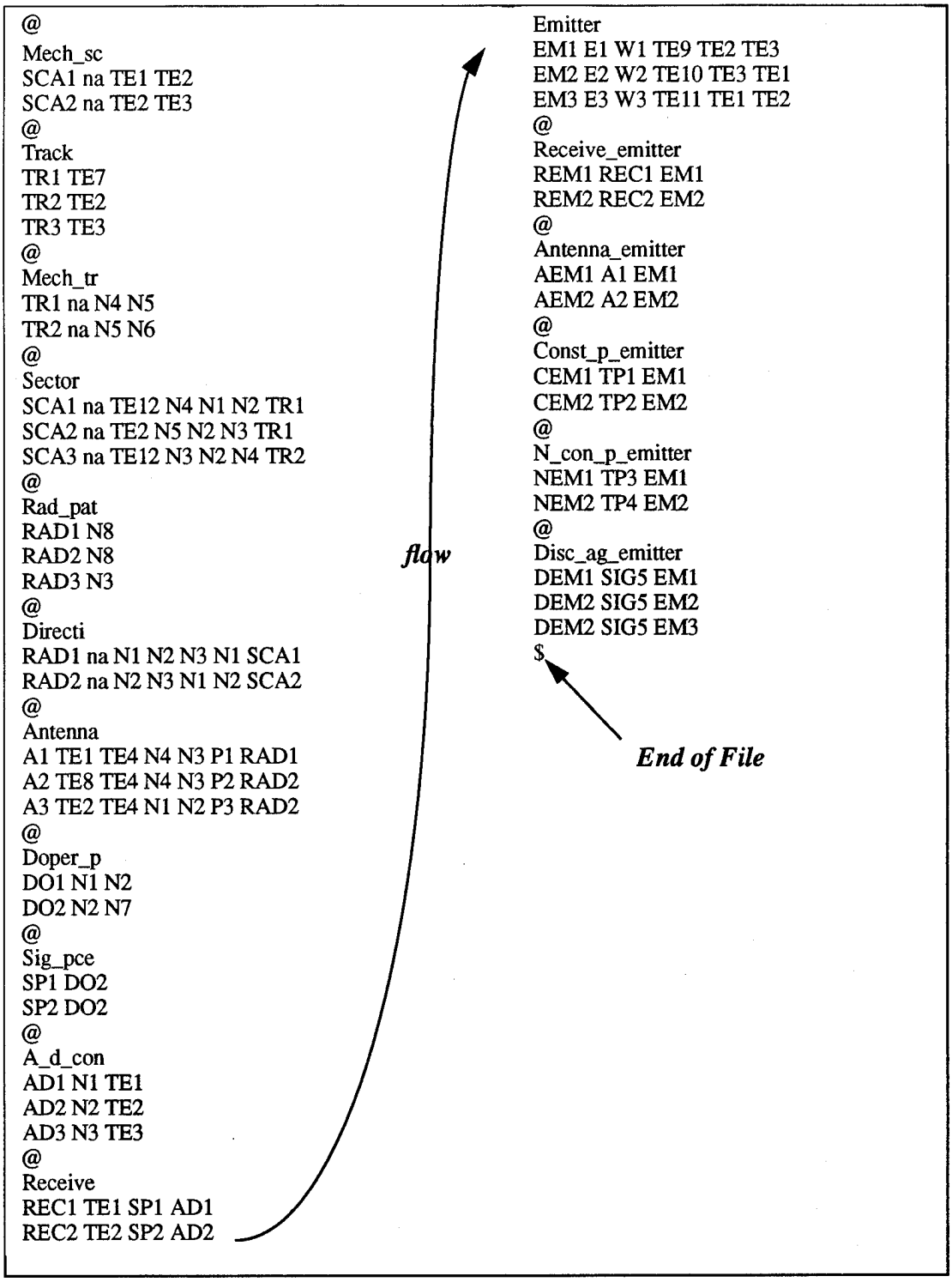


Figure 27(c). The EWIR Record Data.

V. THE MANIPULATION OF EWIR DATA

In Chapter III of this thesis, we derived the new EWIR database from the object-oriented specification of the existing EWIR database. In Chapter IV, the new EWIR database was transformed from an O-ODM specification into a live database using the DDL of the Object-Oriented Interface. In Section A of this chapter, we state the objectives and goals achieved through the execution of transactions on the live EWIR database. In Section B, we present the nine transactions on the live EWIR database using the DML of the Object-Oriented Interface.

A. THE GOALS OF THE EWIR TRANSACTIONS

The transactions or queries on the EWIR database are the final step required to reach our thesis goals and objectives. They provide a basis for evaluating the effectiveness of the Object-Oriented Interface as well as the merits of an object-oriented implementation of the EWIR database. The transactions or queries utilize and test the implemented functions of the Object-Oriented Interface while capturing the EWIR database functionality. In particular, the transactions are intended for the following:

- To test every type of relationship in the database. These relationships include inheritance, 1:1, 1:N and M:N.
- To span the most complex data relationships encountered in the live EWIR database. These include the multiple-level inheritance and multiple-level 1:1 relationships.
- To utilize all available DML operations currently implemented on the Object-Oriented Interface. These operations include *Obj_set*, *Object_refs*, *Find_one*, *Find_many*, *Display*, *Contains*, *And*, *Or*, and the looping structure (*For Each <obj_ref> In <obj_set>*).
- To access and manipulate data from disparate sections of the database. In other words, the information from the antenna, signal, and receiver sections can be accessed and manipulated in a single query.

B. THE EWIR TRANSACTIONS

The queries presented in this section are representative of *the type* of ad-hoc trans-

actions an analyst might submit to the EWIR database. These queries are designed for *testing* the database and do not reflect any strategic relevance. It is also important to note once again that the data values and names used in the new EWIR database *are not* the actual values and names of "real world" parametric data, thereby keeping the research project unclassified.

Nine queries are presented in this thesis. A standard format is used for presenting all the nine queries. The query presentation format is explained in Figure 28. The compiler output is useful for debugging the queries and for following the sequence of execution of each query [Ref. 6].

#. < Subsection Title >

Purpose: The research purpose of the query.

Request: The transaction to be performed stated in English.

Query: The query in the DML of the Object-Oriented Interface.

Result: The values returned by the query.

Remarks: Any remark or figure as required.

Figure 28. The Query Presentation Template.

The limitations experienced in writing the queries are mainly a function of the limitations of the Object-Oriented Interface. The specialization-to-generalization path for an inheritance limits the scope of retrievable information. The lack of an *if-then-else* statement limits the complexity and sophistication of the queries. Given the functions currently available, the following nine queries evaluate the current data manipulation capabilities of the Object-Oriented Interface. These queries also provide enough context to evaluate the effectiveness of an object-oriented approach to manipulation of EWIR data.

1. An Attribute Look-up (Query_1)

Purpose: The primary purpose of Query_1 is to explain the mechanics of how the Object-Oriented Interface processes a query using only the schema and record files. An illustration of the Query_1 execution logic is provided in Figure 29. Query_1 is a simple attribute look-up involving a 1:1 relationship.

Request: Find an antenna with the long-range, anti-air capability.

Query:

```
Query Find_antenna IS
   obj_ref p;
Begin
   p:= find_one Antenna where ant_function.text_dpt = 'lng_rang_AA';
   display(p.ant_type.text_dpt);
End;
```

Result: Text_dpt: *phased_array*

Remarks: When hand-tracing the execution of an EWIR query, we use the schema file and the record file. Refer to Figure 29 when reading the following explanation.

We start with the assignment statement in the body of Query_1. It assigns to the variable p the OID of the *Antenna* whose function is long-range-anti-air (lng_rang_aa). Thus, by first looking at the schema, the *ant_function* attribute is observed to be of type *Text_da*. therefore, we now locate the class *Text_da* and, following the dot notation, locate the attribute *text_dpt*. The *text_dpt* attribute is the first attribute of class *Text_da*. Using this information, we now move to the record file and locate the class *Text_da*. The inspection of class *Text_da* records terminate when we locate the value "lng_rang_aa" in its first attribute field. A match is made with the record having an OID of TE4.

The OID value of TE4 is required as a second attribute in a record of class *Antenna*. Thus, the *Antenna* records in the record file are searched until TE4 is found in the second attribute field. The match is made with the Antenna having an OID value of A1. The Antenna OID value of A1 is now assigned to the variable p.

The display statement now uses the antenna with the OID of A1 to output the

requested information. Note that the display statement in this query also involves dot notation:

display (p.ant_type.text_dpt);

The *p.ant_type* is the first attribute of Antenna. For the antenna with an OID of A1, the *ant_type* attribute is TE1. The record with the OID of TE1 is located. The *text_dpt* attribute is the first attribute field in the record. The *text_dpt* attribute of TE1 has the value *phased_array.* Thus, *phased_array* is the output of the query.

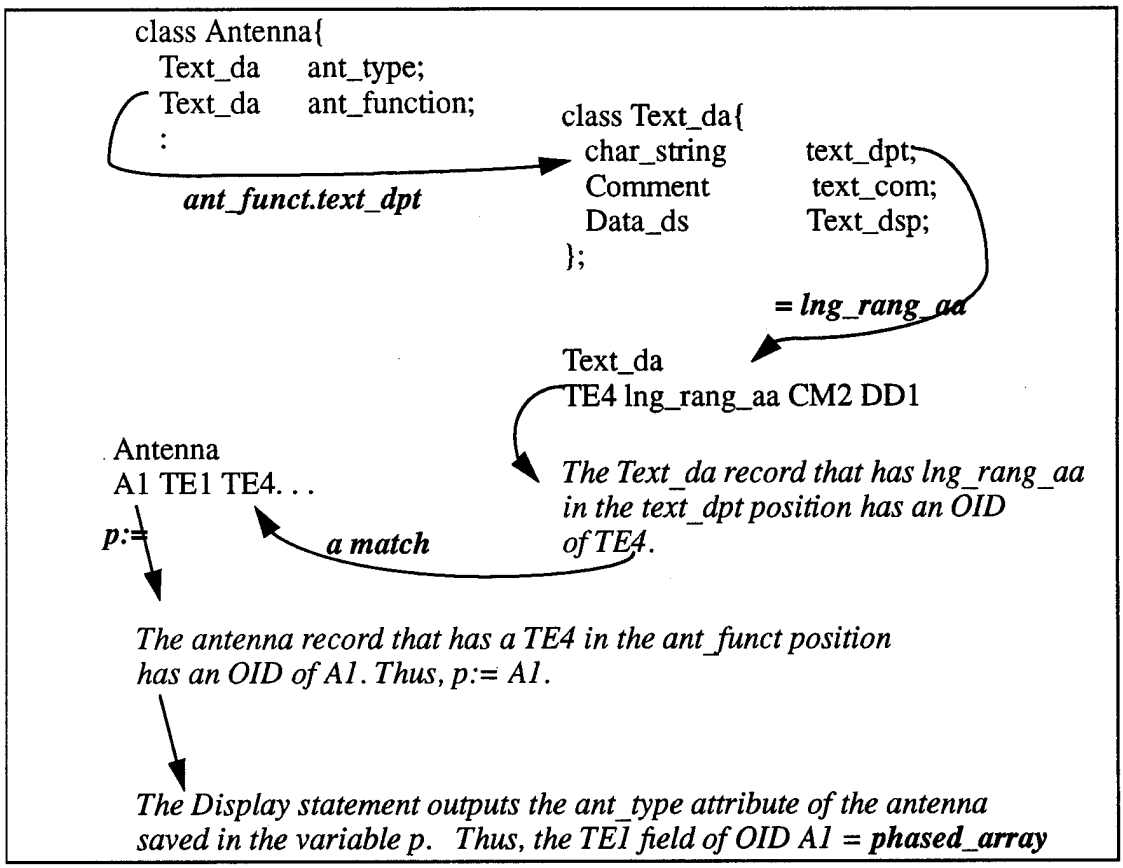


Figure 29. The Query_1 Execution Flow.

2. A Single-level Inheritance and its 1:1 Relationship (Query_2)

Purpose: The purpose of Query_2 is to complete a one-level 1:1, one-level inheritance transaction. Also, new to this query are multiple outputs in a single display statement. The display statement is recoded to handle multiple outputs in a single statement. The previous

version allowed only one output per display statement. The display statement in Query_2 uses the three-level dot notation for its output. A graphic representation of Query_2 is presented in Figure 30.

Request: Find the antenna type, antenna function, contributing agency and last update of an antenna with cross polarization.

Query:

```
Query Cross_polorization IS
  obj_ref p;
 Begin
  p:= find_one antenna where ac_el_pol.polar_data = 'cross_polarize';
  Display (p.ant_type.text_dpt, p.ant_function.text_dpt,
       p.ant_type.text_dsp.ctrib_agency, p.ant_type.text_dsp.last_update)
 End;
```

Result: ant_type: *phased array*
ant_function: *lng_rang_aa*
ctrib_agency: *airforce*
last_update: *08 - 95*

Remarks: The inheritance hierarchy makes the attributes of the inherited classes to appear as if they are part of the inheriting class. This negates the requirement for path statements using dot notation to access multi-level inherited data. It greatly simplifies the query while providing powerful data manipulation functionality.

In Figure 30 is a depiction of Query_2. Notice the depiction of polar_data as an attribute of Circ_or_Ellip_Polarization by virtue of an inheritance. All of the inherited class attributes become "virtual" attributes of the class which inherits.

3. Multi-Level 1:1 Relationships (Query 3)

Purpose: The most complex relationship in the *Antenna Data* section involves four levels of 1:1 relationships plus a one-level inheritance. This query requests information that follows the four-level path statement plus a single-level inheritance. Also presented in query_3 is the logical operator *Or.* It is used for an alternative decision path involving a three-level 1:1. Query_3 also introduces the *find_many* statement resulting in multiple

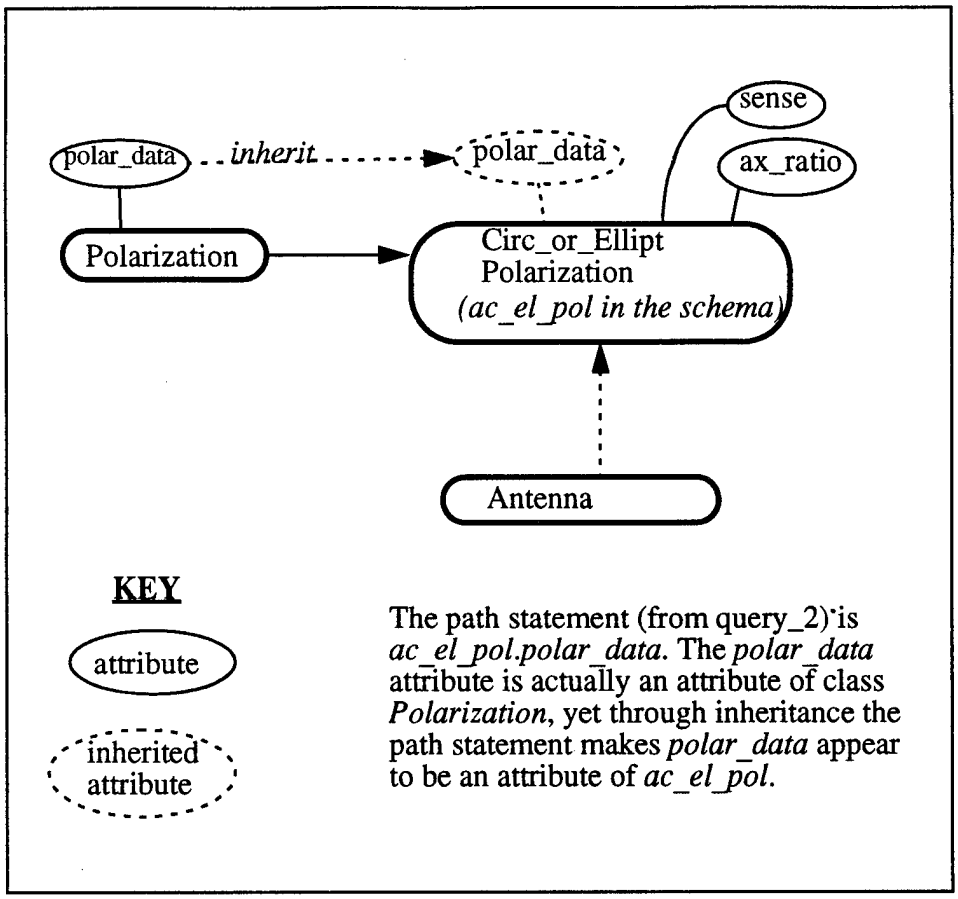


Figure 30. The Logic of Query_2.

object retrievals in a single statement. The output of these multiple objects is achieved using the looping structure *For each <var> in <var>*.

Request: Find the antennas with a tracking plane of 45 degrees horizontal OR a sector scan that is unidirectional.

Query:

```
Query track_plane IS
    obj_ref i;
    obj_set p;
Begin
    p:= find_many antenna where
```

```
    ant_direc.sec_char.sm_track.plane_track.text_dpt = '45_hor'
OR ant_direc.sec_char.sec_type.text_dpt = 'unidirct';

For each i IN p
 display(i.ant_type.text_dpt);
End_loop;

End;
```

Result: Text_dpt: *phased_array*
parabolic
square_sail

Remarks: A graphical depiction of the query is presented in Figure 31. Unlike the inheritance hierarchy, the multiple-level 1:1 structure must be explicitly expressed via a path statement using the dot notation. The statement "p:= find_many antenna" indicates that the class Antenna is the starting point in the path. The statement "where ant_direc.sec_char.sm_track.plane_track.text_dpt" is a listing of the attributes to follow to find the data value '45_hor'.

The 1:1 relationship is useful because it allows a class (or object) to be composed of other classes. This enables high levels of abstraction for the user. For example, a car is a collection of objects (engine, body, seats, etc.). These objects are in turn composed of other objects (engine => block, cylinders, fuel system, etc.). The division of a car into smaller subcomponents continues until it reaches the indivisible (atomic) parts. Thus, it is easier to understand a car as a small collection of major subcomponents (engine, body, etc.) than a huge collection of thousands of small parts (plugs, bolts, fabric, rubber, etc.). An atomic-level component can be identified by successively identifying smaller and smaller subcomponents along a logical path until it reaches the desired atomic-level component.

An example of a path statement using the car example is the following: To find the car with a spark plug gap of 0.05 inches. Starting at the Class *Car*, the example can then traverse the following path:

Find_one Car *where* engine.elect_system.ignition_system.spark_plug.gap = '.05'

This path follows a series of specialization attribute classes until it reaches the target class and the attribute of interest. This logic is employed by the Object-Oriented Interface when

executing multi-level 1:1 relationships. A depiction of the multi-level 1:1 relationship of query_3 is illustrated in Figure 31.

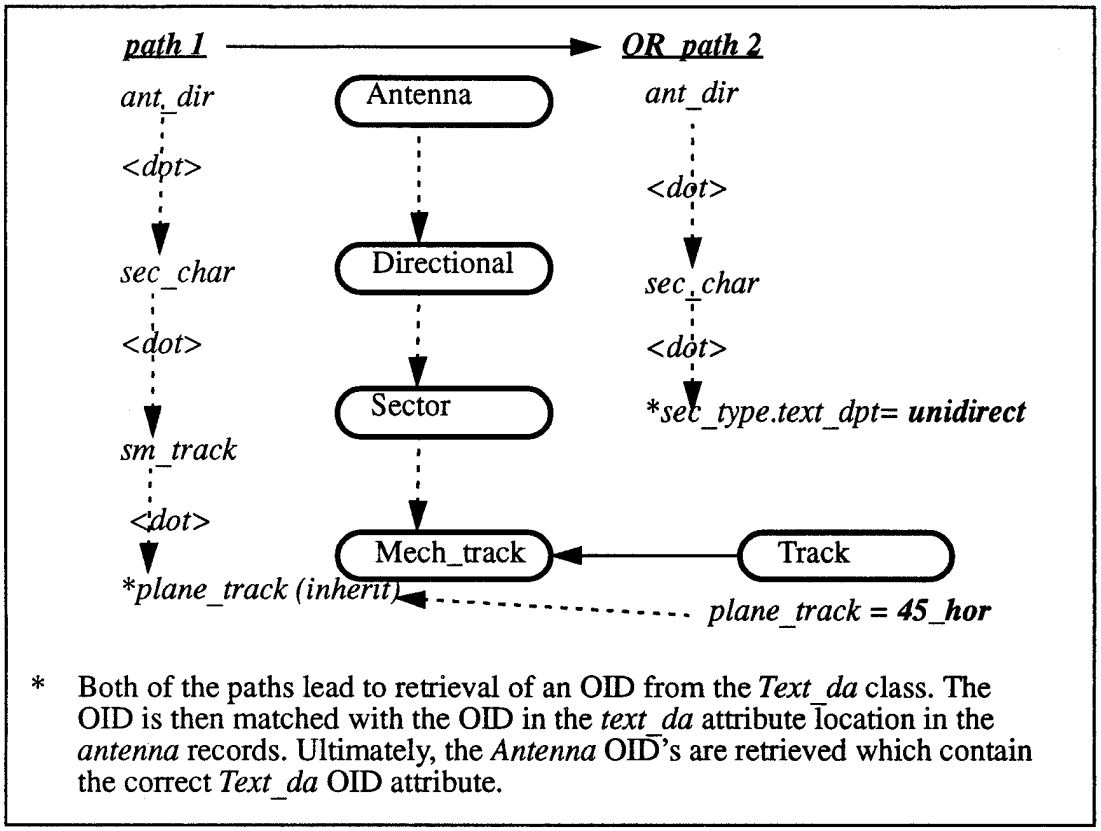


Figure 31. The Multi-level 1:1 Relationship of Query_3.

4. A Two-level Inheritance (Query_4)

Purpose: The purpose of Query_4 is to illustrate a two-level inheritance. Thus far we have done a simple attribute look-up (query_1), a one-level inheritance and 1:1 relationship combination (query_2), a multi-level 1:1 (query_3). The multi-level inheritance of this query completes the testing of basic EWIR data relationships on the Object-Oriented Interface.

Request: What are the sector type and the upper and lower values of the period limits of an antenna with a sample average scan time of 325 ms?

Query: **Query Find_sector IS**

```
    obj_ref p;
Begin
    p:= find_one sector where
    smp_avg_time.value = '325_ms';
    display(p.sec_type.text_dpt,p.per_limits.lwr_value,
            p.per_limits.upr_value);
End;
```

Results: Text_dpt: *unidirect*
Lower_value: *100_ms*
Upr_value: *128_ms*

Remarks: The relative simplicity of queries involved in inheritance hierarchies is a major strength of object-oriented database systems in general and the Object-Oriented Interface in particular. For example, the attribute *smp_avg_time* is not an attribute of *sector*, but *sector* inherits from *Mech_sc*. The *smp_avg_time* attribute is not located in *Mech_sc*, but *Mech_sc* inherits from *Scan*. The attribute is located in *Scan* where a match is made. The inheritance structure does not require the use of any dot notation, thus greatly simplifying the structure of the query. The query_4 logic is illustrated and explained in Figure 32.

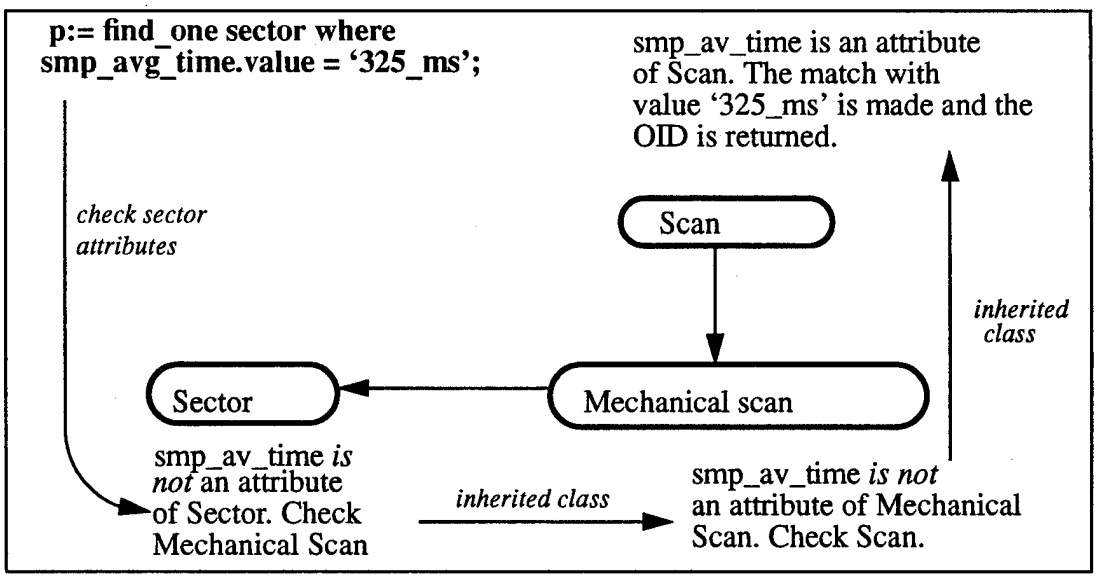


Figure 32. The Two-Level Inheritance of Query 4.

5. A Five-level Inheritance and One-level 1:1 Relationship (Query_5)

Purpose: The purpose of query_5 is the data retrieval from *Signal Data* via the most complex structure available. The complexity is defined in terms of depth of data relationships. This particular query will traverse through five levels of inheritance plus a 1:1 relationship in total. This is the deepest inheritance structure in the entire live EWIR database. Notice that despite the complexity of the query in terms of the depth, the query structure itself is very simple.

Request: What are all modified waveforms (discrete agility) of a signal that is transmitted at a constant peak power of 300 kw?

Query:

```
Query Find_waveforms IS
    obj_set p;
    obj_ref i;
Begin
    p:= find_many disc_ag where
    scon_pwr.pk_pw_txmitter.value = '300_kw';
  For Each i IN p
    display(p.mod_waveform.text_dpt);
  End_loop;
End;
```

Result: Text_dpt: *pulse_wave*
mod_pulse_wave

Remarks: In Figure 33 is a graphical depiction of the path taken through the five levels of inheritance and the 1:1 relationship. The mechanics of inheritance relationships is explained in both Query_2 and Query_4. Query_5 is the initial test of a change to the Object-Oriented Interface code allowing four-levels of an inheritance in a single query. The previous code restricted the inheritance hierarchy to three-levels.

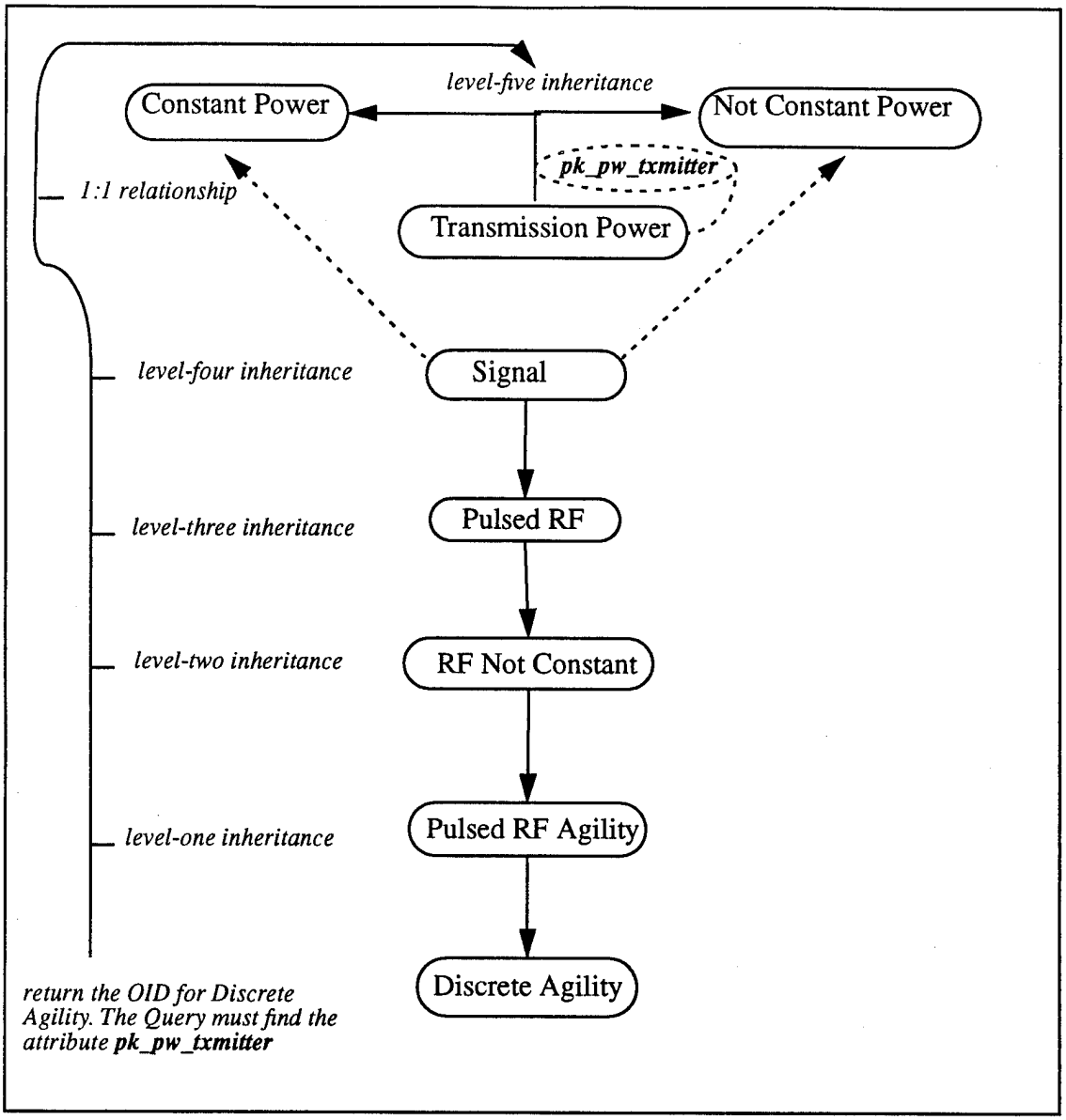


Figure 33. The Multi-level Inheritance Hierarchy of Query_5.

6. The Retrieval of the Emitter ELNOT (Query_6)

Purpose: The ELNOT is the notation used to uniquely identify the emitter in the EWIR database. It is essential to be able to provide ELNOT information to the user, given an emitter's parametric data. When given parametric data, the query_ 6 retrieves the associated emitter ELNOT.

In Query_6 we use the *contains* statement for the first time. The *contains* statement is a set comparison function. It returns the objects in a set which satisfy an expression or match objects in a second set.

Request: What is the ELNOT and the weapon system for an emitter with an antenna having a mechanical scan tracking plane of "45_hor"?

Query:

```
Query find_emitter IS
    obj_ref p,i;
    obj_set q
Begin;
    p:= find_one antenna where
      ant_direc.sec_char.sm_track.plane_track.text_dpt = '45_hor';
    q:= find_many emitter where ant_comp contains p;

   For Each i IN q
     display(q.unique_id.elint_notation, q.weap_system.text_dpt);
   End_loop;
End;
```

Results: Elint_notation: *enot_1*
Text_dpt: *aa_6*

Remarks: In Figure 34 is a depiction of the Query_6 logic. The EWIR schema file shows that the class *Emitter* uses the *set_of* construct for the antenna_component (ant_comp) attribute. The meaning of *set_of* is that one emitter can have many antennas. In this query, all of the antennas of an emitter are searched, using the *contains* statement, for a match with the data contained in the variable *p*. The results are stored in the variable *q* for eventual display.

7. The Retrieval of Parametric Data Given an ELNOT (Query_7)

Purpose: The purpose of query_7 is to retrieve the parametric data of an emitter given only its ELNOT. The ELNOT is the highest-level identification of an emitter and can therefore be used to retrieve any of the related emitter data. This is essentially a reverse operation of Query_6 which *returned* an emitter ELNOT given specific parametric data.

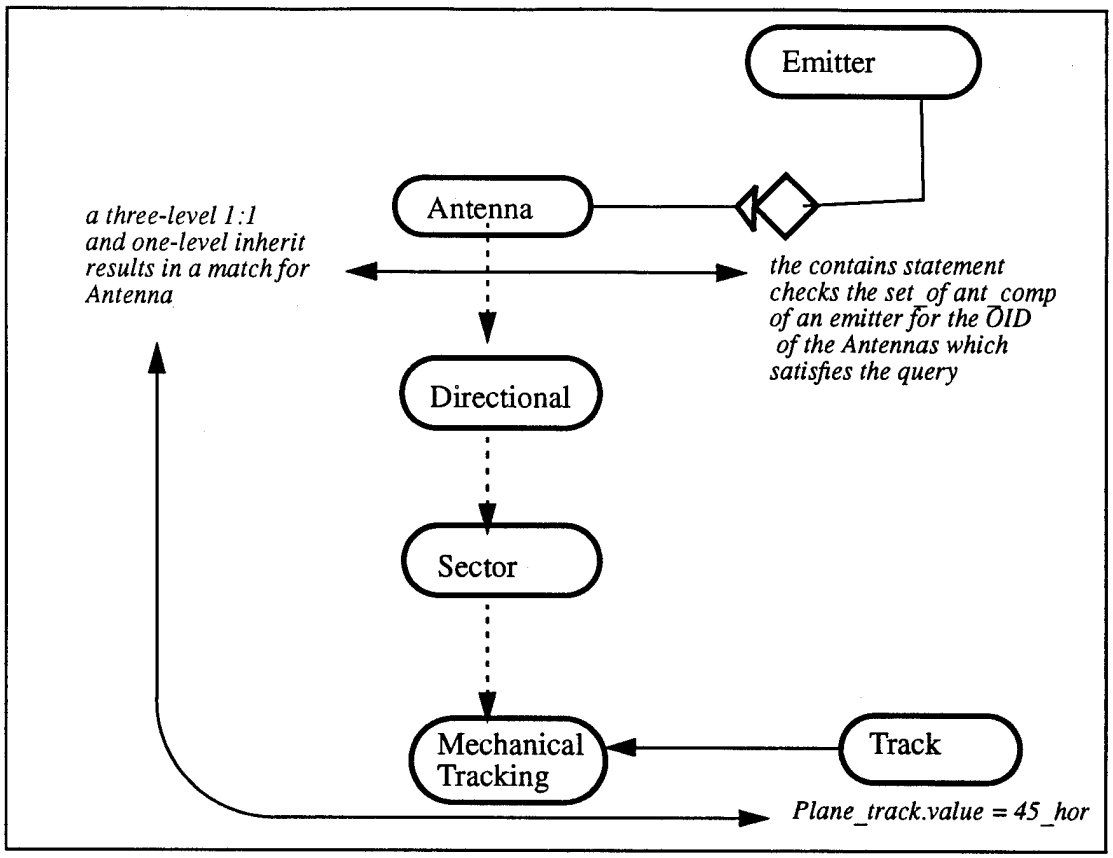


Figure 34. The Retrieval of an ELNOT (Query_6).

This query also utilizes the *inverse_of* function for the first time. The *inverse_of* function creates an M:N capability when used in conjunction with the *set_of* function. Specifically, this query uses the following EWIR classes to create the M:N relationship

```
Class Emitter{
    set_of Antenna      ant_comp;

Class Antenna{
    inverse_of Emitter.ant_comp    anten_comp;
```

Request: Retrieve all weapon systems and the associated antenna type for a given an emitter ELNOT of *e_not1*.

Query:

```
Query find_emitters IS
    obj_ref i;
    obj_set p, q;
```

Begin

```
p:= find_many emitter where
     unique_id.elint_notation = 'e_not1';
q:= find_many antenna where anten_emit contains p;

For Each i IN p
   display(i.weap_system.text_dpt);
End_loop;

For Each i IN q
  display(i.ant_type.text_dpt);
End_loop;
```

End;

Results: Text_dpt: *su_22*
mig_21
mig_23
Text_dpt: *phased_array*
square_sail

Remarks: A visual depiction of Query_7 is illustrated in Figure 35.

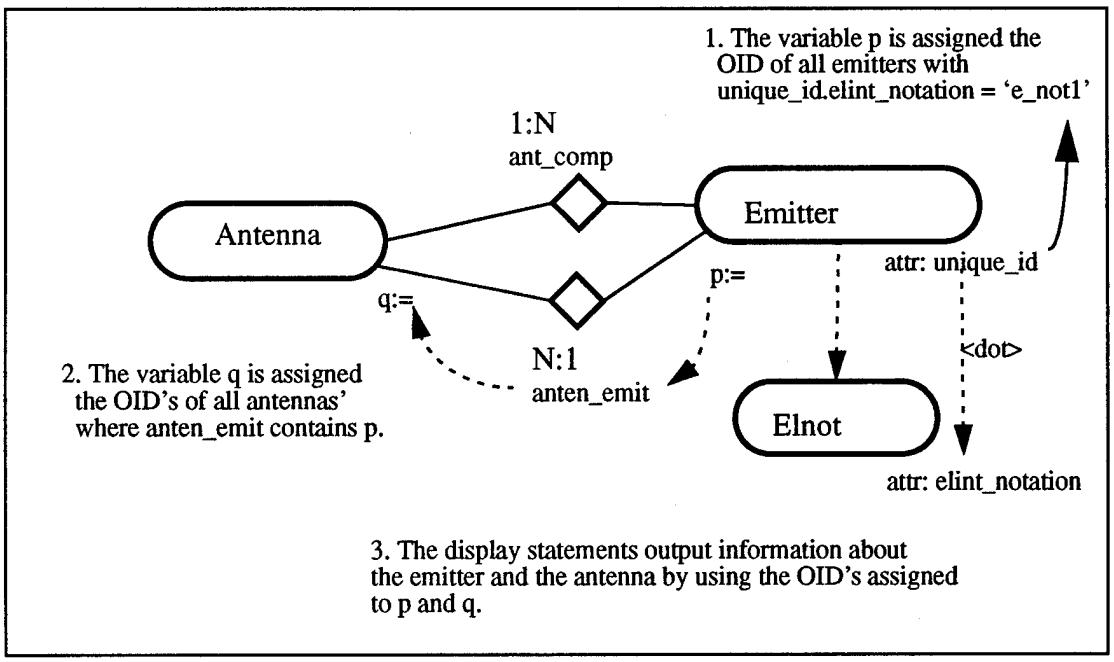


Figure 35. The Retrieval of Parametric Data for a given an ELNOT.

8. Multiple Logical Operators and Changing Data (Query_8)

Purpose: The primary purpose of query_8 is to retrieve data using multiple logical operators, change the value of an attribute, and store the new value in the database. Thus, Query_8 only *replaces* the attribute value of an existing EWIR attribute and no modification in the schema takes place.

Request: Downgrade the pulse wave radar from a long-range-aa to a medium-range-aa platform if the antenna type is phased array and its axial ratio value is '20_db'.

Query:

```
Query Change_antenna IS
    obj_ref p;
 Begin
    p:= find_one Antenna Where ant_function.text_dpt = 'lng_rang_aa'
    AND ant_type.text_dpt = 'pulse_wave'
    AND ac_el_pol.ax_ratio.value = '20_db';
    Display (p.ant_type.text_dpt, p.ant_function.text_dpt);
    p.ant_function.text_dpt:= 'medium_rng_aa';
    Display (p.ant_type.text_dpt, p.ant_function.text_dpt);
 End;
```

Results: Text_dpt: phased_array; lng_rng_aa;
Text_dpt: phased_array; med_rng_aa

Remarks: Figure 36 is the graphical depiction of query_8.

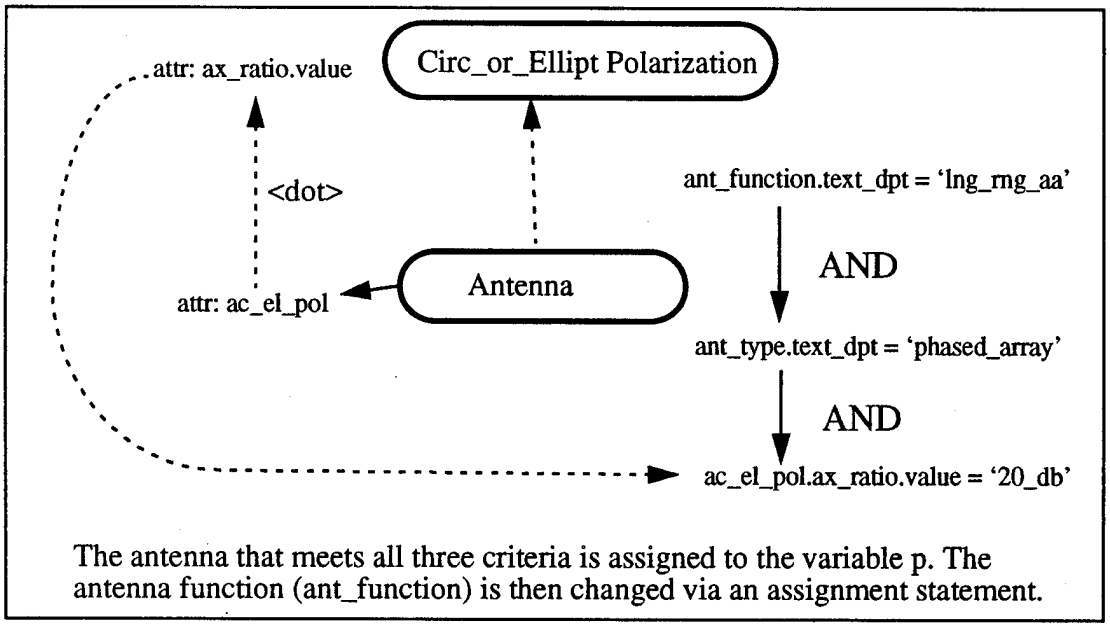


Figure 36. The Changing of a Data Value (Query_8).

9. The Data Retrieval From Multiple Subsections of EWIR (Query_9)

Purpose: The purpose of Query_9 is to retrieve data from all three major subsections of the EWIR database using the logical operators. Query_9 demonstrates that there are no limits to the number of major subsections of the EWIR database in performing a transaction. Without any higher level functions such as *if-then-else*, the complexity of the transaction is limited to logical comparisons of retrieved information.

Request: Determine which weapon systems have a transmission loss on a transmit of 5_db or a receiver with doppler processing of 20_cpi or a directional radiation pattern antenna gain of 10_db.

Query:

```
Query multiple_choice IS

    obj_set q,z,y,p;
    obj_ref i;

Begin

    q:= Find_many disc_ag where scon_pwr.tx_ls_tx.value = '5_db';
    z:= find_many Receive Where
         sig_processor.doppler_calc.num_pulses_cpi.value ='20_cpi';
    y:= find_many antenna Where ant_direc.ant_gain.value = '10_db';
    p:= find_many Emitter where edis_agility contains q OR
       rec_comp contains z OR
       ant_comp contains y;

    For Each i in p
      display(i.weap_system.text_dpt);
    End_loop;

End;
```

Results: Text_dpt: *aa_10*
aa_6
sa_21

Remarks: In Figure 37 is the graphical depiction of Query_9. The variables q, z, and y contain the OID's of the qualifying objects from the Signal, Receiver, and Antenna subsections respectively. The variable p contains the OID's of emitters whose attributes (edis_agility, rec_comp, ant_comp) contain the OID's in q, z, and y.

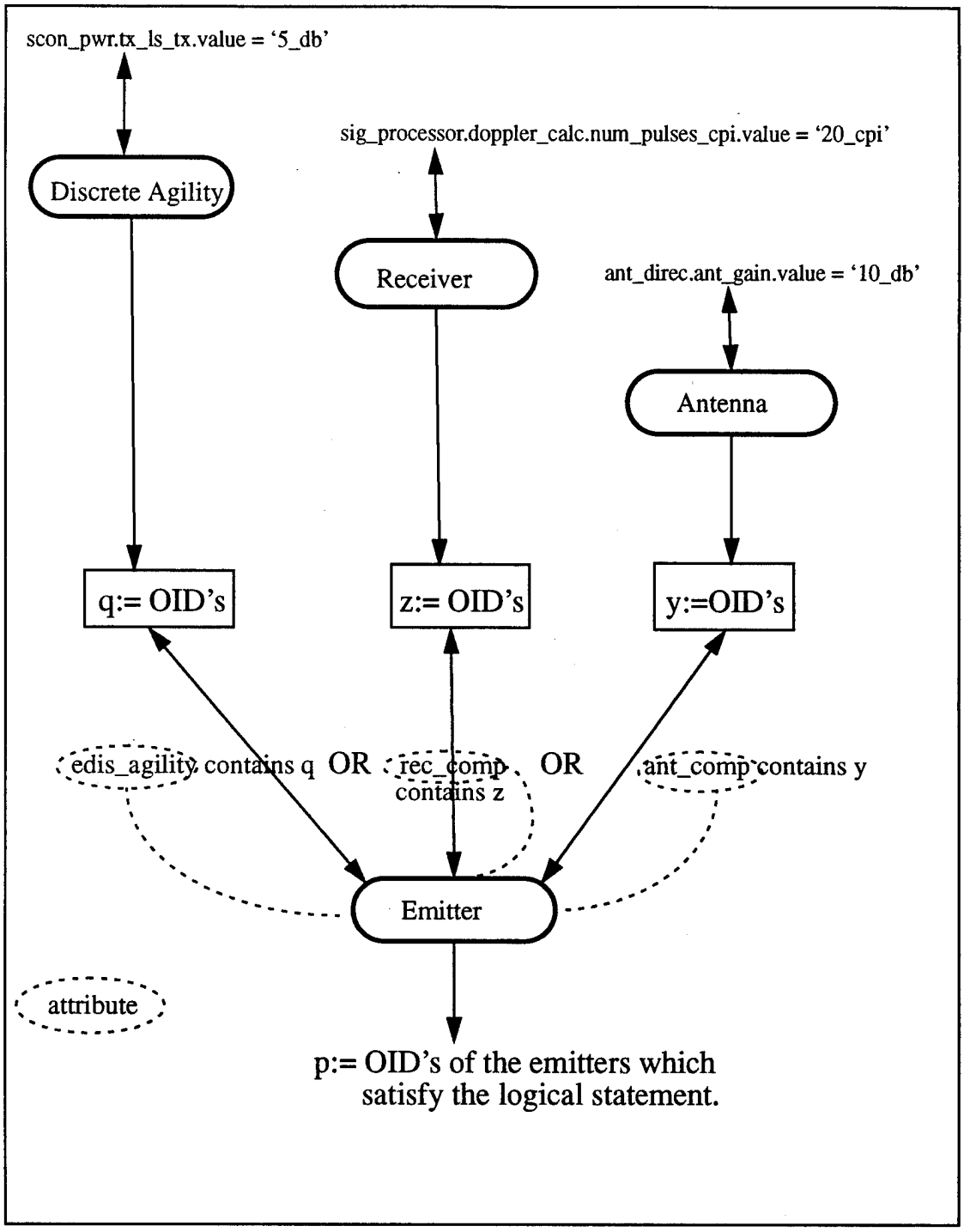


Figure 37. Data Retrieval From Different EWIR Subsections (Query_9).

VI. CONCLUSIONS

In this thesis we first create a live EWIR database given only the object-oriented specification of the existing EWIR data. In Section A is a summary of the methodology used in this thesis to create the live EWIR database. We then create a number of live object-oriented transactions for the purpose of accessing and manipulating the newly created and live object-oriented EWIR database. In Section B is an evaluation of the performance of the Object-Oriented Interface in its support of our new and live transactions. In Section C, there is a conclusive discussion on the merits of the object-oriented implementation of the EWIR database and its transactions.

A. THE THESIS METHODOLOGY SUMMARY

The Electronic Warfare Integrated Reprogramming (EWIR) database is the primary Department of Defense source for technical parametric performance data on non-communications emitters. It has been identified by the National Air Intelligence Center as difficult to use in its current hierarchical database form. There are two problems addressed by this thesis. First, with object-oriented database management being the latest database technology, is an object-oriented EWIR database a superior method for managing complex electronic warfare data collections? Second, is the prototype Object-Oriented Interface of the Multimodel and Multilingual Database Management System developed at the Laboratory for Database System Research in the Naval Postgraduate School capable of supporting a complex object-oriented database such as EWIR and its transactions?

To answer these questions, we first examine the object-oriented specification of the existing EWIR database [Ref. 1] and derive a *subset* for our implementation on the Object Oriented Interface. Using the Object-Oriented Interface DDL, the new object-oriented EWIR database schema and its associated data are stipulated and loaded into M^2DBMS in order to create the live database. Using the Object-Oriented Interface DML, nine object-oriented transactions for EWIR processing are elaborated and executed. The following sections summarize the results of this research.

B. THE EVALUATION OF THE OBJECT-ORIENTED INTERFACE

The EWIR database is the first large, complex database implemented on the Object-Oriented Interface of the M^2DBMS. The EWIR implementation is designed to test the effectiveness, correctness, and efficiency of the Object-Oriented Interface. Specific conclusions on the Object-Oriented Interface include:

- All functions currently implemented on the Object-Oriented Interface work as specified. Some minor code changes have been made during the course of this thesis to correct logic errors or expand the usability of the functions.

- The inheritance relationship allows the data retrieval from a specialization class to a generalization class only. The inheritance functionality should be expanded to include data retrieval capability for the generalization class to specialization class direction.

- The only functions which provide data manipulation capabilities are the *find_one*, *find_many*, *contains*, logical *or*, and logical *and*. These functions restrict transactions to object and object-set comparisons. Additional functions are needed to meet the extensive data manipulation requirements of large, real-world databases such as the full set of EWIR data. Recommended additional functions include *insert*, *delete*, and the *if-then-else* or *case* statement.

- In keeping with the object-oriented paradigm, it is recommended that future students implement *methods*. The methods provide to the user an external interface and also encapsulate the intended data operations for specific objects. The external interface is a key component of a complete object-oriented database system.

- The character-length restriction on class names and attribute names should be eliminated. A large, complex database needs the versatility to clearly name objects. In this thesis, many class and attribute names are cryptic and difficult to understand due to this restriction.

In summary, the Object-Oriented Interface forms an excellent foundation on which to create a complete object-oriented database system. The underlying kernel system is well designed and has a proven ability to support transactions from several different database systems. The prototype Object-Oriented Interface evaluated in this thesis fully supported the complex EWIR database *within the scope of its currently available features*. We see no impediment to the further design and implementation of additional Objection-Ori-

ented Interface functionality. The cross-model accessing capability to the other non-object-oriented databases in the M^2DBMS is also recommended.

C. THE EVALUATION OF THE OBJECT-ORIENTED EWIR DATABASE

As stated in Chapter I, a primary goal of this thesis is to determine whether the object-oriented EWIR database is more effective, efficient, and intuitive for managing complex electronic warfare data collections than the non-object-oriented EWIR database. The evaluation of the object-oriented EWIR database implemented on the Object-Oriented Interface yields the following conclusions:

- The object-oriented specification is easier to understand than the existing hierarchical structure. Data relationships in the object-oriented specification are more naturally depicted and not subject to the user interpretation errors of the existing EWIR specification.

- The Inheritance feature of object-orientation greatly simplified query construction. Inherited attributes are available to a class without a requirement to explicitly define either path statements or relations.

- Accessing specific data elements via the path statement is a simple process. By referencing the object-oriented specification, path construction is no more difficult than following a "roadmap" that leads to the class or attribute of interest.

- The construction of ad-hoc queries is quick and easy. This is due primarily to the ease of understanding the data relationships given a well designed specification. The inheritance, path, and object comparison features streamline the linkage of related data, thus simplifying query construction.

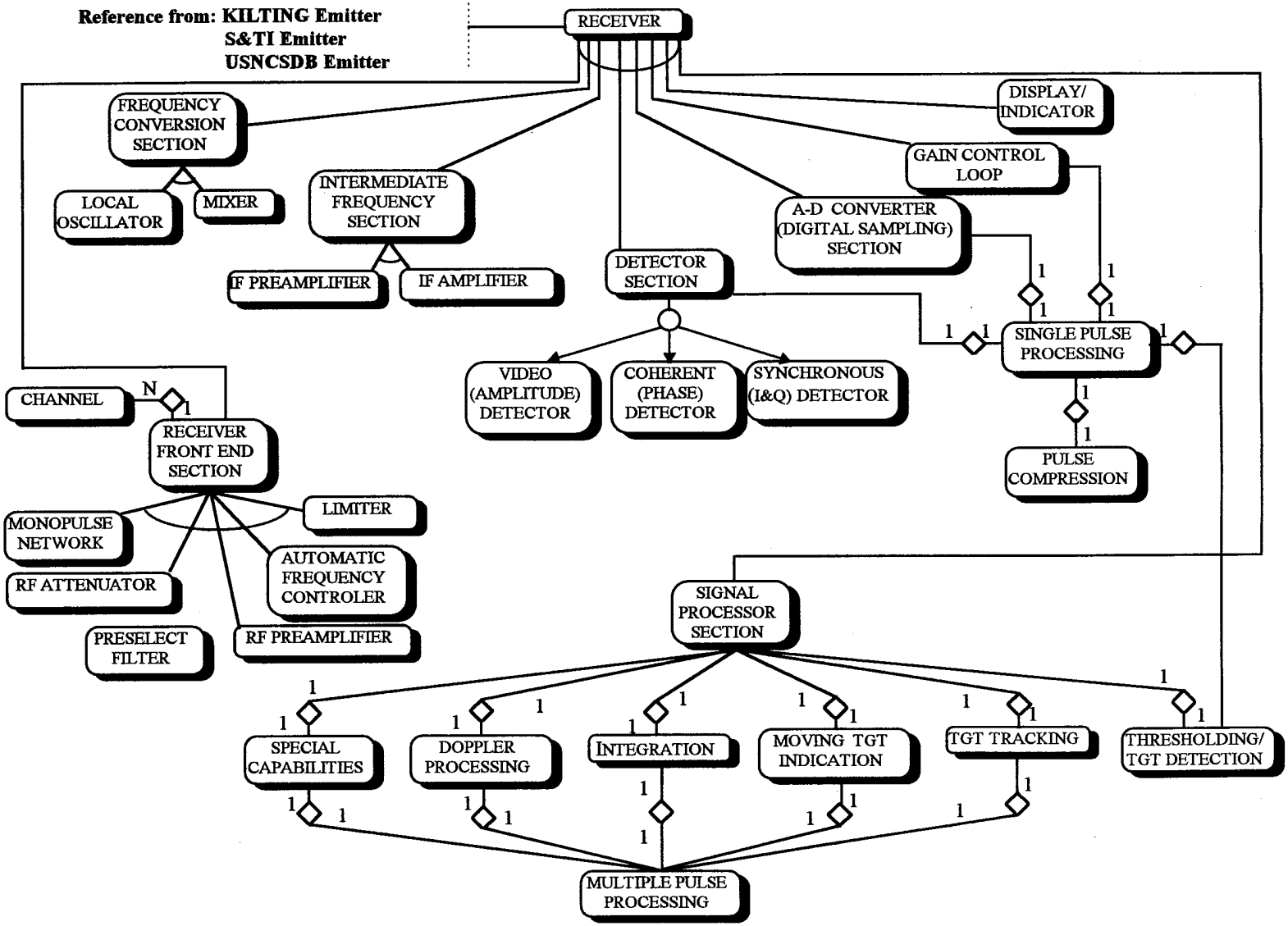
The major complaint from the users of the existing EWIR data is the heavy reliance on the user's understanding of the complex data relationships. From the user's perspective, the object-oriented paradigm is proven to be a superior method of modeling data. It naturally follows that a well designed object-oriented specification of a database would enhance user understanding.

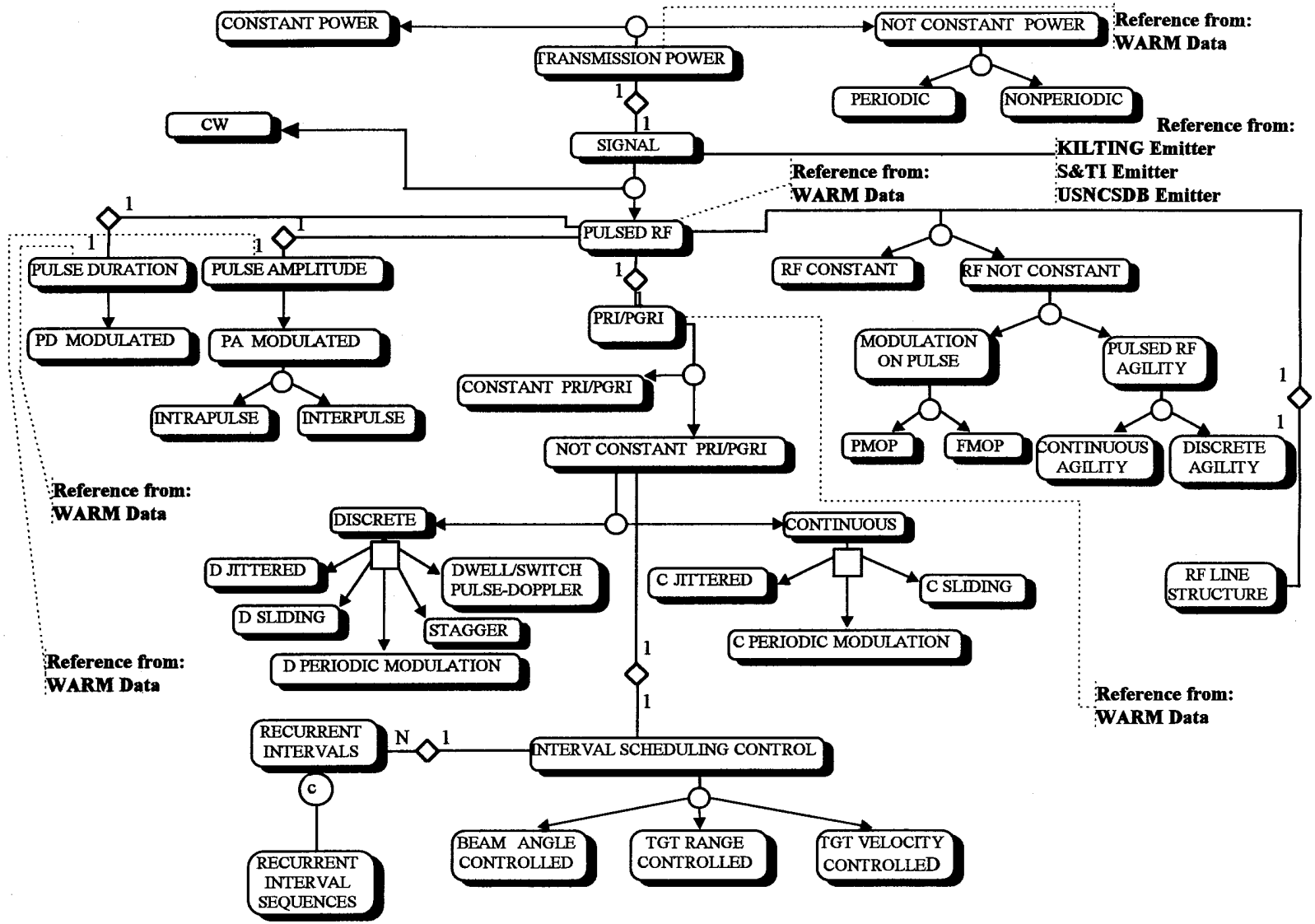
This thesis demonstrated that the object-oriented specification of the EWIR data produces a data model that more accurately reflects the true data relationships. This thesis

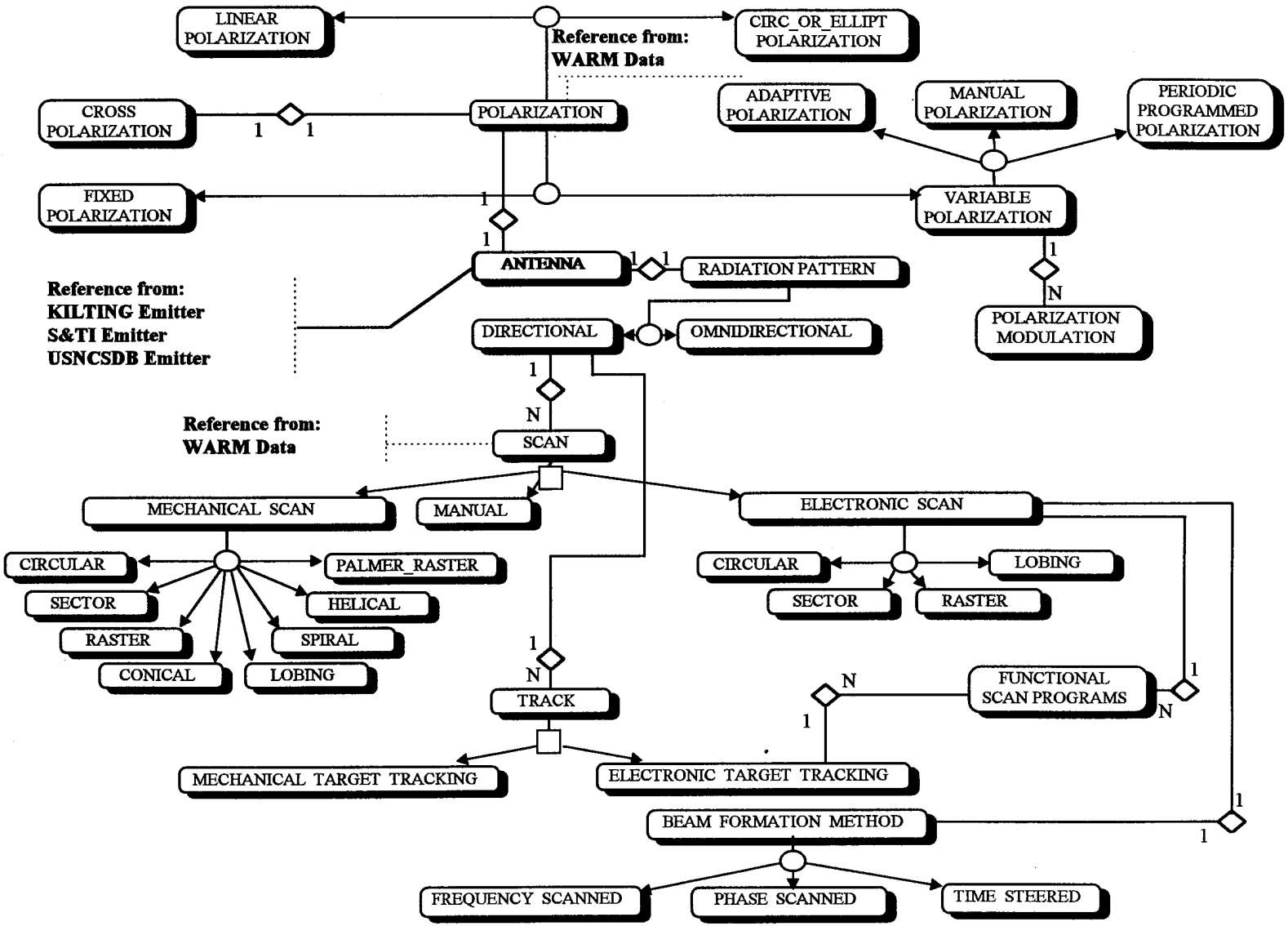
demonstrated that users with little prior knowledge of the EWIR data could take an object-oriented specification and construct a live EWIR database that is accurate in its EWIR data definition. This thesis demonstrated that with little prior knowledge of EWIR, users can quickly construct and execute ad-hoc queries. In summary, a good object-oriented EWIR database specification implemented on a proven object-oriented database system is a viable and preferred option from the perspective of database users.

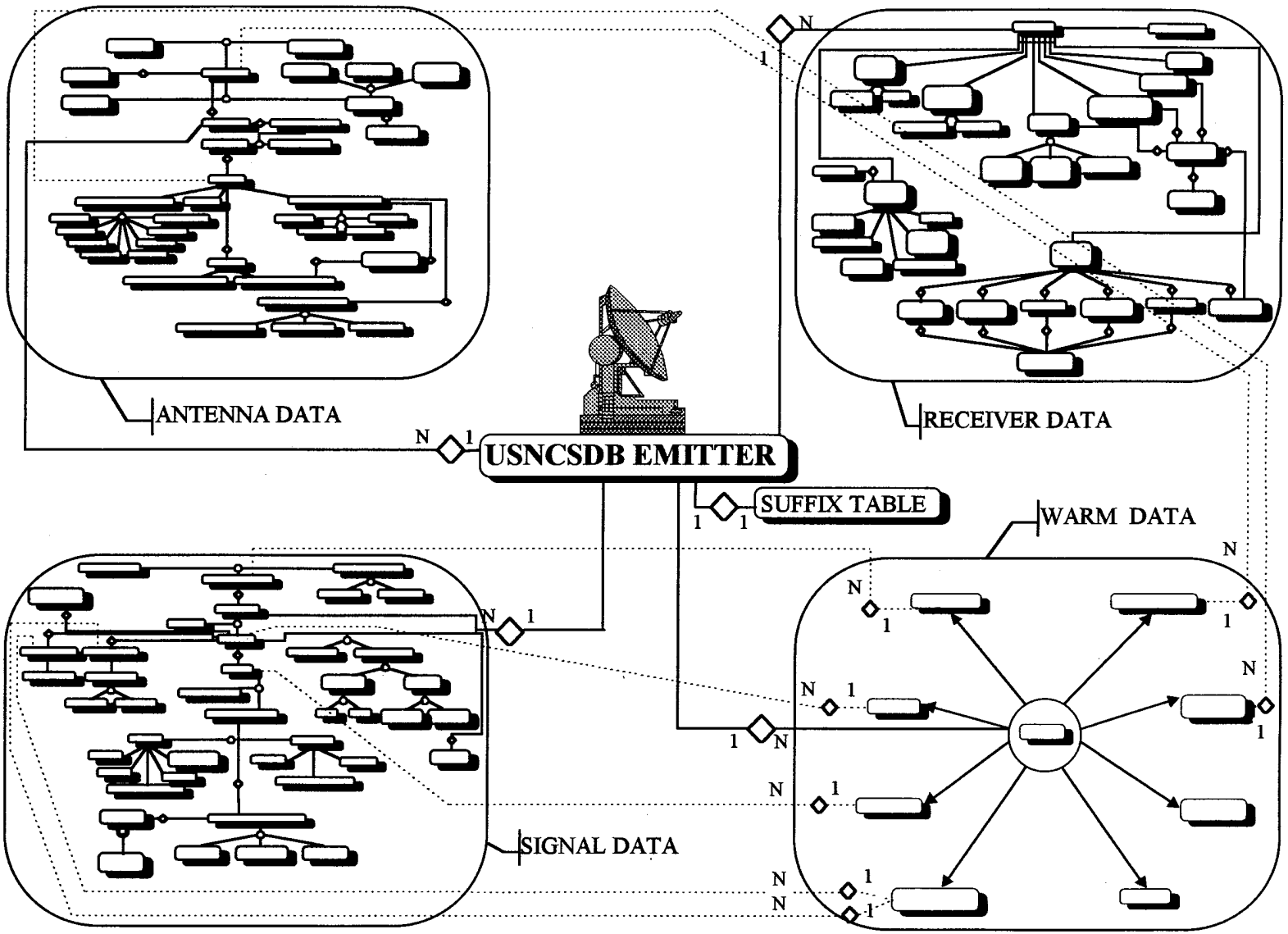
APPENDIX A. THE EWIR SPECIFICATION

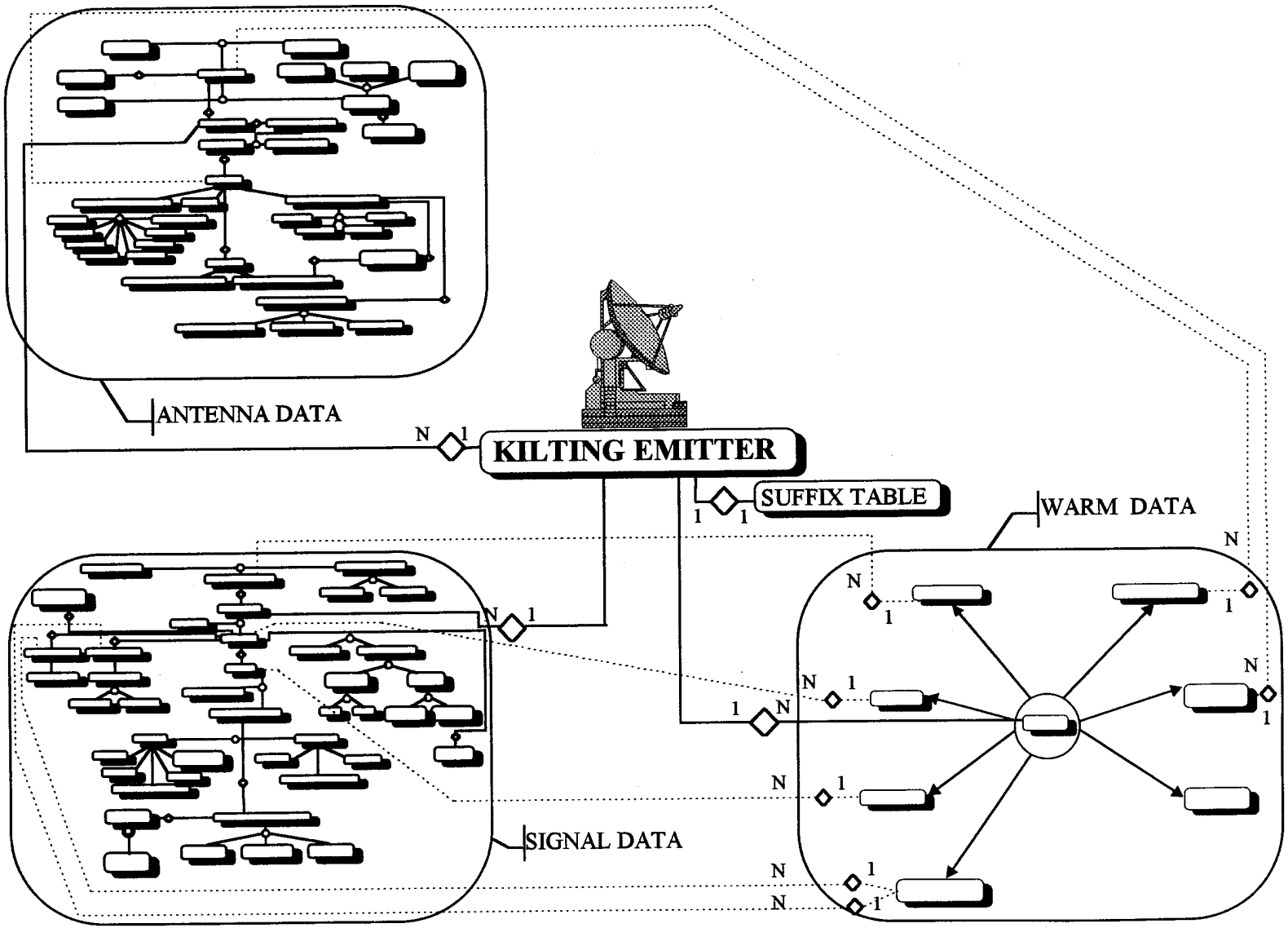
The figures presented in this appendix are the object-oriented specification of the existing EWIR database as presented in Ref. 1. This appendix is used to assist in the comparison of the EWIR specification of Ref. 1 with the derived subset of EWIR as specified in Chapter III of this thesis.

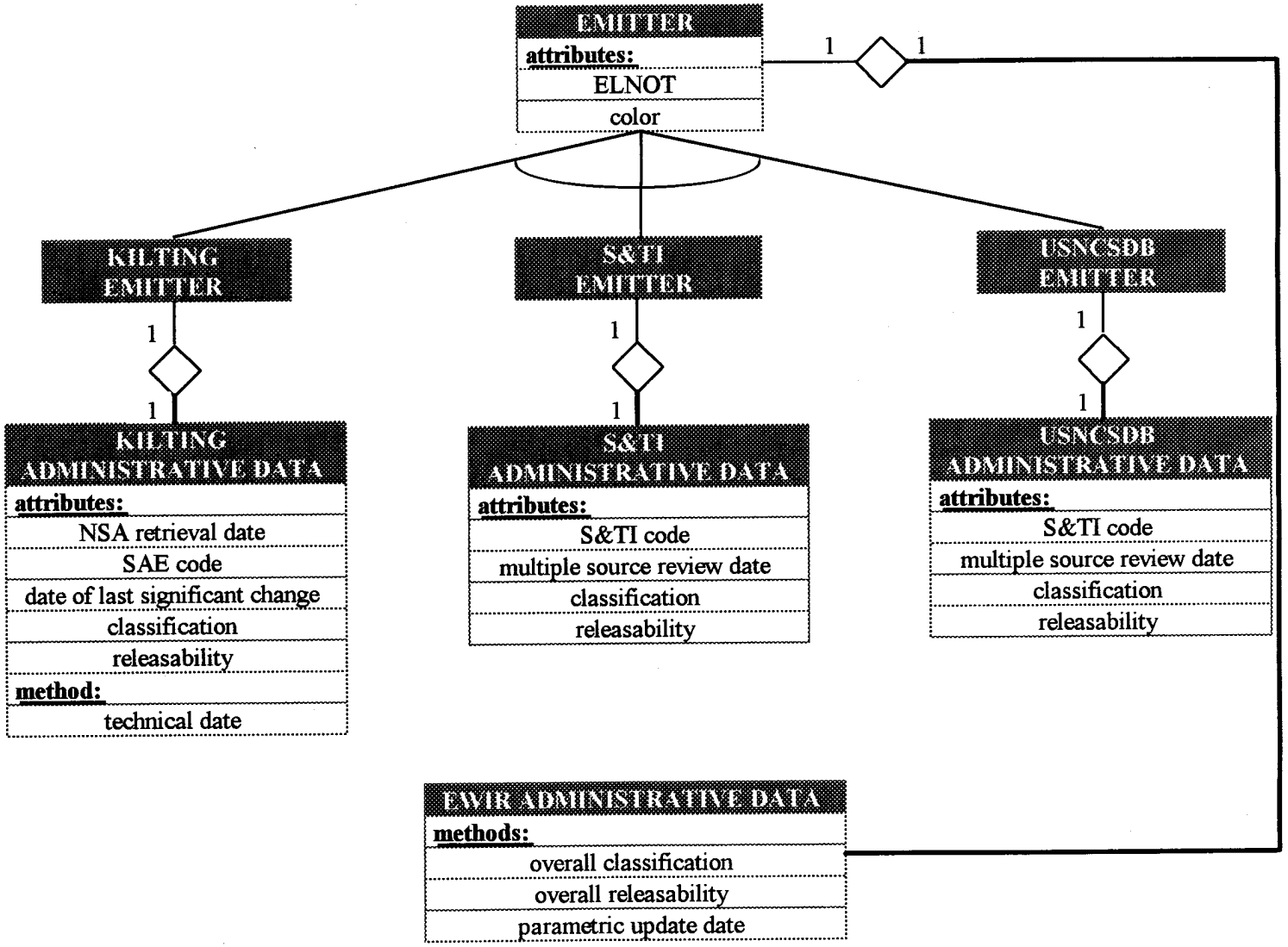

Reference from: KILTING Emitter
S&TI Emitter
USNCSDB Emitter
RECEIVER
FREQUENCY CONVERSION SECTION
LOCAL OSCILLATOR
MIXER
INTERMEDIATE FREQUENCY SECTION
IF PREAMPLIFIER
IF AMPLIFIER
DETECTOR SECTION
VIDEO (AMPLITUDE) DETECTOR
COHERENT (PHASE) DETECTOR
SYNCHRONOUS (I&Q) DETECTOR
A-D CONVERTER (DIGITAL SAMPLING) SECTION
GAIN CONTROL LOOP
DISPLAY/ INDICATOR
SINGLE PULSE PROCESSING
PULSE COMPRESSION
CHANNEL
N
1
RECEIVER FRONT END SECTION
MONOPULSE NETWORK
RF ATTENUATOR
PRESELECT FILTER
LIMITER
AUTOMATIC FREQUENCY CONTROLER
RF PREAMPLIFIER
SIGNAL PROCESSOR SECTION
SPECIAL CAPABILITIES
DOPPLER PROCESSING
INTEGRATION
MOVING TGT INDICATION
TGT TRACKING
THRESHOLDING/ TGT DETECTION
MULTIPLE PULSE PROCESSING


CONSTANT POWER
TRANSMISSION POWER
NOT CONSTANT POWER
Reference from:
WARM Data
PERIODIC
NONPERIODIC
CW
SIGNAL
Reference from:
KILTING Emitter
S&TI Emitter
USNCSDB Emitter
Reference from:
WARM Data
PULSED RF
PULSE DURATION
PULSE AMPLITUDE
RF CONSTANT
RF NOT CONSTANT
PRI/PGRI
PD MODULATED
PA MODULATED
MODULATION ON PULSE
PULSED RF AGILITY
CONSTANT PRI/PGRI
INTRAPULSE
INTERPULSE
PMOP
FMOP
CONTINUOUS AGILITY
DISCRETE AGILITY
NOT CONSTANT PRI/PGRI
Reference from:
WARM Data
DISCRETE
CONTINUOUS
D JITTERED
DWELL/SWITCH PULSE-DOPPLER
C JITTERED
C SLIDING
RF LINE STRUCTURE
D SLIDING
STAGGER
C PERIODIC MODULATION
Reference from:
WARM Data
D PERIODIC MODULATION
Reference from:
WARM Data
RECURRENT INTERVALS
N
INTERVAL SCHEDULING CONTROL
c
RECURRENT INTERVAL SEQUENCES
BEAM ANGLE CONTROLLED
TGT RANGE CONTROLLED
TGT VELOCITY CONTROLLED

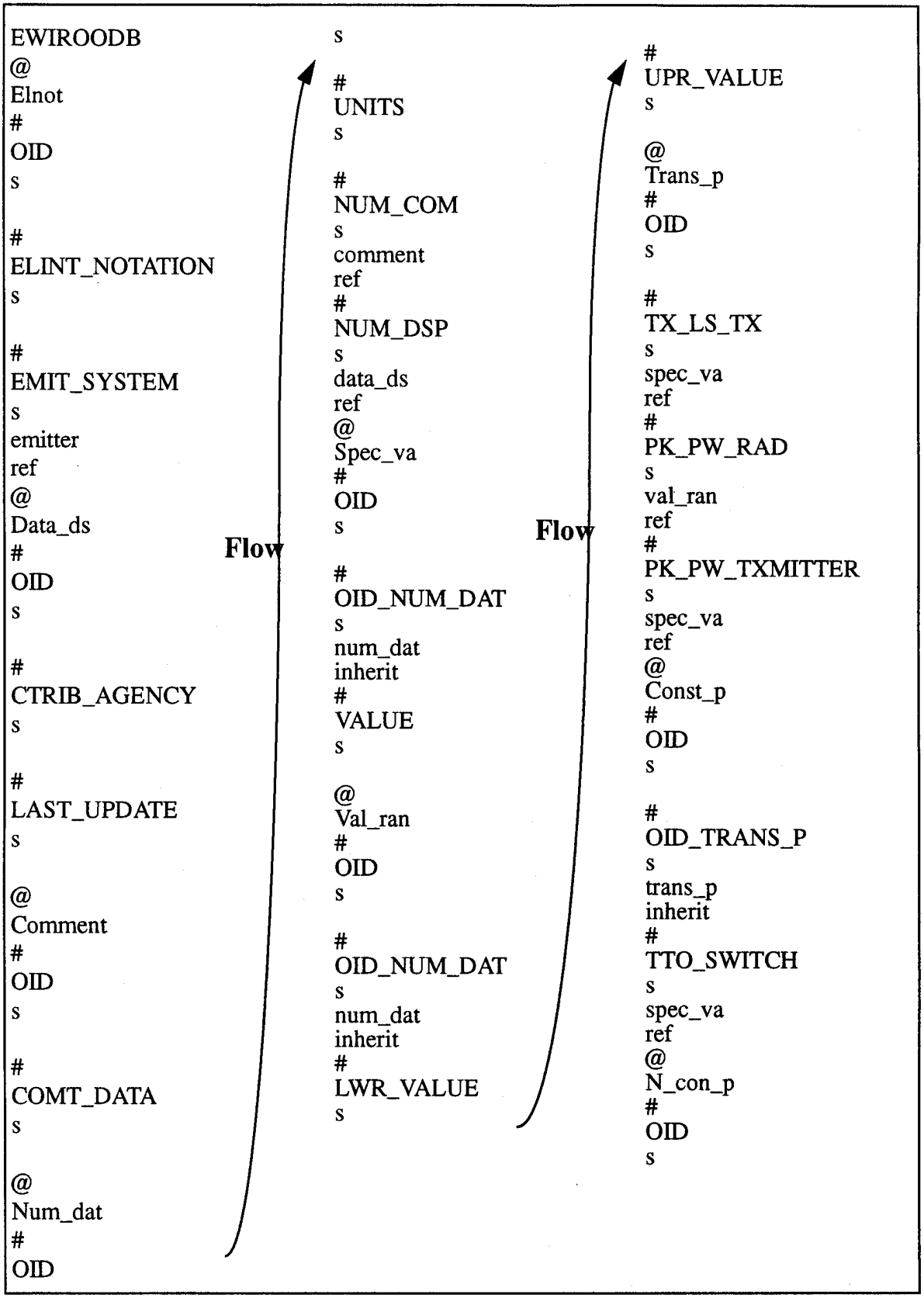

LINEAR POLARIZATION
Reference from: WARM Data
CIRC_OR_ELLIPT POLARIZATION
CROSS POLARIZATION
1
1
POLARIZATION
ADAPTIVE POLARIZATION
MANUAL POLARIZATION
PERIODIC PROGRAMMED POLARIZATION
FIXED POLARIZATION
1
VARIABLE POLARIZATION
1
ANTENNA
1
1
RADIATION PATTERN
Reference from: KILTING Emitter S&TI Emitter USNCSDB Emitter
1
N
POLARIZATION MODULATION
DIRECTIONAL
OMNIDIRECTIONAL
1
N
Reference from: WARM Data
SCAN
MECHANICAL SCAN
MANUAL
ELECTRONIC SCAN
CIRCULAR
PALMER_RASTER
SECTOR
HELICAL
RASTER
SPIRAL
CONICAL
LOBING
CIRCULAR
LOBING
SECTOR
RASTER
1
N
TRACK
N
1
FUNCTIONAL SCAN PROGRAMS
1
N
MECHANICAL TARGET TRACKING
ELECTRONIC TARGET TRACKING
BEAM FORMATION METHOD
1
1
FREQUENCY SCANNED
PHASE SCANNED
TIME STEERED


ANTENNA DATA
RECEIVER DATA
USNCSDB EMITTER
SUFFIX TABLE
WARM DATA
SIGNAL DATA
N
1


ANTENNA DATA
KILTING EMITTER
SUFFIX TABLE
WARM DATA
SIGNAL DATA
N
1


EMITTER
attributes:
ELNOT
color
1
1
KILTING EMITTER
S&TI EMITTER
USNCSDB EMITTER
1
1
1
1
1
1
KILTING ADMINISTRATIVE DATA
attributes:
NSA retrieval date
SAE code
date of last significant change
classification
releasability
method:
technical date
S&TI ADMINISTRATIVE DATA
attributes:
S&TI code
multiple source review date
classification
releasability
USNCSDB ADMINISTRATIVE DATA
attributes:
S&TI code
multiple source review date
classification
releasability
EMIR ADMINISTRATIVE DATA
methods:
overall classification
overall releasability
parametric update date

APPENDIX B. THE NEW EWIR DATABASE DATA DICTIONARY



#
MAX_CHANGE
s
spec_va
ref
@
Signal
#
OID
s

#
SCON_PWR
s
const_p
ref
#
SN_CON_PWR
s
n_con_p
ref
@
Text_da
#
OID
s

#
TEXT_DPT
s

#
TEXT_COM
s
comment
ref
#
TEXT_DSP
s
data_ds
ref
@
Rf_1_st
#
OID
s

#
DB3_S_WIDTH
s
spec_va
ref
#
TXMITTER_TYPE
s
text_da
ref
@
Pulsed
#
OID
s

#
OID_SIGNAL
s
signal
inherit
#
COHERENCE
s
rf_1_st
ref
@
Rf_n_co
#
OID
s

#
OID_PULSED
s
pulsed
inherit
#
DUMMY
s

@
Pul_rf
#
OID
s

#
OID_RF_N_CO
s
rf_n_co
inherit
#
AGT_F_CORREL
s
text_da
ref
@
Disc_ag
#
OID
s

#
OID_PUL_RF
s
pul_rf
inherit
#
MOD_WAVEFORM
s
text_da
ref
#
RF_LIMITS
s
val_ran
ref
#
NO_DISC_STEPS
s
spec_va
ref
@
Warm
#
OID
s

#
PROB_CODE
s

@
Rf_eccm
#

OID
s

#
OID_WARM
s
warm
inherit
#
RF_DISC_AGILITY
set_of
disc_ag_rf_eccm
store
@
Disc_ag_rf_eccm
#
OID
s

#
OID_DISC_AG
s
disc_ag
asc
#
OID_RF_ECCM
s
rf_eccm
asc
@
Polariz
#
OID
s

#
POLAR_DATA
s

@
C_el_po
#
OID
s

#
OID_RF_N_CO
s
rf_n_co
inherit
#
AGT_F_CORREL
s
text_da
ref
@
Disc_ag
#
OID
s

#
OID_PUL_RF
s
pul_rf
inherit
#
MOD_WAVEFORM
s
text_da
ref
#
RF_LIMITS
s
val_ran
ref
#
NO_DISC_STEPS
s
spec_va
ref
@
Warm
#
OID
s

#
PROB_CODE
s

@
Rf_eccm
#

#
OID_POLARIZ
s
polariz
inherit
#
SENSE
s
text_da
ref
#
AX_RATIO
s
spec_va
ref
@
Scan
#
OID
s

#
SMP_AVG_TIME
s
spec_va
ref
#
THRESHOLD_MEAS
s
spec_va
ref
#
PLANE_SCAN
s
text_da
ref
@
Mech_sc
#
OID
s

#
OID_SCAN
s

scan
inherit
#
STP_CG_ABILITY
s
text_da
ref
#
SC_FUNCTION
s
text_da
ref
@
Track
#
OID
s

#
PLANE_TRACK
s
text_da
ref
@
Mech_tr
#
OID
s

#
OID_TRACK
s
track
inherit
#
MAX_R_AZ
s
val_ran
ref
#
MAX_R_EL
s
val_ran
ref

@
Sector
#
OID
s

#
OID_MECH_SC
s
mech_sc
inherit
#
SEC_TYPE
s
text_da
ref
#
PER_LIMITS
s
val_ran
ref
#
SEC_W_AZ
s
spec_va
ref
#
SEC_W_EL
s
spec_va
ref
#
SM_TRACK
s
mech_tr
ref
@
Rad_pat
#
OID
s

#
ANT_GAIN
s
spec_va
ref
@
Directi
#
OID
s

#
OID_RAD_PAT
s
rad_pat
inherit
#
BWDTH_AZ
s
spec_va
ref
#
BWDTH_EL
s
spec_va
ref
#
FIRST_AZ
s
spec_va
ref
#
FIRST_EL
s
spec_va
ref
#
SEC_CHAR
s
sector
ref
@
Antenna
#
OID
s

#
ANT_TYPE
s
text_da
ref
#
ANT_FUNCTION
s
text_da
ref
#
HOR_DIMENSION
s
spec_va
ref
#
VERT_DIMENSION
s
spec_va
ref
#
AC_EL_POL
s
c_el_po
ref
#
ANT_DIREC
s
directi
ref
#
ANTEN_EMIT
inverse_of
emitter.ant_comp
store
@
Doper_p
#
OID
s

#
COH_PCESS_INT
s

spec_va
ref
#
NUM_PULSES_CPI
s
spec_va
ref
@
Sig_pce
#
OID
s

#
DOPPLER_CALC
s
doper_p
ref
@
A_d_con
#
OID
s

#
AD_SAMP_PERIOD
s
spec_va
ref
#
CONV_TRIG_METHO
s
text_da
ref
@
Receive
#
OID
s

#
RECEIVER_TYPE
s

text_da
ref
#
SIG_PROCESSOR
s
sig_pce
ref
#
AD_SECTION
s
a_d_con
ref
@
Emitter
#
OID
s

#
UNIQUE_ID
s
elnot
ref
#
ERF_ECCM
s
rf_eccm
ref
#
REC_COMP
set_of
receive_emitter
store
#
ANT_COMP
set_of
antenna_emitter
store
#
ECON_PWR
set_of
const_p_emitter
store
#
EN_CON_PWR

set_of
n_con_p_emitter
store
#
EDIS_AGILITY
set_of
disc_ag_emitter
store
#
WEAP_SYSTEM
s
text_da
ref
#
EMIT_FUNCTION
s
text_da
ref
#
EMIT_PTF_GEN
s
text_da
ref
@
Receive_emitter
#
OID
s

#
OID_RECEIVE
s
receive
asc
#
OID_EMITTER
s
emitter
asc
@
Antenna_emitter
#
OID
s

#
OID_ANTENNA
s
antenna
asc
#
OID_EMITTER
s
emitter
asc
@
Const_p_emitter
#
OID
s

#
OID_CONST_P
s
const_p
asc
#
OID_EMITTER
s
emitter
asc
@
N_con_p_emitter
#
OID
s

#
OID_N_CON_P
s
n_con_p
asc
#
OID_EMITTER
s
emitter
asc
@

Disc_ag_emitter
#
OID
s

#
OID_DISC_AG
s
disc_ag
asc
#
OID_EMITTER
s
emitter
asc
$

APPENDIX C. THE EWIR TEMPLATE FILE

```
EWIROODB
39
4
Elnot
TEMP s
OID s
ELINT_NOTATION s
EMIT_SYSTEM s
4
Data_ds
TEMP s
OID s
CTRIB_AGENCY s
LAST_UPDATE s
3
Comment
TEMP s
OID s
COMT_DATA s
5
Num_dat
TEMP s
OID s
UNITS s
NUM_COM s
NUM_DSP s
4
Spec_va
TEMP s
OID s
OID_NUM_DAT s
VALUE s
5
Val_ran
TEMP s
OID s
OID_NUM_DAT s
LWR_VALUE s
UPR_VALUE s
5
Trans_p
TEMP s
OID s
TX_LS_TX s
```

PK_PW_RAD s
PK_PW_TXMITTER s
4
Const_p
TEMP s
OID s
OID_TRANS_P s
TTO_SWITCH s
4
N_con_p
TEMP s
OID s
OID_TRANS_P s
MAX_CHANGE s
4
Signal
TEMP s
OID s
SCON_PWR s
SN_CON_PWR s
5
Text_da
TEMP s
OID s
TEXT_DPT s
TEXT_COM s
TEXT_DSP s
4
Rf_1_st
TEMP s
OID s
DB3_S_WIDTH s
TXMITTER_TYPE s
4
Pulsed
TEMP s
OID s
OID_SIGNAL s
COHERENCE s
4
Rf_n_co
TEMP s
OID s
OID_PULSED s
DUMMY s
4

Pul_rf
TEMP s
OID s
OID_RF_N_CO s
AGT_F_CORREL s
6
Disc_ag
TEMP s
OID s
OID_PUL_RF s
MOD_WAVEFORM s
RF_LIMITS s
NO_DISC_STEPS s
3
Warm
TEMP s
OID s
PROB_CODE s
3
Rf_eccm
TEMP s
OID s
OID_WARM s
4
Disc_ag_rf_eccm
TEMP s
OID s
OID_DISC_AG s
OID_RF_ECCM s
3
Polariz
TEMP s
OID s
POLAR_DATA s
5
C_el_po
TEMP s
OID s
OID_POLARIZ s
SENSE s
AX_RATIO s
5
Scan
TEMP s
OID s
SMP_AVG_TIME s

THRESHOLD_MEAS s
PLANE_SCAN s
5
Mech_sc
TEMP s
OID s
OID_SCAN s
STP_CG_ABILITY s
SC_FUNCTION s
3
Track
TEMP s
OID s
PLANE_TRACK s
5
Mech_tr
TEMP s
OID s
OID_TRACK s
MAX_R_AZ s
MAX_R_EL s
8
Sector
TEMP s
OID s
OID_MECH_SC s
SEC_TYPE s
PER_LIMITS s
SEC_W_AZ s
SEC_W_EL s
SM_TRACK s
3
Rad_pat
TEMP s
OID s
ANT_GAIN s
8
Directi
TEMP s
OID s
OID_RAD_PAT s
BWDTH_AZ s
BWDTH_EL s
FIRST_AZ s
FIRST_EL s
SEC_CHAR s

8
Antenna
TEMP s
OID s
ANT_TYPE s
ANT_FUNCTION s
HOR_DIMENSION s
VERT_DIMENSION s
AC_EL_POL s
ANT_DIREC s
4
Doper_p
TEMP s
OID s
COH_PCESS_INT s
NUM_PULSES_CPI s
3
Sig_pce
TEMP s
OID s
DOPPLER_CALC s
4
A_d_con
TEMP s
OID s
AD_SAMP_PERIOD s
CONV_TRIG_METHO s
5
Receive
TEMP s
OID s
RECEIVER_TYPE s
SIG_PROCESSOR s
AD_SECTION s
7
Emitter
TEMP s
OID s
UNIQUE_ID s
ERF_ECCM s
WEAP_SYSTEM s
EMIT_FUNCTION s
EMIT_PTF_GEN s
4
Receive_emitter
TEMP s

OID s
OID_RECEIVE s
OID_EMITTER s
4
Antenna_emitter
TEMP s
OID s
OID_ANTENNA s
OID_EMITTER s
4
Const_p_emitter
TEMP s
OID s
OID_CONST_P s
OID_EMITTER s
4
N_con_p_emitter
TEMP s
OID s
OID_N_CON_P s
OID_EMITTER s
4
Disc_ag_emitter
TEMP s
OID s
OID_DISC_AG s
OID_EMITTER s

LIST OF REFERENCES

[1] Coyne, K., *The Design and Analysis of an Object-Oriented Database of Electronic Warfare Data*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1996 (unpublished).

[2] Badgett, B., *The Design and Specification of an Object-Oriented Data Definition Language (O-ODDL)*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1995.

[3] Stevens, M., *The Design and Specification of an Object-Oriented Data Manipulation Language (O-ODML)*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1995.

[4] Hsiao, David K., "Interoperable and Multidatabase Solutions for Heterogeneous Databases and Transactions", a speech delivered at **ACM CSC '95**, Nashville, Tennessee, March 1995.

[5] Ramirez, L. and Tan, R., M., *The Design and Implementation of a Compiler for the Object-Oriented Data Definition Language (O-ODDL Compiler)*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1995.

[6] Barbosa, C. and Kutlusan, A., *The Design and Implementation of a Compiler for the Object-Oriented Data Manipulation Language (O-ODML Compiler)*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1995.

[7] Senocak, E., *The Design and Implementation of a Real-Time Monitor for the Execution of Compiled Object-Oriented Transactions (O-ODDL and O-ODML Monitor)*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1995.

[8] Kellett, D, and Kwon, T., *The Instrumentation of a Kernel DBMS for the support of a Database in the O-ODDL Specification*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1995.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., S&E
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Code 13
   Naval Postgraduate School
   Monterey, CA 93943-5101

3. Chairman, Code CS 2
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943

4. Dr David K. Hsiao, Code CS/HS 2
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943

5. Dr C. Thomas Wu, Code CS/KA 2
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943

6. LCDR Thomas D. McKenna 2
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943

7. LtCol J.J. Lee 2
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943

8. Ms Doris Mleczko, Code p22305 1
   Weapons Division
   Naval Air Warfare Center
   Pt. Mugu, CA 93042

9. Ms Sharon Cain 1
MAIC/SCDD
4115 Hebble Creek Rd
Wright Patterson AFB, OH 45433-5622