USAISEC
US Army Information Systems Engineering Command
Fort Huachuca, AZ 85613-5000

U.S. ARMY INSTITUTE FOR RESEARCH
IN MANAGEMENT INFORMATION,
COMMUNICATIONS, AND COMPUTER SCIENCES

ARMY NONPROGRAMMER SYSTEM FOR WORKING ENCYCLOPEDIA REQUESTS PHASE II FINAL REPORT (ASQBG-I-90-005)

DECEMBER 1989

AIRMICS
115 O'Keefe Bldg
Georgia Institute of Technology
Atlanta, GA 30332-0800

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704--188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION / AVAILABILITY OF REPORT N/A
2b. DECLASSIFICATION / DOUWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ASQBG-I-90-005	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION AIRMICS 6b. OFFICE SYMBOL (if applicable) ASQBG - I	7a. NAME OF MONITORING ORGANIZATION N/A
6c. ADDRESS (City, State, and ZIP Code) 115 O'Keefe Bldg., Georgia Institute of Technology Atlanta, GA 30332-0800	7b. ADDRESS (City, State, and Zip Code) N/A
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AIRMICS 8b. OFFICE SYMBOL (if applicable) ASQBG - I	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N/A
8c. ADDRESS (City, State, and ZIP Code) 115 O'Keefe Bldg., Georgia Institute of Technology Atlanta, GA 30332-0800	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 62783A PROJECT NO.: DY10 TASK NO.: 04-01 WORK UNIT ACCESSION NO.:

11. TITLE (Include Security Classification)

Army's Nonprogrammer System for Working Encyclopedia Requests (ANSWER) Phase II Final Report (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)

Dr. Karen Ryan, Cho-Li Hou, Datta Shetti

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
	FROM 06/16/89 TO 12/15/89	December 15, 1989	52

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Data Encyclopedia, Very Large Database, Distributed Query Processing, Metadata

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report describes research efforts into the access of distributed heterogeneous databases through an encyclopedia facility. Specifically, several data management tools that have been prototyped to date are described to include database registration, schema integration, browsing, an AI based standard data element naming tool, and an Information Resource Dictionary System (IRDS) repository. This toolset has been integrated under the X-windows interface management system. Plans for future efforts to include additional AI techniques for data management, security, distributed query formulation, and distributed query processing are also discussed.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED / UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
CPT Joseph J. Nealon	(404) 894-3110	ASQBG - I

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

This research was performed by Honeywell Federal Systems, contract number DAKF11-88-C-0024, for the Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS), the RDTE organization of the U.S. Army Information Systems Engineering Command (USAISEC). The report represents a 6 month effort, the second phase of a four phase effort. Requests to view a demonstration of the prototype may be made by contacting CPT Joe Nealon at 404/894-3110. This research report is not to be construed as an official Army position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited. Not protected by copyright laws.

THIS REPORT HAS BEEN REVIEWED AND IS APPROVED

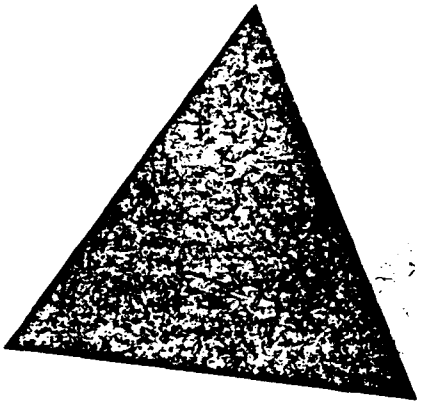
s/ Glenn Racine

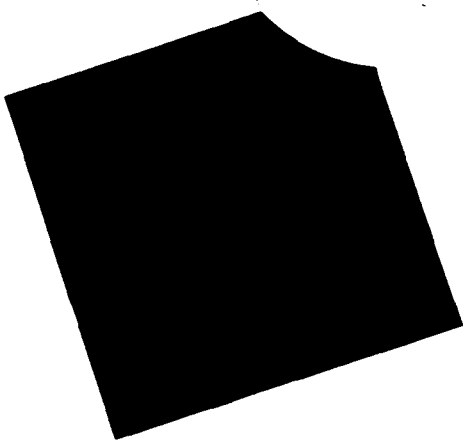
Glenn Racine, Chief
Computer and Information
Systems Division

s/ James T. Ritchie LTC, AV
for John R. Mitchell
Director
AIRMICS

Honeywell

ANSWER Phase II Final Report





Contract No.
DAKF11-99-C-0024

December 15, 1989

Sensor and System Development Center
1000 Boone Avenue North • Golden Valley • Minnesota • 55427

Honeywell

ANSWER
Phase II Final Report

Contract No. DAKF11-99-C-0024

Prepared for:

AIRMICS
Under Subcontract to
Honeywell Federal Systems Inc.

Honeywell Inc.
Sensor and System Development Center
1000 Boone Avenue North
Golden Valley, Minnesota 55427

December 15, 1989

G89381

Table of Contents

Table of Contents

Figures

Section 1
Introduction

This report covers the Phase II activities of the ANSWER program from June 16 to December 15, 1989.

Not another database management system (DBMS), repository or query language, ANSWER is a set of tools to support integrated data management specialized to the Army's needs. In the long term, ANSWER will provide support for building integrated views and definitions of data, graphical tools to manipulate data and query formulation facilities that can be used with existing distributed query processing environments.

In Phase I of the ANSWER program we developed a prototype schema integrator and graphical browser, and an implementation of a data definition repository. The schema integrator reads and writes entity-category-relationship schemas from an IRDS repository. In addition, the schema integrator creates a single integrated schema from two or more individual schemas. The integrated schema can be used as the basic description of data available at a single site in support of distributed query processing at that site. The IRDS repository is an ANSI and FIPS standard for data definition repositories. All ANSWER tools will eventually share information through the IRDS repository. The browser developed in Phase I presents a graphical display of schemas and related data definition information, following AR 25-9. The browser in Phase I was not integrated with IRDS. All the schema information in the Phase I browser prototype was represented in data structures local to the browser.

The overall ANSWER architecture as envisioned at the end of the program is shown in Figure 1-1. The deliverables from Phase I are highlighted in Figure 1-2.

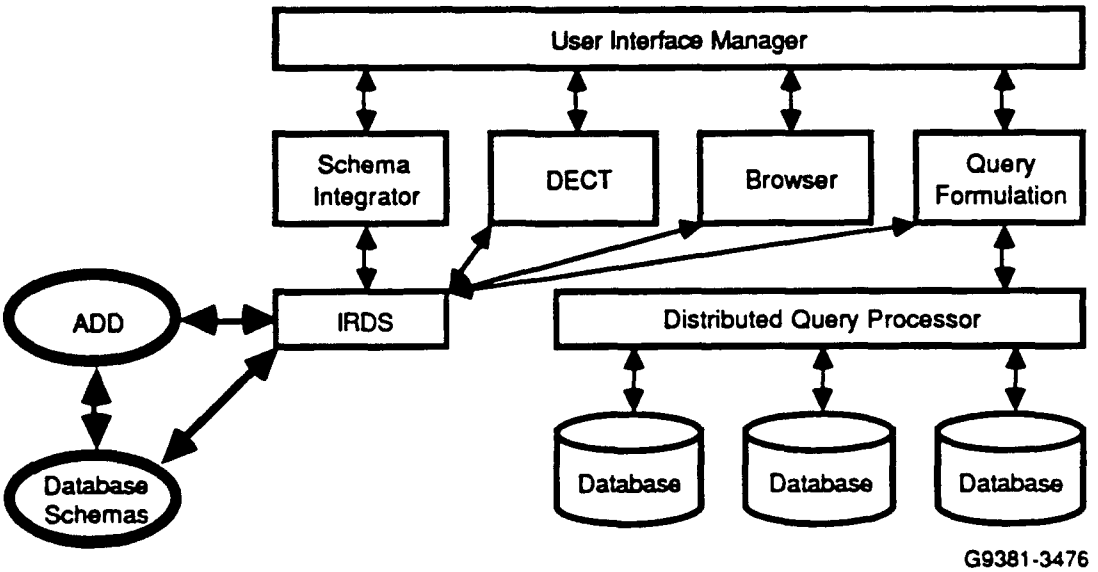


Figure 1-1. ANSWER Architecture

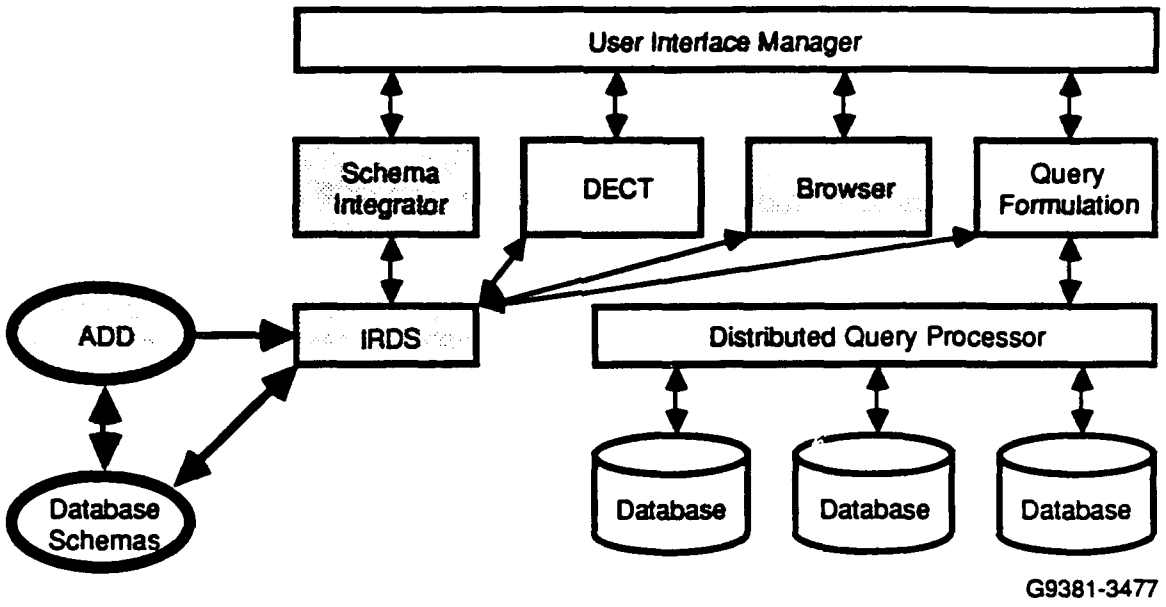


Figure 1-2 ANSWER Architecture: Phase I Deliverables

Phase II was a six month effort whose major goals were to produce a uniform interface and execution environment for all the ANSWER tools and to make modifications to the schema integrator and browser as required. The Phase II deliverables are highlighted in Figure 1-3.

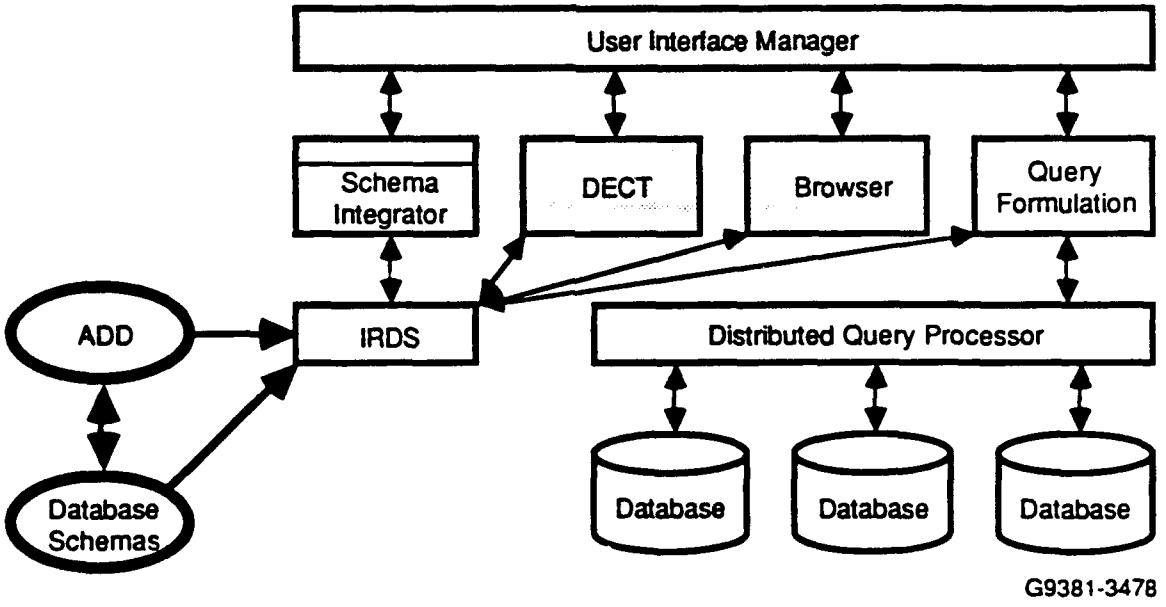


Figure 1-3. ANSWER Architecture: Phase II Deliverables

The schema integrator interface was significantly modified in Phase II. Additional functions were added to the browser capability and the browser was modified to access IRDS structures for schema information. An additional tool prototype, a data element creation tool (DECT) was built during Phase II. The DECT, based on a prototype done by MITRE, guides a user through the creation of a data element name in accordance with AR 25-9. We reimplemented the basic MITRE functionality on a different platform and developed additional functionality to demonstrate the feasibility of simple synonymy detection for data element names.

The specific Phase II tasks are described below.

1.1 User Interface

We have implemented an X-Windows-based user interface that allows the user to invoke the ANSWER tools, including the schema integrator tool, the browser, IRDS and the data element creation tool through a uniform interface. The interface allows a series of tools to be suspended and resumed with other tools being invoked in the interim through a seamless interface.

1.2 Database Registration Automation

In this task we made modifications to the schema integrator user interface. The Phase I interface was a menu-driven interface. The Phase II interface is a graphically oriented interface.

1.3 AI Technique Implementation

In this task we have developed a prototype data element creation tool (DECT) that provides automated support for creating data element names in conformance with AR 25-9.

1.4 Browsing

We added several additional functions to the browser interface. It is now possible to graphically add new nodes to an existing graph and edit some aspects of the node description for existing graph structures. The browser functions are used to support the creation of new schemas through a graphical interface.

This report discusses the results of each of the four tasks.

Section 2
User Interface Manager

The ANSWER user interface system is designed to be an extensible interface support graphical and text interaction with tools registered in the ANSWER system. In this section we present an overview of the user interface manager followed by a more detailed discussion of each component.

In constructing a user interface manager we had two major goals:

- Consistent information presentation and tool invocation facilities for end users,
- Convenient modular architecture for ANSWER tools that allows tools to be added or removed from the system.

The X-Windows environment used as the basis for the user interface manager will support tools running on a distributed hardware platform. This means not all tools need to reside on the same workstation. A network of tools can be supported that are integrated through a consistent user interface and integrated through a common data dictionary (implemented in IRDS).

2.1 Architecture Overview

The basic architecture for the user interface manager assumes that tools are described as scripts with associated interaction objects. Each script defines states of the tool that have specific I/O requirements or control requirements to which the user interface manager needs access. A script represents an abstraction of the tool that can be used by the interface manager to control:

- Presentation of tool output information to the end user,
- Collection of user input information,
- Presentation to the user of control options at various points in the tool's execution.

An example of a script for part of the data element creation tool is shown in Figure 2-1.

The tools input and output requirements are encoded as interaction objects. Each object defines the type of information being manipulated at a state and the user interface requirements for that information. Examples of interaction objects include:

- Transient menus,
- Persistent menus,

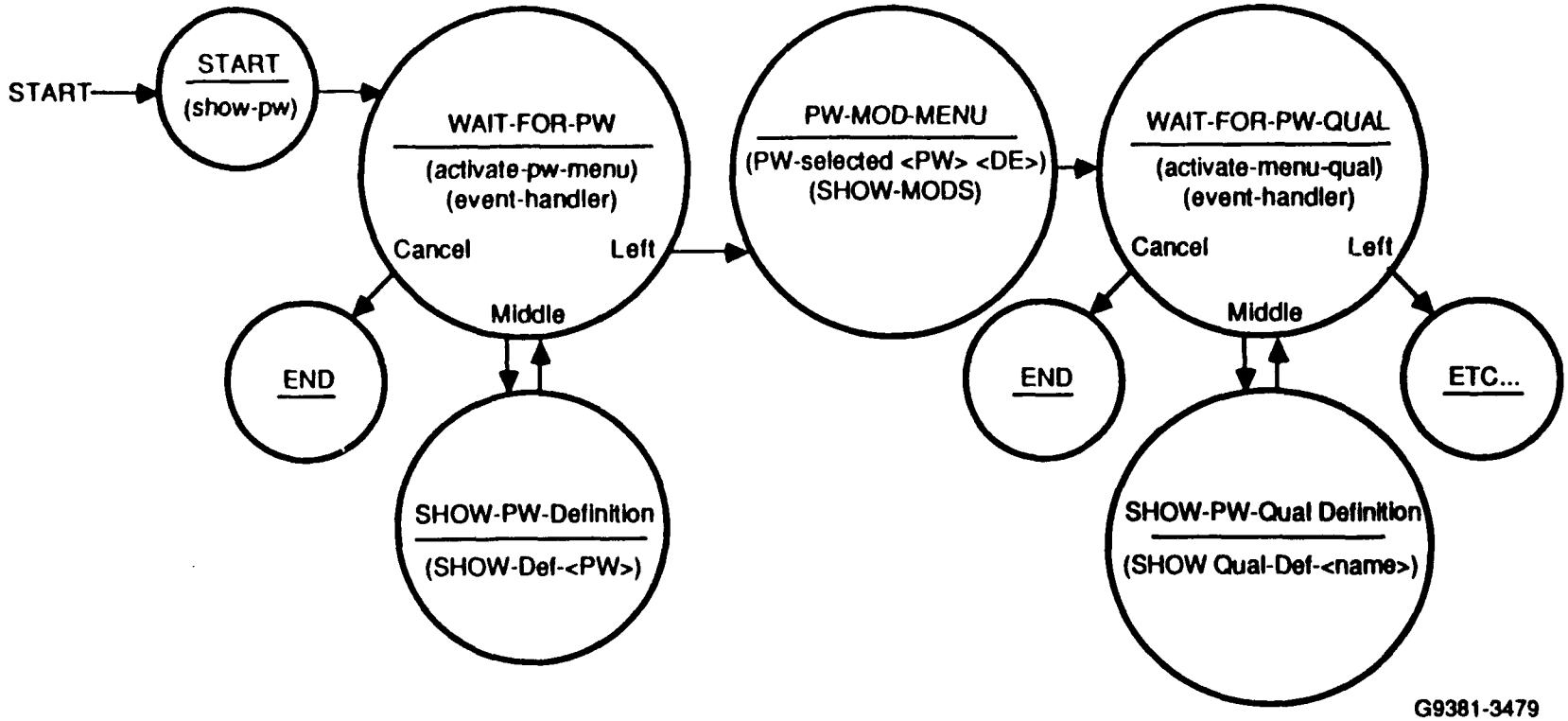


Figure 2-1. Data Element Creation Tool Script Fragment

- Graph nodes with appropriate functions,
- Radio buttons (for indicating choices in a list),
- Dialog boxes which present status messages to the user.

Transient menus present menu choices to the end user but do not remain on the screen once the choice has been selected. Transient menus are used for I/O where the I/O is meaningful only at individual states of a script. Persistent menus remain on the screen once a choice has been made. Persistent menus are used for situations where choices are meaningful throughout a number of states. Graph nodes are used extensively in the presentation of database schemas and Army Data Dictionary information. The graph nodes must support edit operations as well as display operations that may be used by tools manipulating schemas or Army Data Dictionary information.

The basic architecture for the user interface manager is shown in Figure 2-2.

The script manager tracks the current states of individual scripts. Only one script may be active at a time. All other scripts are either inactive or suspended. The scripts may be invoked using a single thread of control. This means that once a script has been invoked, it may be suspended and other scripts may be called but the resumption of any script must follow a stack protocol (i.e., last suspended, first resumed). The reason for this is the application area does not indicate a high degree of concurrency. In addition, portability to other software platforms would be compromised if the user interface manager were implemented using a multiple control thread strategy. In particular, the current version of LISP being used has some support for multiprocessing, but that support is not common across all environments.

The single thread of control strategy used by our user interface manager is the same strategy used by numerous PC- and MacIntosh-based interfaces. As each tool is invoked, it must be "closed" before processing can continue with a different tool.

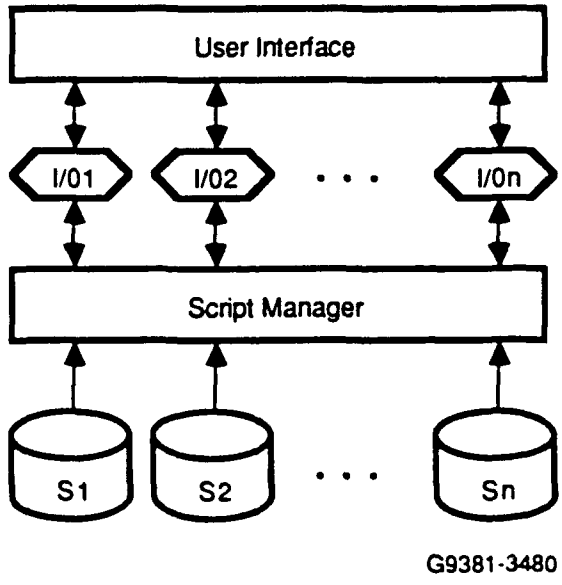


Figure 2-2. User Interface Manager Architecture

The script manager invokes interaction objects as indicated by states in the currently active script. The interaction objects read and write information to the user interface layer. The user interface layer consists of a display and associated control information for the windows, which remain on the screen throughout all possible script invocations. The user interface consists of:

- Two graphics windows.
- A set of global command buttons available during any tool invocation and a text interaction window.
- All other interaction objects display information within this user interface environment. The initial display of the user interface is shown in Figure 2-3.

The graphics windows and the global command buttons are resources shared by all tools being managed by the user interface manager. The text interaction window and other windows that appear during the course of an individual tool's execution are local to the tool.

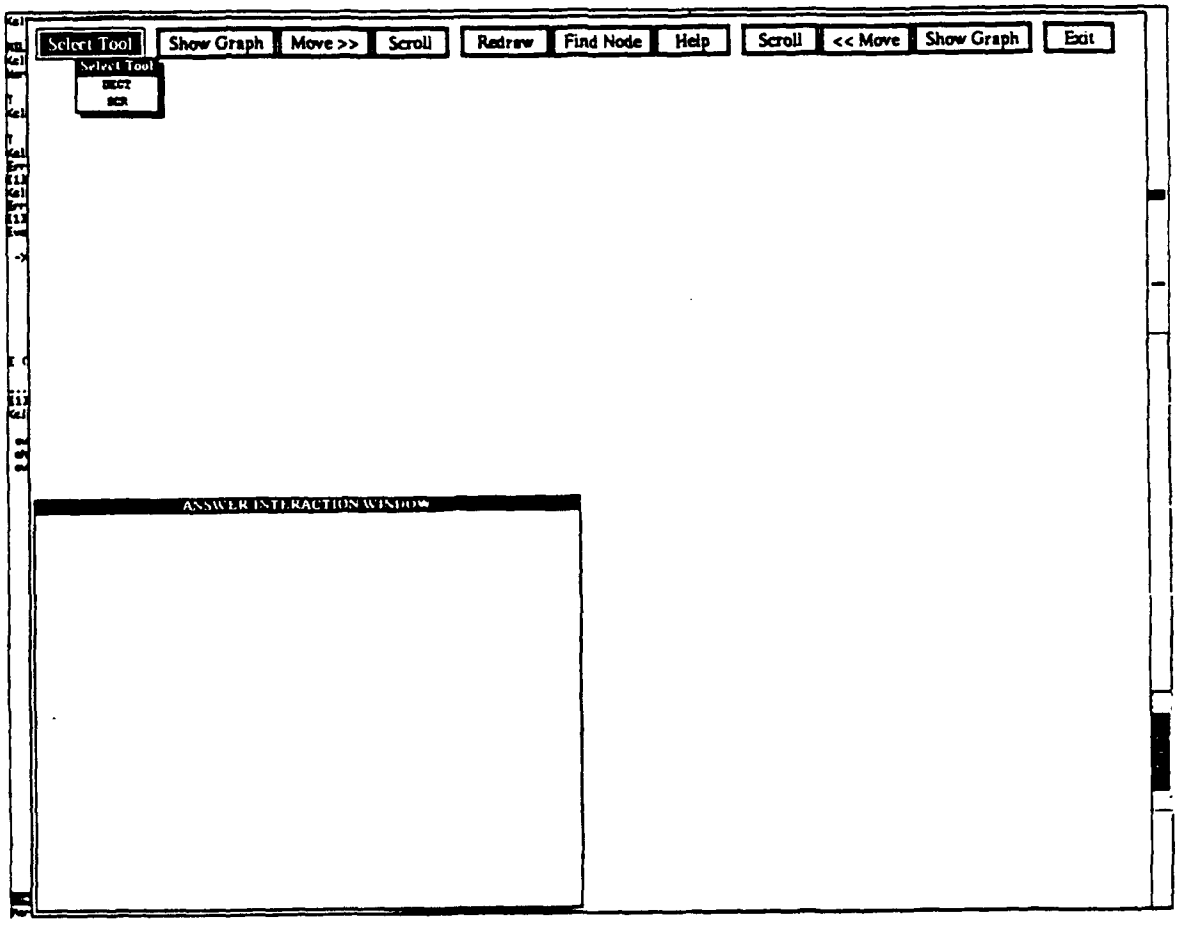


Figure 2-3. Initial Display

Each major component of the user interface manager will now be discussed in more detail.

2.2 Scripts

A script is the mechanism for describing the functionality of a tool managed by the user interface manager. The user interface manager uses script information to monitor the current state of a tool and as a guide to presenting options to the user as to which tools may be invoked at a given state.

A script describes the functionality of a tool at a level of abstraction that is dictated by the I/O requirement of the tool. A script is divided into individual states where each state includes the following information:

- Initialization information,
- Functions to be computed by that state,
- Interaction objects used by the state,
- Pointers to states reachable from the current state and state transition requirements.

A script controls tool execution by passing information from the user input to the tool associated with the script. The tool executes computation associated with the user input and typically produces some output to be passed back to the user. The script manager updates the active state as the tool computation proceeds. In our implementation, the start of each state corresponds to presenting some output information to the user. Each state ends with a request for new input from the user. Figure 2-4 shows the relationship between subfunctions of the tool and states of the script describing the tool.

States are structured this way because input points are the major points where users can switch between tool invocations. Again, this is similar to the strategy used by Macintosh style interfaces, which wait for mouse clicks or key strokes before transitioning to the next state or tool.

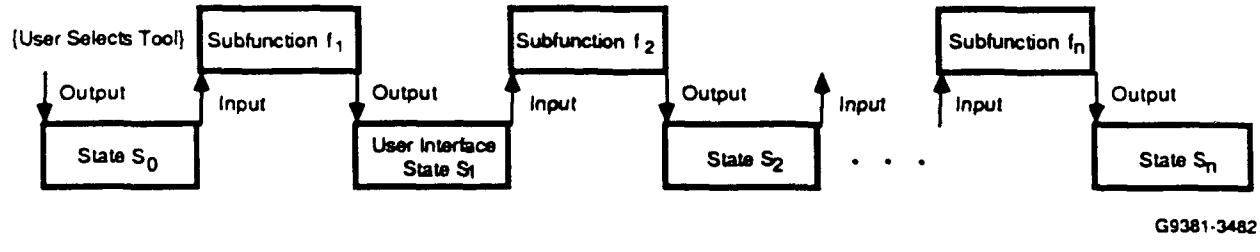


Figure 2-4. Relationship Between Subfunctions of the Tool and States of the Script Describing the Tool

A complete description of scripts for all the tools used in Phase II can be found in Appendix A.

The use of scripts as a tool abstraction mechanism allows the ANSWER system to gracefully add or remove tools from the tool set. As new tools are added, new scripts will be added. New tools create new possibilities for tool interaction not anticipated when existing tools were implemented. Since the tools are represented by scripts, modifications to the scripts will allow the user to suspend existing tools and invoke new tools at points that may not have originally supported suspend operations. Modifications to existing scripts for existing tools will allow the coherent use of additional tools not anticipated in the original tool design.

2.3 Script Manager

A script manager, which monitors the current state of tools and coordinates the execution of interaction objects used by each state uses the scripts as data structures. The script manager is an event-driven system. It continuously monitors the user interface environment. At each wait-for-input an event is trapped. The event type will either be an event to be handled by the user interface manager directly or it will be an event to be handled locally by an interaction object within an individual script.

The types of events which may occur are:

- Global events,
- Script-specific events.

Events require the following information:

- Window (location where the event occurred),
- Type (global or local to a script),
- Button type and button state (middle, left, right, ...).

Event processing is managed through a semaphore that indicates whether an event is currently being handled or whether other processing is enabled. When an event occurs, the semaphore blocks all other processing until the event has been completely handled. When the event handling finishes, other processing may resume again.

2.4 Interaction Objects

Interaction objects encode the information about the types and interface mode of user input and output. Interaction objects describe the type of data being presented or collected from the user and the form in which that information is collected or presented to the user.

Interaction objects are subdivided into two different resource types:

Blocking resources require the user to respond before the current state in a script can be exited. An example of a user interaction implemented using a blocking resource is an interaction box requesting that the user confirm a choice before allowing processing to proceed as shown in Figure 2-5.

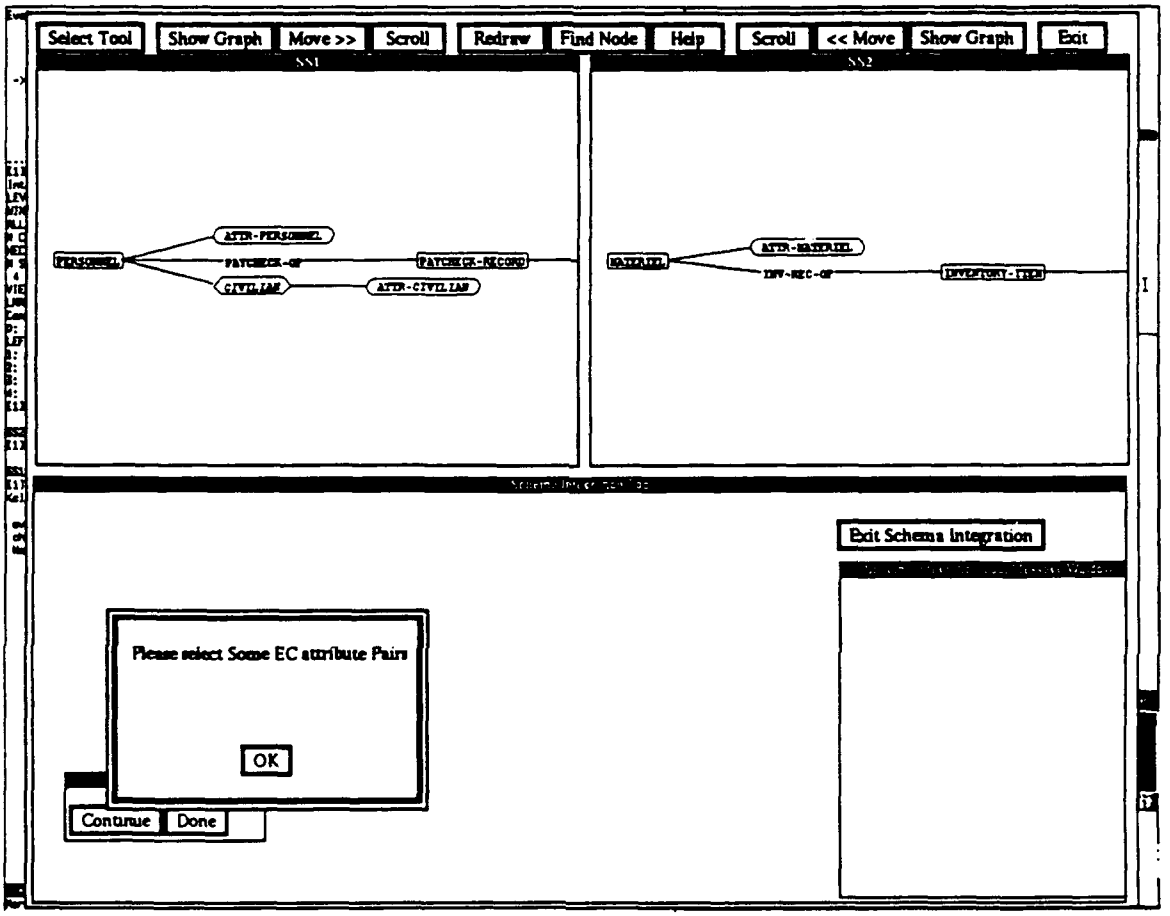


Figure 2-5. Blocking Resource Interaction

Blocking resources are used when there is critical information in the current state that cannot be saved if the script manager transitions to a new script. Blocking resources disable the choice of any global event types.

Nonblocking resources do not require that the user provide some input in response to the nonblocking interaction object before the transition to the next state. Nonblocking resources are used for options of user input that remain valid over a large number of states. An example of a nonblocking resource is an interaction that requests the user to confirm that a standard data element associated with an integrated schema attribute should be changed or remain the same, as shown in Figure 2-6.

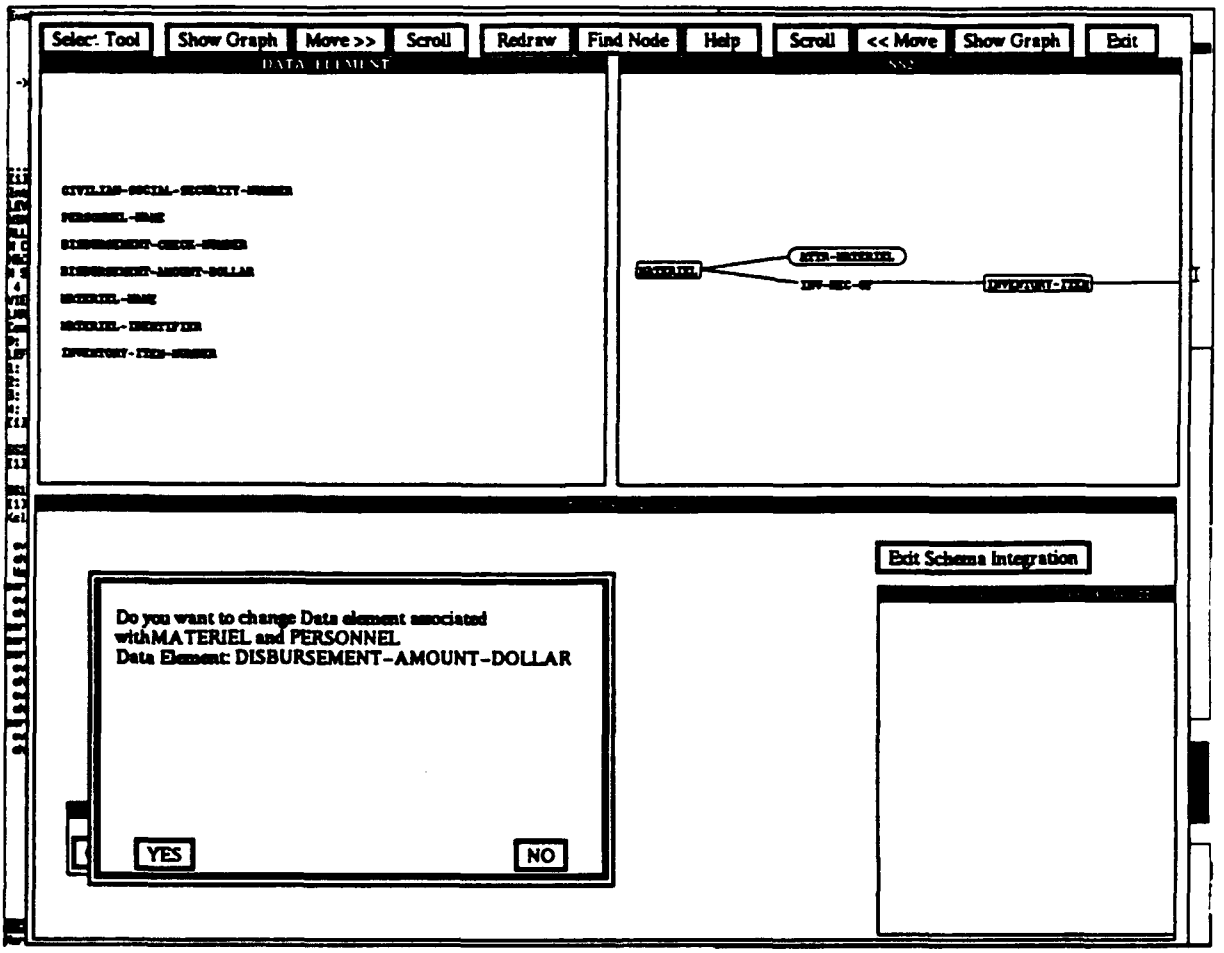


Figure 2-6. Nonblocking Resource Interaction

It is important in interactions such as this one to allow the user to invoke another tool, for example the data element creation tool. If the user decides that the standard data element needs to be changed, the data element creation tool can be invoked to support the user in defining a new standard data element. This requires the user interface to suspend execution of the schema integrator script and to begin invocation of the data element creation tool script. Once the data element creation tool finishes execution, the schema integrator script can be resumed at the current state

Interaction objects require the following information to be defined:

- Return value/call method,
- Window resource,
- Event handler specializations.

Interaction objects allow much of the interface resources to be shared between existing tools. The same type of interaction object, a dialog box for example, need be defined only once. Multiple instances of the object may be created to be used by different tools. The use of interaction objects also allows a tool's computation to be separate from the interface

requirements of the tool. Changes can be made to the interface form or style without affecting changes to the actual tool code.

2.5 Communication Subsystem Architecture

The tools whose interfaces are managed by the user interface manager must communicate with that manager by sending packets of information to the manager to be structured and manipulated by interaction objects. There are two major issues involved in the communication between tools and the user interface manager:

- The protocol to be used for interprocess communication,
- The internal structure of the data buffers being sent between the tools and the interface.

The IPC protocol is required because some of the tools in the ANSWER tool set run in C while the user interface manager and other tools run in LISP. The user interface manager communicates with the C tools through UNIX sockets. The sockets are statically allocated at the time that the ANSWER system is initially started up. The schema integrator and various IRDS data access routines make up the C-based tools that require the use of sockets. The complete list of tools running as separate processes in C is:

- IRDS-info—Retrieves information about dictionary names, schema names and schema history information from IRDS at the request of the user interface manager;
- Browser-info—Schemas are extracted from IRDS at the request of the browser;
- Schema-creation—The user interface manager sends information about interactively created schemas to IRDS for storage in IRDS;
- Schema-integration—A variety of requests for reading and writing information to and from IRDS in support of the execution of the schema integrator tool;
- IRDS—Gets concurrency status and establishes a socket connection for direct IRDS access.

The internal representation of data in the tools is not the same as the representation of that data as required by the user interface manager. Each communication between a C process and the user interface manager requires a buffer with information structured for that communication type. For example, schema information that will be sent from IRDS to the user interface manager for user by the browsing functions will have a buffer structured as shown in Figure 2-7.

Each communication buffer includes a number at the beginning of the message indicating the total number of buffers and an EOT at the end of the buffer indicating the end of the

buffer. The structure of the buffer depends on the type of communication. For the example in Figure 2-7, the buffer structure is organized into information about entities and categories (the OFLIST), information about attributes (the AFLIST), and information about relationships (the RFLIST). This particular structure reflects the order in which the information is retrieved from IRDS. A complete description of the structure of all communication buffers is given in Appendix B.

The communication architecture requires only standard UNIX as a software environment. It is the critical underpinning allowing some tools to be running in C and some tools to be running in LISP. It is also the critical component that allows for concurrent execution of multiple tools.

```
TOTAL # OF _BUFFER  ((BROWSER_SCHEMA_INFO <Schema_name>
                    (OFLIST (<ename> <object_type> <purd>) . . . )
                    (AFLIST ( <ename> < attr_name> <attr_type> <key> <sde>)
                    (RFLIST ( <rname> <ename1> <ename2> <obj1_maxcard>
                              <obj2_maxcard> <obj1_mincard> <obj2_mincard>
                              <role1> <role2> <type1> <type2>) . . . )
                    ) EOT)
```

G9381-3483

Figure 2-7. Browser Buffer Structure

Section 3
Database Registration Automation

The database registration task has focussed on additional development of the schema integrator prototype originally developed in Phase I. The schema integrator automates the production of integrated schemas from unintegrated individual schemas. The schema integrator automatically reads schemas from IRDS and interacts with a user to collect information about equivalence relationships between individual entities and attributes. Based on the information collected interactively, the schema integrator synthesizes a complete integrated description of the input schemas. The integrated schema may include additional entities not present in the original schema as a result of the generalizations discovered during the integration process.

The schema integrator will support the creation of integrated views of data contained in databases and flat files maintained at installations throughout the Army. The schema integrator will make it possible to:

- Add new types of information without disrupting data integrity or existing application functionality;
- Integrate "legacy" systems with new systems;
- Support the retirement or redevelopment of "legacy" systems;
- Create integrated views of data that will support distributed query access.

The results in Phase I provided basic schema integrator functionality in a prototype that used a menu-driven interface based on the CURSES window environment. The major goal in Phase II was to revise the schema integrator architecture and provide a graphical interface for the schema integrator consistent with the user interface manager architecture in an X-Windows environment. This section of the final report discusses the development of the graphical user interface for the schema integrator and its integration with the user interface manager.

3.1 Schema Integrator Script

To integrate the schema integrator under the user interface manager, it was necessary to define interaction objects and a script for the schema integrator. The script will be discussed in this section. The interaction objects are discussed in Subsection 3.2. The major steps of schema integrator processing are shown in Figure 3-1.

The schema integrator collects assertions from a user about the equivalence of attributes and uses that information to guide the collection of information about the equivalence of entities, relationships and categories. The possible assertions include equality, inequality, subset, superset, disjoint and integrable, disjoint and not integrable. Once all local

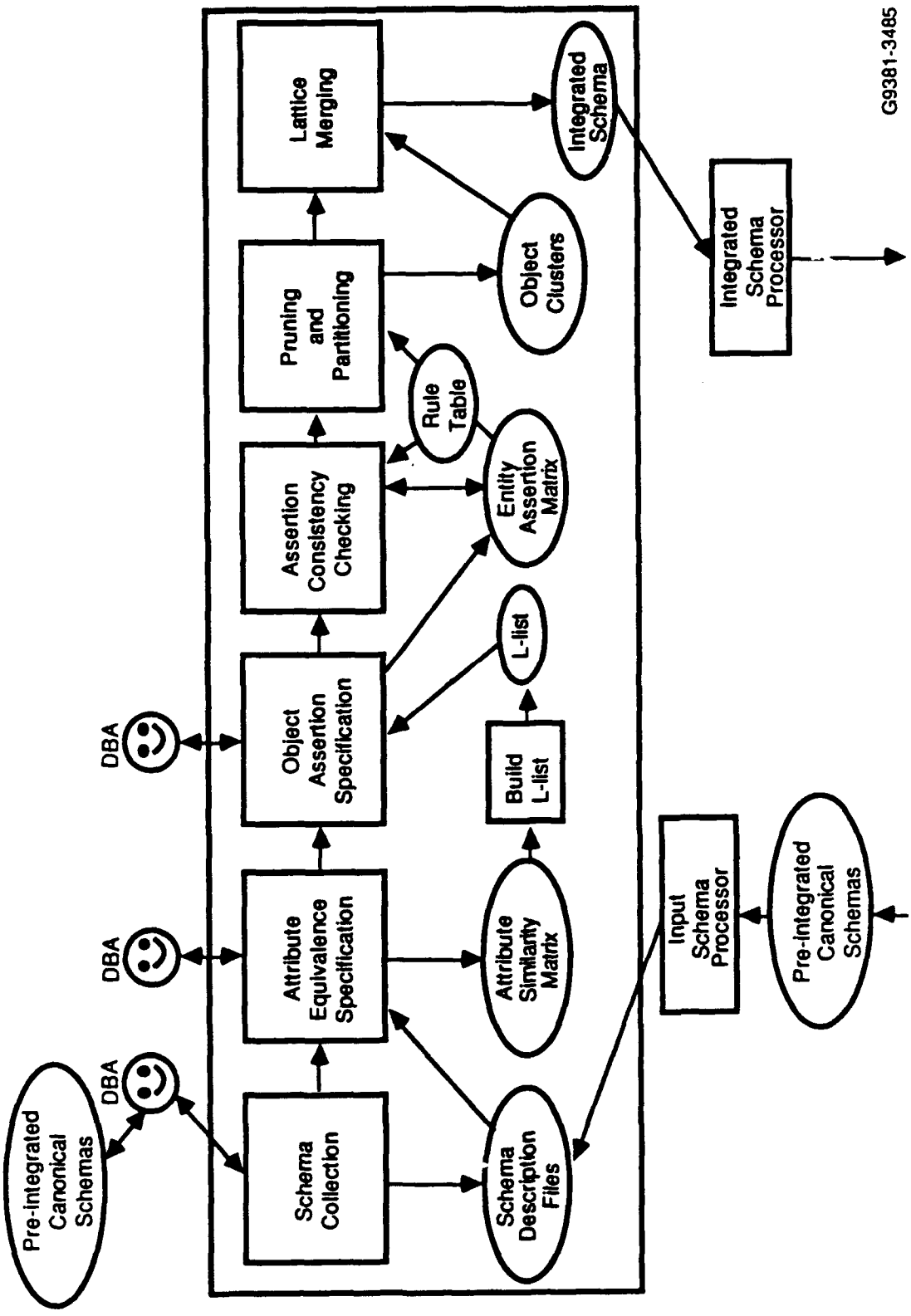


Figure 3-1. Schema Integrator Architecture

assertions have been collected, the schema integrator derives additional assertions, which are logically implied by the original assertion set, and checks for consistency of the assertion set. For example, it is not possible for two objects A and B to be both equal and disjoint at the same time.

Once all assertions have been collected, derived and checked for consistency, the schema integrator identifies distinct equivalence classes for integration. Each cluster is integrated independently. Once each cluster has been independently integrated the lattice merging function takes each cluster and generates a lattice based on subset-superset relationships between objects. Only one of a pair of objects asserted to be equal will occur in the final lattice. This completes the integration process. The user is free to view the results and commit them to storage in IRDS if the results are satisfactory.

The schema integrator script follows the breakdown of processing represented in Figure 3-1. The script is represented at a much finer level of detail to track individual user interactions and support suspension and resumption of the schema integrator at various points in the processing.

The schema integrator script is broken into a number of subscripts following the structure of choices implied by the original menu-based interaction. The top-level schema integrator script is shown in Figure 3-2.

Each node in the script is actually the name of a subscript. Several nodes in this figure have more than one possible entrance or exit. For example, the node RelAttrSelection may be invoked from either the TopLevelMenu state directly or it may be invoked after the completion of the ECAssertion state. The final state in this script, CommitOrExit may be exited by either taking the commit arc to a final state or taking an arc back to the TopLevelMenu state.

Each one of the nodes in Figure 3-2 actually names a subscript with more nodes. For example the subscript associated with ECAttributeSelection is shown in Figure 3-3.

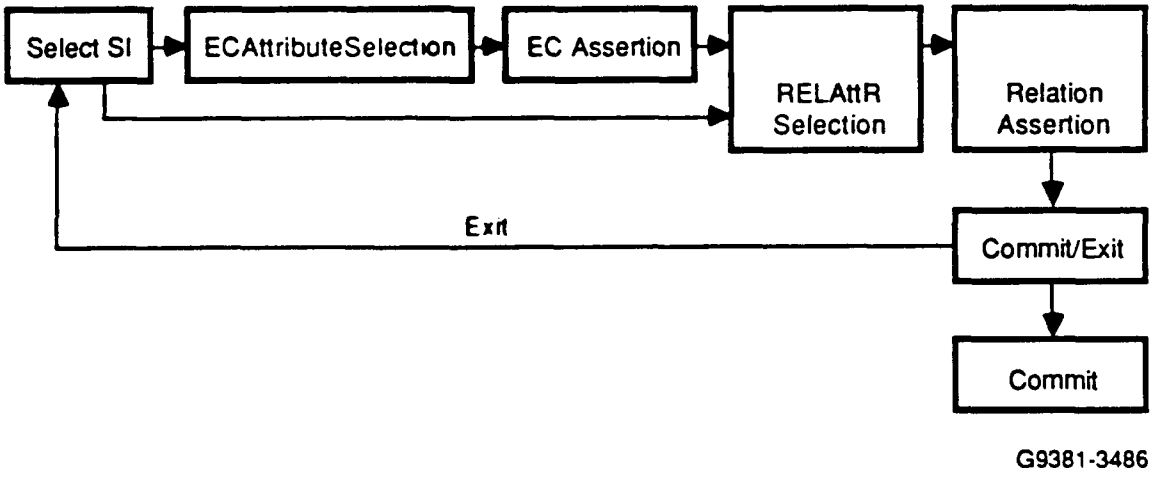


Figure 3-2. Schema Integrator Script

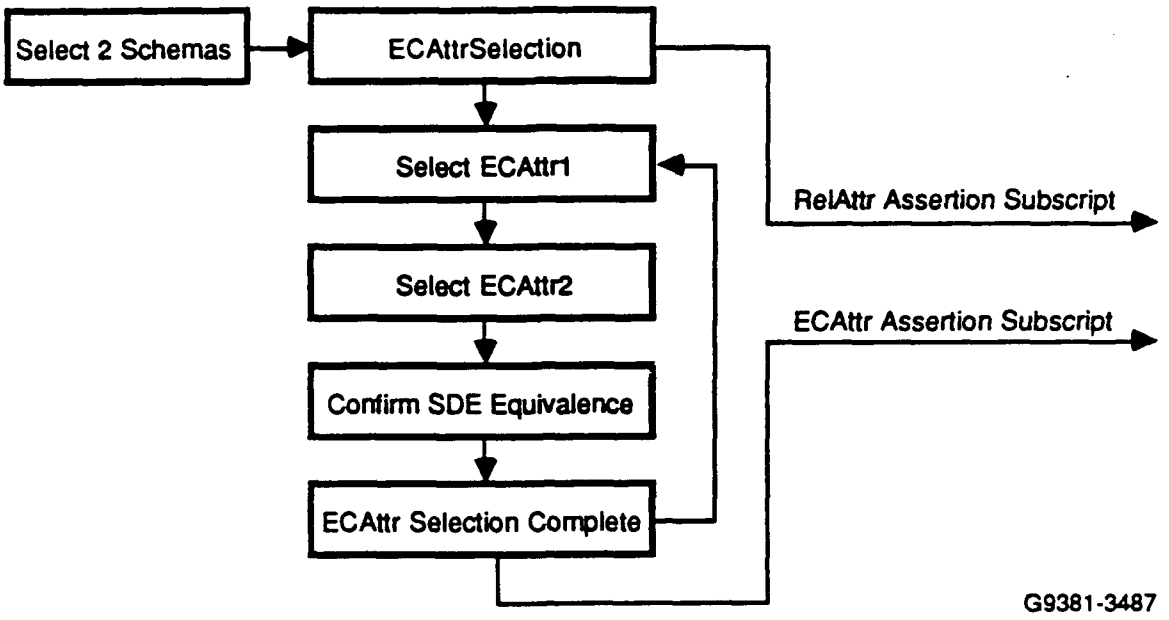


Figure 3-3. ECAttributeSelection Script

There are two possible exists from ECAttributeSelection: one to SelectECAttribute1 and one to the RelAttrSelection subscript. Each exit from the state is associated with an event, in particular a mouse click on a button specified in the state. The SelectECAttribute1 unconditionally transactions to SelectECAttribute2 and to the state ConfirmEquivalence. The state ConfirmEquivalence has two possible exits; one to ECAttrSelectCompleted and one to a different tool, the DECT tool, which creates new standard data elements.

The method of representing a tool's processing as a script allows the insertion of suspension points such as in this example at any point in the development of the script that explicitly offers the choice of invoking a specified tool. As new tools are developed, new suspensions points may be introduced into an existing script to allow controlled use of multiple tools in the same environment. In this case, if the user selects the DECT option in the script, the schema integrator processing will be suspended and the DECT will be invoked. The user may then use the DECT to create a new standard data element to associate with the schema currently being examined by the schema integrator.

The script allows the tool interactions to be easily configured and reconfigured as new tools are developed or old tools are removed from the ANSWER tool set. This is a crucial element of the ANSWER tool environment support for data administration and query support.

This explicit inclusion of points where other tools can be invoked should not be confused with the facility to invoke other tools at any point where the current interaction object is a non-blocking resource. At such points the user may also elect to suspend processing and invoke another tool.

3.2 Schema Integrator Interaction Objects

The schema integrator uses both blocking and nonblocking resources. Blocking resources do not allow any other processing to occur until the current interaction is completed. Non-blocking resources allow other interactions even if the interaction for the nonblocking interaction object has not yet been completed.

In the example script fragment in Figure 3-1, both blocking and non-blocking resources are used. Blocking resources are used for dialog boxes at ECAAttrSelection . Nonblocking resources are used for interaction objects at the ECAttrSelection state, and the ConfirmEquivalenceSDE state. and the states SelectECAtrribute1 and SelectECAtribute2.

At the SelectECAttribute1 and SelectECAttribute2 states, the fact that the interaction is nonblocking allows the user to browse through the schemas before selecting attributes to be equivalenced. This is an important facility since the user may view arbitrary parts of the schemas before determining the correct relationship between two attributes.

At the state ConfirmEquivalence/SDE, the user may determine from examination that the attributes being displayed do not have the correct standard data element associated with them. It is possible that a standard data element created in the context of an individual schema may no longer be the appropriate standard data element for the new integrated schema. The data administrator may want to change the standard data element by changing the modifiers of the prime word or the modifier of the class word to more accurately reflect the new semantics of the standard data element in an integrated context. The ANSWER tool set will allow the user to invoke the DECT tool or other tools as appropriate to make the best determination on the correctness of the attribute equivalence assertion and the associated standard data elements.

Section 4
Browser

The Phase I browser functionality has been used as the basis of the development of the User Interface Manager. The Phase I browser functionality has also been expanded in Phase II to provide support for:

- Adding nodes,
- Deleting nodes,
- Editing node definitions,
- Integration of browser with IRDS.
- Integration of the browser with the user interface manager.

The Phase I browser functionality included the ability to display and manipulate graph structure as well as managing text and mouse interaction facilities in a single environment. The initial browser user interface from Phase I is shown in Figure 4-1.

The Phase II browser has maintained the same basic user interface with (1) two fixed location graphics windows, (2) the ability to scroll, index nodes by name, and create new links between nodes. All of the commands that were originally part of the browser functionality have been included as global commands in the new user interface manager.

Additional facilities that have been added to the browser are actually represented as global commands available through the user interface. Each command may be individually suppressed depending on the current state of the user interface manager. For example, in some cases the user should not be allowed to edit nodes in a schema. That operation, which will typically be globally available, will be suppressed at that time. For example, editing a node in a schema currently being used as input for schema integration while schema integration is active cannot be allowed.

The editing and add node operations, which are new in Phase II allow the user to manually create a new schema if desired or to edit characteristics of existing nodes in a schema. For example, the definition of an attribute or the data type of an attribute may be changed by the edit node facility.

The browser is also fully integrated with IRDS in Phase II. This means that the user may request schemas to be read in from IRDS into local memory for manipulation by the browsing commands. The user may also create new schemas and commit those schemas to IRDS for storage.

The interface for the browser in Phase II is the same as the interface for the user interface manager, as shown in Figure 4-2.

The functionality of the browser is essentially a subset of the functions supported by the user interface manager. The browser functions are not selected by first choosing a tool

using the select tool command. Instead the browser functions are available as the buttons across the top of the screen for moving, finding, and scrolling, as well as functions available by selecting a node in the graphical display with a mouse click.

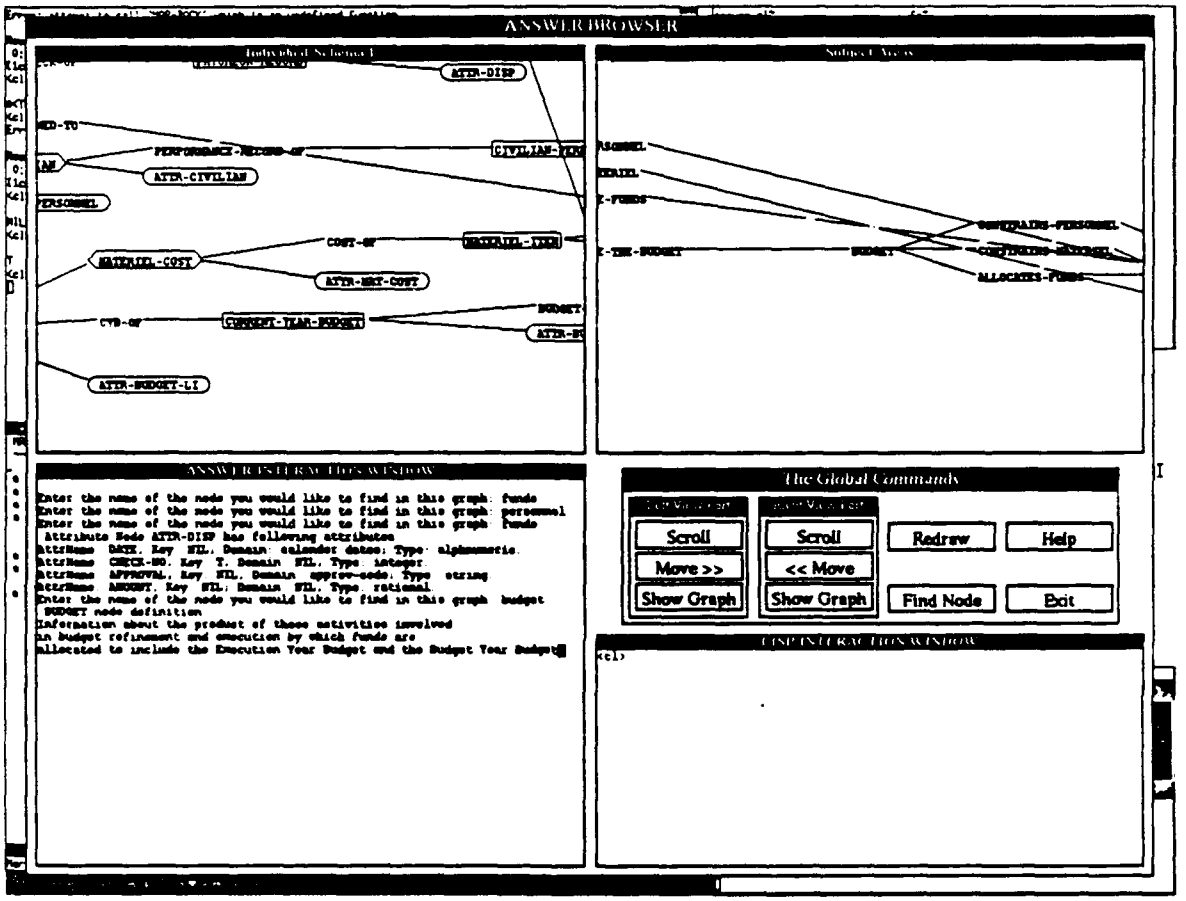


Figure 4-1. Phase I Browser Interface

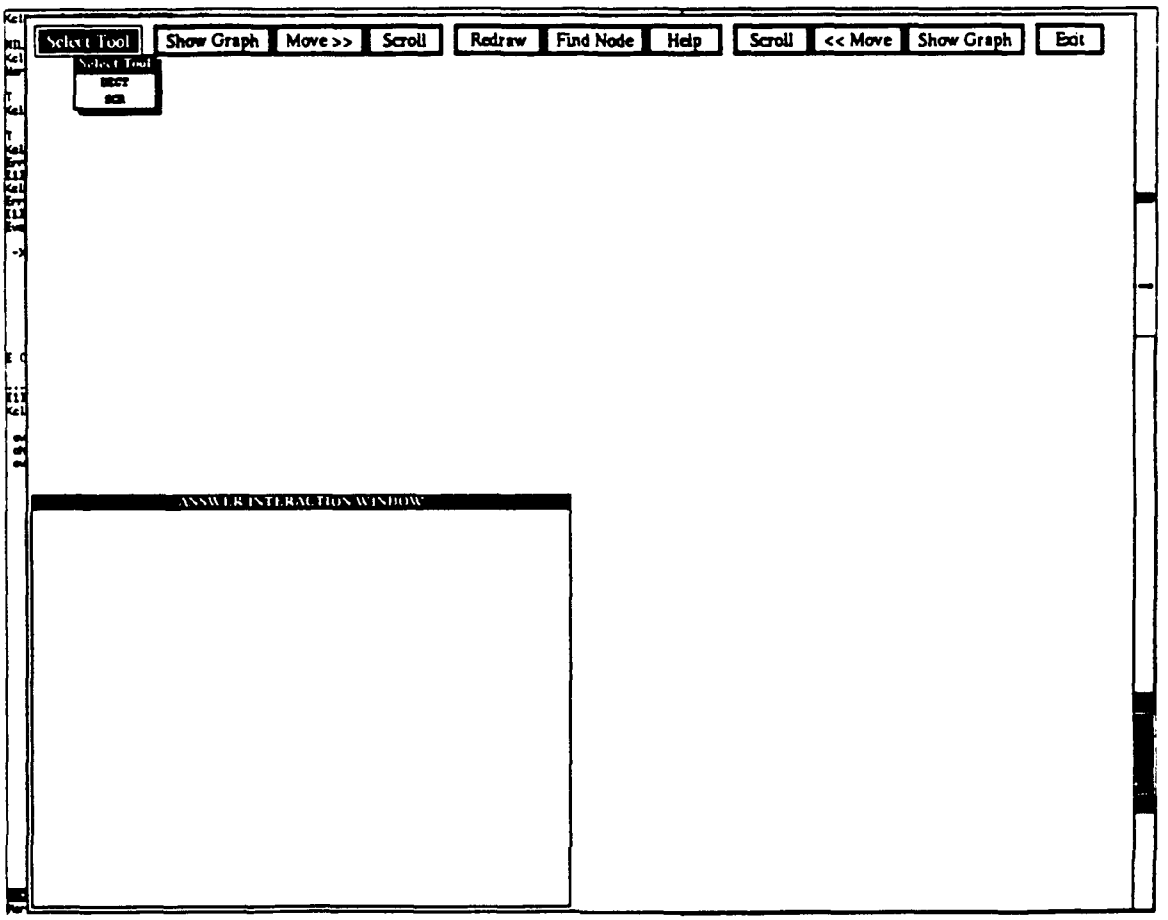


Figure 4-2. Browser Interface

Section 5
Data Element Creation Tool

Data element names must be syntactically and semantically correct to be useful for data administration. Syntactic correctness for data element names in ANSWER is specified according to AR 25-9. Semantic correctness will be ultimately insured through the Army's approval process of individual data elements. Initial checks for semantic correctness can be done prior to the final official approval by users preparing data element names for approval.

The data element creation tool (DECT) provides interactive assistance to a user to create standard data element names in accordance with AR 25-9. The tool is an aid to automatic enforcement of AR 25-9 by only allowing data element names to be created which follow the approved syntax specified by AR 25-9.

DECT also provides assistance by leading the user through a series of questions as a data element name is created to insure that the intended semantic distinctions are being captured by the new data element name. The questions allow the user to implicitly or explicitly traverse a structure that represents a semantic model of the prime words and class words as used in AR 25-9.

DECT also provides additional aid to the user by identifying data element names similar to the new name being created. The user may choose to use a similar existing name in place of a newly created name if the similar name accurately describes the data element in question. For example, if the new standard data element name developed by the user were "materiel purchase date," an existing standard data element name of "acquisition purchase" might be a more appropriate choice. The DECT would aid the user in identifying the existence of the name "acquisition purchase" by limiting the amount of search the user would need to perform.

The remainder of Section 5 discusses the architecture of the DECT and the integration of the DECT with the user interface manager.

5.1 DECT Architecture

The DECT maintains separate models of the prime words and class words used to guide the question answering process for creating a standard data element and for classifying a standard data elements to identify synonymous elements. Enhanced with additional nodes to improve the semantic discrimination capability, the models are structured following the Army Data Architecture. For example, in the Class Word model, the class word 'length' is related to the class word 'area' since they both measure spatial extent. Additional nodes can be added to the hierarchies to improve the discriminating capability of the models.

Each model is structured as a hierarchy with the following information contained in each node:

- A short description of the node's contents,
- A long definition following official Army definitions where available,
- An optional list of contrasts with other nodes at the same level (e.g., distinguishing characteristics of the subject areas Materiel and Acquisition),
- A list of data elements which have been classified under the node.

The model has been constructed for use with the prototype. Enhancements to the model will directly affect the effectiveness of the model in distinguishing semantic characteristics of various standard data element names. Over time, the model could be refined to specific requirements reflecting the context of use of the DECT. For example, to distinguish standard data element names, a data administrator at the installation level may have different needs than the needs of a data administrator at the ODISC4 level.

Portions of each model are shown in Figure 5-1 and Figure 5-2.

Definitions of individual words are maintained through pointers to nodes in the hierarchy. Some words will relate to more than one area of a hierarchy. For example, the prime word 'materiel' will be associated with the area of the prime word hierarchy dealing with material and with the area dealing with 'acquisition' since the definition of 'acquisition' as a subject area includes the acquisition of materiel. The definitions of prime words, expressed in terms of the relationship of the prime word to this hierarchy, will aid the user in identifying the best possible choice of a prime word.

5.2 DECT Processing

The DECT will aid the user in creating a new standard data element, generating a definition for that data element, and checking to see if any synonymous data elements have already been generated.

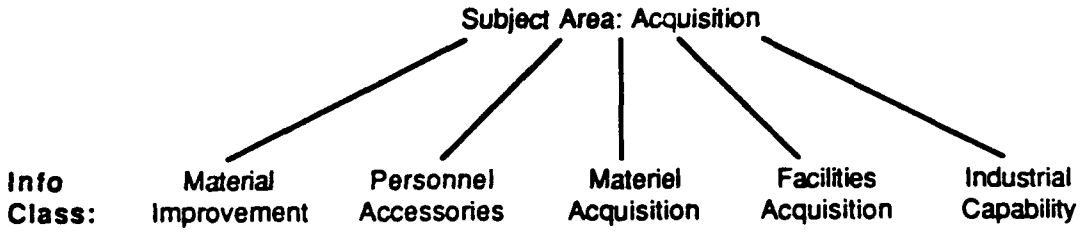


Figure 5-1. Prime Word Hierarchy

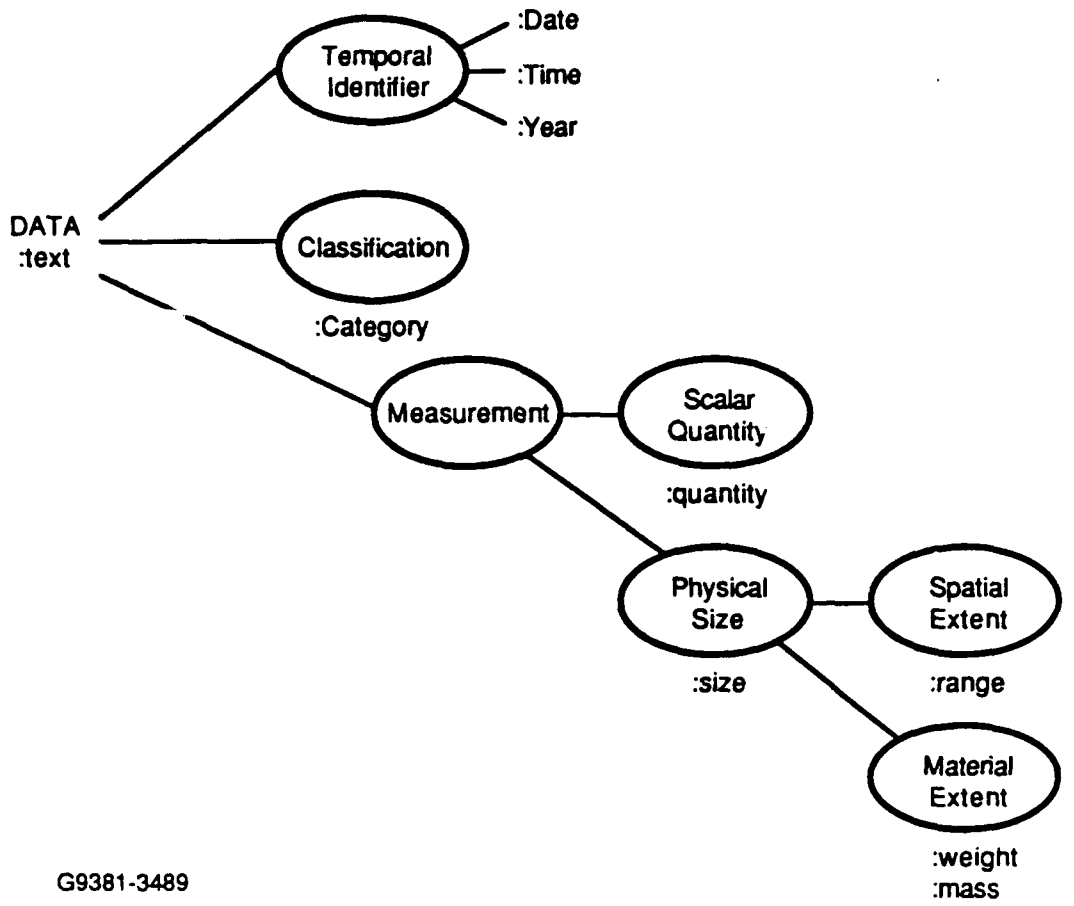


Figure 5-2. Class Word Hierarchy

The four components of the data element (DE) name are entered through menu-based interactions (this is similar to the protocol used in the MITRE software). The system asks the user questions about the DE to help classify it in each of two semantically-based indexes, one for class words and the other for prime words. This is how the system organizes its knowledge about data elements and keeps track of semantically similar data elements. When necessary, the system suggests an alternative prime word or class word that it judges more appropriate based on the user's answers to the system's questions. The user can rename the DE, if desired. The system suggests potentially similar DEs (using the semantic indexes). The user can select one of these previously defined DEs or stay with the new one. The DE definition is generated based on the answers to questions and context-sensitive synonyms.

Prime and class words point to nodes in their respective indexes. Words such as "materiel" are easily misused since there is more than one subject area that deals with materiel (e.g., Acquisition and Materiel). These words will have multiple pointers, referencing nodes in different parts of the hierarchy. Such words will cause the system to begin asking the user questions that will determine the proper classification of the DE in the indexes, identifying which use of the word was intended. The possible answers to these questions will be

supplied in menus built from the short descriptions. At any point the user can ask to see the long description of any node that is currently an option. The user can also ask to see all the nodes at the current decision point, to move up one level in the hierarchy, or to supply a different prime or class word.

Once the user has settled on an index node as the proper classification, the system will use that node in the following ways: it will suggest a more appropriate prime or class word if one exists as determined by the node and its ancestor nodes; it will use the DEs stored in the chosen node and closely related nodes as possibly synonymous DEs to present to the user. It will use the node in choosing synonyms for use in the data element definition.

5.3 Definition Generation

The DECT generates definitions for standard data elements once they have been created. Generating appropriate definitions requires an algorithm to generate appropriate syntax for definitions and a considerable amount of information about the semantics of the words being used in the definition to avoid odd or bizarre sounding definitions. For example, it is sufficient syntactically to say that the definition must consist of some descriptive noun phrase followed by a prepositional phrase of the form <<OF/FOR a Prime Word (or Prime Word synonym). Consider for example, definitions generated by the MITRE prototype for definition generation:

- SDE: military-personnel-authorized-leave-category,
  DEF: permitted absence class of a military person,

- SDE: military personnel-proficiency-pay-category,
  DEF: ability wage class of a military person,

- SDE: military-personnel-service-category,
  DEF: duty class of a military person,

- SDE: military-personnel-separation-category,
  DEF: withdrawal class of a military person.

The definitions all follow the general form of << (synonym for class word modifier) (synonym for class word) OF A (synonym for architectural modifier) (synonym for prime word)>>. For example: <permitted absence> is a synonym for the class word modifiers <authorized leave>; <class> is a synonym for <category>; <military> is the architectural modifier; <person>is a synonym for the prime word <personnel>.

This approach to definition generation will always generate a syntactically well formed string (i.e., a noun phrase followed by a prepositional phrase) but the simple one for one substitution of synonyms for modifiers, prime words and class words will result in semantically odd definitions in many cases. For example, the following definitions are also generated by the MITRE prototype:

- SDE: personnel-birth-location,
  DEF: delivery site of a person,

- SDE: personnel-patient-code,
  DEF: patient symbols of a person.

In the first definition the use of the term <site> is pragmatically odd when speaking of an individual's birthplace. In the second example, the definition does not convey any additional meaning to the standard data element. In that case, the definition should convey something about the intended range of code values.

The approach used for definition generation in DECT attempts to ameliorate some of these problems by employing a more highly structured representation of the synonyms for modifiers. As with the class word and prime word hierarchies this represents a first cut and can be enhanced as the tool is used more. The DECT uses a modifier dictionary to guide the definition generation process.

The modifier dictionary contains the context sensitive synonyms for DE modifiers. By context sensitive we mean that a different synonym can be used for a modifier depending on the surrounding words and on the Prime Word and Class Word Index nodes. For example, in the data element PERSONNEL-SECURITY-CLEARANCE-CHECK-CATEGORY we might want VERIFICATION as a synonym of CHECK. This same synonym would be inappropriate for the data element DISBURSEMENT-CHECK-NUMBER. Context sensitive synonyms allow this distinction to be made by stating that VERIFICATION should be used when SECURITY-CLEARANCE precedes CHECK, and that some other synonym (or perhaps CHECK itself) should be used when the prime and class words are DISBURSEMENT and NUMBER, respectively.

Consider some additional examples of standard data elements where it would be important to be able to store context sensitive synonyms for modifiers:

- personnel-social-SECURITY-number-code.

  Note: Security here should be treated as part of the single word social security number and that word should have synonyms and not security by itself.

- facility-SECURITY-unit.

  Note: Here the best synonym for security might be something like guard or defense or police.

- personnel-loan-SECURITY-code.

  Note: Here the best synonym for security might be something like collateral.

It is clear from the examples, that a context-sensitive definition is important.

5.4 DECT Integration with the User Interface Manager

The DECT is integrated with the user interface manager. The user can invoke the DECT at any point allowed by the user interface manager. To be integrated with the user interface manager, the DECT has its own script that describes the possible states of the DECT. Figure 5-3 shows a portion of the script for the DECT.

The fragment of script shown for the DECT controls the initial selection of prime words and qualifiers for those prime words. Each state in the DECT script specifies computation to be performed at that state and events that must occur to transition to a new state. Not all transitions are governed by external events. Some states support unconditional transition from one state to the next.

The first state, WAIT-FOR-PW, controls the display of prime words and prime word definitions and the selection of a prime word for the data element name being created. Two different functions are called from this state, one to create and display the menu and one to return control to the event handler to wait for a mouse click. There are three possible exits from this state, based on three events: CANCEL, MIDDLE-BUTTON, and LEFT-BUTTON. The events are all mouse button selections. CANCEL will allow transition to an END state. MIDDLE-BUTTON will allow transition to a state; SHOW-PW-DEFINITION1. That state displays the definition of a prime word in isolation. The only transition from that state is back to the originating state, WAIT-FOR-PW. LEFT-BUTTON is the event for the third possible transition from WAIT-FOR-PW to the state PW-QUAL-MENU. The state PW-QUAL-MENU supports display of the qualifiers for the prime word that has been selected. It also stored the results of the prime word selection by calling the function (pw-selected <pw><de>) with the parameters <pw> and <de> appropriately bound based on the previous state. It supports an unconditional transition to the state WAIT-FOR-PW-QUAL, which again has transitions governed by mouse click events.

5.5 DECT Support of AR 25-9

The DECT can be used to support the enforcement of data element naming standards as outlined in AR 25-9. The tool can be used at the installation level to create new candidate standard elements, and used at higher levels to verify the candidate or reject it as a standard element. At the installation level, a data administrator may use the tool to create a syntactically well formed name that is not ambiguous or synonymous with any other data element names known at that installation. Higher levels of authority may use a version of the DECT as an aid to verifying the absence of synonymous elements and to determine that the candidate standard element is not ambiguous. The DECT is an important tool to support AR 25-9.

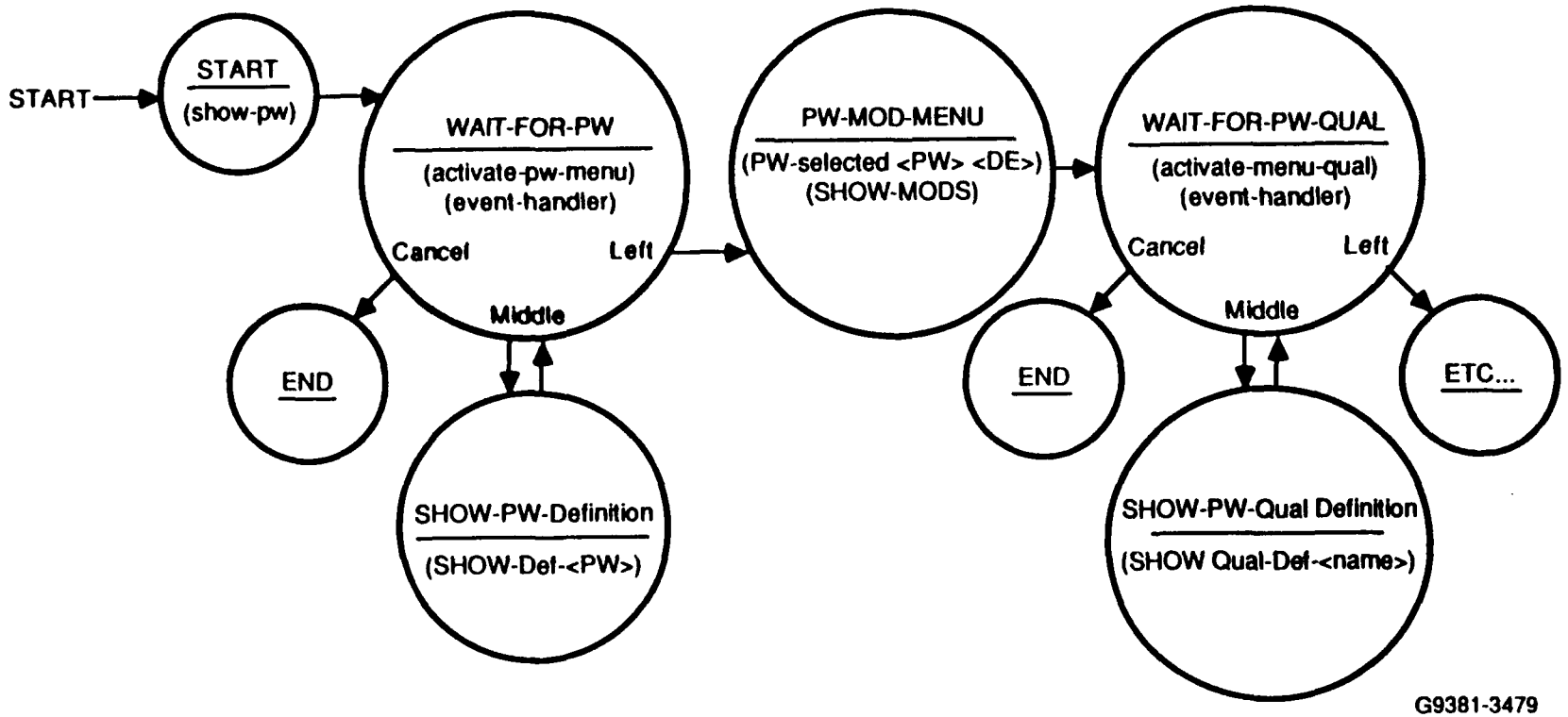


Figure 5-3. Data Element Creation Tool Script Fragment

Section 6
Scenario for Phase II Tool Use

Data has a distinct life cycle from creation and implementation through possible integration and standardization. A rough representation of the life cycle of data is shown in Figure 6-1.

This life cycle follows the system development life cycle as described in DODD 7920.1 and presented in AR 25-9, with the exception that the design phase may be paralleled by an integration phase. Depending on the state of the data (i.e., already present in an existing system or about to be introduced in a new system) the life cycle may follow one or both of the design and integration paths. In fact, in an environment where data from existing systems is being integrated with other existing systems and new systems, the life cycle may iterate between the design and integration phase a number of times.

As discussed in AR 25-9, the life cycle for standard data element development parallels the system design life cycle. If the data life cycle presented in Figure 6-1 is matched against the standard data element development life cycle following AR 25-9 the result is as shown in Figure 6-2.

In this case, the element requirements definition phase of standard data element development overlaps with both the design and integration of data. The ANSWER tools are designed to support both the schema design and integration phases and the standard data element requirements definition phases.

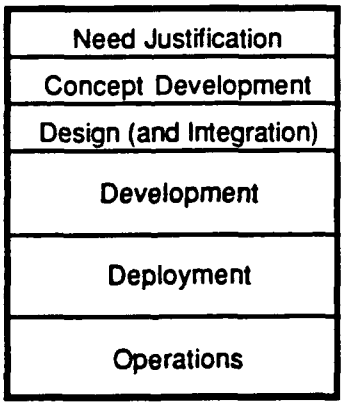


Figure 6-1. Data Life Cycle

There are a number of individuals who must be involved in developing and maintaining integrated views of information and data element standards. These include:

- Requirements developers,
- Organizational data administrators,
- Database administrators,

- Information class proponents,
- Army data encyclopedia administrator,
- Army data manager,
- End users.

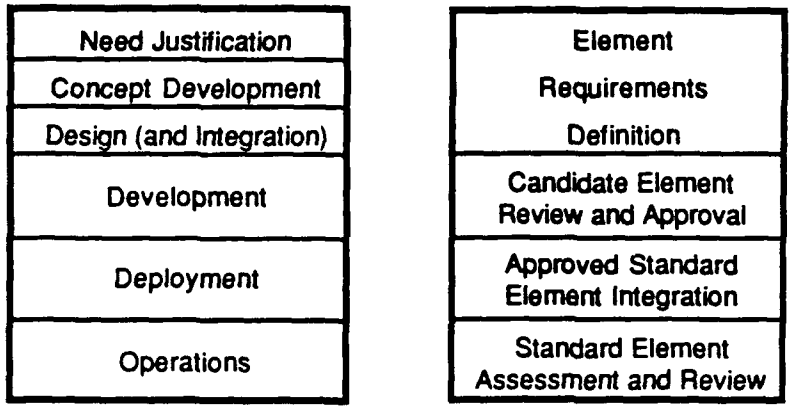


Figure 6-2. Standard Data Element Life Cycle Matched with Data Life Cycle

Each of these classes of individuals may use some of the ANSWER tools at various points in the life cycle of data.

At the point of schema design and integration, requirements developers, organization data administrators and database administrators must work together to produce a schema design which correctly reflects identified needs and in which appropriate data elements have been assigned standard data element names.

The schema integrator and browser support the process of initial schema design and schema integration. A schema may be manually created through the the ANSWER tool set graphical interface and committed to IRDS. Existing schemas stored in IRDS may be read into the ANSWER tool environment and the database administrator may use the schema integrator to develop an integrated view of the existing schemas. The integrated view can serve as the basis of distributed query processing against those existing systems.

As integrated schemas are developed, requirements developers must identify any data elements which need to be standardized. As discussed in AR 25-9, the requirements developers must work together with the organization data administrator to review the inventory of existing Army standard elements. The ANSWER tool set can support this process with the browsing and graphical schema display facilities. The set of existing standard elements can be displayed with the browser and any existing mappings to schemas stored in IRDS may also be graphically examined. This information will provide assistance to the requirements developer in preparing documentation for a new candidate element. The data element creation tool may be used by the requirements developer to create a candidate element consistent with AR 25-9.

The organization data administrator may review the submitted candidate as discussed in AR 25-9. The review process can be supported by the DECT. The organizational data administrator may use the synonymy detection facilities of the DECT to identify any existing elements similar to the candidate data element. The data administrator may also use the browsing facilities to investigate existing data elements for similarity with the candidate element.

Information class proponents, the Army data encyclopedia administrator, and the Army data manager may similarly use the browsing and data element creation tools of ANSWER to support the candidate element review process. It should be noted here that the specific information on schemas and standard data element instances will vary through organization levels. The specific standard data element instances available to a requirements developer for inspection will probably be much more restricted than the set of instances available to an information class proponent, as shown in Figure 6-3. The same tools may be used to aid the review process at each level in the organization. Only the contents of the Army Data Dictionary information accessible by the individuals may change.

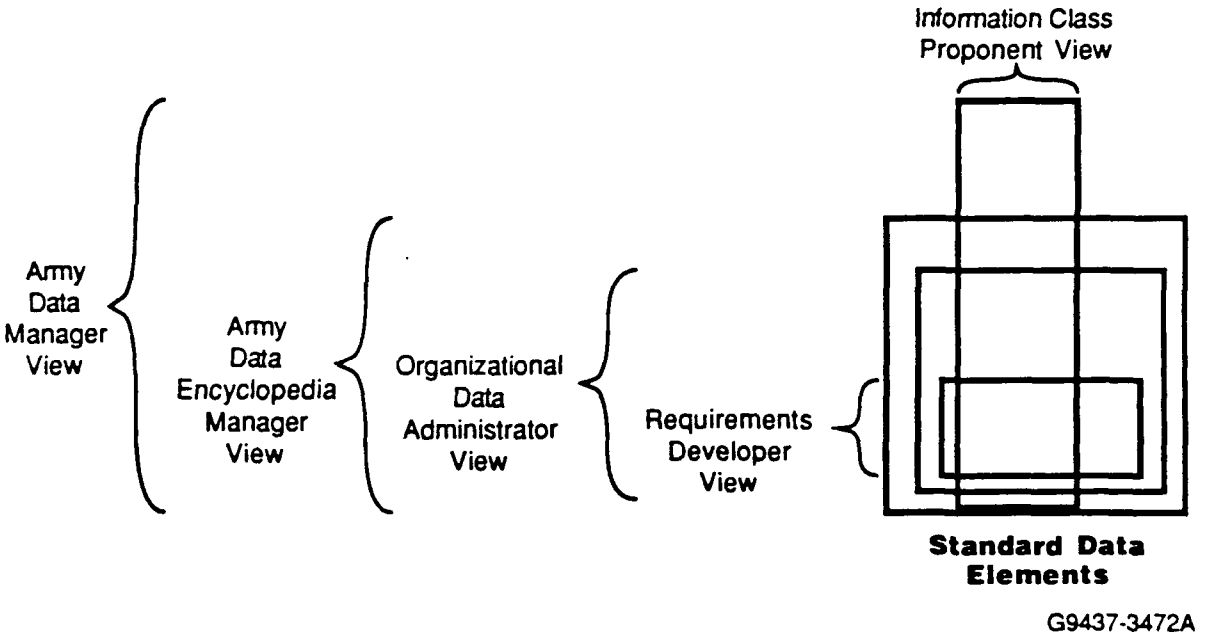


Figure 6-3. Views of Data Elements

The end user will benefit in part from the availability of the browsing facilities provided in ANSWER Phase II. The end user will be able to use the deliverables of ANSWER Phase III and IV much more directly. The tasks that will be supported in Phase III and IV correspond to Phase 5, operations, in the data life cycle. These tasks include formulating:

- Distributed queries in terms of Army Data Dictionary concepts,
- Distributed query processing of queries in terms of Army Data Dictionary concepts.

The basic tasks supported by the ANSWER tool set across the life cycle of the data and repeated by various individuals using different instances of data includes:

- Developing and maintaining integrated views of data to support distributed access and design of integrated distributed systems from existing systems;
- Developing and managing the creation of standard data element names for selected data elements;
- Supporting browsing of Army data through the use of Army Data Dictionary concepts to aid in identifying requirements.

Section 7
Installation Requirements and Testing Scenario

The ANSWER tool set may be installed for testing at individual sites. The tools should be treated as alpha test tools. The site using the tools should be interested in developing real test cases for the schema integrator and the data element creation tool and would work together with the ANSWER team to test the software on realistic Army problems. The ANSWER team will take the results of the testing and modify the software as required to meet the functional needs of the site.

Specific enhancements and customization will be required for testing the data element creation tool in the area of the class word hierarchy and the prime word hierarchy. The hierarchies must be customized to the individual site.

The schema integrator requires schemas expressed in terms of the Entity-Relationship formalism. The schemas may be stored and accessed from IRDS or they may be created through the ANSWER graphical user interface.

Schemas and Army Data Dictionary elements to be used must be stored in IRDS. The installation and testing would follow a twelve month schedule including the following tasks:

- **January 1990**—Identify target applications and schemas to be used for testing; develop test requirements and success measurements.
- **February 1990**—Software installatin at site. Site personnel populate IRDS with specific schemas and ADD elements; initial training on tool use; ANSWER team performs initial testing in cooperation with site personnel.
- **March 1990**—Test results evaluated by ANSWER team and Army; necessary modifications to software identified; additional tests identified to be performed by site personnel; ANSWER team begins software modifications.
- **April to June 1990**—ANSWER team modifies software; ANSWER team and site personnel evaluate results of additional tests.
- **June to October 1990**—ANSWER team identifies performance issues and optimizes code to meet performance requirements.
- **November to December 1990**—ANSWER team and site personnel develop demonstrations for other Army personnel.

The hardware and software requirements for installation of Phase II software include:

- A Sun 3/60 with 16 megabytes of RAM
- A 352-megabyte hard disk with an 80-megabyte swap space,

- A UNIX 3.5 (A port of Phase II software to 4.0.2 could also be done if necessary),
- An Oracle RDBMS, SQL*Plus and ProC,
- An Allegro Common LISP, Common Windows, Allegro Composer.

It is important to note that individual tools could eventually be fielded independently of the entire ANSWER system. Each tool individually would require considerably less hardware resources.

Section 8
Phase III and IV Plans

The tool set developed in Phase II addresses issues involved in the development of new schemas and the integration of existing schemas and the creation of standard data elements. Phase III and IV of the ANSWER program are planned to address the query formulation and query processing issues. The current task for Phase III and Phase IV are as follows.

8.1 Phase III—Duration: December 16, 1989 to December 15, 1990

This phase now includes work from the original Phase II and work from the original Phase III. These tasks are assigned a new task number with the original task number presented in parentheses. Some of the original tasks overlap between the new Phase II and Phase III. For example, the browsing task appears in both the new Phase II and the new Phase III. This is because browsing was originally planned as a 12-month effort and has now been broken into an initial effort to be completed in six months with the remainder of the effort to be done sometime during the subsequent 12-month period.

8.1.1 Task 3.1—Implement Enhancements

This task was originally defined to be a small effort to implement browser enhancements identified in Phase II. We completed the browser enhancements early, as part of Phase II. Because of that we will be able to perform some additional work under this task. In this task we will implement enhancements to the user interface manager developed in Phase II. We will also investigate issues associated with implementing the user interface and schema integrator on a large realistic model.

8.1.2 Task 3.2—Query Formulation Implementation

This task is the major focus of Phase III. In this task we will analyze approaches for query formulation, make recommendations for implementation and review this with the Army representatives to select and implement a prototype of an appropriate approach.

8.1.3 Task 3.3—AI Techniques Implementation

This task is a continuation of the new Phase II task for identifying additional tools for database registration. In this task we will continue to develop the data element creation tool (DECT).

8.1.4 Task 3.4—Security Study

In this task we will develop an overall approach for security enforcement in ANSWER, and make recommendations for implementation and exploration of key concepts.

8.1.5 Task—3.5 Demonstration and Training

In this task we will design and implement scenarios to display the improved browsing prototype, new database registration tools developed under Task 3.3 and the use of the query formulation aids. We will train Army personnel in the use of these tools.

8.1.6 Task 3.6—Distributed Query Processing

In this task we will identify existing commercially available distributed query processing capabilities and design the integration of ANSWER tools with the distributed query processing capability.

8.1.7 Phase III Deliverables

The following deliverables will be completed as part of Phase III:

- Browser enhancements identified as part of Phase II.
- Selected query formulation techniques and additional database registration tools identified in Phase II.
- Interim and final demonstrations of software deliverables to AIRMICS.
- Up to three demonstrations of software deliverables to groups selected in cooperation with AIRMICS.
- Training (2 to 3 days) in the use of software deliverables to groups identified by AIRMICS.
- Interim oral review.
- Final report summarizing results of software development efforts, security study, and distributed query processing study.
- Final oral review.

8.2 Phase IV—Duration: December 16, 1990 to May 16, 1991

Phase IV represents the remaining six months of the original Phase III. The major task in Phase IV is the implementation and integration of the ANSWER tools with existing Army distributed query processing facilities. The scope of these tasks will be discussed with the ARMY in 1990.

8.2.1 Task 4.1—Distributed Query Processing

This task will implement the integration of ANSWER tools with a distributed query processing facility.

8.2.2 Task 4.2—Demonstration and Training

Install the ANSWER system on hardware at an ARMY installation. Conduct training sessions in the installation, use and maintenance of the ANSWER system.

8.2.3 Phase IV Deliverables

The deliverables for Phase IV include:

- The ANSWER tool set integrated with a distributed query processor running at an ARMY location on ARMY hardware.
- Interim and final demonstrations to AIRMICS of contract progress
- One to two demonstrations of Phase IV software to groups identified by AIRMICS
- A final report summarizing the results of Phase IV.