REFERENCE MODEL FOR DBMS USER FACILITY

by

User Facility Task Group
of the ASC X3/SPARC Database System Study Group

Authors:

John Gersting, Indiana Univ.-Purdue Univ. at Indianapolis
Kate Kinsley, Datawise, Inc.
Nancy McDonald, Computer Technology Planning
John North, AT&T Technologies
Mark Sastry, Honeywell
Edward Stull, GTE Governmental Systems

PREFACE

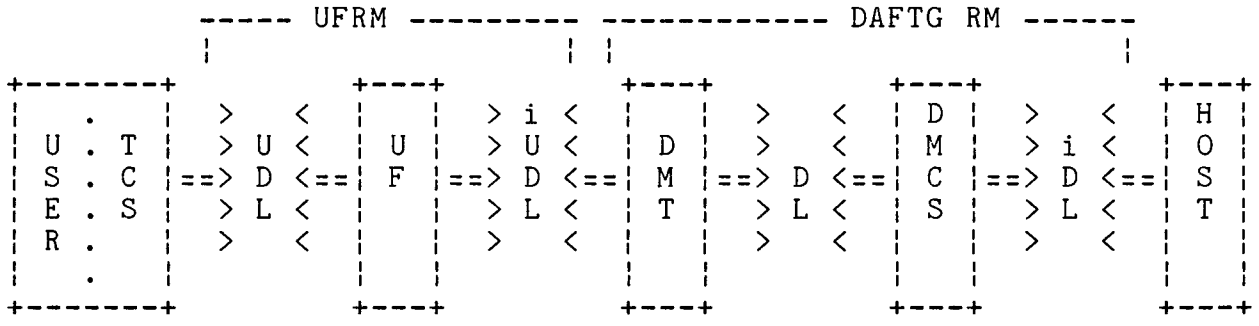
This report is a contribution to X3/SPARC Database system Study Group (DBSSG). The technical work represents the careful distillation of the contribution of the authors and others who have participtated in the meetings and studies of the User Facility Task Group (UFTG) which is a sub-group of the DBSSG.

The UFTG invites comments and suggestions for improvement. Please forward your comments to the chairperson of the UFTG:

Dr. Nancy McDonald
Computer Technology Planning
10014 N. Dale Mabry Suite 101
Tampa, FL 33618
(813) 968-2660

EXECUTIVE SUMMARY

In this report we present the following reference model for the user facility portion of a database management system.



The User Facility Reference Model (UFRM)

This figure is a representation of a database management system beginning with the user at the left and ending with the computer system on the right. The model depicted here is composed of two major parts, the User Facility Reference Model (UFRM), which is the topic of this report, in conjunction with the DAFTG (Database Architecture Framework Task Group of the DBSSG) reference model. There are three components of the UFRM, the User Facility (UF), and the interfaces; the User Data Language (UDL) and the Internal User Data Language (iUDL).

The UDL is the single interface between the user environment and the UF. It provides a uniform mechanism through which user requests and resulting responses can be posed.

The UF contains components that process messages from and presents output to the user. It also sends and receives information from the DBMS.

The iUDL is the single interface between the UF and the Data Management Tool (DMT) layer of the DBMS. It provides a mechanism to categorize the connections needed by various tools, thus providing for reusability of UF implementations.

The UFRM offers a separation between the Data Management Tools and the User. The UDL and iUDL are candidates for "standardization". Since the DMT are continually evolving and current technology often embeds a user interface in each tool, the process of "standardization" must be staged. The work here indicates that the UDL can provide a focus for the start of this process. This report also provides a theoretical basis from which a UDL could be developed.

1 INTRODUCTION

1.1 PURPOSE AND SCOPE

This report describes the aggregate work of the User Facility Task Group (UFTG) of the Database System Study Group (DBSSG) in developing a Reference Model for a database management system user facility.

1.2 STRUCTURE OF THE REPORT

This chapter gives a short definition of the term "reference model" and indicates how one may be used. In addition, a brief review of other models examined by the UFTG is given.

Chapter 2 looks at the user and various devices that might be at the user's disposal.

Chapter 3 suggests a User Facility Reference Model in terms of component and functional descriptions.

Chapter 4 summarizes, concludes, and discusses avenues for future work.

1.3 EXPLANATION OF THE TERM "REFERENCE MODEL"

The term Reference Model (RM) [FON86] is defined in the literature as a conceptual framework for a subject area. The main use of an RM is to divide standardization work into manageable pieces and to know at a general level how these pieces are related to each other. This definition is especially appropriate for this effort that seeks to provide some order and structure to the User Facility (UF) of a Database Management System (DBMS).

1.4 USES OF A USER FACILITY REFERENCE MODEL (UFRM)

There are numerous benefits that will be gained from the development and acceptance of a UFRM for users, purchasers, vendors, designers, servicers, teachers, and students of DBMSs. Some of these benefits are discussed below.

* Promote understanding of DBMS.

  By separating the DBMS functions from the user facility functions and highlighting the interfaces between these two sets of functions, many people will obtain a better understanding of the DBMS inner workings as well as the manner in which those workings are reflected to the user populations.

* Help maintenance and control activities

  Through this type of top-down division of systems, developers and users will obtain an increased understanding of functions, which leads to easier maintenance and stronger control mechanisms.

* Provide guidelines for modularization of functions and standardization of interfaces.

  The identification of pieces and their interrelationships will establish a framework for standardization. Such a framework will enhance portability of software, improve human productivity, reduce training requirements, provide comparability, and potentially reduce costs of products.

* Aid in Comparison and Selection of Systems

  The categorization of the UFs is the first step toward their qualification. This in turn aids the efforts of organizations to choose, change, train for, and maintain a new UF and/or DBMS.

1.5 RELATIONSHIP OF USER FACILITY REFERENCE MODEL TO OTHER DATABASE REFERENCE MODELS

The DBSSG/UFTG reviewed several models related to the user facility area. The most important of these was the DBSSG/DAFTG (Data Architectural Framework Task Group) Reference Model since it is used to establish the relationship between the user facility and the ISO reference model. Other models examined were Foley's model of communication for the human elements of the problem, the actor/act/action model for its humanistic appeal, and the fundamental theory of computation for its computer orientation.

1.5.1 The DAFTG Reference Model

As will be seen, the User Facility Reference Model (UFRM) will serve as a front-end to the DAFTG DBMS Reference Model; these together form a functionally compatible means of describing interaction between the user and a database system. (See Figure 1.1.)

The DAFTG Reference Model itself consists of three components, each communicating via a language interface. Starting from the right in Figure 1.1, the Operating System (OS) supports the DBMS and communicates with the Data Mapping Control System (DMCS) through the internal Data Language (i-DL).

The DMCS is the "core" DBMS and provides operators for data manipulation and data description. The Data Language (DL)

interface is the data manipulation language for the DMCS data model. The Data Management Tools (DMT) represent software tools that support high-level functions such as high-level query languages, graphics-systems, report writers and database design tools. There, tools communicate with the DMCS through the DL interface.

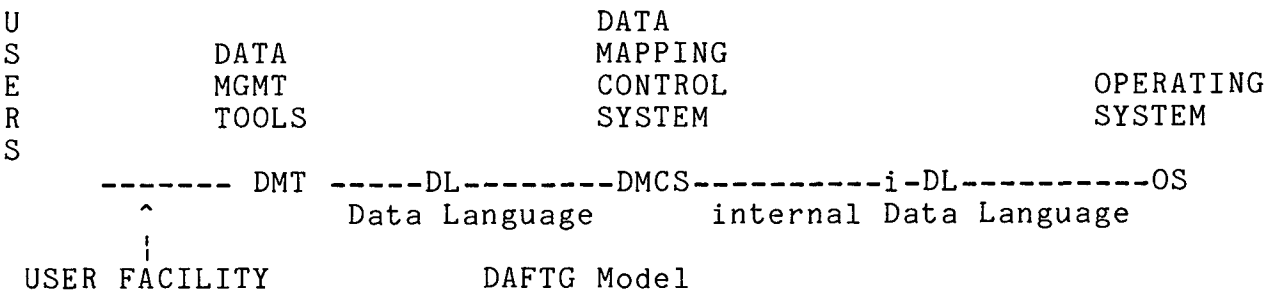


Figure 1.1 - DAFTG DBMS Reference Model

1.5.2 The Communications Model

The Communications Model [FOL81] consists of four layers. At the highest layer, the conceptual layer defines the key application concepts of objects, relationships, and operations on these objects in terms understood by the user. The semantic layer defines items needed for each conceptual operation as specified by the user to the machine and the machine to the user. The syntactic layer defines sequences or a grammar of the tokens exchanged between the user and the machine. The lexical level associates the primitive operations of the support hardware with the tokens of the syntactic level.

1.5.3 The Actor/Act/Action/Message Model

The Actor/Act/Action/Message Model (A3M) [HEW73], described in this paper, is a modeling approach that is based on artificial intelligence concepts and provides a humanistic model. Described here is a perception where database actions are considered to be the result of database acts being carried out by database actors.

The meta-model applied here for studying behavior in user facilities extends the actor system of C. Hewitt's and the concept of object programming. Actors have the following properties:

- An actor may perform any number of acts. The behavior of an actor results from the "actions" taken as it performs

an act.

- An actor embraces the internal knowledge about itself.
- For communication, an actor may monitor and send messages to other actors over one or many communications channels.

Actors are agents that carry out acts resulting in actions as pictured in Figure 1.2. An actor specified as an example of one or more other actors may inherit properties from these actors. Lateral actors are autonomous and may only communicate through messages.

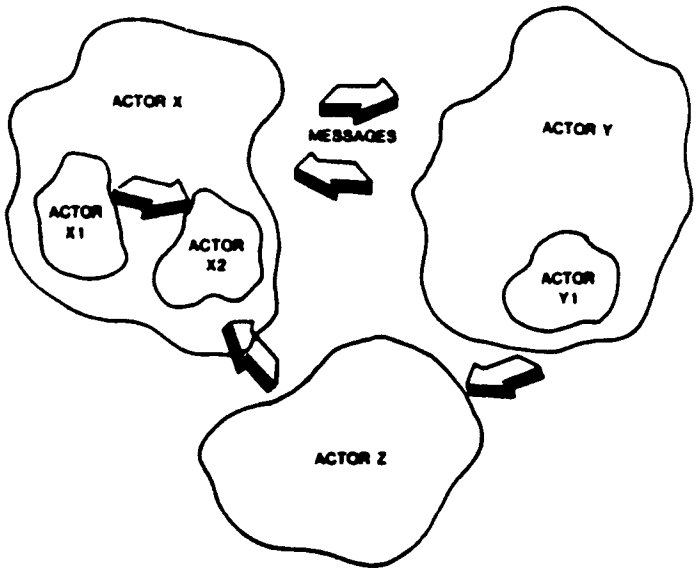


Figure 1.2 - Communicating Actors

1.5.4 The Fundamental Theory of Computation Model

The fundamental theory of computation (FTOC) Model is simply that a machine, concrete or abstract, produces a result based on descriptions (e.g., data, instructions) operated upon by a set of description-manipulating functions (sometimes called the interpreter).

CHAPTER 2 BACKGROUND

2.1 INTRODUCTION

Three areas for collection of background information surfaced early on:

- the user environment
- modes of interaction with the DBMS (primarily hardware devices)
- "facilities" available in current DBMS implementations

These areas were examined in some detail. This chapter contains the salient results of these examinations.

Little is included on the latter two areas. These are mostly subsumed in the DAFTG, which became available during the course of the UFTG's studies, and by the User Facility Reference Model itself. The user is still the most important element of the study and will be the focus of the next sections.

2.2 THE USER ENVIRONMENT/USER FACILITY

The UFTG's decision to address the user facilities' needs for DBMS and not the general user facilities for all of the user's computing needs was appropriate for a subgroup of the DBSSG. However, the UFTG quickly discovered that limiting the scope of the project to DBMSs did not really limit the scope of the user facilities since DBMSs are rapidly becoming the backbone to most computer application systems.

Reviewing past research in related areas proved only partially helpful. The existing literature about DBMS user facilities mainly concentrates on such narrow aspects of the user's interaction with DBMSs as query languages or experiments with "novice" users.

The work of [MCD82] and [VAS82] developed classification schemes for the functionality of query languages and for query language users. They then developed a set of tables to aid the users in a preselection of query languages based upon the user's interaction capabilities, the structure of the task under consideration, and the functionality of the query language. The final selection is most often based on other indirect criteria such as cost and existing hardware and software compatibility. Even though the cited work addresses only a portion of the user facilities, query languages, it has aided us in defining and combining the various user attributes and in developing the table that associates user roles with user attributes.

The British Computer Society, first through its Query Language Group and now through its End User Systems Group and some of its

members, has produced several related papers, [BCS81], [TAG84], and [TAG85]. This work initially concentrated on query languages, but now is aimed at models for user interactions with computing systems. The UFTG members have used these ideas in our discussion for relating user facilities to database architecture. The use of a dialog manager and service manager interfacing to the data access routines and application programs supports the separation of the UF from the DMT layer in our work.

2.2.1 The DB User Task Translation Sequence

A user facility must provide good support for aiding the user in performing a specific task with the DBMS. The task will differ considerably depending upon the user's current role. It varies from defining additional data structures for the database, to building a new application program for performing a monthly summary, to browsing through a portion of the database to help make a business decision. But there is a common thread for all the tasks independent of the user's role - the way a user approaches the translation from his mental representation to the physical sequence of operations that interact with the I/O system or the user facility. The User Task Translation Sequence is depicted in Figure 2.1.

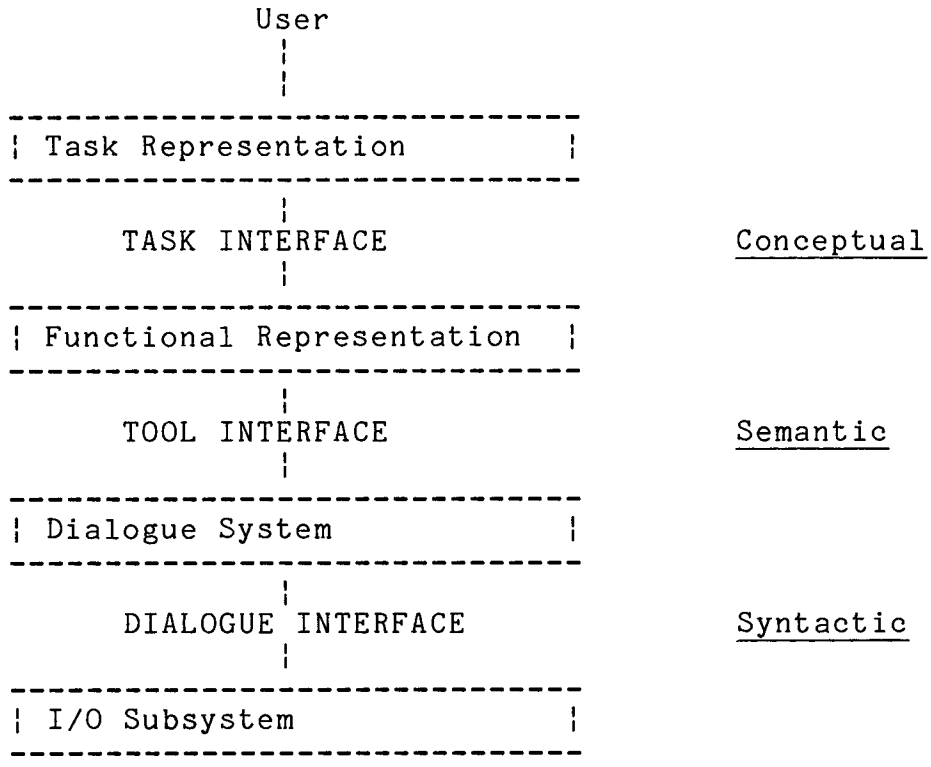


Figure 2.1 - User Task Translation Sequence

The user has a conceptual model of the various objects, their relationships, and the manipulations necessary to solve a task. This Task Representation is usually in terms familiar to the user (e.g. a business model of the enterprise and its goals). This is typically too conceptual for immediate translation to the system.

Thus the user first translates the Task Representation into a Functional Representation that the system knows. This Functional Representation includes the various data structures and commands that the database management system uses. But even this set of objects and commands must be translated into terms that the Dialogue System understands before the user can interact with the system.

The Dialogue System supports the user with many different modes including menus, forms, command prompts, icons, etc., depending upon the user's previous knowledge of the system, the task, and the environment. Finally, the user translates a request from the Dialogue System into physical movements or operations such as entering keystrokes, picking an item on a menu, or speaking the request to a voice recognizer.

As projected in Figure 2.1, the user performs a sequence of mappings that maps the user's mental task representation into the physical interactions with the user facility. We have indicated the relationship to the conceptual, semantic, and syntactic notions along the right side of Figure 2.1. The conceptual mapping translates the user's mental Task Representation into the functional representation the system knows; the semantic mapping maps the database objects and functions to the various representations available to the user in the Dialogue System; and the syntactic mapping allows the user to interact with various dialogue representations by manipulating the physical devices of the user facility.

2.2.2 User Roles

In the early days of DBMSs the user classes of database management systems were more distinct - the database administrator was responsible for maintaining all the data objects associated with the database management system as well as monitoring and optimizing system performance; the application programmer developed the transactions via screens for capturing the information entered by the data clerks to be stored on the database; and the end-user made use of reports generated from the application programmer's programs.

Today, the use of DBMSs has migrated from mainframe systems, through departmental computers, to single user workstations and computers. The single user computer is often used by small businesses to run one or more applications using a DBMS. This migration has clouded the distinction of user classes so much that it is best to consider users in the roles they are executing

at a moment in time.

The major roles that will be considered in this reference model include the system programmer, the application programmer, the database administrator, and the end user. Obviously, there are many different subroles under each major role. Each role has associated with it attributes that are necessary for the successful execution of the role. Besides the human roles that we usually associate with information systems, we must also consider the virtual user. This is a computer system that has been developed and implemented to stand-in as an agent for the human. Figure 2.2, User Roles, shows the simple analysis of four specific roles, the interface tool required, and the system component needed.

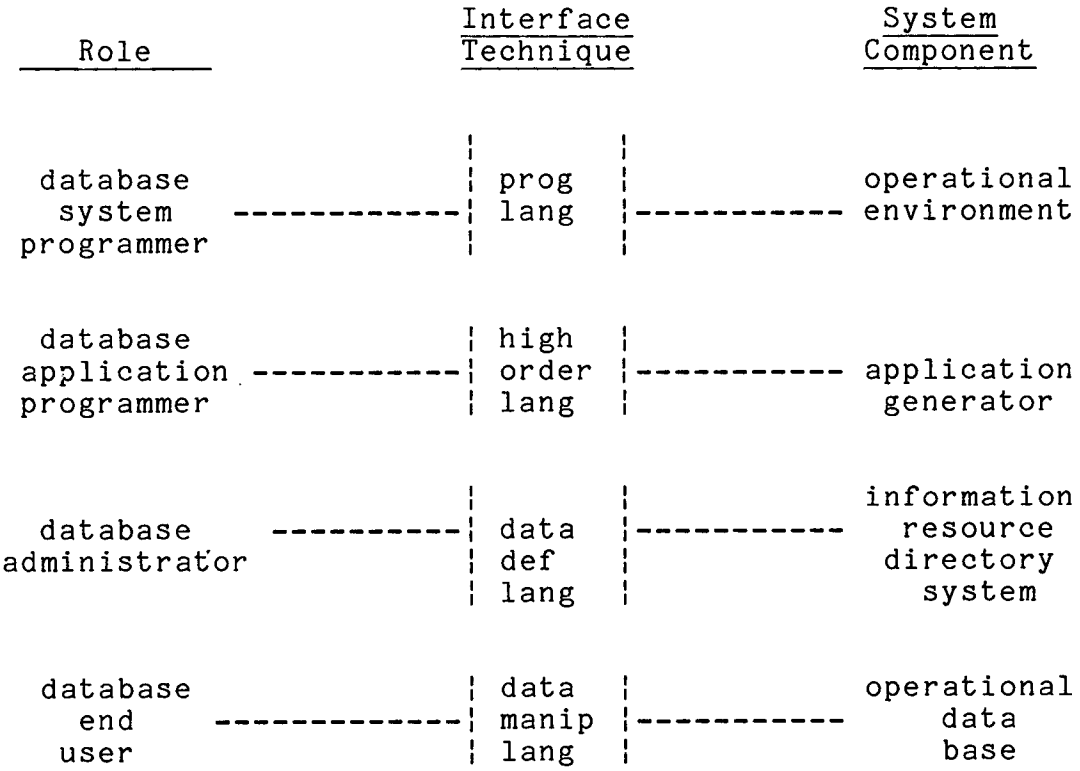


Figure 2.2 - User Roles

The system programmer role deals with either the DBMS software itself or utility programs written to support the administration of the database. There are three major subroles in this group - the DBMS system programmer, the DBA system programmer, and the tool builder. Each of these subroles demands a high level of knowledge of the system's data structures, the hardware, and the interface and functions of the operating system. People in these roles direct the action of the system.

The DBMS system programmer develops interfaces to the internal access methods, command language interpretor, etc.. The DBA

system programmer subrole writes utilities to aid in tuning the system, in performing the backup and recovery of DB files, or in restructuring the DB. The tool builder creates a level of abstraction for application programmers to use when they want to interface with the DBMS.

The application programmer role deals with extending the range of data covered in the DB while incorporating new functions within the information systems for the enterprise. This application programmer often uses a programming language interface with the DBMS, writing much of the application code in procedural computer languages. These applications tend to deal primarily with transaction oriented production systems that have regularly scheduled reports. In the last several years there has been a push to infiltrate the data processing department with 4th generation languages to improve the application programmers' productivity.

The DBA role deals with the metadata for the DBMS. This includes the definition of the data elements, the enterprise and logical structure of the data, different user views, and permissions for security and privacy. Some organizations separate this role into a DBA subrole that is concerned with interfacing to the application programmer and the data administration (DA) role that is concerned with the enterprise model. The DBA has a high level of knowledge about the metadata and medium knowledge about the real world data; this is reversed for the data administrator since he must model the enterprise.

The end-user role is the fastest growing segment of DBMS users. There are two groups of users, based on the frequency with which they use the DBMS. The casual user group consists of the infrequent data entry person, the intermediary for an organization or management, and management itself. This group requires a "user-friendly" front end to a general purpose query language typified by natural language, graphics, or menus and forms. The daily user group contains the regular data entry/inquiry person who, being familiar with the system, can bypass some aspects of the restrictive front end, and the analysis/researcher who uses the computer as an integrated tool for compiling specific information relative to a job function. These users utilize more command oriented query languages, forms, non-procedural report languages, and specialized 4th generation languages.

2.2.3 User Attributes

Each user role has a set of attributes associated with it as depicted in Figure 2.3. The first attribute is domain knowledge. Each user has some knowledge of the domain within which the user is working. This knowledge can vary from just enough to perform a data input operation to a level high enough to realize the ripple impact of changing one data field. Domain knowledge is

typically associated with level of experience and training. As an example, a DBA programmer writing a utility for restructuring the data within a database could have little understanding of the real world data but would have a very good understanding of the data dictionary metadata. People typically dealing with metadata receive much more training than many of the end users dealing with the real world data.

ATTRIBUTES

		skill level				dialog style		
ROLES	domain knowl	data str	appl str	hdw	os	sys dir	user dir	resp time
system pgmr								
DBMS sys pgmr	L	H	L	H	H	L	H	L
DBA pgmr	L	H	M	H	H	L	H	L
tool builder	M	H	H	H	H	L	H	L
application pgmr								
DP dept pgmr	M	M	H	M	M	L	H	L
info cntr spec	L	M	L	M	M	L	H	M
dept support	M	M	H	M	M	L	H	M
database admin								
DBA	H	H	M	M	M	L	H	M
DA	H	H	M	M	M	M	M	M
end-user								
casual users								
inf.input	L	L	L	L	L	H	L	H
intermediary	M	M	M	M	M	M	M	M
management	H	M	L	L	L	M	M	M
daily users								
reg input	L	L	L	M	L	H	L	H
anal/rschr	H	M	M	M	M	L	H	H

L = low, M = medium, H = high

Figure 2.3 - User Roles to Attributes Matrix

The skill level attributes help differentiate user roles. Skills encompass familiarity with and ability to manipulate the data structure (data str), the application structure (appl str), the hardware (hdw) used to interface the user to the computer system, and the underlying operating system (os).

The expertise needed to manipulate the data structure involves knowing the data elements, the records, and the relationships between the records. A programmer usually has more knowledge and concern about the data structure than an end-user who relies on the system (e.g. help screen, on-line tutorials, forms, graphics) to help him remember all the data structure attributes.

The application structure skill level requires knowledge about the operations, the relationship between the operations and programs, and the transactions, forms and various output reports. A frequent user may tire of too much system direction (sys dir) found in rigid prompts, menus, or forms; rather, he may prefer a user directed (user dir) dialog found in most command oriented interfaces. The tool builder on the other hand needs different tools to extend the operations and ensure that extensions don't interfere with the existing operations.

The ability to use the hardware is tightly tied to the frequency of use. Again, the highest level of expertise is typically associated with the people building or enhancing the software capabilities of the system. The end-user, on the other hand, has benefitted from the drive for "user-friendly" hardware devices such as pick devices (mouse, light pen, touch screen), multi-color output devices, high-resolution graphics, etc. Though the end-user at his own workstation or personal computer has gained control over the time when data is input, the structure of reports, and the overall flexibility in dealing with the data, he no longer can depend on the operators and support staffs found in large data processing organizations to insulate him from hardware issues and problems.

Knowledge about the operating system is an important attribute for users. The programmers must be able to invoke other commands besides those associated with the DBMS since they deal with existing data files and generating data files to be used by other processes on the computer. The end-user prefers to have all these operating system functions integrated into his interface facility to minimize his education and memorization needs.

The Dialog Style attribute has been divided into two parts - system driven, and user driven. The notion put forth by this attribute is that there is a low, medium, or high utilization of dialogue styles by user roles. Certain dialog styles are easier to learn, more flexible, and/or easier to use for certain roles.

A system directed dialog style typically has less flexibility. The system must track its current state and determine a list of possibilities from this state. System directed dialogs are typically associated with either the casual user or a frequent data entry person who has just a few possible tasks to perform. The more knowledge of the domain and the higher level of skills a user has, the more likely that this user will become bored with a system directed dialog style.

On the other hand, programmers and daily users needing flexibility will choose a user-directed dialog style. This style is typified by command languages often supported with macros and/or command procedures to execute often run sequences of operations. These users are often concerned about the extendability of the command language so they can tailor their

own work environment. One of the recent user-driven dialog styles is the desk-top metaphor that permits multi-tasking.

Response time requirements are based both on the user's real-time needs and the user's expected response time. Often the end-user's expectation of response time cannot be met. Some users lack an understanding of the total number of operations, the impact of shared resources, and the audit and recoverability issues needed to achieve a successful completion of a function. Hence, many users are looking to dedicated workstations to solve this problem of variability of response times when they use shared computer systems. Dedicated systems definitely provide a more consistent response time operating in a single process environment, but the use of multi-layer interfaces on these workstations will cause the response times to elongate. The user who chooses a dedicated workstation must also become responsible for providing the audit and recoverability of functions, the data to other users, and the concern for performance.

2.3 INTERACTION DEVICES

Another of the initial concerns of the UFTG was how the user interacts with the DBMS. Lists of devices were compiled, including the obvious items of keyboard, light pen, mouse, voice, etc. These lists seemed to require continual updating. In any case, from a reference model point of view, the problem became intractable (should defining a standard mouse interface be part of the UFRM development?).

Figure 2.1 suggests that the task, not the device, is the important item. For example, the user might be requested to make a forced choice among, say, five items. This choice involves "traffic" over the user interface. Such traffic should be independent of the mode of presentation of the choice (e.g., menu, command line, and so forth) and of the device used to make the choice (e.g., light pen pick, key stroke, and so forth). In this case, the UFRM proposed later in this work might take on the form of 5 items passing toward the user and one of the 5 returning.

In order to render the hardware problem tractable, the proposed "interface" will be between the devices available to the user and the rest of the DBMS, as opposed to an interface between the user and the devices.

2.4 EXISTING DBMS FACILITIES

The third area of background investigation was the facilities available in current DBMS implementations. Again, the limitation of a database orientation did not really limit the scope of the problem because a graphics display system or a statistical package can present equally valid user facilities for databases. The problem was to separate the user facility from a particular user interface. Feature analysis techniques for some selected

systems were developed, but the results were difficult to evaluate.

When the DAFTG Reference Model became available, it provided a solution to the problem in the form of the DMT, Database Management Tool Layer, a "top layer." The natural place for the user interface is between the user and the DMT. The next chapter presents these ideas in more detail.

CHAPTER 3 USER FACILITY REFERENCE MODEL

3.1 INTRODUCTION

The User Facilities Reference Model is defined to be a conceptual categorization of functions/facilities that a database user would like to use while interacting with a DBMS. It will provide a common basis for the coordination of standards development, while permitting existing standards to be placed into perspective. The model has sufficient flexibility to accommodate advances in technology and expansion in user demands. Accordingly, the model should permit the phased transition from existing implementations to new DBMS standards.

The database system is an agent that carries out a variety of tasks on behalf of its user. It is from this perspective that the database system must be studied to determine the support it is to provide a user facility.

Figure 3.1 depicts the unification of the previously mentioned models (DAFTG, Communications, A3M, FTOC) in the framework proposed for the UFRM. The diagram itelf shows the form of the proposed UFRM along with the DAFTG model. In the top of the blocks are DAFTG terms (conceptual, syntactic, tool command, database, operating system). In the lower portion of the blocks are the Communication Model's terms (semantic, lexical,...). The D and I indicate data values and interpreters, i.e., that local data and computing power exist at each stage. The A3M model is used to describe how the "traffic" flows through the various stages of the model.

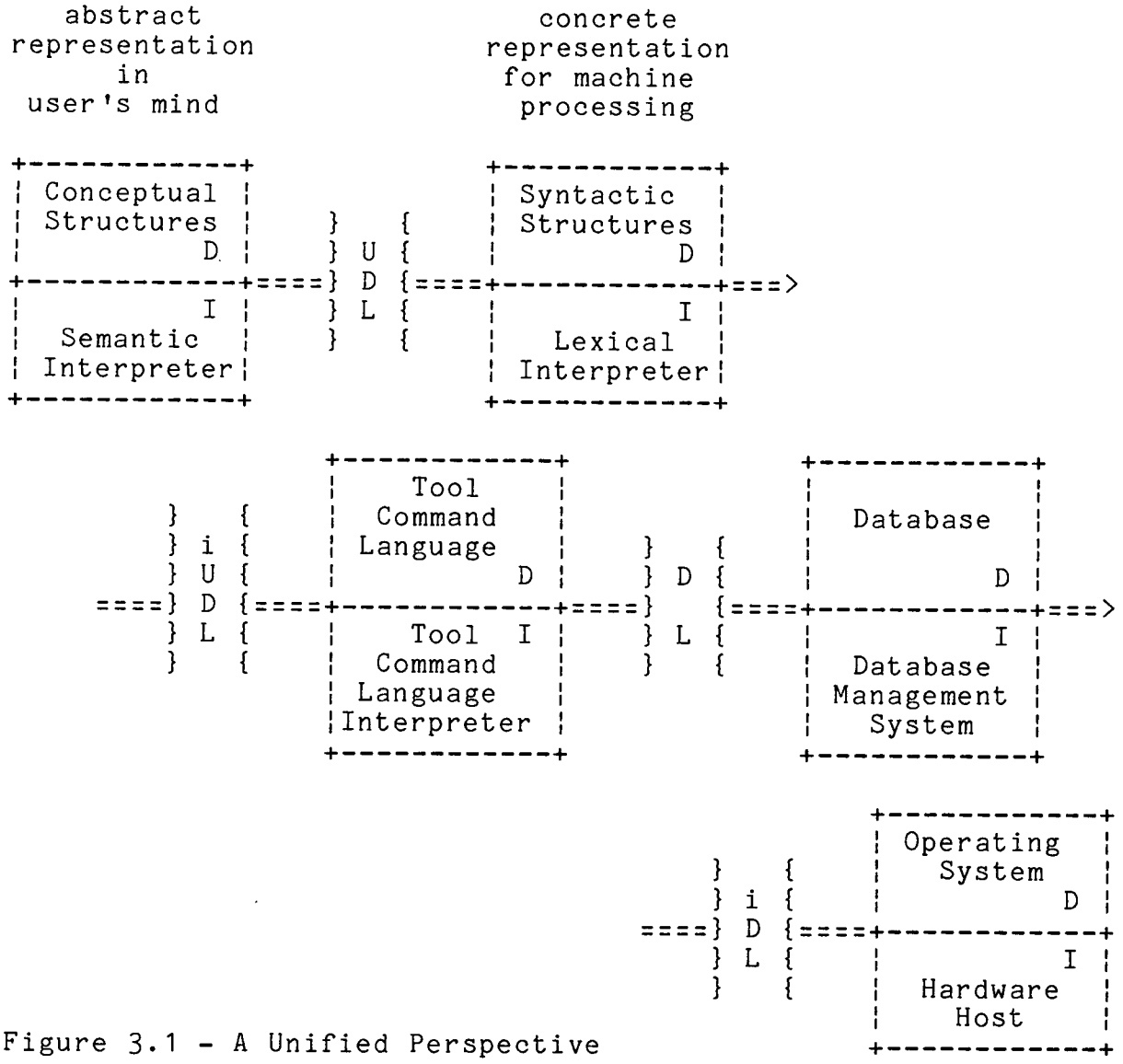


Figure 3.1 - A Unified Perspective

3.2 FUNCTIONAL DESCRIPTION OF THE USER FACILITY

The User Facility Reference Model serves as a front end to the DAFTG-DBMS Reference Model and together forms a functionally compatible means of describing interaction between the user and the system.

A User Facility: (1) transforms a user's request (for information from a database) into functional requests for data management tools and (2) maps the output from data management tools into a presentation format to the user.

3.3 COMPONENTS OF THE UFRM

Figure 3.2 positions the UFRM within a database-oriented system. This work focuses on the three components of the UFRM, called the UF, UDL, and iUDL. The remaining portion of Figure 3.2 is the DAFTG Reference Model. For completeness, all 9 of the elements of Figure 3.2 are described below, first the processes (boxes) and then the interfaces, denoted by > < .

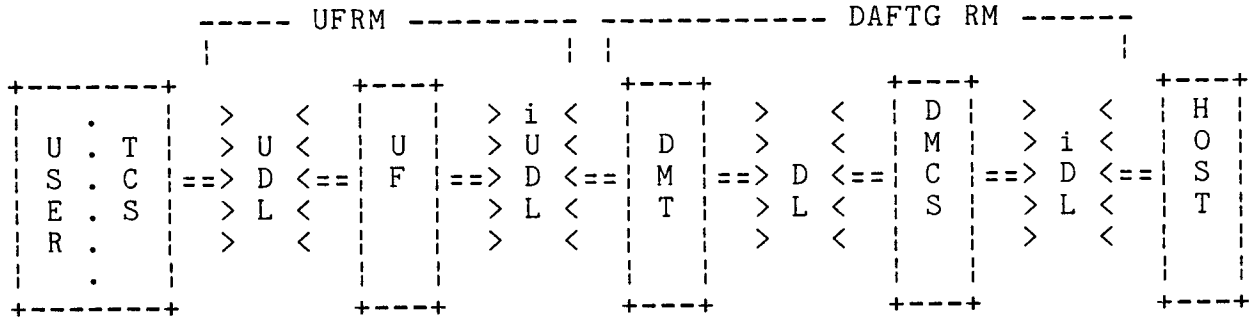


Figure 3.2 - The User Facility Reference Model (UFRM)

Processes:

- The term "user" represents a user environment that interacts with a DBMS. This includes the human component examined in Chapter 2 and may include hardware devices in the form of the TCS, Terminal Control System.

- The User Facility (UF) contains components that process messages from and presents output to the user and in turn directs and accepts data to and from the database management system as defined by the DAFTG of the DBSSG.

- The Data Management Tools (DMT) processor, as defined by the DAFTG, contains software components that are plug-compatible with the DMCS through the DL-interface. A simple DMT could be a data definition language macro facility which would transform DDL statements into sequences of DL statements.

- The Data Mapping Control System (DMCS), as defined by the DAFTG, is a "core" database management system (DBMS) supporting and enforcing the ANSI/SPARC dimension of external-, conceptual-, and internal-schemas and the orthogonal intension-extension dimension of data model-, data dictionary-, and application schemas, and application data. An active/dynamic data dictionary system is an integral part of the DMCS. Although the DMCS would be typically a one-data-model DBMS, it is the core of a potentially multi-data-model DBMS.

- The Host is the computational host for the DMCS. The DAFTG defines Operating System (OS) as part of the environment of a DBMS which here is considered to be part of Host. The operating system services in many existing systems are either too slow or inappropriate to support the DBMS. Therefore, many current DBMSs provide their own operating system environment. Future operating system designers must recognize such needs of the DBMS; these needs are to be specified in the i-DL interface.

Interfaces:

- The User Data Language (UDL) is the interface to the UF. UDL is an interface between the user and user facility to transcribe user requests and present the DBMS responses.

- The internal User Data Language (iUDL) is the interface to data management tools so that the user is not required to learn what may be the specialized language requirements for each data management tool.

- The Data Language (DL) is the single interface to the services offered by the DMCS. It is the data manipulation language for the fundamental data model of the DBMS. The fundamental data model is a data model that supports the established data models such as the relational-, the entity-relationship-, the network-, the object-role models. Because of the intension-extension dimension of the data description and because the fundamental data model is used at all levels of this dimension, the DL supports both data and meta-data management. The data model schema is self-describing and thus all data including data descriptions are defined, retrieved, and manipulated through the DL interface.

- The internal Data Language (iDL) is the single interface between the DMCS and the OS through which all data is passed whether real data, meta-data, meta-meta-data, etc.

3.4 POTENTIAL COMPONENTS FOR STANDARDIZATION

In keeping with previous standards practices, interfaces within reference models are likely candidates for standardization. For the UFRM this would be the User Data Language and the Internal User Data Language.

3.4.1 The User Data Language Interface (UDL)

Communication between the user and the DBMS may be perceived as an ordered series of assertions and interrogatives from one to the other (Figure 3.3, User to DBMS Communication). Both interrogatives and assertions are imperatives but here imperatives are all those messages that are neither interrogatives nor assertions. An interrogative message is defined to be any action intended by the transmitting actor to only examine knowledge available to the receiving actor. An assertional message is any action intended by the transmitting actor to cause a change to the knowledge available to the actor receiving the message. In the most general sense, both assertions and interrogatives of an actor may result in further assertion and interrogative messages to the corresponding actor. Note that both the transmission and receipt of a message are actions resulting from an actor performing an act.

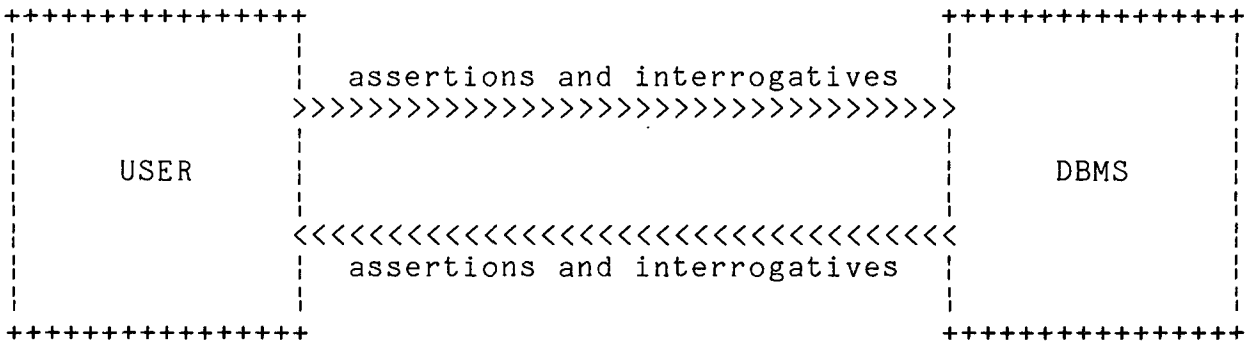


Figure 3.3 - User to DBMS Communication

Assuming the DBMS is in operation, it somehow prompts, that is interrogatives, the user for some act to be performed and upon receipt, attempts to perform the act resulting in a corresponding action possibly with side-effects to cause modification to the database and advise (i.e., assert to) the user of its action. Whether the user or the DBMS is in control is not important here but rather the investigation of how it is that the user may cause the DBMS to perform the correct act through a natural and minimal effort on the part of both the user and the DBMS.

The following two definitions are provided from the database system perspective:

1) An interrogative act is an act that examines the content of the database. The database may optionally respond to the user with the results of the examination.

2) An assertional act is an act that causes a change in the state-of-affairs of the database. It typically would be a statement of new information with an embedded or implied modification-type command.

In the most general sense, assertional acts may result in other assertional acts as well as interrogative acts.

Note: Composite operations, such as audit trailing user queries, are considered a composite of both assertional and interrogative acts.

All assertions belong to one of four major classes:

1) extend - The extend assertion adds new knowledge (extensionally) to the knowledge base.

2) update - The update assertion modifies existing knowledge. The modification identifies the

previous knowledge as being out-of-date and revokes it. Depending on the intent of the roles involved, this may initiate other appropriate actions such as notifying the change.

3) correct - The correct assertion, like the update assertion, modifies existing knowledge and revokes it. However, the modification identifies the previous knowledge as being incorrect and, depending on the intent of the roles involved, may initiate other appropriate actions such as signaling an error and resolving problems specified in the cognitive models that used the incorrect knowledge.

4) revoke - The revoke assertion also modifies existing knowledge. Revoking explicit knowledge may or may not cause deletion from the information store depending on the intent of the roles involved as in the update and correct assertions. For example, revoking implicit knowledge (i.e. information which does not exist physically but must be computed) causes knowledge to be added to the information store that initiates failure of the inference (i.e. computed information) for the implicit knowledge being revoked.

Any system that assimilates new knowledge must both provide and enforce semantics for the effects of assertion. This is perhaps one of the weakest properties in many current knowledge base systems. Often new rules and facts are admitted and stored regardless of whether they make any sense in terms of the existing knowledge. A sensible universe of discourse must always be maintained and enforced between the external world and the DBMS.

There are four important classes of interrogation:

1) The literal interrogation is an act that attempts to solve a problem literally as stated typically using only that inference necessary to search the store of information. Consider the fact: CPU is part of computer. The fact employs the domain of CPU, a relation of "is part of" and a codomain of computer. A literal interrogation of this fact would succeed only if the problem were stated using any of these three components (e.g., <X> is part of computer, CPU <Y> <Z>).

2) The explicit interrogation is a relaxation of the literal interrogation. The relaxation is to permit a limited inference over any or all of the components of the problem. Consider the problem: <X> has CPU. With a small amount of inference, it may be reasoned that <X> has CPU could be stated as CPU is part of <X> thus leading to the solution of CPU is

part of a computer.

3) The implicit interrogation is the antithesis of literal and explicit interrogation in that it attempts to solve a problem by using implied information. Even though the information store contains the fact above, implicit interrogation would only succeed if it could somehow be implied that a CPU is part of a computer.

4) The general interrogation is simply a combination of all possible solutions generated by interrogating literally, implicitly and explicitly using whatever information can be discovered to contribute to the discovery of another solution. For example, a literal interrogation may lead to discovering an implicit solution, and so on.

3.4.2 THE INTERNAL USER DATA LANGUAGE INTERFACE (iUDL)

The iUDL provides the interface between the user facility (UF) and the data management tools (DMT). The iUDL provides a mechanism to categorize the connections needed by various tools, thus providing for reusability of UF implementations.

The existence of the iUDL makes the assumption that the data management tools can be accessed directly by the iUDL, i.e., remotely from the user. This would provide access to data dictionary tools or graphics tools from the UF. Embedding the details of such accesses at the iUDL, the user need only see the UDL view of these tools while benefitting from their full capabilities.

Currently many data management tools must be invoked directly by the user. Providing categorization for the interfaces of the wide variety of tools that will become available is an area for future work.

3.4.3 THE DATA LANGUAGE (DL) AND INTERNAL DATA LANGUAGE (iDL)

The remaining interfaces in Figure 3.1 are the DL and the iDL. These are part of the DAFTG Reference Model.

3.5 EXAMPLE

Figure 3.4 demonstrates how the User Facility Reference Model integrates with the DBMS Reference Model. For example, a row could be inserted using a software tool such as Query-By-Example.

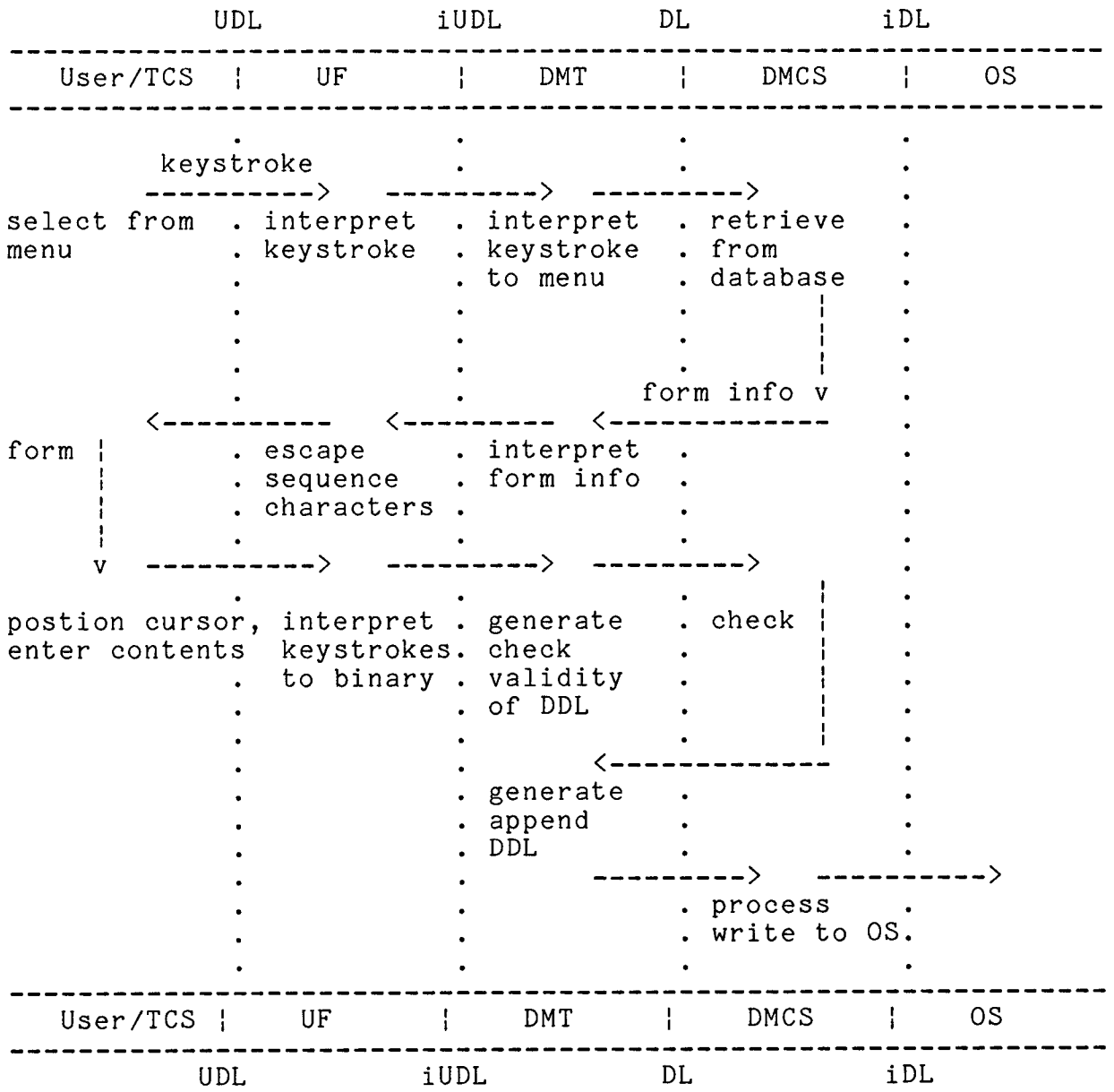


Figure 3.4 - Example Flow in UFRM

The UDL interface accepts the signal from the input device--whether it is a keyboard, light pen, audio or video response, etc.--and sends it to the UF. The UF then interprets this signal into the binary code that represents the action, such as the number "1" selected from a menu, and passes it to the DMT through the internal-Data Language (i-UDL) for processing. The i-UDL is the interface through which all data is passed between the DMT and the UF. In a similar fashion, the UF relays the correct control sequences to cause the output device to display, print, etc., according to the information passed to it by the DMT via the i-UDL.

3.6 SUMMARY

The goal of the UFRM is to provide a framework within which transportable components for a user facility can be constructed. These components fit between the user environment and the DBMS, more specifically between the user and the DMT.

A mechanism, the A3M, is proposed as a starting point for the UDL. The A3M would be used to "phrase" the traffic between the user and the UF across the UDL interface.

The figures below restates the problem. Reviewing current technology indicates that most systems expect the user to interact with the DMT layer directly. The tools have "built in" user interfaces. Figure 3.5 shows such a situation.

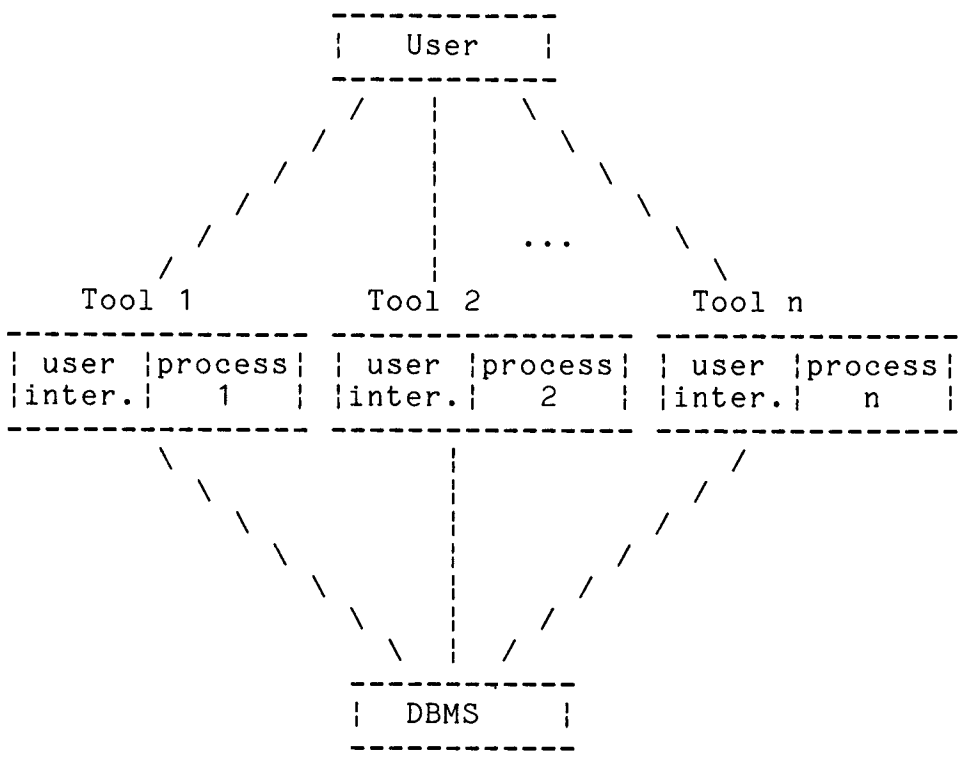


Figure 3.5 Organization of current tools

The UFRM and DAFTG RM suggest the configuration shown in Figure 3.6.

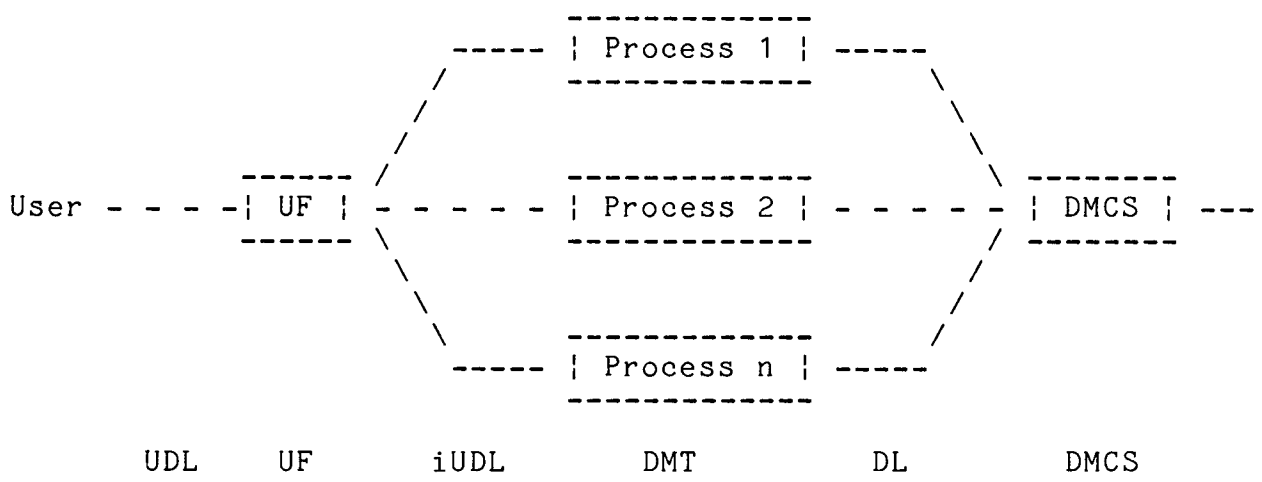


Figure 3.6 UFRM/DAFTG RM configuration

The configuration in Figure 3.6 provides for portability and modularity of system components without a large impact on the user.

CHAPTER 4 CONCLUSION AND FUTURE WORK

The UFRM described and discussed in this document is a consolidation of models from graphics, cognitive psychology, database technology, and theoretical computer science thinking. The UFTG has attempted to mesh the concepts from many disciplines due to the multi-faceted subsystem, the user interface, we were trying to model. Of major consideration was a consistency with the DAFTG Database Reference Model since that was our application focus. We believe that the UFRM is a natural outgrowth of the general database reference model. Throughout this report, we have attempted to explain how our thinking was influenced by the various models we analyzed and utilized to achieve our result.

Future activities are expected to revolve around two general areas: 1. review of existing products, and 2. design and development of new UF products. Initially, we believe an analysis of the effectiveness of present systems and subsystems with unmodularized UFs will be useful to better understand the features that are most useful for certain actions and to extrapolate into a single UF. From that experience, we anticipate some interesting development activities that will incorporate the lessons learned.

It is expected that the UDL and iUDL will be standardized to facilitate future DBMS development.

REFERENCES

[BCS81] British Ccomputer Society, "Query Languages - A Unified Approach", Report of the British Computer Society Query Language Group, Heyden & Son Ltd., Cambridge, Great Britian, 1981.

[FOL80] Foley, James D., "The Structure of Interactive Command Languages", Methodology of Interaction, Guedj (ed.), North Holland Publishing Compny, IFIP, 1980.

[FOL74] Foley, Jaames D. and Victor L. Wallace, "The Art of Natural Graphic Man-Machine Conversation", Proceedings of the IEEE, Vol. 62, No. 4, April 1974.

[FON86] Fong, E.N. and D.K. Jefferson, "Reference Models for Standardization", Institute for Computer Sciences and Technology, National Bureau of Standards, Proceedings of Computer Standards Conference, May 1986.

[HEW73] Hewitt, C., P. Bishop and R. Steiger, "A Universal Modular Actor Formalism for Artificial Intelligence", Stanford University, August, 1973.

[MCD82] McDonald, N.H. and J.P. McNally, "Query Language Feature Analysis by Usability", International Journal of Computer Languages, Vol. 7, Nos. 3/4, 1982.

[TAG85] Tagg, R.M, "Object Modeling in a Generalized End-User System", Unpublished paper, 1985.

[TAG84] Tagg, R.M. and M.Sandford, "Where to now the Mouse has Arrived?", Computer Bulletin (UK), December 1984.

[VAS82] Vassiliou, Y. and M. Jarke, "A Framework for Choosing a Database Query Language", NYU symposium on User Interfaces, May 1982.

GLOSSARY

A3M - see Actor/Act/Action/Message Model

act - that which is carried out by an actor to perform an action (see Actor/Act/Action/Message (A3M) Model)

action - the result of an actor carrying out an act (see Actor/Act/Action/Message (A3M) Model)

actor - that abstration which carries out an act resulting in an action (see Actor/Act/Action/Message (A3M) Model)

Actor/Act/Action/Message (A3M) Model - a model based on artificial intelligence concepts. Actors are agents which carry out acts resulting in actions. Each actor embraces the internally knowledge about itself. The behavior of an actor results from the "actions" taken as it performs an act. An actor may perform any number of acts. For communication, an actor may monitor and send messages to other actors. An actor specified as an example of one or more other actors may inherit properties from these actors. Lateral actors are autonomous and may only communicate through messages.

ANSI - the American National Standards Institute

ASC - Accredited Standards Committee

BCS - the British Computer Society (BCS)

Communications Model - J. D. Foley defines a linguistic approach broken into four levels of abstraction: the conceptual, the semantic, the syntactic, and the lexical levels. The Foley Model consists of four layers. As the highest layer, the conceptual layer, defines the key application concepts of objects, relationships, and operations on these objects in terms understood by the user. The semantic layer defines the services needed for each conceptual operation as specified by the user to the machine and the machine to the user. The syntactic layer defines sequences or the grammar of the tokens exchanged between the user and the machine. The lexical level associates the primitive operations of the support hardware with the tokens of the syntactic level.

DA - see Data Administrator

DAFTG - the SPARC/DBSSG Database Architectural Framework Task Group.

Data Administrator (DA) - Some organizations separate this role into a DBA subrole that is concerned with interfacing to the

application programmer and the data administration (DA) role that is concerned with the enterprise model. The DA must model an enterprise with his understanding of the real world data and its relationships as opposed to the Database Adminstrator (DBA) who has a high level of knowledge about the metadata and medium knowledge about the real world data.

Data Language (DL) - the single interface to the services of the Data Mapping Control System (DMCS). It is the data manipulation language for the fundamental data model of the DBMS. The DL supports both data and meta-data management.

Data Management Tools (DMT) Layer - Data Management Tools processor is a collection of software components that are plug-compatible with the DMCS through the DL-interface. A sample DMT could be a data definition language macro facility which would transform DDL statements into sequences of DL statements. The DMT supports high-level functions such as high-level query tools. Each tool may have its own user interface.

Data Mapping Control System (DMCS) - a "core" database management system (DBMS) providing operators for data manipulation and data description, supporting and enforcing the ANSI/SPARC dimension of external-, conceptual-, and internal-schemas and the orthogonal intension-extension dimension of data model-, data dictionary-, and application schemas, and application data. An active/dynamic data dictionary system is an integral part of the DMCS.

Database Administrator (DBA) - deals with the metadata for the DBMS. This includes the definition of the data elements, the enterprise and logical structure of the data, different user views, and permissions for security and privacy. Some organizations separate this role into a DBA subrole that is concerned with interfacing to the application programmer and the data administration (DA) role that is concerned with the enterprise model. The DBA has a high level of knowledge about the metadata and medium knowledge about the real world data as opposed to the data administrator who must model an enterprise with his understanding of the real world data and its relationships.

DBA - see Database Administrator

DBMS Reference Model - consists of three components, each communicating via a language interface. The Operating System (OS) supports the DBMS and communicates with the Data Mapping Control System (DMCS) through the internal Data Language (i-DL). The most recent model was proposed by the DBSSG/DAFTG and is currently under review by ISO for international standardization.

DBSSG - the X3/SPARC Database System Study Group

DL - see Data Language

DMCS - see Data Mapping Control System

DMT - see Data Management Tools

end-user - the fastest growing and largest segment of DBMS users typically considered to be composed of two sub-segments depending on frequency of use: the casual user and the daily user.

extensional information - the information being described by intensional information.

FTOC - see fundamental theory of computation

fundamental data model - a data model that supports the established data models such as the relational-, the entity-relationship-, the network-, the object-role models.

fundamental theory of computation (FTOC) - is simply that a machine, concrete or abstract, produces a result based on descriptions (e.g., data, instructions) operated upon by a set of description-manipulating functions (sometimes called the interpreter).

Host - the computational host for the DMCS. The DAFTG defines Operating System (OS) as part of the environment of a DBMS which here is considered to be part of Host.

Host Environment Actor - a more general concept for the DAFTG Operating System. Here, the OS is consider data being operated upon by the hardware. (Note, the firmware and microcode levels are considered part of the hardware for simplicity.) This actor also closely corresponds to Foley's lexical layer. There is no good correspondence to the OSI except it might be said that it would cover the OSI session through physical layers.

internal Data Language (iDL)- the single interface between the DMCS and the OS through which all data is passed whether real data, meta-data, meta-meta-data, etc.

internal User Data Language (iUDL) - the single interface to data management tools thus, the user is not required to learn what may be the specialized language requirements for each data management tool. The iUDL provides the interface between the User Facility (UF) and the Data Management Tools (DMT). The iUDL provides a mechanism to categorize the connections needed by various tools, thus providing for reusability of UF implementations.

intensional information - the information that is description of a description

ISO - International Standards Organization

message - see Actor/Act/Action/Message (A3M) Model

OSI - see Open System Interconnection Reference Model

Open System Interconnection Reference Model (OSI) - the 7 - layer architecture for autonomous communicating systems as specified by the International Standards Organization (ISO).

reference model (RM) - conceptual framework for a subject area, divides standardization work into manageable pieces and to know at a general level how these pieces are related with each other.

SPARC - the Standards Planning and Requirements Committee of the accredited standards committee Information Processing Systems (X3) of ANSI.

Terminal Control System (TCS) - see User Facility (UF)

UFTG - see User Facility Task Group

user - the UFTG's abstraction to represent and describe any user of the user facility; here, only those properties of a user involved in using a database management system are considered.

User Data Language (UDL) - the single interface to the UF. Most DBMSs have user languages built-in to the DBMS. Under this model, the user language is externalized relative to the DBMS. Other languages are possible or an existing language may be extended or adapted in the UF to accommadate the user. The UDL interface accepts the signal from the input device--whether it is a keyboard, light pen, audio or video response, etc.--and sends it to the UF.

User Facility (UF) - contains software components which process messages from the user and in turn directs and accepts data to and from the database management system as defined by the DAFTG of the DBSSG. It is the major component of the User Facility Reference Model and is the control system for the input and output devices.

User Facility Reference Model (UFRM) - serves as a front-end to the DBMS Reference Model and, together form a functionally compatible means of describing interaction between the user and the system. It is composed of five processors and four interfaces. The processors are: the User, the DMT, the DMCS, and the Host. The interfaces are: the UDL, the iUDL,

the DL, and the iDL.

User Facility Task Group (UFTG) - a task group of the SPARC/DBSSG developing a reference model for the user facilities of a data base management system.

virtual user - a software/hardware agent of some human or group of humans. The range of this user is limited only by intelligence built into the processes which represent this type of user.