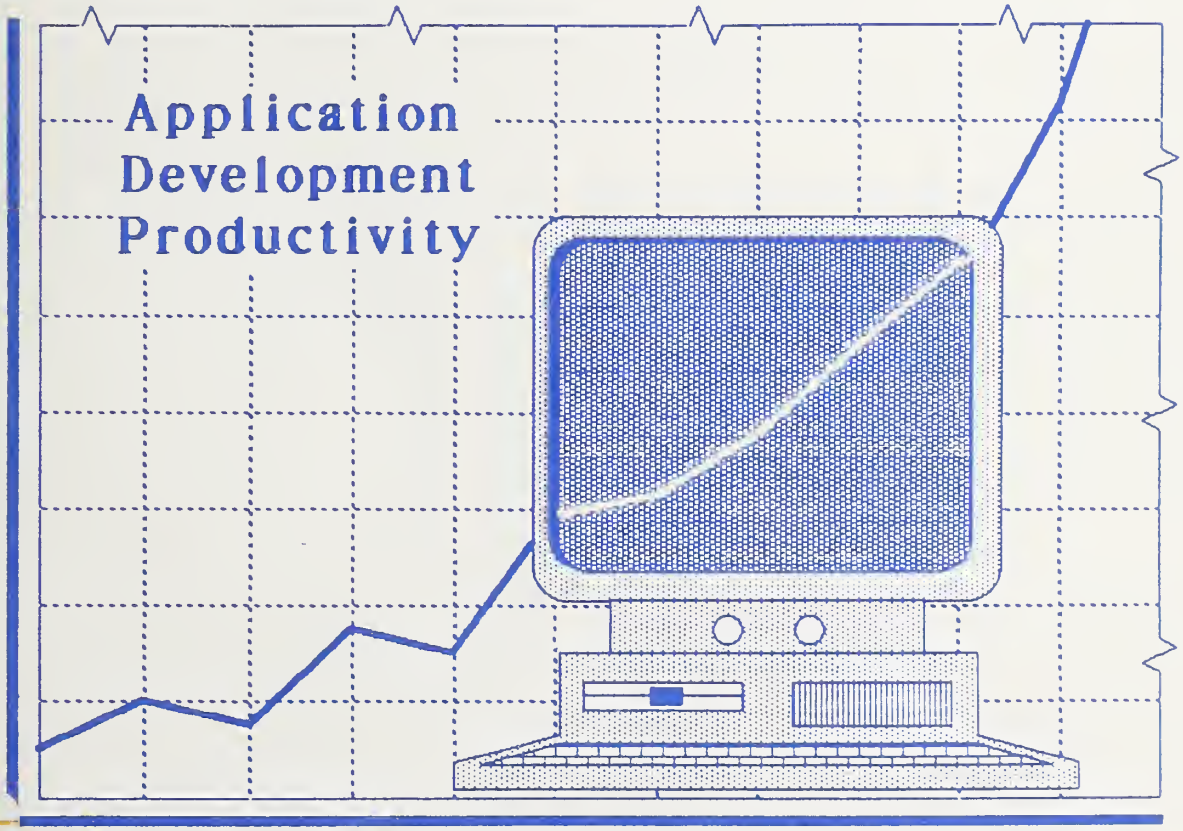
Computer Science and Technology

NBS Special Publication 500-138

A Functional Model for Fourth Generation Languages

Gary E. Fisher



The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, the Institute for Computer Sciences and Technology, and the Institute for Materials Science and Engineering.

The National Measurement Laboratory

Provides the national system of physical and chemical measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; provides advisory and research services to other Government agencies; conducts physical and chemical research; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

- Basic Standards[2]
- Radiation Research
- Chemical Physics
- Analytical Chemistry

The National Engineering Laboratory

Provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

- Applied Mathematics
- Electronics and Electrical Engineering[2]
- Manufacturing Engineering
- Building Technology
- Fire Research
- Chemical Engineering[2]

The Institute for Computer Sciences and Technology

Conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

- Programming Science and Technology
- Computer Systems Engineering

The Institute for Materials Science and Engineering

Conducts research and provides measurements, data, standards, reference materials, quantitative understanding and other technical information fundamental to the processing, structure, properties and performance of materials; addresses the scientific basis for new advanced materials technologies; plans research around cross-country scientific themes such as nondestructive evaluation and phase diagram development; oversees Bureau-wide technical programs in nuclear reactor radiation research and nondestructive evaluation; and broadly disseminates generic technical information resulting from its programs. The Institute consists of the following Divisions:

- Ceramics
- Fracture and Deformation [3]
- Polymers
- Metallurgy
- Reactor Radiation

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Gaithersburg, MD 20899.
[2]Some divisions within the center are located at Boulder, CO 80303.
[3]Located at Boulder, CO, with some elements at Gaithersburg, MD.

Computer Science and Technology

NBS Special Publication 500-138

A Functional Model for Fourth Generation Languages

Gary E. Fisher

Center for Programming Science and Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, Maryland 20899

Issued June 1986

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

Library of Congress Catalog Card Number: 86-600545
National Bureau of Standards Special Publication 500-138
Natl. Bur. Stand. (U.S.), Spec. Publ. 500-138, 36 pages (June 1986)

U.S. GOVERMENT PRINTING OFFICE
WASHINGTON: 1986

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

PREFACE

This report has been prepared by the Institute for Computer Sciences and Technology in response to requests for information and reference materials made by Federal Government and private sector organizations over the past year. ICST formulated a schedule of tasks designed to organize and research the area of **Fourth Generation Language (4GL)**, and provide the needed information. Two products of these tasks are a forthcoming **guide on** the **selection and use of 4GL**, and a **functional model** designed to solidify the concept of 4GL into an objectively definable entity.

A workshop on **Application Development Productivity** was held at the National Bureau of Standards in Gaithersburg, Maryland, on November 13-15, 1985. The first day of the workshop was a plenary session on Fourth Generation Languages. Much of the discussion centered on an urgent requirement for information and guidance on 4GL. This impetus set the stage for accelerating the dissemination of information and eventually led to the publication of this model.

This version of the Functional Model is a result of long and arduous discussion and evaluation by my fellow members of ICST, Martha M. Gray, James Hall, David Jefferson, Elizabeth Fong, and Joan Sullivan. I would like to thank them for their help and guidance.

EXECUTIVE SUMMARY

This report proposes a **Functional Model for Fourth Generation Languages (4GLs).** The purpose of this functional model is to define Fourth Generation Language in a manner similar to specifying the functions of a specific software application. This definition process allows managers, technical personnel, and end-users to refer to a **commonly understood terminology** in the 4GL context. In addition, the interfaces between 4GLs and external entities (i.e. humans, operating systems, peripheral devices, and other application systems) can be identified and studied for research purposes and possible standardization.

The capabilities provided by 4GLs are grouped into three major areas based on similarities in overall function. They are--

- User functions;
- Data management functions; and
- System functions.

User functions define those capabilities necessary to provide a high level dialogue between the 4GL and users of the 4GL. Users of 4GLs may include humans and other systems.

Data management functions provide capabilities to describe, store and retrieve, and perform ancillary tasks in the management and safekeeping of application data.

System functions provide the support services necessary to allow the user of 4GLs to define and access applications in relation to the constraints of the environment in which the 4GL operates.

1. INTRODUCTION

1.1 What is a Functional Model?

A **functional model** objectively defines the generic concepts of an entity in terms of functions and services provided by the entity. Fourth Generation Languages are at least one level of abstraction removed from specific applications implemented in 4GL. Therefore the Functional Model is conceptually at a much higher level than functional specifications used to define applications. Functional model relates to functional specification in the same way that the term "transportation" relates to automobile or airplane.

The 4GL Functional Model provides a basis for defining a commonly **understood terminology** in referring to 4GL, and for describing the services and functions provided by a 4GL to external users. The users of 4GL may include humans and software systems. The functional model, in this case, is specified by a list of functions and services that 4GL is expected to provide. This list is organized into meaningful groupings corresponding to interfaces between the model and specific types of users. The International Organization for Standardization's (ISO) Open Systems Interconnection layered architecture is an example of the functional model approach [Inte84] where open systems interconnections (communications interfaces) are modeled.

1.2 Objectives of the 4GL Functional Model

The objectives of this functional model are--

- to serve as a basic **definition** of 4GL concepts;
- to define the major **capabilities** of 4GL;
- to facilitate the **training** of personnel by providing a common framework for describing 4GL;
- to allow objective **classification** of vendor implementations of 4GL; and
- to aid in **reviewing, evaluating,** and **introducing** 4GL into an organization.

Although the model itself is not a proposal for a standard, it provides a framework for future study and possible standards research. Important benefits may be achieved through standardization of languages, product features, and interfaces between products. It is up to users, vendors, managers, and teachers to

become aware of potential applications of these standards. Potential benefits of standardization include, but are not limited to, the following:

- o **Transportability** of applications through common languages, database management systems, user interfaces, and operating systems.
- o Improved staff **productivity** and lowered training costs from a reduction in required training and retraining time.
- o **Simplification** of 4GL evaluation and selection.
- o Increased feasibility of **data interchange** between applications.

1.3 Scope of the 4GL Functional Model

Fourth Generation Languages encompass many diverse mechanisms for organizing, entering, and retrieving data in the domain of interactive on-line business data processing applications. The functional model is designed to give managers technical exposure to the capabilities of 4GL. As a consequence of this direction, the functions and services are described from a perspective that includes both an end-user's and professional programmer's views.

4GL may be used in those applications that are classed as management support systems, transaction processing systems, and other general business applications. Several vendors are in the process of adding components to their 4GLs to allow 4GL use in other types of applications that are far removed from typical business applications, such as patient monitoring, or command and control systems. This report will concentrate on 4GL definition in the business data processing systems domain.

1.4 Organization of the Report

This report provides a description of the 4GL Functional Model, background information, and the 4GL environment. Section one described introductory material on the form of the functional model.

Section two provides general background information on the definition of language generations, and what a 4GL is and does. Section three formally presents the 4GL Functional Model. Specific layers, components, and interfaces of the Model are described. Section four concludes the report with a summary, and expectations for the future of 4GLs.

2. WHAT IS A 4GL?

2.1 Language Generations

There is no general agreement on what constitutes a language generation. A 4GL may not even be "fourth generation" depending on whose definition of language generation is used. Some experts call machine code the first generation, while others describe assembly languages as the first.

Sippl defines machine code as a "programming language" because it can be used to represent a program [Sipp72]. Jean Sammet, however, left machine code out of the realm of programming languages by defining "programming language" as a "set of characters with rules for combining them" [Samm69]. One characteristic of this set of rules is that knowledge of machine code is unnecessary. In this definition of language, Sammet has explicitly excluded machine code.

The **American National Dictionary for Information Processing Systems** by contrast refers to machine code as the instruction set executed on a computer. No reference to language is made [ANSI82].

James Martin defines language generations as follows:

- First - machine language
- Second - assembly language
- Third - machine-independent [standardized] languages (COBOL, PL/I, BASIC, etc.)
- Fourth - nonprocedural end-user oriented languages (RAMIS II, MANTIS, IDEAL, etc.) [Mart82]

Another aspect of the debate on language generations stems from the relatively parallel development of languages and hardware. First generation languages were used on first generation computers (vacuum tube memories), second generation on transistorized computers, third on integrated circuit computers, and fourth generation on very large scale integrated circuit machines. Although first, second and third generation languages are also used on fourth generation machines, the argument can be made for classifying languages based on the generation of hardware in which each language reached prominence.

The term **Fourth Generation Language** is somewhat misleading. 4GLs are referenced by various authors as programming languages, system specification languages, nonprocedural languages, query

languages, end-user languages, application development languages, and so on. This multiplicity of terms stems from the diverse features offered by vendors, and different uses of 4GLs in the overall scheme of software evolution. A survey of 4GL definitions presented by various authors [Auer85, Inte84b, Inte84c, Mart84] shows a few common ideas, but the uncommon ideas lead to further confusion. Some of the recurring concepts in 4GL definitions found in computing literature are--

- an integrated database management system;
- a nonprocedural language subset geared for non-technical end-users;
- a nonprocedural and procedural language superset geared for professional data processing personnel;
- a screen generator;
- operating system independence (or transparency);
- a 10 to 1 improvement in programming productivity as measured in lines of code when compared to third generation languages (Harel and McClean contend that a 4GL will produce the same application in less code than will COBOL, but that COBOL will use less resources than the 4GL application in executing [Hare82].);
- a report generator; and
- a query language.

Atre [Atre85], in an article published on languages beyond third generation, presented another twist by asserting that the software industry is not quite at the Fourth Generation Language stage. Almost all 4GLs purport to be "user-friendly" when described in the end-user context, but Atre maintains that what really exists are 3.5, 3.7, and 3.9 generation languages which are "applications-programmer-friendly," "professional-friendly," or "non-professional-friendly" languages.

Unfortunately, most of these characteristics cannot be used in objectively defining the functions of a 4GL. Therefore, characteristics like "nonprocedural," "10 to 1 productivity increase," "query language," and "database management system" are not included in this model. Instead, **functions and services that may be provided by components of a 4GL are used to define the model.** A discussion on some of these general characteristics will illustrate the difficulties encountered in including them as part of the functional model.

2.2 General Characteristics of 4GL

Tools, by design, are suited to particular functions and needs. For example, a wrench is not used to drive wood screws, nor is a screwdriver suited to sawing wood. Each tool has its specific application domain, even though one may attempt to try to perform tasks with the wrong tools. The same is generally true of 4GLs.

Fourth Generation Languages are eminently suited to the development of interactive on-line business data processing systems. Budget, payroll, order-entry, and general accounting systems are prime candidates as the types of applications that currently fit the 4GL domain.

The objective of a 4GL is to hide many of the housekeeping chores of programming from the user. On the other hand, this same feature will inevitably provide many stumbling blocks for those applications that require unusual manipulation and computations.

Complexity of the application has an effect on the ability of a 4GL to handle the application. Most 4GLs operate up to a vague level of complexity, beyond which an application is usually less difficult to develop in a lower level language. This level of complexity is defined by the amount of application detail hidden in the 4GL. One of strong points of a 4GL is its ability to let the user view an application at a higher and more abstract level than is possible with most third generation languages. However, because application details are hidden by the abstraction process designed into the 4GL, a 4GL, is out of necessity, more rigid and inflexible in defining an application than are lower level languages. The insulation that protects the user from the system also protects the system from the user [Inte84a].

2.2.1 Compiled or Interpreted Code

The importance of executing compiled or interpreted code that is produced by a 4GL will depend on the type of application generated. For example, some 4GLs produce COBOL, PL/I, or other third generation code which is then **compiled** into machine code for execution. Others generate internal proprietary code which is **interpreted** by a run-time system. Usually, compiled code executes faster on a particular machine than does the interpreted code [Hare82].

Interpreters offer other capabilities that are not found in most compilers. For example, interactive debugging is inherent in many interpreted languages. This capability allows the user to execute a program, suspend it at an appropriate point, change data element values or control structures, and continue execution. Interpreted code may be better for those applications that are classified as "quick and dirty," or developmental, while compiled

code may be more appropriate for production systems. Thus, the applications may well dictate the selection of a 4GL.

Whether a 4GL produces compiled or interpreted code is not a suitable element to include in a functional model. The production of one form of code or another is purely a matter of implementation and not of function.

2.2.2 Nonprocedural or Procedural

"Procedural" is defined in the programming sense as the describing of the course of action taken for the solution of a problem in a finite series of steps. "Nonprocedural" states that the solution of a problem is not dependent on the order in which these steps are executed. These definitions reduce more or less to the following:

- **Procedural - "how" a problem is to be solved**
- **Nonprocedural - "what" problem is to be solved**

If one uses the definition that nonprocedural means "non-order dependent problem-solving," there are no nonprocedural 4GLs. In every case, the order in which the problem is stated does make a difference. For example, virtually all database management systems require knowledge of the structure of the user's database before data may be stored or retrieved. Those that do not require this have built in knowledge of a data structure and force the user to organize the data in this structure.

This characteristic is not included in the functional model. Intuitively, a user may be able to state that a 4GL is nonprocedural, but there is currently no meaningful measure that can be applied to this term to determine if a 4GL meets the requirements of "nonprocedurality."

2.2.3 Productivity Improvement

James Martin [Mart82] and others assert that 4GL leads to productivity improvement in the evolution of new systems. This may be true when source code is used as the basis of productivity measurement. (See [Hare82].) However, there is some difficulty in accepting this as the primary measurement of productivity. One programmer's source code may be another programmer's maintenance nightmare. Producing code can be very different from producing good, clean, efficient, maintainable code.

Other factors in software evolution such as elapsed time of development, execution error rate, dollars spent, etc. may all be

just as valuable in measuring productivity. These factors are still saddled with the problem that there is no objective way of defining absolute reference points (standards) against which productivity can be measured. Relative measures based on any of these factors can indicate whether or not wise decisions have been made in the evolution process, but a functional model must include only those criteria that are, at a minimum, based on general acceptance by the user community.

2.3 Users of 4GL

A Fourth Generation Language may be used by humans in the context of application evolution, or in the actual processing of data. In addition, other application systems may also be characterized as users of a 4GL. This characterization is necessary in order to differentiate between applications or systems that interact with 4GLs at a conceptually high level, and operating systems that interact with 4GL at a lower level in providing services to the 4GL.

2.3.1 Human Users

Human users of 4GL fall into two broad categories: technical data processing professionals and nontechnical end-users. Nontechnical end-users view 4GL as a day-to-day working tool used to support information needs as they arise during the course of business. It is not uncommon for a trained end-user to generate a formatted report in less than a day. Queries may be generated in hours or minutes.

Professional data processing personnel may take a different view of 4GLs. The greatest benefit of a 4GL, in the eyes of the professional data processor, may be its ability to quickly create the logical structure of a system. However, as is the case with its 3GL cousins, a 4GL lacks components to handle specific aspects of system evolution that are outside the area of program/code generation, such as requirements analysis, testing, and maintenance. A 4GL is useful in structuring one or two phases of the software life cycle, but it is not the total solution to software evolution problems.

The task of a data processing professional is to use available techniques to effectively and efficiently build software systems that support the real needs of end-users. These end-users should not be concerned with how the professional data processor performs development tasks. Instead, they should concentrate on what their real information needs are and develop the ability to communicate those needs. A 4GL offers a conduit to help in this communication process.

2.3.2 Other Users

4GLs may act in concert with other applications and the operating system. These applications may use a 4GL to retrieve files of data explicitly generated by the 4GL for this purpose. An example of this is illustrated in the extraction of data from a database by the 4GL for downloading to another computer.

The operating system may use a 4GL to act as the primary means of communicating with end-users. For example, this interaction may exist specifically to allow novice end-users to access an on-line database of operating system tutorials or to provide command menus.

3. 4GL FUNCTIONAL MODEL

Figure 3-1 illustrates the functional view of a 4GL. The three major functional areas are--

- o **User functions;**
- o **Data management functions;** and
- o **System functions.**

Each of these areas is discussed in the sections that follow. Implementation of specific functions is not the task of this report. However, where applicable, specific examples may be used to illustrate features that may be found in implementations that are currently available.

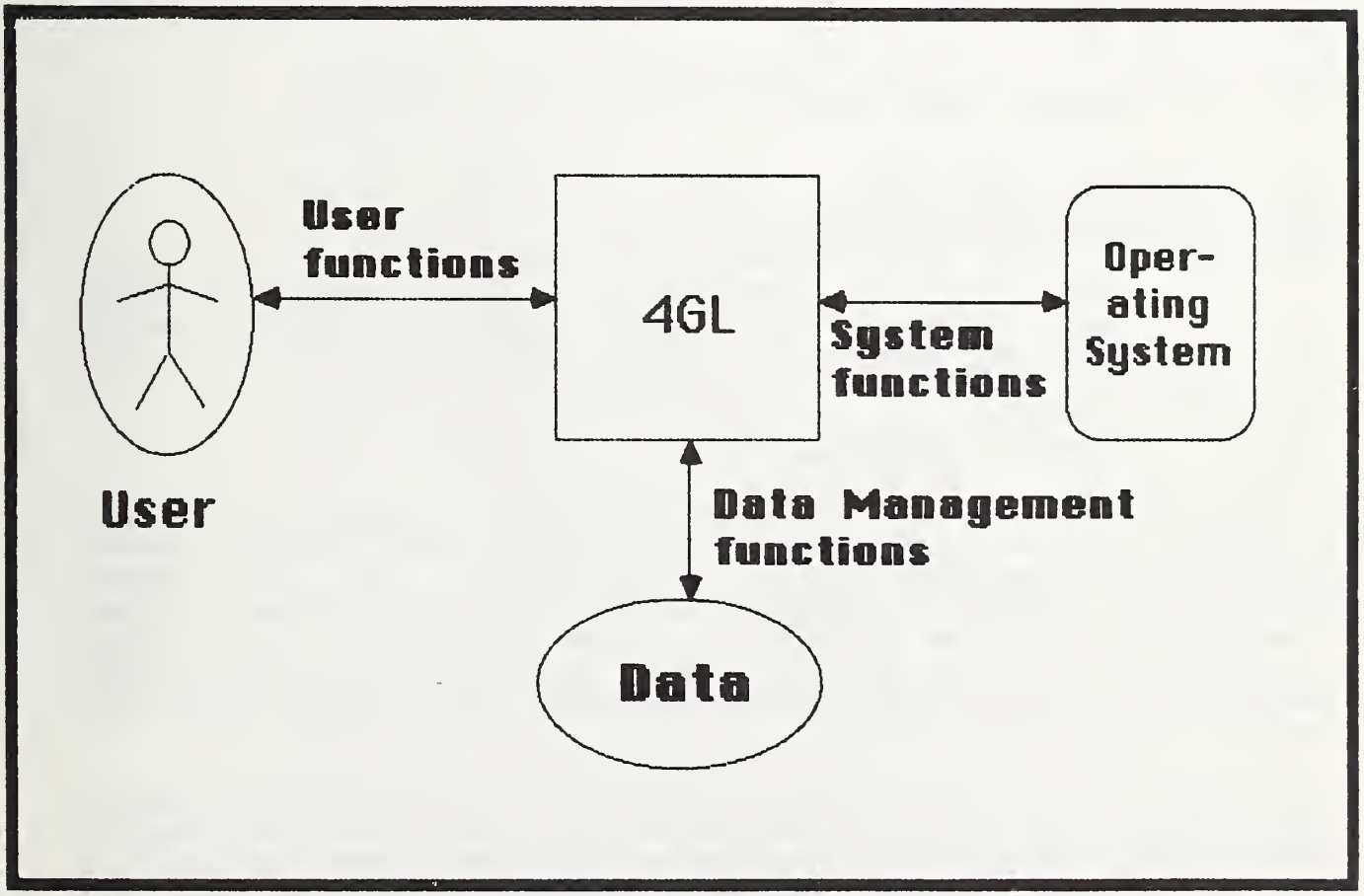


Figure 3-1. Functional View of 4GL

3.1 User Functions

The subjective characteristics of a user (i.e. whether or not the user is a 4GL novice, how experienced the user is in the application domain, etc.) influence how that user will interact with a system. For example, a novice end-user may want to rely on full menus and help screens to perform tasks using the 4GL. A professional data processor may not want to wade through many levels of menus, and may choose a more cryptic form of command entry instead. Still other users may feel comfortable with command entry by mouse in some 4GL components, but may revert to keyboard entry in others.

User functions define those capabilities and services provided by a 4GL to address the interaction between system users and the 4GL. These functions define a high level dialogue management capability in the sense that much of the housekeeping concerning the interaction between the 4GL and the user is managed and performed by the 4GL. This area is further broken down into the following specific functions:

- Screen formatting
- Menu management
- Message prompting
- Logical device management for devices such as light pen, touch-screen, mouse, graphics tablet, remote sensors, etc.

3.1.1 Screen Formatting

There are numerous ways of formatting a display screen. Examples include scrolled question-answer dialogue, full-screen cursor positioning, graphic symbol manipulation, etc. The screen formatting function concerns itself with only the logical description of the screen (i.e. the description of screen formats, the types of data that are displayed, etc.), and not the transformations necessary to produce what the end-user physically sees on the screen.

A screen's logical description may consist of information such as the name of the character set used, the type of terminal, screen dimensions, positioning coordinates for each item displayed on the screen, actual data values or literals to be displayed, and physical field attributes such as format and justification.

Examples of capabilities that implement some of the screen formatting functions include the following:

- **Field character validation** (only valid numbers may be entered in numeric fields and only alphabetic characters in alphabetic fields)

- **Bell/buzzer** - recommended for all autoskip fields and error occurrences

- **Field mask** (specific character strings or a mixture of character types may be specified as the only acceptable entries; for example, a code identifier may be defined as "@###", where the first character may be defined as alphabetic and the last three as numeric)

- **Required fields** (the cursor will not pass a field that requires an entry until one is made)

- **Filled fields** (the cursor will not leave a field that must contain a minimum number of characters till that field has been entered)

- **Autoskip** (the cursor automatically jumps to the first character of the next input field once the current field is filled)

- **Video intensity** (reverse video, bold versus faint characters) - particularly useful to form visual cues in guiding the user on screens that are not frequently used

- **Computed fields** (field is computed and filled-in based on other entries made by the user or default values) - recommended for applications that require time-consuming computation or complex entries

- **Default field values** (saves time in exception oriented data entry)

- **Screen/data field refresh** (the last entry in each field stays visible until specifically erased by the user)

- **Inter-field checks** (entries in fields are checked against entries in other fields or separately defined tables to maintain consistency in the data)

- **Inter-record checks** (contents of a screen and the associated action - add, change, delete - are checked against records in the data files to insure that the action results in consistent data)

3.1.2 Menu Management

Menus are conceptually a subset of formatted screens, and may be used in contexts different from those in which formatted screens are used. Menus generally are used to channel users into specific task performance, whereas formatted screens are associated more with data entry and display. Limited selection of actions is a typical feature of menus.

3.1.3 Message Prompting

Not all terminals are equipped with CRT screens. Many use thermal, electrostatic, impact, or other types of printers to display information on paper. Screen formatting in such cases is not feasible. Instead, one-line messages may provide a better format for information display. The user may make various types of entries based on the response required.

3.1.4 Logical Device Management

The growth of peripheral input/output devices has grown to the extent that it is now feasible to use multiple types of devices in concert with applications without incurring large expense. Examples of such devices include voice recognition hardware, touch sensitive screens, light pens, mice, graphics tablets, and remote sensors.

Each of these types of devices produces characteristic data that may be translated from analog to digital form and fed through an electronic medium directly or indirectly to an application. 4GLs allow these devices to be connected to applications through logical interfaces provided for that purpose. To a 4GL, these differing devices should appear to be the logical equivalent of other input/output ports such as those associated with the keyboard or disk drives.

3.2 Data Management Functions

The area of data management in the context of 4GL includes capabilities necessary to describe data structures, store and retrieve instances of data, and provide facilities to secure the content and integrity of the data. Functions in this area include the following:

- Logical data structure management
- Data storage and retrieval
- Archiving and restoration

- Auditing
- Data security

Most of these functions are predominantly found in database management systems (DBMS). However, the functional model makes no assumptions about the implementation of these functions in actual usage.

3.2.1 Logical Data Structure Management

A 4GL must provide facilities for recording, storing and processing descriptions of data structures to be used in an application. The mechanism for this is usually located in the DBMS. The importance of this function to the 4GL and the user is that data structures need be defined only once for numerous applications or uses within an application. Once defined, these data structures may be included automatically in retrievals, dialogues, and procedures simply by referencing data item names within the structures.

The inclusion of this function in a 4GL will allow the user to ascertain other important information about the application. For instance, modifications to data structures may cause ramifications throughout an application. The user should be able to identify how and where these changes will affect the application by listing references to data structures in association with the modules that use these structures.

Data structure management also includes the ability to copy one structure into another, rename structures, delete structures, and reorganize data structures application-wide. Generally, vendors implement these functions in the component that has become known as a data dictionary.

3.2.2 Data Storage and Retrieval

Data storage and retrieval functions include the capabilities to add, delete, and modify instances of data in primary and secondary storage (i.e. floppy diskette, magnetic tape, rigid disk, etc.); and to perform transformations on this data for displaying it in forms other than those available through the user functions. These forms include printed reports, plotted graphics, and other output media.

Data storage and retrieval may be performed through components, such as query languages and report generators as implemented by many 4GL vendors, or through a single command language that

contains capabilities of performing the actions of several components. These capabilities should include functions similar to those described in the following paragraphs.

With a report generator, a user may define report layouts in several different ways: interactively through a question and answer dialogue at a terminal, through a predefined command file, or by other means. The report may be the result of executing a program generated by the report generator, or interpreting the report commands directly. The ultimate goal is that the 4GL produces printed user-defined reports based on data that is available within the application and described through catalogued logical data structures.

Query languages are used primarily for querying or browsing through the data (hence, the name) in an ad hoc manner, but the implementation of query language in a particular 4GL context may include data modification capabilities (e.g. joining tables, creating temporal relationships, updating, deleting, etc.).

3.2.2.1 End-user Language

Part of the meaning of language in the term **Fourth Generation Language** is based on the existence of a programming language designed for use specifically by 4GL end-users. This language may overlap somewhat with capabilities provided by the user functions area. Typically, an end-user will not differentiate between the command language used to operate a 4GL and the language used to execute reports, display screens, and define procedures.

Examples of the types of commands found in end-user languages are--

- **COMPUTE, ADD,** and **SUM** for performing numerical computations;
- **SELECT, JOIN, and DISPLAY** for retrieving and presenting data in a relatively straightforward form such as would be found in many query components; and
- **PRINT, SUBTOTAL, and TABLE LOOKUP** for retrieving and presenting data in tabular formatted form such as in printed reports.

3.2.2.2 Professional/Technical User Language

Many Fourth Generation Languages available today include a more comprehensive language for use by professional programmers in creating extensive applications with the 4GL. The constructs available in this language are not application specific and

usually require much more technical expertise in application evolution than is observed in most 4GL end-users.

The capabilities of this language include mechanisms for manipulating virtually all components of the 4GL. It is here that many 4GL vendors implement the commands necessary to perform screen formatting, report generation, and procedure definition.

3.2.3 Archiving and Restoration

4GLs must possess the capability of archiving copies of all data and application code to a secure medium. These copies will compose the backup of the application and system data in the event that a system failure occurs, or the system must be restarted at some previous instant in processing. Either the 4GL must allow the user to define the archival procedure specifically, or the 4GL must perform periodic checkpoint dumps of data to meet this requirement.

Conversely, a 4GL must allow the user to invoke utilities to restore the application or data from the archived information. Presumably, the 4GL will include enough information in the archive to allow most, if not all, of the system recovery to take place automatically. This may be effected through specific automated procedures, or through manual procedures that can be executed by end-users (i.e. procedures that do not require professional assistance).

3.2.4 Auditing

The ability to reconstruct the status of an application at a particular instant of time through a repeatable procedure has legal impact on many organizations. Accounting systems, as well as many other types of applications require the capability of verifying system status through external audits of the system. A 4GL may provide the functions necessary to record and track changes to the data. Normally, this is done through automatic logging of transactions as they are processed by the application. In addition, system accesses may be logged by the 4GL to allow analysis of when and by whom these changes were made.

3.2.5 Data Security

Security includes the concepts of data protection and confidentiality, and data integrity. Data integrity is included because it exists in close kinship with data protection, although it may be implemented in components along with data storage and retrieval functions, or user functions.

As a minimum, a 4GL should provide data protection at the file, record, or table level. Two common methods of implementing this requirement are password protection and data encryption. Data integrity may be implemented partially through data storage and retrieval, such as in the use of primary and secondary keys and indexes; and partially in user functions, such as in formatted data entry screens.

3.3 System Functions

The environment in which a 4GL operates may contain more capabilities than those available as part of the 4GL. The ability to access these capabilities through the 4GL is an integral function of 4GLs in general. These functions include, but are not limited to, file handling, job control, communications, and other applications. Each of these is described in the following paragraphs. The minimum requirement of the 4GL in this particular case is that it be able to access system capabilities that are not directly a part of the 4GL.

3.3.1 File Handling

File handling includes file management, transfer of files, file editing, and other actions that affect files as a whole. File management at the system level allows the user to set and retrieve pertinent information about files such as file access modes (i.e. read, write, extend), buffer size, file size, file location, directory entries, etc. Files may be transferred from one device to another such as in copying from fixed disk to a removable disk. Operating systems may include programs to edit files based on physical structure rather than logical structure as is most often found in 4GLs.

3.3.2 Job Control

A 4GL will generally allow the user to execute other applications that are accessible only through the operating system. These would include applications that have no direct access from the 4GL. For example, the user may use a 4GL to maintain statistical data files, and a separate application package to actually perform statistical calculations. The user may store the data using the 4GL in a form that is usable by the statistics package, and then suspend 4GL execution by calling the appropriate statistics routines through the operating system. Once the statistics routines have finished, the operating system would return control to the 4GL to perform the next task.

Using predefined job control routines, the 4GL user may invoke totally independent background tasks to execute outside of the

4GL in a timesharing or distributed computing environment. The user may continue executing the 4GL while the background tasks are executing in batch mode on the same machine, or on another machine in a network.

3.3.3 Communications

Communications includes two overlapping areas: communicating over dedicated networks, and communicating over multi-purpose networks such as voice-grade telephone systems. Both types of network require specialized hardware and software to implement communications between the network nodes. The hardware is used to transform digital information into analog counterparts for transmission over wires, microwaves, fiber optics, or radio/TV transmitters. The software codes and decodes the digital information transmitted and received by each computer in the communication exchange.

A communications capability is essential for implementing distributed applications in a 4GL, or for enabling developers to send and receive information on individual workstations through electronic mail software. Application evolution can be enhanced through the sharing of common data and utilities that are accessible to many users through the communications network.

Communications capabilities may be implemented in various methods which are dependent on the 4GL vendor's concept of communications. Some 4GLs provide a direct interface to communications ports, while others provide a separate utility that is accessible through the 4GL's system function interface.

3.4 Advanced 4GL

The functions described in the previous sections represent a minimal 4GL. This does not prevent a 4GL from providing additional capabilities, such as the following:

- graphics output
- programming language interface (PLI)
- command language for direct access to the operating system
- program/data/text editing capabilities
- debugger/compiler
- real-time control language functions and services

- office automation facilities
- word processing

A 4GL that contains additional capabilities above those in the minimal 4GL may be classified as an **advanced 4GL.**

4. CONCLUSION

4.1 Summary of 4GL Functional Model

A Fourth Generation Language is not a language in the classical sense of programming languages. Instead it is a system of integrated tools designed to assist end-users in developing interactive on-line business applications with a minimum need for knowledge of the technical aspects of data processing.

A 4GL is also suited for use by professional application developers through a high level language that permits rapid development of code and performs much of the general housekeeping associated with programming in a lower generation language.

A minimal 4GL provides the following functions and services:

- User functions
- Data management functions
- System functions

An advanced 4GL provides the minimum set of functions and services plus additional functionality, such as command language processing, graphics manipulation, decision support modeling, artificial intelligence, and other special purpose capabilities.

4.2 Typical Implementation of 4GL

Figure 4-1 illustrates a typical Fourth Generation Language architecture for a component implementation. The components are--

- Query language;
- Screen formatter;
- Report formatter;
- Procedural language;
- Data dictionary; and
- Database management system or file handler.

The user functions generally are implemented across the combination of query language, screen formatter, report formatter, and procedural language components. This is a conceptually easy-to-understand method of providing the required functionality.

However, the functional model does not specify how user functions should be implemented.

For example, a massive all-encompassing procedural language could contain all user functions so the user could interact with just this one part. In this case, the query language is subsumed into the procedural language so there is no need for a query language component. The screen and report formatters change complexion to become processors of commands submitted by the procedural language component. There is then no need to have the individual screen and report components do any more than execute these commands.

The data management functions are implemented almost universally in the combination of data dictionary and database management system components. System access functions, however, may be spread throughout the 4GL with each component capable of gaining its own access to other components and providing sentries to determine if other components may have access.

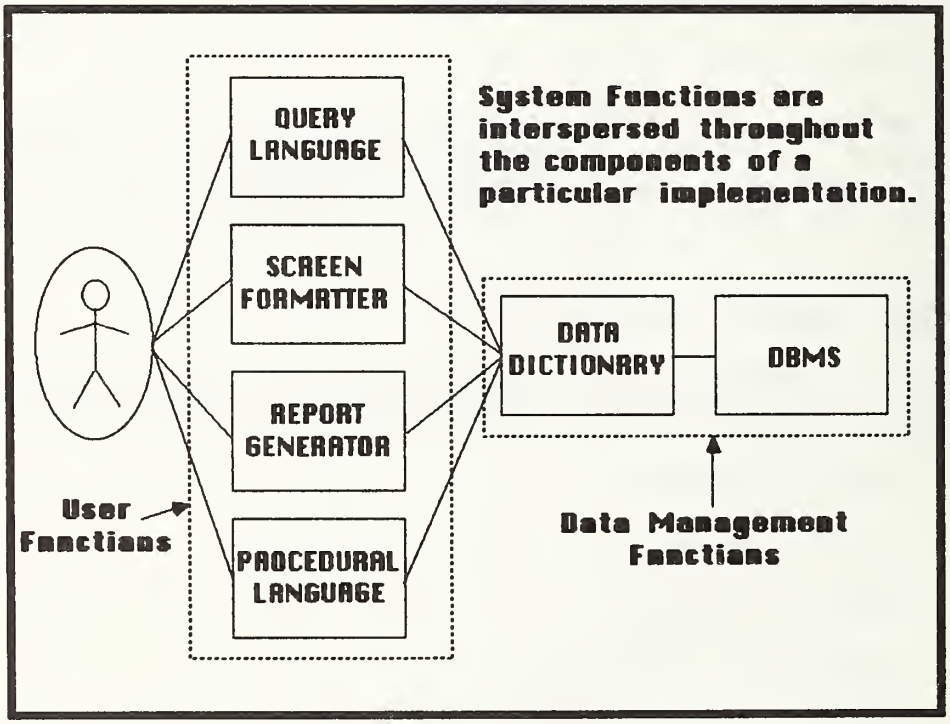


Figure 4-1. Typical 4GL Architecture

4.3 The Future of 4GLs

The 4GLs available today have benefitted from many years of research on the software evolution process. Human factors

engineering has been incorporated into many of the newer products available today, so users with virtually no data processing experience can accomplish work using a 4GL. Database techniques have been incorporated in many 4GLs to assist users in the process of cataloguing their information resources and controlling parts of the software evolution process. A 4GL may be used to quickly generate and modify functioning models of systems to allow users to exercise ideas about information needs.

An excellent use of 4GLs which has found a significant following within the last two or three years is in the evolution of new systems specifications. Using the 4GL as a prototyping tool, a useful model or prototype of a proposed system can be developed quickly. The developer can immediately view the prototype's results, make modifications and enhancements, and repeat the process until the prototype embodies most, if not all, of the concepts that the developer has in mind.

Upon acceptance by the user, the data structures, screen formats, report commands, and other system elements defined in the prototyping process then become the basis of the new system's specification. Evolution may proceed using the 4GL, a lower generation language, or a combination of both. References [Boar84, Conn84, EDPA84, EDPA85] propose methods for implementing prototypes in a 4GL and discuss the prototyping process in detail.

As a prototyping tool and as a day-to-day working tool, a 4GL can be used by both technical and nontechnical users. With this dual utility comes the problem of resource control. Data processing managers are learning how to implement controls on the 4GL-based software evolution process. Difficulties arise, however, in introducing these controls to end-users. Controls such as documenting programs, reports, screens, and data files; configuration management; and code optimization are new concepts to nontechnical users. They do not see these controls as necessary in view of the primary responsibilities of their jobs. Instead, these controls may be seen as arbitrary and sometimes politically motivated by the data processing organization. DP managers are faced, not only with the problem of controlling resources, but of educating the nontechnical user on the value of data controls.

There are numerous documented cases to support the need for controls in using 4GLs [Inte84b, EDPA85, EDPA84]. Large mainframes have been "brought to their knees" under the deluge of applications developed by nontechnical end-users and professional data processors in unrestrained use of 4GLs. There are also many cases that show the benefits of controlling computing resources in similar situations.

As more is learned about software evolution, the concept of a **Fourth Generation Language** will evolve into a broader spectrum.

Today's market is proceeding in several directions at once with the suggestion of new tools and capabilities to be added to the 4GL concept in the not too distant future. Included among these are expert system components to assist developers and users in the requirements and design phases of system development, testing methodologies and tools to assist in determining reliability of the software produced, and truly nonprocedural methods of specifying systems utilizing graphics interfaces and very high level language components.

Fourth Generation Languages will have a profound effect on applications evolution for a long time to come. Perhaps they cannot accommodate the whole range of applications needed in today's information systems. They are, nevertheless, important and formidable tools in the evolution of data processing systems.

REFERENCES

[Amer82] American National Standards Institute Committee X3 - Computers and Information Processing, "American National Dictionary for Information Processing," X3 **Technical Report** X3/TR-1-82, Computer and Business Equipment Manufacturers Association (CBEMA), Washington, DC, 1982.

[ANSI85a] X3 Committee, H2 Subcommittee on Database Standards, "Database Language SQL, ANSI Draft Proposed American National Standard X3.xxxx-198x," American National Standards Institute, March 1985, New York, New York.

[ANSI85b] X3 Committee, H4 Subcommittee on Information Resource and Dictionary System Standards, "American National Standard Information Resource Dictionary System: Parts 1, 2, 3, and 4," American National Standards Institute, New York, New York, 1985.

[ANSI85c] Database System Study Group, Database Architecture Framework Task Group, American National Standards Institute, X3 Committee/SPARC, "Reference Model for DBMS Standardization," **NBSIR 85-3173**, National Bureau of Standards, U.S. Department of Commerce, May 1985.

[Atre85] Atre, Shaku, "Will the Real Fourth-Generation Language Please Stand Up," **Computerworld**, Vol. XIX No. 24, June 17, 1985, pp. 57-70.

[Auer85] "Data Base Management," **Report No.** 23-02-04, Auerbach Publishers, Inc., Pennsauken, New Jersey, 1985, p. 4.

[Boar84] Boar, Bernard H., "Application Prototyping," John Wiley and Sons, Inc., New York, New York, 1984.

[CODA85] CODASYL Screen Management Committee, "CODASYL Screen Management System Journal of Development," V1.0, July 1, 1985.

[Conn84] Connell, John, and Linda Brice, "Rapid Prototyping," **Datamation**, August 15, 1984, pp. 93-100.

[EDPA84] "Fourth Generation Languages and Prototyping," EDP **Analyzer Special Report**, Canning Publications, Inc., Vista, California, 1984, pp. 32.

[EDPA85] "Speeding Up Application Development," **EDP Analyzer**, Vol. 23, No. 4, April 1985, pp. 1-16.

[Gold85] Goldfine, Alan, and Patricia Konig, "A Technical Overview of the Information Resource Dictionary

System," **NBSIR 85-3164**, National Bureau of Standards, Institute for Computer Sciences and Technology, 1985.

[Gran85] Grant, Dan, "Hammers, Monkeys, and the Fourth Generation," **Information Center**, Vol. 1 No. 7, July, 1985, pp. 16-17.

[Hare82] Harel, Elie C., and Ephriam R. McLean, "The Effects of Using A Nonprocedural Computer Language on Programmer Productivity," Information Systems Working Paper #3-83, Computers and Information Systems Research Program, Graduate School of Management, University of California, Los Angeles, California, November 1983, pp. 26, 26 refs.

[Heff85] Heffernan, Henry, "Trade-offs Found in FOCUS, COBOL Test," **Government Computer News**, Vol. 4 No. 11, June 21, 1985.

[Inte84] International Organization for Standardization, "Information Processing Systems - Open Systems Interconnection - Basic Reference Model," **ISO 7498**, First Edition--1984-10-15.

[Inte84a] International Data Corporation, "Application Generators," **IDC #2545**, Framingham, Massachusetts, October 1984, pp. 48.

[Inte84b] International Data Corporation, "Fourth Generation Languages: Information Generators to Meet Information Needs," **IDC #2563**, Framingham, Massachusetts, October 1984, pp. 41, 23 refs.

[Inte84c] International Data Corporation, "New Programming Languages," **Research Memorandum IDC #2483**, Framingham, Massachusetts, May 1984, pp. 22.

[Mart82] Martin, James, "Application Development Without Programmers," Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[Mart84] Martin, James, "Ideal DP Development Facility for Commercial DP," **Report on High-Productivity Languages**, Technology Insight, Inc., 1984, pp. 1.3/1+.

[Samm69] Sammet, Jean E., "Programming Languages: History and Fundamentals," Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969.

[Sipp72] Sippl, C. J., and C. P. Sippl, "Computer Dictionary and

Handbook," Second Edition, Howard W. Sams and Co., Inc., Indianapolis, Indiana, 1972.

A. GLOSSARY

Advanced Fourth Generation Language -- A Fourth Generation Language that has specific capabilities in addition to those required of a minimal 4GL.

Application Generator -- A 4GL that produces functional programs by interpreting and directly executing very high level commands specified by the user. No source code is produced.

Code Generator -- A 4GL that produces program source code in a third, second, or first generation language. This code may then be compiled and executed.

Data Dictionary -- A database that specifically organizes documentary information about an application or organization. This information is then used by other applications as a source of data for specific purposes in application evolution and execution, such as for providing descriptions of data used in reports and transactions.

Data Management Functions -- A required function of 4GLs to provide capabilities to describe, store and retrieve, and perform ancillary tasks in the management and safekeeping of application data.

Decision Support System -- Application systems that are used in the high level management support processes of an organization rather than in the day-to-day business operations. Used to abstract data into trends to support planning for future operations.

Dialogue Manager -- A special purpose computer program that is used to manage and control interactions (the dialogue) between human users and the applications that execute on the computer.

End-user Language -- A high level programming language that allows end-users who are not data processing professionals to define applications and interact with a computer in a nonprocedural fashion.

First Generation Language -- The languages originally used to program computers. They were defined at the machine or hardware level of the computer and were therefore called machine languages. They consisted of strings of zeros and ones in specific patterns that represented operations and operands to the central processing unit. Virtually all present-day computers still use machine languages, but programming is done in high level languages and automatically translated or compiled into machine language.

Functional Model -- A representation of an object or concept through abstract symbols and narratives that denote each specific function or service to be provided by the object or concept.

Menu -- A tabular list of choices and responses displayed by a program on a terminal screen, and that controls the sequence of events chosen by a human user in interacting with the program.

Minimal Fourth Generation Language -- A high level nonprocedural language that provides basic user functions, data management functions, and system functions for defining and interacting with data processing applications.

Nonprocedural Language -- Programming language based on manipulation of objects and specification of what a program should do rather than how the program should do it.

Procedural Language -- Commonly used programming languages that specify a program in discrete steps or algorithms defining how the program should behave.

Program Generator -- A special purpose computer program that is used to generate source code in a third or second generation language from high level procedural or nonprocedural statements. The program generator may use libraries of preprogrammed subroutines or blocks of commonly used source code statements to produce the final generated program.

Query Language -- A high level language designed specifically for interacting with a database management system (DBMS) to retrieve and manipulate data stored within the DBMS.

Report Generator -- A special purpose computer program with which a user may create and modify reports to be automatically generated by computer.

Screen Formatter -- A special purpose computer program used for defining terminal display screen layouts and for defining the types of interaction that can occur between the user and the screen.

Second Generation Language -- A computer programming language that consists of symbolic names and numbers which are translated into first generation language instructions by means of an assembler; generally called assembly language.

Software Development Productivity -- The relative capacity of combinations of organization, technique, methodology, and automated tools to produce and maintain software.

Software Life Cycle -- The definition and organization of phases through which a software system goes in its lifespan. Typically, these phases are requirements analysis, functional specification, design, code, test, installation, and maintenance.

Software Prototype -- A model or less-than-complete version of a proposed software application that is developed to verify and validate user requirements before major specification work is done.

System Functions -- Functions that provide the support services necessary to allow the user of a 4GL to define and access applications in relation to the constraints of the 4GL's environment. Examples are file management functions, communications, peripheral device control, etc.

Third Generation Language -- Languages that consist of high level procedural and declarative statements which are compiled into second or first generation code. Examples of 3GLs include COBOL, ADA, C, and Fortran.

User Functions -- Those functions that define those capabilities and services necessary to provide a high level dialogue between the 4GL and users of the 4GL. Users of a 4GL may include humans and other application systems.

* U.S. Government Printing Office: 1986 0-491-070 (40092)

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBS/SP-500/138	2. Performing Organ. Report No.	3. Publication Date June 1986

4. TITLE AND SUBTITLE

Computer Science and Technology:
A Functional Model for Fourth Generation Languages

5. AUTHOR(S)

Gary E. Fisher

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

National Bureau of Standards
Gaithersburg, Maryland 20899

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number 86-600545

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

The Fourth Generation Language (4GL) functional model places 4GL in the context of programming language evolution, and describes the functions provided within this context. An example implementation of a 4GL is also presented.

A 4GL is a software system that provides integrated functions for developing interactive on-line data processing applications. These functions are defined as: 1) user functions that define those services and capabilities necessary to provide a high level dialogue between the 4GL and users of the 4GL; 2) data management functions that provide capabilities to describe, store and retrieve, and perform ancillary tasks in the management and safekeeping of application data; and 3) system functions that provide the support services necessary to allow the user of 4GL to define and access applications in relation to the constraints of the 4GL's environment.

A typical implementation of 4GL distributes pieces of these functions over various components, such as a DBMS, query language, data dictionary, screen formatter, report generator, and high level procedural language.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

application generation; code generation; end user computing; Fourth Generation Language; high level language; nonprocedural language; 4GL

13. AVAILABILITY

☒ Unlimited
☐ For Official Distribution. Do Not Release to NTIS
☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

36

15. Price

USCOMM-DC 6043-P80

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NBS Technical Publications

Periodical

Journal of Research—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Bureau of Standards
Gaithersburg, MD 20899