AN EVALUATION OF LANGUAGES FOR THE IMPLEMENTATION OF INFORMATION STORAGE AND RETRIEVAL SYSTEMS

C. J. Crouch
Center for Informatics Research
University of Florida

INTRODUCTION

Some of the most important and interesting questions that arise during the development of any mechanized information storage and retrieval system relate to the programming language(s) to be used for the implementation of that system. The term "programming language" as used here reflects Sammet's [4] definition; thus it implies the use of a higher level language. Assuming that a system is to be designed to fulfill a specific storage-retrieval requirement, the system designer is faced with a number of questions with regard to existing languages, such as:

(1) Do any programming languages exist which can satisfactorily provide all features desired in the implementation of this system?

(2) If so, does one particular language provide facilities which make it more applicable to the implementation of the system than any other language under consideration?

(3) If so, can each system component be implemented most effectively in this language, or can the requirements of these individual components best be met by different languages?

The answers to these questions may be of varying interest depending on such factors as system complexity and the availability of the language(s) in question. The more complex a retrieval system in terms of the features it provides, the more relevant and important this area of inquiry becomes (assuming the availability of several possible implementation languages so that a choice can be made).

What criteria can be applied in order to determine which programming language is best suited for the implementation of a particular information storage and retrieval system? The following approach results in a quantitative measure of a language's ability to provide the necessary features. Steps taken include:

(1) Development of a model of the system.

(2) System analysis, based on the model.

(3) Quantitative evaluation of a language's ability to provide basic capabilities.

(4) Application of an algorithm to determine the effectiveness of a language for implementing the system.

These steps are illustrated by applying them to a generalized information storage and retrieval system.

GENERALIZED ISR MODEL

The problems involved in selecting a programming language for implementation purposes can be demonstrated by investigating a generalized document or reference retrieval system. A model of such a system appears in Figure 1. This system is of special interest because of the many distinct tasks which must be performed within it. The terms "information storage and retrieval (ISR) system" and "document retrieval system" are used synonymously within this paper; both refer to a system which retrieves documents or document references in response to a specific request.

The ISR system may be further described as follows. The User (generally assumed to be unfamiliar with mechanized ISR systems or digital computers) inputs his query to the system. The query is taken by the Logical Processor (or Query Formalizer) which operates on the query and outputs to the Selector the query in terms of descriptors or index terms. The Selector uses the descriptors to search the Descriptor (or Index) File. The resultant specifications, i.e., pointers to those documents which have successfully satisfied the search according to some pre-established criteria, are returned to the Selector. The Selector, which may or may not operate on these specifications, sends the final selected specifications to the Locator, which uses this information to search the Document File. The documents themselves are returned first to the Locator and from there to the User.

In addition to that information input by the User, document data may enter the system through the Data Block. This information is taken by the Analysis Block, which operates upon it and produces two

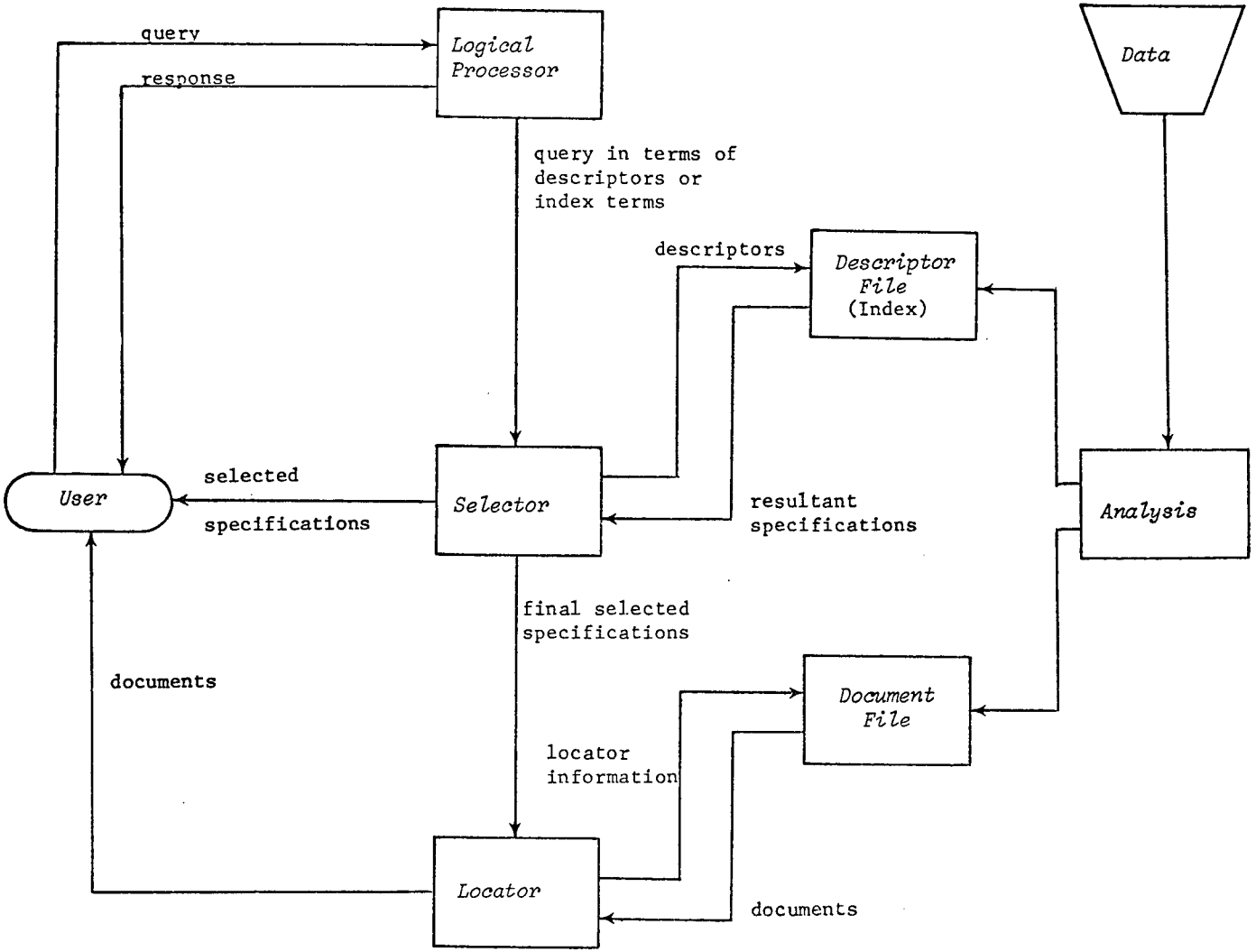

Figure 1. Model of a Generalized Document Information Storage and Retrieval System

outputs--a representation of the document in terms of descriptors, to be stored in the Descriptor File along with a pointer to the document in the Document File, and a reference to the document itself (i.e., an identifier) to be stored in the Document File.

Two important possible feedback loops in the system are:

(1) From the User to the Logical Processor and back to the User, and

(2) From the User to the Logical Processor to the Selector and back to the User.

In the first case, the Logical Processor is asking the User to re-formulate, clarify, or augment his query. In the second case, the Selector is asking the User to give his approval of the selected specifications, to choose from amongst them those which best suit his needs, etc.

This description of the operation of an ISR system reveals the many different types of activities that occur within its confines. The system under consideration is assumed to be more than rudimentary; for example, it might accept a query in some semi-English form and allow query clarification through system-user interaction. Factors affecting the choice of a language (or languages) for such a system include:

(1) Natural language processing (Analysis)

(2) Interactive man-machine dialogue (Logical Processor)

(3) File organization (Analysis in forming Document and Descriptor files)

(4) File search (Selector and Locator).

The problem is to establish the relationship between these factors and the choice of a programming language.

SYSTEM ANALYSIS

Using the model of the generalized ISR system as a basis, an analysis of any particular system proceeds as follows:

(1) Designation of the subsystems to be included

(2) Identification of the tasks to be accomplished within the component parts and subsystems of the total system

(3) Detailed description of these tasks in a defined notation (a functional description or definition of the components) [3]. Such a description reveals the basic structure of the system by requiring an exact specification of its operation at every point.

We contend that basic system structure is revealed in terms of system components, the tasks and subtasks which must be accomplished within each component, the various information structures which may be employed in the accomplishment of each task, and the operators which are required at that point in the system [1]. Thus the system may be represented in an hierarchical arrangement or tree structure (Fig. 2). The root of each level 2 subtree is represented by the system component and the terminal nodes are represented by the operators and/or data structures which are required to accomplish each specific task of that component.

LANGUAGE EVALUATION

In considering the applicability of various programming languages for the implementation of a system, the ability of the language to provide specific operators and data structures must be determined for the tasks to be accomplished. The necessary operator capabilities must be available in that language; the data structure requirements are less stringent since different data structures frequently may be employed to accomplish the same task.

How can the ability of a programming language to provide specific operator and data structure capabilities be measured? One such measure is the number of statements (or SEU's [4]) required in that language to provide a necessary facility. The minimal number of SEU's necessary to provide a specific operator capability (if the operator is not already a feature of the language) may be estimated without great difficulty. The situation is much more difficult to assess with regard to the data structures; a certain data structure capability either may or may not be present in a language. In the latter case, it can generally be simulated.

The match between ISR system requirements and language capability in terms of operators and data structures can be evaluated qualitatively or quantitatively. We attempt a quantitative approach using the following estimation procedure:

(1) For operators--obtain a normalized estimate, a value between 1.0 and 0.0 for each operator, depending upon the estimated number of SEU's required. A bias may be added, representing the cost of implementation. Thus the more SEU's required to provide a specific operator, the closer its value lies to 0.

(2) For data structures--a value of 1.0, .5, or 0.0 is assigned, depending on whether that data structure is present, can be simulated, or is not present within the language.

Note that we have not described our approach as objective, for a certain degree of subjectivity certainly remains. However, our ultimate objective is to provide a comparative evaluation, not an absolute measure; hence the crucial point is not

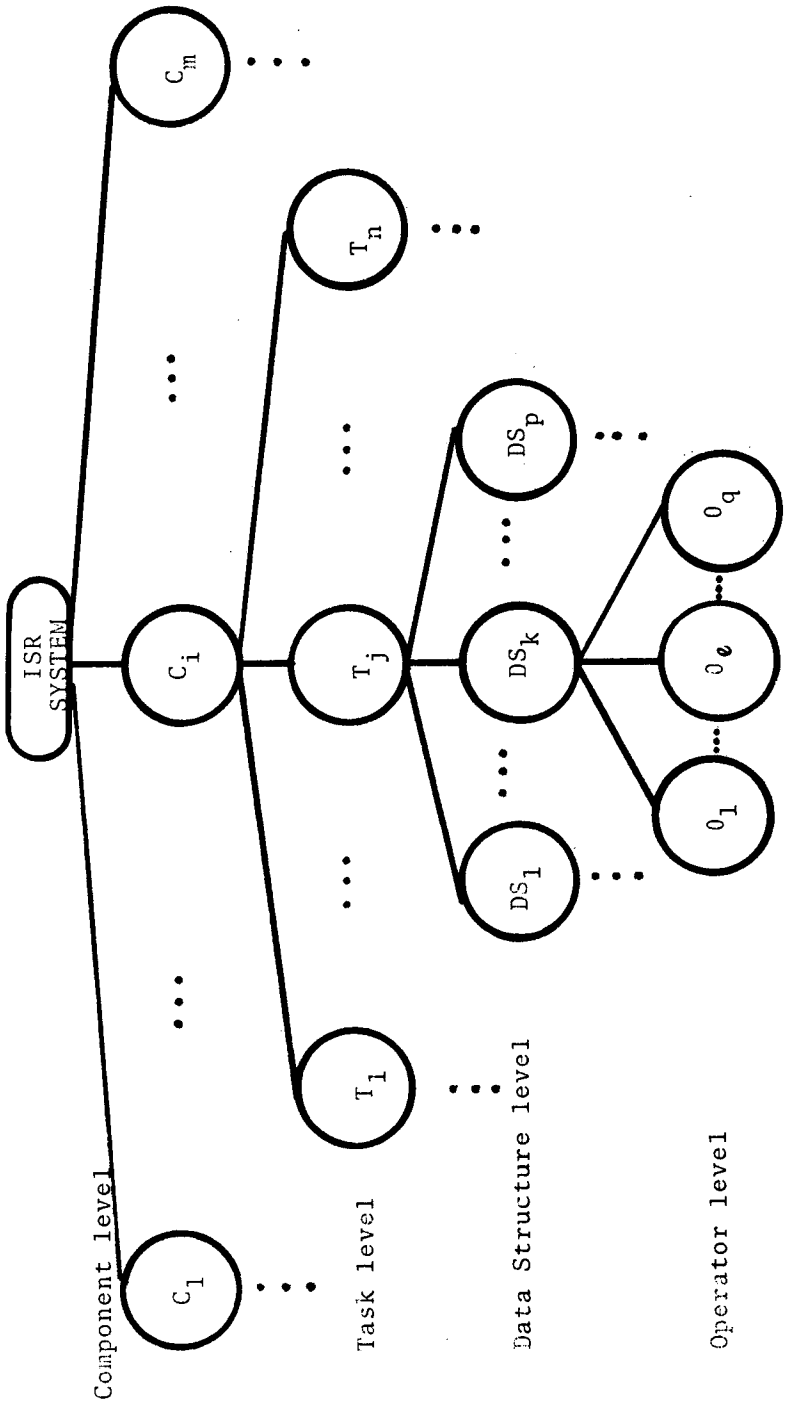

Figure 2. Hierarchical Representation of ISR System

to allow subjectivity to affect the evaluation of one language more than another.

ALGORITHM FOR LANGUAGE SELECTION

The systems analyst is allowed to assign weights to the various levels of the tree which represent the ISR system (namely, at the component, task, and structure level) on a basis that reflects the design of his system. The value of each terminal node can be determined by multiplying the weight of each node in the branch by its level number (with the component at level 1) and forming the sum of these numbers. (Weights may be assigned on an arbitrary basis of say, 0 to n, with the restriction that weights at the same level of each subtree should all sum to n.)

Values are now available by means of the SEU estimates and the biasing procedure, which represent the ability of a language to provide certain operator and data structure capabilities. Values which represent the system's requirements in terms of operators and data structures are also available (via the hierarchical structure of the system and the weighting procedure). Using these values, two vectors are created: one represents the operator requirements of the system and the second represents the ability of a possible implementation language to fulfill these requirements. The dot product of these two vectors yields a number which represents the ability of the language to provide the necessary system operator requirements. The same general procedure is followed in arriving at a similar measure of a language's ability to provide the necessary data structure capabilities. See [1] for more detailed discussion.

RESULTS AND CONCLUSIONS

This approach has been investigated by applying it to 4 operating retrieval systems, namely the Query, GIPSY, BIRS, and SMART systems [2]. All of these systems fall within the confines of the generalized system. The languages in question are FORTRAN IV, SNOBOL4, and PL/I. The experimental data and results may be found in [1]. Analysis of these results shows that an equal operator capability is provided by SNOBOL4 and PL/I, with a lesser capability being provided by FORTRAN IV for each of the systems investigated. The results are similar with respect to the data structure capability of each language, with the exception of the SMART system. In this case, PL/I provides a capability superior to that of SNOBOL4.

The results affirm that as the retrieval systems become increasingly complex, they require more in the way of language capabilities. Whereas a comparatively simple ISR system may be implemented by a variety of different programming languages with little loss of efficiency, a complex system is more efficiently implemented by a language with a greater capability in terms of data structures and operators. Additional testing shows that the results are not unduly sensitive to variations in the input parameters, i.e., the estimates derived by the system designer.

The technique described within this paper is an attempt to determine the applicability of a programming language to (any) ISR system implementation in terms of a measurable quantity. Additional quantitative factors in the choice of a language might include such considerations as average time of compilation and execution and economy of storage. Qualitative considerations would include the training and abilities of the programmer(s) concerned, availability of the language, efficiency of the implementation, ease of conversion, the design purpose of the language, special features such as debugging aids available, the physical environment in which the language is to be used, file management facilities, etc. [4]. Some of these factors may override other considerations.

Sammet [4], in discussing factors in the choice of a language, says that there is currently no scientific or even logical way to choose the best programming language for a particular situation. She attributes part of the difficulty to the fact that the situation itself is usually poorly defined. The approach described within this paper relies upon a complete functional definition of the system in question. Initial investigations indicate that it is a possible approach in solving the problem.

REFERENCES

1. Crouch, C. J. Language Relations in a Generalized Information Storage and Retrieval System. Unpublished Ph. D. Dissertation, Southern Methodist University, 1971.

2. Crouch, C. J. and Crouch, D. B. An Analysis of Document Retrieval Systems Using a Generalized Model. In: Software Engineering, Vol. 3, Julius Tou, Ed. Academic Press, New York, 1973.

3. Nance, R. E. and Crouch, C. J. A Functional Representation of a Generalized Information Storage and Retrieval System. Technical Report CP 72006, Southern Methodist University, 1972.

4. Sammet, J. E. Programming Languages: History and Fundamentals. Englewood Cliffs, N. J., Prentice-Hall, 1969.